# Collective Decision Making through Self-regulation

## Mechanisms and Algorithms for Self-regulation in Decision-Theoretic Planning

Scharpff, J.C.D.

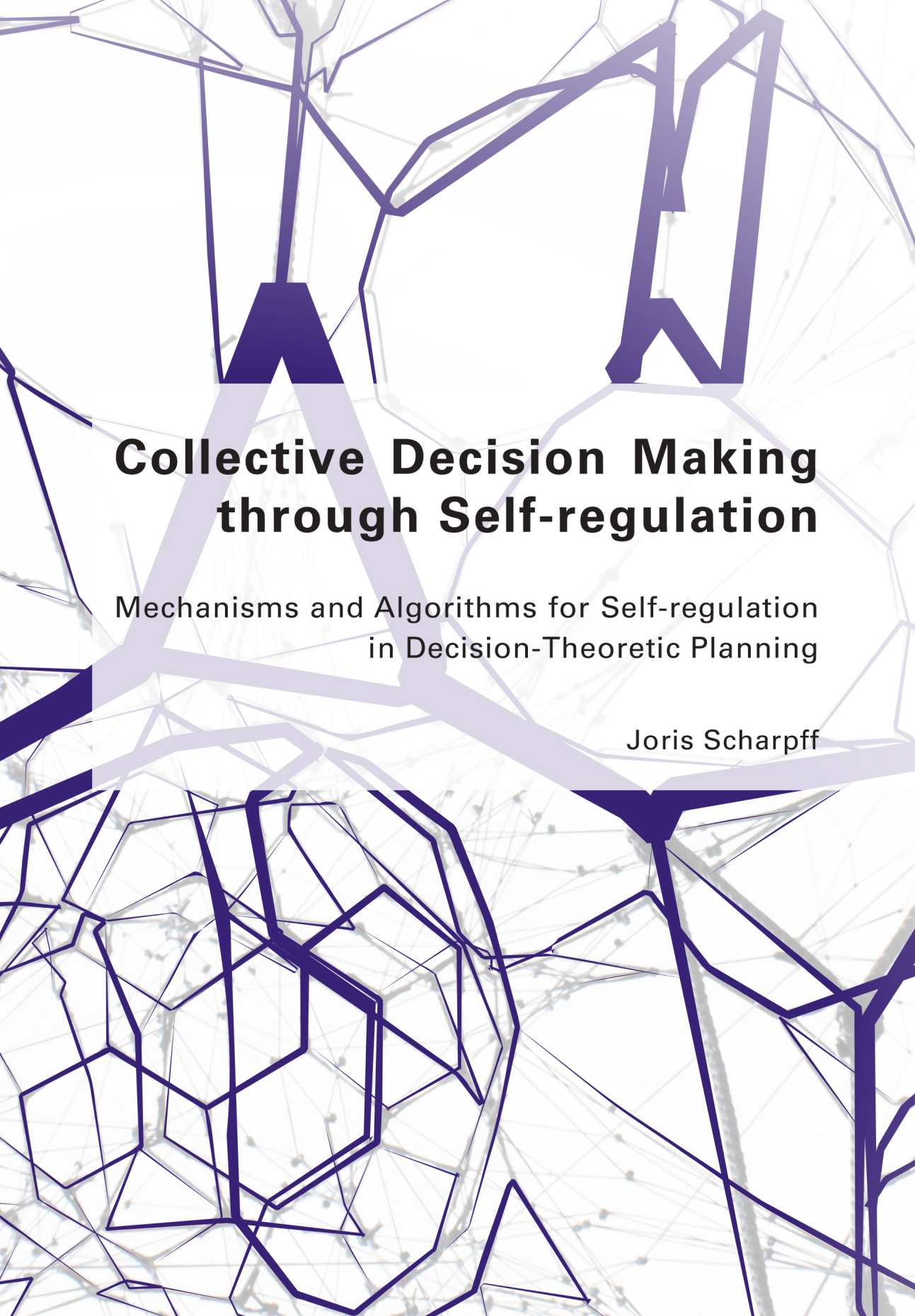
**DOI**
10.4233/uuid:63d60259-e0bf-4852-b38b-c1157c390b0d

**Publication date**
2020

**Document Version**
Final published version

**Citation (APA)**
Scharpff, J. C. D. (2020). *Collective Decision Making through Self-regulation: Mechanisms and Algorithms for Self-regulation in Decision-Theoretic Planning*. [Dissertation (TU Delft), Delft University of Technology]. TRAIL Research School. https://doi.org/10.4233/uuid:63d60259-e0bf-4852-b38b-c1157c390b0d

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.



# Collective Decision Making through Self-regulation

## Mechanisms and Algorithms for Self-regulation in Decision-Theoretic Planning

Joris Scharpff

# Collective Decision Making through Self-regulation

Mechanisms and Algorithms for Self-regulation in Decision-Theoretic Planning

## Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof. dr. ir. T.H.J.J. van der Hagen;
Chair of the Board for Doctorates
to be defended publicly on

Friday 20 November 2020 at 12:30 o'clock

by


## Joris Carl Derk SCHARPFF

Master of Science in Computer Science, Delft University of Technology, the Netherlands
born in Leiden, the Netherlands

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

| | |
|---|---|
| Rector Magnificus, | chairperson |
| Dr. M.M. de Weerdt, | Delft University of Technology, promotor |
| Dr. M.T.J. Spaan, | Delft University of Technology, promotor |
| | |
| *Independent members:* | |
| Prof. dr. ir. A.R.M. Wolfert, | Delft University of Technology |
| Prof. dr. R.R. Negenborn, | Delft University of Technology |
| Prof. dr. M.H.M. Winands, | Maastricht University |
| Dr. E.H. Gerding, | University of Southampton |
| | |
| *Other members:* | |
| Dr. A.W. Stam, | Almende B.V. |




This research is supported by NGInfra and Almende BV (03.21.ALM "Dynamic Contracting in Infrastructures"), NWO DTC-NCAP (#612.001.109) and NWO VENI (#639.021.336).

| | |
|---|---|
| *Keywords:* | Self-regulation, decision-theoretic planning under uncertainty, dynamic mechanism design, serious gaming |
| *Printed by:* | Haveka BV, the Netherlands |
| *Front & back:* | Sjoerd van der Vlugt |

TRAIL Thesis Series no. T2020/17, the Netherlands TRAIL Research School

TRAIL
P.O. Box 5017
2600 GA Delft
The Netherlands
E-mail: info@rsTrail.nl

ISBN: 978-90-5584-274-2



An electronic version of this dissertation is available at http://repository.tudelft.nl/.

# Summary

Over the two last decades, performance-based contracting has become the prevalent approach in tenders for service delivery in public-private partnerships, changing the way in which public organisations outsource their operations. Whereas traditional regulatory contracts exhaustively stipulate the work, performance-based contracts state only the desired output and the payment mechanism that is used. Under such an agreement the service providers are free to plan and execute the work when and how they see fit, while their reward (or fine) depends on their (negative) contribution to the contracted objective. In other words: their performance. This approach to contracting offers many promising advantages over complete governance such as increased flexibility, preservation of autonomy and authority, stimulation of performance and innovation, shared responsibility, less demand on governmental resources and, as a consequence, better use of public funding.

The practical successes, however, have so far been limited to bilateral agreements. It is currently not clear how to realise these results in group contracts with competitive service providers. In an endeavour to transfer the aforementioned advantages to multilateral agreements, both industry and academics now shift their attention to the potential of performance-based contracting in group tenders. Of particular interest is the use of monetary incentives to not only maximise performance but also instill *self-regulation*, that is, achieving "an organization regulating itself without intervention from external bodies"[1]. If applied well, self-regulation implements the key ideas of performance-based contracting, i.e. letting the service providers 'do what they do best' and account them for their performance, while additionally stimulating them to coordinate their operations amongst themselves. Nonetheless, the limited control over the group decision-making process and the self-interest of service providers lead to greater uncertainty regarding the outcomes thereof, an increased potential for opportunistic behaviour, a misalignment of (societal) objectives and an overall greater complexity experienced by group members and current planning algorithms. The central challenge for this thesis is to overcome these obstacles that prevent a transfer of the benefits and successes of bilateral performance-based contracting into group tenders. This is formulated in the main research question as "Can algorithmic techniques be employed to efficiently coordinate planning in self-regulating contracts and ensure successful outcomes while preserving the autonomy and interests of the agents?".

[1] *Self-regulation, Oxford Dictionary.*

This thesis approaches the main challenge from a computer science perspective, using a combination of the fields of decision-theoretic planning, game theory and serious gaming. In particular, the challenge is addressed in two parts: 1) the design of incentives that instill self-regulation and counter opportunistic behaviour by aligning interests, and 2) solving the decision-coordination problem that service providers face when these incentives are implemented in group contracts. As motivating example, a characteristic problem from the domain of infrastructural maintenance planning is used. This problem models the scheduling of maintenance activities for a highway network by a team of service providers and highlights the tension between the objectives of maximising individual profit and minimising the traffic hindrance. The latter objective results in fines for the service providers relative to the traffic hindrance they *jointly* cause, thereby creating a dependency between service providers and making it in their best interest to coordinate their decisions. This problem is formulated as a multi-agent optimisation problem, the MAINTENANCE PLANNING PROBLEM (MPP), and used in the chapters throughout this thesis to demonstrate the algorithms and techniques contributed by this work.

Chapters 3 to 5 address the coordination of the service providers, assuming the existence of a performance-based incentive mechanism. The focus in these chapters is on producing *decision policies* during a preliminary planning phase that, if followed during execution, in expectation optimise the total value realised from the execution of contracted work. Chapter 3 presents a mathematical formulation of the MAINTENANCE PLANNING PROBLEM and demonstrates how it can be encoded as a sequential decision-making problem in the *Markov Decision Process* (MDP) model. This encoding enables a vast range of existing techniques to produce optimal decision policies for MPP. Additionally, an approximate Monte-Carlo Tree Search approach is discussed to produce decision policies more efficiently, albeit at the loss of quality. That is, these decision policies are not guaranteed to maximise the overall expected value when implemented but they can be developed significantly faster. Chapter 4 resumes the pursuit of optimal decision policies and presents a more efficient approach that exploits the structure of self-regulating planning problems such as MPP. More specifically, by grouping result-equivalent decision sequences, the structure of the reward function in problems such as MPP can be represented more compactly. This in turn allows for a more efficient policy search algorithm, which is the heart of the `Conditional Return Policy Search` (`CoRe`) solver. This novel multi-agent MDP solver outperforms the current state-of-the-art on MPP problems and enables finding optimal decision policies for instances that were previously deemed impossible to solve.

Thereafter Chapter 5 addresses the multi-objective nature that is inherent to MPP. While Chapters 3 and 4 implicitly assumed that the objectives of maximising revenue and minimising traffic hindrance can be operationalised into a single monetary value, Chapter 5 models these goals as two distinct optimisation criteria that need to be balanced explicitly through a linear *scalarisation function*. This function captures the relative importance of both objectives and allows decision makers to specify (and alter) the *objective weights* during the execution of the decision policy. This flexibility is paired with a substantial increase in computational complexity, however, because it requires finding a set of decision policies that optimises the total reward for every possible combination

of objective weights. While restricting to only linear scalarisation functions significantly reduces the weight combinations that need to be considered, the computational effort required to produce an optimal *Convex Coverage Set* (CSS) is still prohibitively large for problems such as MPP. Therefore Chapter 5 proposes two algorithms to approximate this set, the `Approximate Optimistic Linear Support` (`AOLS`) and `Scalarised Sample-based Iterative Improvement` (`SSII`). The former is an approximate version of the existing `OLS` algorithm that can use any approximate $\epsilon-$MDP solver to produce a solution set with a value that is guaranteed to be at least $(1-\epsilon)$ times the value of the optimal CCS. `SSII` employs sampling to iteratively improve its approximation of the optimal CCS in particular areas of the search space, thus often leading to better approximations within a specific region of focus albeit without theoretical guarantees.

Chapters 6 and 7 shift their attention from decision coordination to the self-interested nature of agents. As opposed to earlier chapters, these chapters assume that the coordination of decisions is performed by the agents themselves and focus on approaches that address the self-interested and autonomous nature of agents. Chapter 6 presents two methods to overcome different hurdles of self-interest. The first, the *Dynamic Maintenance Mechanism*, uses incentives to in expectation maximise the value of contracted work and prevent opportunistic behaviour, i.e. manipulation of the mechanism to increase personal gain. This mechanism provides the strongest guarantees but also has steep requirements: it must compute optimal decision policies many times during its execution and demands from agents that they disclose their exact decision-making model. The former condition is hard to satisfy due to the complexity of MPP and the second due to the sensitive nature of this information. Hence a second approach is proposed based on best-response planning. In this approach service providers iteratively submit their own decision policy as a response to the current joint decision policy of other agents until the joint policy is acceptable to all. Computing single-agent decision policies is less demanding, only requires sharing the decision themselves and not the underlying information, and opportunistic behaviour can be mitigated by responding with a countermanding policy in the next iteration. The trade-off is that this mechanism only guarantees that eventually the agents will settle on a joint policy, but the quality thereof may be arbitrarily poor.

Chapter 7 evaluates the concept of self-regulation in a setting with human decision makers through the use of a *serious game*. The "Road Maintenance Game" simulates the MAINTENANCE PLANNING PROBLEM in a self-regulating group contract based on monetary incentives and lets human players play as service providers that need to plan their maintenance work and coordinate with other players to optimise their profits. Then, by analysing the decision made by the players and comparing them to their a priori preference, the effectiveness of monetary incentives to influence behaviour and the role of social relationship on this influence are investigated. The observations and measurements made over the course of seven gaming sessions show that incentives are an effective means to influence decision making but sometimes leads to unintended behaviour of the players and can result in undesirable competition within the group. When social relations between players are stronger, however, the same incentives do successfully incite self-regulation. Although these results are not yet sufficient to con-

clude that self-regulation always ensures satisfactory outcomes in group tenders, they provide strong evidence for its potential in particularly collaborative settings such as strategic partnerships or alliances. Moreover, the experiments once more confirm the paramount role of relationships in partnerships and suggest the social dimension as a key enabler for self-regulation.

The conclusion that follows from Chapters 3 to 7, and hence the answer to the main research question, is that no one-size-fits-all method exists to implement self-regulation in contracts that simultaneously satisfies all conditions. Further research is certainly needed to bring the tools, techniques and learnings contributed by this thesis into real-world contracts and should be complemented by counselling from other disciplines such as contracting theory, (public) network management, legal studies, behavioural psychology and social sciences. Notwithstanding, this thesis lays the mathematical foundation for the implementation of self-regulation in contracts through monetary incentives, guides the design of incentive mechanisms to realise self-regulation, contributes coordination methods to optimise the value of self-regulating contracts and demonstrates the potential of monetary incentives to incite self-regulation in human decision makers. Concurrently, the decision coordination algorithms presented here advance the current state-of-the-art in sequential decision making, solving instances of planning problems that have so far been considered intractable, and offer new angles for future research. Promising next steps are applying the encoding and 'flattening' of multi-agent MDPs to other problems and developing efficient representations, extending the potential of `Conditional Return Policy Search` to a broader model and incorporate pruning, heuristic or approximate techniques, a hybrid algorithm for approximate multi-objective planning and approximate mechanism design to counter strategic behaviour in complex sequential multi-agent decision-making problems. In parallel, the serious game offers an empirical framework to researchers and professionals for further (automated) exploration of agent strategies, opportunistic behaviour and contracting mechanisms with the ultimate goal of bringing self-regulation into real-world group tenders.

# Samenvatting

Dit proefschrift onderzoekt het samenwerken van meerdere partijen op basis van zelfregulering, een aanpak die recentelijk zeer relevant is geworden in de context van prestatiegericht contracteren met meerdere partijen. De populariteit van prestatiegericht contracteren is in de laatste twee decennia zodanig gegroeid dat het tegenwoordig geldt als voorkeursmethode bij de aanbesteding van diensten in (met name) publiek-private samenwerkingen. Anders dan traditionele, gereguleerde contracten waarin de werkzaamheden volledig worden voorgeschreven, beschrijven prestatiegerichte contracten alleen de gewenste eindresultaten en het betalingsmechanisme dat gehanteerd wordt. Binnen een dergelijk contract zijn de dienstverleners vrij om het werk naar eigen inzicht te plannen en uit te voeren, maar de vergoeding die ze voor het werk ontvangen wordt bepaald aan de hand van hun (negatieve) bijdrage aan het gecontracteerde doel. Met andere woorden, hun *prestaties*. Door dienstverleners af te rekenen op hun prestaties bieden dergelijke contracten substantiële voordelen ten opzichte van volledig gereguleerde contracten aan beide partijen. Dienstverleners ervaren een hoge mate van autonomie, flexibiliteit en eigen verantwoordelijkheid, waardoor ze efficiënt te werk kunnen gaan om het gecontracteerde doel te realiseren op hun eigen wijze. Innovatief werken wordt zodanig impliciet gestimuleerd: een dienstverlener kan immers meer verdienen door op een slimme manier het gestelde doel te realiseren, wat typisch ook weer voordelig is voor de contracterende partij. Daarnaast kan de contracterende partij zich volledig op de gewenste uitkomst concentreren zonder de exacte invulling van de werkzaamheden te bepalen, plannen en besturen. Daardoor hoeft de contracterende partij niet meer over specialistische kennis te beschikken en wordt de verantwoordelijkheid voor het resultaat grotendeels verlegd naar de uitvoerende partij. Onder meer deze wenselijke eigenschappen van prestatiegericht contracteren hebben geleid tot een veelvoudige en succesvolle inzet van prestatiegerichte contracten in de praktijk.

Echter, tot dusver zijn de successen van prestatiegericht contracteren voornamelijk gelimiteerd tot bilaterale contracten tussen de aanbesteder en een enkele dienstverlener. De belangrijkste open vraag op dit moment is dan ook hoe prestatiegerichte contracten succesvol ingezet kunnen worden in aanbestedingen met meerdere dienstverleners. De interesse gaat daarbij met name uit naar het gebruik van financiële stimuli om niet alleen prestaties te maximaliseren maar ook *zelfregulering* te realiseren, d.w.z. het bewerkstelligen van "een organisatie die zichzelf reguleert zonder beïnvloeding van

buitenaf"[2]. Een goede implementatie van zelfregulering binnen een contract realiseert de kernideeën van prestatiegericht contracteren, oftewel de dienstverleners laten doen waar ze goed in zijn en ze afrekenen op basis van hun prestatie, en stimuleert bovendien de dienstverleners om hun werkzaamheden onderling af te stemmen. Dit laatste kan gerealiseerd worden door dienstverleners te belonen of beboeten op basis van hun *gezamenlijke prestaties*, maar een dergelijke aanpak is niet zonder obstakels. Het verlies van controle over het beslisproces van de groep, de dynamiek van de interacties tussen meerdere partijen en het eigenbelang van dienstverleners brengen uitdagingen met zich mee die nieuw zijn ten opzichte van bilaterale contracten. Wanneer er meerdere partijen betrokken zijn neemt de onzekerheid wat betreft de uitkomst van een aanbesteding sterk toe. Daarnaast zijn de belangen van de partijen zelden gelijk waardoor mogelijk andere doelen nagestreefd worden dan gewenst. In het slechtste geval kan dit zich uiten in opportunistisch gedrag van de dienstverleners waarbij ze hun eigen winst proberen te vergroten ten koste van de aanbesteder, het gecontracteerde doel of andere dienstverleners. Tenslotte is de dynamiek tussen de partijen zeer complex voor zowel menselijke actoren als ook huidige planningsalgoritmen waardoor het optimaliseren van de waarde van een aanbesteding een bijna onmogelijke opgave is. De belangrijkste uitdaging voor dit proefschrift is derhalve het overwinnen van deze obstakels die het nu onmogelijk maken om de successen van bilaterale prestatiegerichte contracten over te brengen naar aanbestedingen met meerdere partijen. In het bijzonder wordt onderzocht hoe technieken en ideeën uit de informatica gebruikt kunnen worden om de voorgenoemde obstakels te adresseren. Dit is geformuleerd in de hoofdvraag als "kunnen algoritmische technieken worden ingezet om planningen efficiënt te coördineren in contracten met zelfregulering en verzekeren dat uitkomsten succesvol zijn waarbij de autonomie en de belangen van agenten behouden blijven?".

Dit proefschrift benadert de hoofdvraag vanuit een informaticaperspectief, waarbij gebruik wordt gemaakt van een combinatie van besliskundig plannen, speltheorie en serious gaming. Dit wordt gedaan in twee delen: 1) het ontwerpen van financiële stimuli die leiden tot zelfregulering en opportunistisch gedrag ontmoedigen door belangen te verenigen, en 2) het oplossen van het coördinatie probleem waar de dienstverleners mee te maken krijgen als stimulatiemechanismen worden geïmplementeerd in groepscontracten. Bij het presenteren van de bijdragen in beide delen wordt gebruik gemaakt van een karakteristiek probleem uit het domein van infrastructureel onderhoud, het MAINTENANCE PLANNING PROBLEM (MPP) ofwel het "onderhoud planningsprobleem". Dit probleem modelleert het plannen van onderhoudswerkzaamheden aan een wegennetwerk door een team van dienstverleners waarbij het conflict tussen het maximaliseren van individuele winst en minimalisatie van gezamenlijke verkeershinder centraal staat. Verkeershinder resulteert in dit model tot boetes voor de dienstverleners waarvan de hoogte wordt bepaald door de mate van hinder die ze gezamenlijk veroorzaken, m.a.w. hun gezamenlijke prestaties. Door dit boetemechanisme wordt het afstemmen van planningen onderling in het eigenbelang van de dienstverleners omdat ze individueel meer kunnen verdienen door gezamenlijke overlast te minimaliseren. Het MPP wordt geformuleerd als een wiskundig optimalisatieprobleem met meerdere agenten in hoofd-

[2] *'Self-regulation', Oxford Dictonary, vertaald uit het Engels.*

stuk 3 en wordt vervolgens in alle hoofstukken gebruikt om de algoritmen en technieken te demonstreren die door dit proefschrift worden bijgedragen.

Het eerste deel van het onderzoek, bestaande uit hoofdstukken 3 tot en met 5, richt zich op het tweede deel van de hoofdvraag: de coördinatie tussen de (planningen van) dienstverleners. In dit deel wordt dus nog niet onderzocht hoe betalingsmechanismen te ontwerpen, maar hoe de waarde van een aanbesteding gemaximaliseerd kan worden als een dergelijk betalingsmechanisme gebruikt wordt. In andere woorden, deze hoofdstukken behandelen planningsalgoritmen die het doel hebben een plan te produceren dat in verwachting de waarde van een aanbesteding optimaliseert zodanig dat er rekening wordt gehouden met de prestatieboetes en -beloningen. Hoofdstuk 3 presenteert een wiskundige formulering van het MAINTENANCE PLANNING PROBLEM en demonstreert hoe dit probleem kan worden getransformeerd tot een sequentieel beslisprobleem in het *Markov Decision Process* (MDP) model. Deze modellering maakt het mogelijk om gebruik te maken van een breed scala aan bestaande technieken te gebruiken om optimale plannen produceren voor MPP. Tevens wordt er een benaderingsaanpak besproken op basis van Monte-Carlo Tree Search om plannen efficiënter te vinden, ten koste van de kwaliteit van de oplossing. Dat wil zeggen, van deze plannen kan niet worden gegarandeerd dat ze de waarde van een aanbesteding maximaliseren maar ze kunnen wel significant sneller worden gevonden.

Hoofdstuk 4 vervolgt de lijn van optimale planning en introduceert een efficiëntere aanpak die gebruik maakt van de structuur van zelfregulerende planningsproblemen zoals MPP. Door beslispaden die tot gelijke beloningen leiden te groeperen, kan de structuur van de beloningsfunctie van problemen zoals MPP compacter worden opgeslagen. Op haar beurt kan deze compacte structuur worden gebruikt als basis voor een efficiënter algoritme voor het vinden van plannen, dat de kern vormt van `Conditional Return Policy Search` (`CoRe`). Dit nieuwe multiagent MDP zoekalgoritme presteert beter dan bestaande algoritmen op MPP problemen en maakt het mogelijk plannen te ontwikkelen voor instanties die voorheen niet opgelost konden worden.

Vervolgens gaat hoofdstuk 5 in op de multidimensionale aard van MPP. In eerdere hoofdstukken was (impliciet) aangenomen dat het maximaliseren van de winst en minimaliseren van de overlast samen uit te drukken zijn in één enkel totaalbedrag. In hoofdstuk 5 wordt deze aanname opgeheven en worden beide criteria gemodelleerd als twee separate doelen die expliciet afgewogen moeten worden middels een lineaire *wegingsfunctie*. Deze functie beschrijft het relatieve belang van beide doelen en maakt het mogelijk voor planners om *gewichten* toe te kennen aan doelen en deze aan te passen tijdens de uitvoering van het plan. Deze extra flexibiliteit gaat echter gepaard met een substantiële toename van de computationele complexiteit aangezien nu een oplossing gezocht wordt welke voor alle combinaties van gewichten een plan bevat dat de waarde van de aanbesteding maximaliseert. De toename in complexiteit kan gedeeltelijk worden beperkt door het limiteren tot alleen lineaire wegingsfuncties. Dit type functie wordt veelvuldig gebruikt in de praktijk en omschrijft afwegingen zoals een prijs per stuk of relatieve belangen tussen criteria. Door te beperken tot deze set functies wordt het aantal gewichtscombinaties dat moet worden beschouwd significant kleiner, maar het produceren van een optimale verzameling van plannen voor deze subset van problemen, bekend als de *Convex Coverage Set* (CSS), blijft ook met

de extra restrictie ondoenlijk voor complexe problemen zoals MPP. Om deze reden worden in hoofdstuk 5 twee approximatiealgoritmen geïntroduceerd die deze verzameling benaderen: `Approximate Optimistic Linear Support` (`AOLS`) en `Scalarised Sample-based Iterative Improvement` (`SSII`). De eerste is een benaderingsvariant van het bestaande `OLS` algoritme dat gebruik kan maken van elk $\epsilon - MDP$ algoritme om een CCS te produceren waarvan de waarde tenminste $(1 - \epsilon)$ maal de waarde van de optimale CSS heeft. Hierdoor is gegarandeerd dat het gevonden plan maximaal een factor $\epsilon$ minder waarde realiseert ten opzichte van het optimale plan. Het tweede algoritme, `SSII`, maakt gebruik van monsters om iteratief de benadering van de optimale CCS te verbeteren in vooraf bepaalde regionen van de zoekruimte. Dit algoritme biedt geen theoretische garanties maar resulteert in praktijk vaak tot betere benaderingen in het specifieke interessegebied. Dit algoritme is te prefereren in situaties waarin bijvoorbeeld vooraf bekend is wat de waarden van de gewichten ongeveer zullen zijn of wanneer deze maar een beperkte bandbreedte hebben.

Vanaf hoofdstuk 6 verandert de focus van coördinatietechnieken naar het ontwerp van de financiële stimuli om zelfregulering te realiseren, ofwel het eerste deel van de hoofdvraag. In dit hoofdstuk worden twee methoden geïntroduceerd die ieder verschillende aspecten van eigenbelang behandelen. De eerste methode, het *Dynamic Maintenance Mechanism*, maakt gebruik van financiële stimuli om de waarde van de aanbesteding in verwachting te maximaliseren en opportunistisch gedrag te voorkomen. Dat wil zeggen, dit mechanisme ontmoedigt het manipuleren van de uitkomsten voor persoonlijk gewin door dit ongunstig te maken voor de manipulator. Dit mechanisme biedt de beste theoretische garanties maar stelt ook flinke eisen: het moet vele optimale plannen berekenen tijdens gebruik en het vergt van de agenten dat ze hun volledige beslismodel kenbaar maken. Aan de eerste conditie is moeilijk te voldoen vanwege de complexiteit van MPP, aan de tweede vanwege de gevoeligheid van de informatie die moet worden prijsgegeven. Om deze redenen wordt in dit hoofdstuk een tweede aanpak voorgesteld op basis van best-response planning. In deze methode stellen dienstverleners iteratief een plan op voor hun eigen beslissingen als reactie op het huidige gezamenlijke plan. Anders gesteld, om de beurt krijgt elke agent de mogelijkheid een nieuw plan in te dienen voor zijn werkzaamheden, welke wordt geïntegreerd in het gezamenlijke plan. Het resulterende plan met de nieuwe planning voor de activiteiten van de laatstgenoemde agent wordt vervolgens aan de volgende agent voorgelegd, waarop deze de mogelijkheid krijgt om te reageren op het nieuwe plan. Dit proces gaat door totdat het samengestelde plan acceptabel is voor alle deelnemers. Het opstellen van een plan voor één enkele agent is minder belastend, vergt alleen het delen van de planningskeuzes en niet de informatie op basis waarvan deze tot stand is gekomen, en eventueel opportunistisch gedrag kan tegen worden gegaan door te reageren met een mitigerend plan in een volgende iteratie. De afweging is dat dit mechanisme alleen kan garanderen dat uiteindelijk de agenten het eens worden over een gezamenlijk plan. Over de maximale waarde die behaald kan worden aan de hand van het resulterende gezamenlijke plan kan vooraf niets gezegd worden.

Hoofdstuk 7 evalueert het concept van zelfregulering wanneer het wordt geconfronteerd met menselijke actoren door middel van een *serious game* (educatief spel). Het "Road Maintenance Game" simuleert het MAINTENANCE PLANNING PROBLEM

binnen een groepscontract en laat mensen spelen in de rol van dienstverleners. De spelers krijgen de opdracht hun onderhoudswerk naar eigen inzicht te plannen, waarbij ze de mogelijkheid hebben om te coördineren met andere spelers om hun opbrengsten te optimaliseren. Het coördineren wordt echter niet verplicht of gefaciliteerd door het spel verder dan het geven van inzicht in de verwachte consequenties. Integendeel, coördinatie zou vanuit de groep zelf moeten ontstaan als gevolg van de financiële stimuli van het contract. Met andere woorden, er wordt onderzocht of de stimuli effectief zijn in het uitlokken van zelfregulering. Hierbij wordt extra aandacht besteed aan de invloed van relaties tussen de spelers omdat eerdere onderzoeken uit wijzen dat deze een grote rol speelt in het succes van samenwerkingsverbanden. De observaties en metingen aan de hand van zeven gespeelde sessies laten zien dat het gebruik van financiële stimuli een effectief middel is om beslissingen te beïnvloeden maar dat deze aanpak soms leidt tot onbedoelde gedragswijzigingen. Dit kan resulteren in een ongewenste competitie binnen de groep wat weer kan leiden tot onverwachte of ongewilde uitkomsten. In groepen met sterke sociale relaties tussen de spelers leidden dezelfde stimuli in de experimenten wel tot de beoogde zelfregulering. Ondanks dat deze resultaten nog niet voldoende zijn om te kunnen concluderen dat zelfregulering altijd leidt tot tevredenheid over de uitkomsten in groepsaanbestedingen vormen ze een sterk bewijs van de potentie van zelfregulering in met name coöperatieve samenwerkingsverbanden zoals strategische partnerschappen en allianties. Daarnaast bevestigen de experimentele resultaten nogmaals het enorme belang van relaties in samenwerkingsverbanden en tonen ze dat de sociale dimensie een onmiskenbare factor is van zelfregulering.

De conclusie die volgt uit hoofdstukken 3 tot en met 7, en daarmee het antwoord op de hoofdvraag, is dat er geen alomvattende methode bestaat om zelfregulering te implementeren in contracten zodat aan alle gestelde criteria tegelijk voldaan wordt. Verder onderzoek is noodzakelijk om de gereedschappen, methodes en lessen die worden bijgedragen door dit proefschrift toe te passen in groepsaanbestedingen in de praktijk. Dit verdere onderzoek zou gepaard moeten gaan met advies vanuit andere disciplines zoals contracttheorie, (publiek) netwerk management, wetgeving en gedragspsychologie en sociale wetenschappen. Desondanks legt dit proefschrift een wiskundige basis voor de implementatie van zelfregulering in contracten door middel van financiële stimuli gelegd, begeleidt dit werk het ontwerpen van zelfregulerende, prestatiegerichte mechanismen en draagt het coördinatietechnieken bij ten behoeve van het maximaliseren van de waarde van aanbestedingen of vergelijkbare problemen. Daarnaast demonstreert dit werk het potentieel van financiële stimuli om zelfregulering te bewerkstelligen bij menselijke actoren. De algoritmen van dit proefschrift brengen de huidige 'state-of-the-art' in sequentiële beslisproblemen een stap verder, waardoor oplossingen gevonden kunnen worden voor problemen die werden beschouwd als ondoenlijk. Ook suggereren ze nieuwe invalshoeken voor volgende onderzoeken, zoals het toepassen van een slimme MDP codering op bekende complexe problemen of het uitbreiden van `Conditional Return Policy Search` naar andere problemen en domeinen. Parallel hieraan biedt het educatieve spel een empirisch platform aan onderzoekers en professionals dat gebruikt kan worden voor de verdere (geautomatiseerde) verkenning van agent strategieën, opportunistisch gedrag en contractmechanismen met het einddoel zelfregulering over te brengen naar actuele groepsaanbestedingen.

# Preface

Finally, after a long period of almost 10 years I can truthfully say that I am very close to completing my PhD, a statement that I have made many times over the last years but only now has become a reality. Indeed it has been an incredible journey for me. In my opinion there is no other professional position with a similar degree of freedom in the work, nor one that provides so many opportunities for personal growth, to expand boundaries both literally and figuratively, and to come in touch with so many inspiring people, ideas and environments. I have very much enjoyed my time as PhD student everywhere: as a young researcher in the Almende office in Rotterdam, as a member of the Algorithmics group at the University of Delft and even as a presenter at international conferences a few times. Perhaps I have enjoyed my time as a PhD student – and the activities typical to one of such stature – slightly too much, causing my journey to last a bit longer than your average PhD[3] and making it necessary to take up a position in industry in 2015 while the thesis was not completed yet.

Now whereas a switch from full-time student to part-time researcher working from home and part-time software architect at Divider BV is not really beneficial to the pace of writing a dissertation, working as a full-time manager of a software development department that grows from 3 to 20 developers in the middle of a take-over by KPN, two relocations and several reorganisations while maintaining a social life is certainly detrimental to its progress. Still, having 'two jobs' concurrently gave me the opportunity to experience both sides, i.e. academics and industry, and draw interesting parallels between theory and practice. For instance, I have observed that many interactions in a corporate environment such as that of KPN follow game-theoretical models or that my style of management much resembles that of self-regulation (and that this should in my opinion be the preferred style of all management). On the other hand, working in industry has given me a result-driven attitude that certainly helped me to complete this thesis. It must also be said that while doing a PhD might sometimes feel stressful, especially when paper-deadlines are due, I have experienced it as a walk in the park compared to the pace of a competitive business environment. But then again, that may be why I took slightly longer.

Although there has not been a single moment in which I really considered quitting, I do have a piece of first-hand advice for those just starting out. If you are reading this

[3] *Ten years is not much longer than the average* 8.2 *years according to the New York Times, see* https://www.nytimes.com/2007/10/03/education/03education.html.

and thinking about doing a PhD yourself or if you are currently at the start of your own amazing journey, I can definitely recommend finishing your PhD before starting a new job. I am certain that while I have (tried to) put in a lot of effort during the evenings and weekends, the effective work produced thereby during the last five years could have been done in approximately 2 or 3 months of consecutive writing, if that time would have been available. It is hard to keep up the writing pace with just a couple of hours per week. A large part of these hours is 'wasted' on catching up or go unused entirely after intensive days at work. Moreover, the nagging feeling of guilt that you have when deciding to spend time on things other than writing the thesis is definitely one I am not going to miss. Handing in this thesis after almost ten years will be paired with an enormous feeling of relief, that is for sure.

Regardless of the total duration and hardships it has been a wonderful experience, one that has been made possible by a lot of people that I would like to thank here. Foremost my gratitude goes out to my promotors at the TU Delft, Mathijs and Matthijs, and my former promotor Cees for all the fruitful discussions we have had, your ever critical but always constructive feedback and your patience and continued belief. I would like to thank my counsellors Andries and Hans from Almende who helped me to keep in mind the practical side and relevance of the research and gave me the opportunity to work in an inspiring environment of young researchers and professionals (and fusball). Also I am very grateful for the extension of the deadline by 5 months that you have made possible. Even though I was not able to complete the thesis in this extra time, it allowed me to get sufficiently far as to not give up when I had to combine the PhD with my new job at Divider. Speaking of which, I want to thank also the people at Divider – and later KPN – that have been very supportive in my efforts to complete 'the last mile', in particular Arno, Pieter and especially Kim who convinced me to request leave and get it over with and who made me promise her that she would know the moment I submitted my final version. Of course, my gratitude also goes out to all the other beautiful colleagues I have met in all three of my working environments.

A special thanks I would like to extend to my co-authors who have been paramount in getting the ideas of this thesis published. Leentje and Daan for our combined work on the dynamic contracting framework and the serious game, Diederik for the many interesting sparring sessions on decision-theoretic planning that culminated into two excellent papers, and the many others that contributed their exceptional knowledge and experience to this academic endeavour of which I would like to name a few: Frans Oliehoek, Shimon Whiteson, Paulien Herder, Martin de Jong and Monica Altamirano. Additionally, my appreciation goes out to all the members of the User Advisory Board who have brought into this research many of the practical considerations and lead to the conception of the serious game. Your contributions have been essential in the development and validation of the serious game. Equally I want to thank all of the participants of the serious gaming sessions and associated questionnaires who have provided us with so much valuable data.

Finally my very personal thanks goes out to three people in particular, the first two of which are my parents who have always supported me to make the most out of all my opportunities and be the best version of myself I can be. While my mother taught me

the virtues of working hard, caring for people and staying humble and compassionate, my father has been my principal inspiration to intellectually challenge myself and to pursue the highest possible level of education, amongst many other things. Last but not least is of course my own Laura, to whom I own both my colossal gratitude as well as an extensive apology for the many hours that we could not spend together because of this dissertation. I want you to know that without your continuous support and understanding I would not have been able to complete this thesis and I am happy to say that I told you I would finish it before marrying you.

Joris Scharpff

Zoeterwoude, 30 December 2019

# Table of Contents

**Summary** i

**Samenvatting** v

**Preface** xi

**1 Decision Coordination through Self-regulation** 1
- 1.1 Road Maintenance Planning . . . 5
- 1.2 Implementing Self-regulation: Dynamic Contracting . . . 11
- 1.3 The Challenges of Self-regulation in Contracts . . . 15
- 1.4 Outline and Contributions . . . 23

**2 Stochastic Planning using Markov Decision Processes** 27
- 2.1 Markov Decision Processes . . . 27
- 2.2 Finding Optimal Policies . . . 33
- 2.3 Factored MDPs . . . 35
- 2.4 Partial Observability . . . 37
- 2.5 Planning with Multiple Agents . . . 43
- 2.6 Gaining Traction on Dec-POMDPs . . . 50
- 2.7 Planning with Multiple Objectives . . . 58

**3 Solving the Maintenance Planning Problem** 65
- 3.1 The Maintenance Planning Problem . . . 67
- 3.2 Solving MPP with Dynamic Programming . . . 77
- 3.3 Maintenance Planning as a Markov Decision Process . . . 78
- 3.4 Approximation of Maintenance Plans . . . 89
- 3.5 Empirical Evaluation of MPP . . . 93
- 3.6 Further Discussion . . . 97

**4 Maintenance Planning with Multiple Agents** 99
- 4.1 Returns in MDP . . . 102
- 4.2 Conditional Return Graphs . . . 105
- 4.3 Policy Search based on Returns . . . 114
- 4.4 Experimental Evaluation of CoRe . . . 120
- 4.5 Further Discussion . . . 123

**5 Maintenance Planning with Multiple Objectives** **127**
5.1 Multi-objective Planning with Unknown Weights . . . 130
5.2 Approximate Optimistic Linear Support . . . 134
5.3 Scalarised Sample-based Iterative Improvement . . . 138
5.4 Comparison of Multi-objective Algorithms . . . 142
5.5 Further Discussion . . . 147

**6 Maintenance Planning with Self-interested Agents** **149**
6.1 A Dynamic Mechanism Approach to Maintenance Planning . . . 150
6.2 Selfish Best-response Maintenance Planning . . . 165
6.3 Further Discussion . . . 171

**7 The Game of Maintenance Planning** **173**
7.1 The Road Maintenance Game . . . 175
7.2 Gaming Results . . . 183
7.3 Evaluation of Gaming Results . . . 189
7.4 Further Discussion . . . 191

**8 Discussion and Conclusions** **193**
8.1 The Challenges of Self-regulation . . . 194
8.2 Conclusion . . . 197
8.3 Implications and Next Steps . . . 198

**Bibliography** **205**

**Publications and Supplementary Material** **226**

**TRAIL Thesis Series** **228**

**Appendices** **231**
A Proofs . . . 232
B Computing Game Scores . . . 243
C Game Session Outcomes . . . 250
D Computational Complexity Theory . . . 259
E Game Theory and Mechanism Design . . . 262

# Chapter 1

# Decision Coordination through Self-regulation

Central to this thesis is the conflict of interest that is typical to many group decision-making problems: the misalignment between the goals of the individual group members and that of the group as a whole. In many day-to-day situations, people are grouped together to complete complex tasks that can not or may not be completed by any of them individually, or these tasks are simply not in their best interest. Examples thereof are students doing a group assignment, colleagues combining skills to complete a project or competing construction companies cooperating to realise a real-estate project. Although the group members share a common goal of completing or optimising a complex task, they are typically autonomous, self-interested entities that strive to maximise their personal gain from participating. Oftentimes the group members are not really interested in performing the joint task the best they can; they are contracted to participate and only the compensation for their contribution is what motivates them to do the work. For example, the students are primarily interested in their own grade and will put most of their effort in the parts they will be accounted for personally. The same may apply to the team of colleagues: they are likely to focus most on the work that is demanded or monitored by their line manager, which is not necessarily in the best interest of completing their joint project. In the scenario of the construction companies, they will likely plan their activities to maximise their own profit without regard for other contractors or the overall project schedule.

Naturally, it is not uncommon for group members to be self-interested. On the contrary, most models of decision making consider *agents* such as the students, colleagues and companies as autonomous, rational entities with personal goals and interests that they seek to optimise. In particular decision theory and game theory, two of the major strands of research on decision making, are both founded upon the model of an agent choosing its *actions* to maximise its expected *utility*, i.e. the gain it expects to obtain based upon its knowledge of the current and anticipated future *states* of the environment. This model is known as *rational decision making*. While in practice the assumption of agent rationality is typically too strong, it does approximate the

decision-making process of a single agent and is therefore used as theoretical basis in most multi-agent decision-making research focusing on decision-making strategies. Indeed, the model of rational decision making can be used to optimise the goal(s) of a single agent by computing the maximal expected gain over all possible decisions and consequential futures within the model. If a group of such agents has to solve a common task, however, optimising the utility of all agents simultaneously is impossible unless their interests align, their personal goals are irrelevant or they are non-autonomous. In some situations this is inherently the case: an alliance of taxi companies jointly scheduling trips (aligned interests), a team of firefighters seeking to exterminate all fires as fast as possible (no personal interest) or factory robots jointly planning their operations to maximise total production output (no autonomy). In many other group decision-making situations, however, autonomy and misalignment need to be addressed if the aim is to optimise a global goal. This thesis focuses on problems in which agents are requested to solve a common task but the interests of agents do not align.

In particular, this thesis addresses *sequential* decision-making processes in which groups of autonomous agents have to make multiple decisions over time with potentially uncertain outcomes to optimise a global goal, known as *collaborative multi-agent planning under uncertainty* or simply *collaborative planning*. A practical example of collaborative planning is joint maintenance of the national highway network in the domain of infrastructural maintenance, the motivating domain at the origin of this research. But the problem of planning in the presence of conflicting interests is certainly not limited to this domain. Occurrences can be found in a multitude of settings: scheduling the loading and unloading of vessels in the harbour with the global goal of optimising the transfer of goods [82], optimisation of a supply chain where parts of the chain are controlled locally [136] and dial-a-ride coordination where individual taxi drivers seek to maximise their fares but serve customers collectively to get more jobs with less mileage [260]. Indeed, collaborative planning with a misalignment of interests occurs in many domains and settings, and therefore it is of great academic and industrial interest.

Currently the most prevalent approach to align the goals of group members is through governance, also known as (complete) *regulation*. In this approach a coordinator or director is appointed to coordinate the decisions of the group members, typically by the party that defined the group goal, in order to prevent individual interests from harming the group goals. Reconsidering the examples of conflicting interests given at the beginning of this chapter, in the case of the students this director could be the teacher that forces students to work together in order to succeed in their assignment. The co-workers may decide to elect a coordinator/manager to lead their project team and coordinate tasks between team members. In the situation of partnering construction companies, the contracting party that procured the maintenance may choose to act as a coordinator and impose a joint maintenance plan. The companies must then adhere to this schedule, otherwise they will be fined or even disqualified from further participation.

Although regulation can overcome the misalignment of objectives through enforcement of (joint) decisions, it is paired with significant effort and responsibility on the behalf of the director. First of all, complete regulation requires the director to fully understand the decision-making model of the agents to produce a coordinated schedule

that optimises the group goal. This implies that the director has to know the capabilities and skills of all group members and match their level of expertise in order to understand the choices available to the agents. Consequentially, the director is solely responsible for dealing with aspects like risk-management, liability, planning robustness, inter-agent communication and coordination. For the agents similar arguments can be made against complete regulation. Agents are required to fully disclose private information regarding for instance costs, resources, material, risks, preferences, etc. to the director; something that especially commercial contractors are unwilling to share because it might undermine their competitive position. The agents will have to completely submit their autonomy to the joint schedule imposed by the director. Finally, even if a complete model of all agents would be available to the director, finding a high-quality joint schedule poses a non-trivial computational challenge to automated planning support tools (as becomes clear in Chapter 3).

For these reasons significant attention has recently gone towards *performance-based* approaches that do not seek to control the decision-making process of the group members, but instead allow the agents to make their own planning decisions while accounting them for their (negative) contribution to the global goals [41, 45, 218]. The key idea is that through e.g. monetary performance incentives, the agents are rewarded exactly when they contribute to the global goals thus aligning both interests. Put differently, a successful performance-based incentive scheme ensures that the agents profit most when the global goal is achieved. Of particular interest and true to the core ideas of performance-based methods are approaches that strive to incite *self-regulation* within the group [75, 124, 255], i.e. achieving "an organization regulating itself without intervention from external bodies" [2]. The main idea of such approaches is that if a group is self-regulating, group members will actively seek to coordinate their decision making to achieve the global goal without the support or interference of any director. The key to successful implementation of self-regulation is hence to provide the right incentives to the agents, based upon *their performance as a group*, so that it becomes in their own best interest to both optimise the global goal as well as coordinate their joint efforts internally.

Revisiting the previous examples one last time, self-regulation in the case of the school project can be based upon individual contributions towards the success of the project. The less effort a student puts in the project – arguably the interest of most students – the lower its grade will be. However, if the final grade of the group members is set to the lowest in the team they can be stimulated to work together and motivate each other to do a (better) job as it is in the interest of the entire group to make the lowest grade as high as possible. To get the team of co-workers to collaborate better and complete a project faster, a financial bonus could be given to all co-workers but only when the entire project is completed before the deadline so they are incited to ensure a swift joint completion. For the construction alliance a penalty can be given to all team members for every day the construction project continues past the due date, even if they have completed their part already. Note that these are just examples of incentive schemes; many other self-regulation mechanisms may be employed by a contracting party to achieve their desired results.

If applied well, self-regulation preserves the autonomy of the agents and lets them 'do what they do best', making the most out of their skills, capabilities, resources and expertise, while implicitly optimising the global goal. However, the limited control over the group decision-making process may lead to greater uncertainty regarding the outcomes thereof, an increased potential for opportunistic and self-interested behaviour, a misalignment of (societal) objectives and an overall greater complexity experienced by the group members. Even though the first cautious applications of self-regulation are being seen in contracts [34, 43, 57, 76][4], still much is unknown regarding the exact opportunities, risks and outcomes associated with this novel approach to contracting. As of now, no guidelines have yet been proposed on how to implement self-regulation within contracts and no guarantees can be given with respect to the outcomes of such a contracting approach. The concept of self-regulation is promising but neither does there exist a theoretical framework for its implementation nor is it feasible to experiment with these ideas in real-world contracts due to the high risks and costs associated with typical multi-agent service delivery projects. In other words, how to successfully implement self-regulation in contracts is currently unknown.

This thesis addresses this problem through theory and techniques from computer science, with a focus on the domain of *road maintenance planning*; a practical and intuitive example domain in which self-regulation is being investigated for its potential [73, 115, 197]. More specifically, the literature of game theory is applied to model the execution of road maintenance contracts as a game that can be analysed mathematically. Given such a game, techniques from mechanism design are employed to design incentive structures that will ensure desired outcomes when implemented in the game. That is, mechanism design provides the guidelines on how to implement incentives that achieve self-regulation and guarantee successful delivery of road maintenance projects. Complementary to the design of incentives, algorithmic techniques from the domain of multi-agent decision making are employed to coordinate planning decisions while accounting for the incentives. Automated planning methods help both the director as well as the agents to manage the complexity of the decisions in this domain and maximise their obtained value from the contracted work. Finally, serious gaming offers the toolbox to build a simulated environment of road maintenance planning and experiment with self-regulation contracts without the risks and costs of real-world implementation.

In summary, this thesis employs algorithmic techniques from aforementioned fields to support the implementation of self-regulation within contracts by contributing tools for the design, implementation and validation of self-regulating incentives in the context of road maintenance planning. The next section introduces the road maintenance domain and focuses in particular on the exemplifying problem of this thesis, the MAINTENANCE PLANNING PROBLEM. In this problem a group of service providers is responsible for the scheduling of their maintenance activities while their payments are relative to their joint impact on traffic, thus making their rewards interdependent and hence necessitating coordination amongst the service providers if they are to optimise their gain. In Section 1.2 it is discussed how innovative contracts could be employed

[4] *Albeit that most current contracts assume collaborative parties with joint decision coordination, not the self-interested and fully autonomous parties that makes self-regulation different from strategic alliances.*

to maximise the gains in tenders for such problems, building upon ideas of previous work in the fields of contract management, (public) network management[5], decision-theoretic planning and mechanism design. This section also identifies several gaps in the current literature that will have to be addressed before self-regulation can be successfully employed in realistic tenders, which are presented in Section 1.3. The chapter is concluded with an overview of the contributions of this work and a reading guide in Section 1.4.

## 1.1 Road Maintenance Planning

The source of inspiration for the research performed and presented in this thesis originates mainly from the domain of infrastructural maintenance and, in particular, the challenges that arise when trying to plan road maintenance operations optimally. The elegance of planning problems from the domain of infrastructure is that while it is relatively easy to formulate interesting and intuitive problems, they typically involve complicated interactions and dependencies between self-interested agents, complex tasks with uncertain outcomes and long project durations. Furthermore, the autonomy of agents advocates decentralised approaches in the sense that every agent should be able to make its own planning decisions independently, adding to the problem difficulty. All these ingredients combined makes infrastructural maintenance planning very hard to optimise, even despite the availability of automated planning tools. On the other hand, many people experience or have experienced the effects of (poor) road maintenance planning at some point – some even on a daily basis – and can therefore easily relate to the problem as well as its economical, societal and personal impact.

By no means is the topic of infrastructural maintenance or in particular its impact on society a new one. However, over the last decade the nature of infrastructural maintenance projects has changed significantly for both public institutions as well as private companies. At the origin of this change is the combination of an increase in network usage, thus (further) stressing the capacity of the network and increasing the impact of maintenance, with a funding that does not scale accordingly [10]. Considering the case of the Dutch national highways, traffic intensity has grown from 2,089 in 2011 to 2,261 vehicles per hour on average in 2017 on the same highway segments [3], car ownership has increased from 10.8M in 2015 to 11.5M cars owned in 2019 [4], and the total distance travelled on the national highway network has risen from 128.3B in 2011 to 137.1B KM in 2018 [5]. In contrast, the budget that is allocated to the Dutch national road authority Rijkswaterstaat has barely increased in the recent years [1] and will not expand in the near future [6]. This matches the current trend of road authorities reporting that their budget is no longer sufficient to achieve satisfactory maintenance levels over all parts of the infrastructure [10]. Hence innovative approaches to road maintenance planning are being sought that make better use of the limited resources and funds that is available.

[5] *The field of network management is concerned with how to manage a group of (contracted) agents, not the actual management of an infrastructural network (see Remark 7.1).*

One approach that has become exceptionally popular since the turn of the century is performance-based contracting [41, 45, 113, 218, 268], also referred to as value-driven or best-value contracting [239, 246], especially in the context of *public-private partnerships* (PPP) [75, 200] that characterise the relation between road authority and service providers in the infrastructural maintenance domain. Performance-based contracts change the way maintenance works are tendered from an exhaustive project specification into an output-driven agreement. Whereas traditional contracts describe every element of the work, including prices, resources and planning, a performance-based contract simply specifies the desired result and an associated pricing scheme and leaves the planning and execution of work to the service providers. This pricing scheme or *incentive mechanism* rewards contractors for positive contribution to the goals and/or penalises them when performance is not adequate. Hence, the road authority does no longer have to govern maintenance work, instead it monitors performance and rewards or fines service providers accordingly. The service providers, on the other hand, experience more freedom in implementing the contract in the best way they see fit and will not be 'held back' by the resources or expertise of the road authority. Overall, performance-based contracting offers many promising advantages over complete governance: increased flexibility, preservation of autonomy and authority, better use of expertise and skill, more innovation, higher level of performance of each individual participant and, consequentially, better use of public funding [7, 47, 141, 152, 233]. Furthermore, by transferring control and responsibility to the contractors, asset management can be outsourced [156, 100] and the burden on governmental resources can be greatly reduced [75, 124].

Performance-based contracting has recently proven itself as a successful approach in practice. Incentive mechanisms are a valuable tool in achieving favourable outcomes [41, 45, 218] and have demonstrated their worth in actual tenders [43, 62, 268]. These successes, however, are currently limited to bilateral agreements, i.e. partnerships between the road authority and a single service provider. While this is without a doubt an accomplishment in the domain, it fails to incorporate the characteristic 'network aspect' of road maintenance planning. That is, typically in road maintenance multiple works are being performed concurrently by multiple of service providers on the same infrastructure, leading to dependencies and interactions between them. Coordination of these works is essential to minimise the impact of maintenance on the network throughput [115, 197, 236]. Of course, this network-level coordination could be performed by the road authority but that implies a regression to traditional governance models. Therefore, road authorities have recently begun to explore the potential of performance-based contracts to group tenders with a particular interest to instill *self-regulation* on the network level. The key idea is simple: by incorporating the network-level dependencies in the incentive mechanisms of performance-based contracts, service providers are inherently motivated to coordinate their activities. The design and implementation of such a network-level incentive scheme, however, is not. Without the complete control offered by governance, the benefits promised by performance-based contracting are accompanied by more complexity, greater uncertainty, increased potential for opportunistic, self-interested behaviour and possible misalignment of (societal) objectives [11], and may thus result in sub-optimal network performance or even total

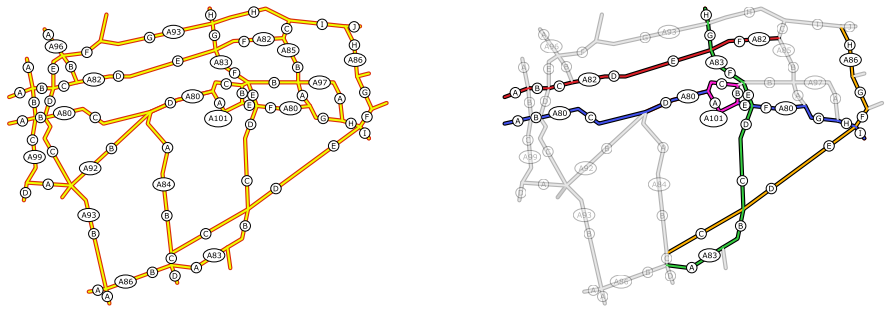
failure when not properly implemented [109, 125]. Given that the annual maintenance budget of the Dutch national highway authority approximates 1 billion euro every year [6], the cost of failure in this domain can be substantial. Recent practical applications of these new approaches show that information asymmetry, lack of transparency and distrust result in parties falling back to traditional control-oriented relationship between road authority and service providers [239], resulting in costly governmental interventions [90, 100]. Naturally this problem is not limited to the Dutch road authority, or the domain of infrastructural maintenance, many related and similar domains face the same challenge with multilateral agreements. Examples can be found in large-scale maintenance projects [83, 235], the construction sector [121], manufacturing industry [122], the energy sector [74], system-support engineering [174] and pollution control initiatives [123]. The main question is common to all these domains: how to implement the benefits of performance-based contracting in a group contract while avoiding the potential pitfalls thereof?

This thesis approaches that question from an algorithmic point of view, using a mathematical formulation of a road maintenance planning problem that exemplifies the core of this question. This problem, called the MAINTENANCE PLANNING PROBLEM (MPP), is a decision-theoretic formulation that models the planning of maintenance work and incorporates the network-level dependencies between service providers. The MAINTENANCE PLANNING PROBLEM was part of the work of Altamirano et al. [9] (although no name was given to the decision problem) to study opportunistic behaviour of contractors in a road maintenance planning game called "Road Roles". This problem is by no means a complete model of the real world, neither does it encompass every aspect or challenge from the domain; it is however an illustrative and accessible formulation of a planning problem that captures the characteristic complexities of the infrastructural domain and is representative for many similar collaborative planning problems. Furthermore, its mathematical formulation enables reasoning about (joint) strategies to optimally achieve planning goals, i.e. the coordination of maintenance work. But that is the subject of subsequent chapters, here the problem of maintenance planning, with its challenges, is illustrated by example.

At the core of road maintenance planning is the infrastructure itself, which is comprised of a collection of *assets* such as road segments, traffic signs, bridges, tunnels, etc. The assets are generally owned by a public institution, often a national or federal government, and managed by another, like the national highway authority or local government. These two are commonly referred to as the *asset owner* and the *asset manager* (AM) respectively, where the latter is responsible for the upkeep of the assets owned by the former. Therefore the role of the AM is to identify the components of the infrastructure that require maintenance, formulate corresponding maintenance projects and ensure successful completion of these projects, taking into account the demands of the asset owner and the users it represents. As such, the asset manager has a responsibility towards society (the asset user) not only to maintain a high-quality network, but also to prevent or minimise other negative impact like project delays, environmental harm and traffic hindrance as a result of performing maintenance.

Although the AM is responsible for the outcomes of aforementioned projects, the actual maintenance is rarely performed by the asset manager itself. More commonly

this is outsourced to *service providers* (SPs), commercial companies that are contracted to perform the maintenance projects identified by the asset manager. Their best interest, however, is not necessarily similar to that of the AM: the service providers focus primarily on maximising their profits and are not inherently motivated to consider the other goals of the asset manager, i.e. low hindrance or environmental impact.[6] This misalignment in objectives may lead to undesirable outcomes of maintenance projects with potentially severe economic impact to society, in particular when dealing with a group of such self-interested SPs. This is illustrated through the following example of road maintenance planning in the context of a realistic highway network.



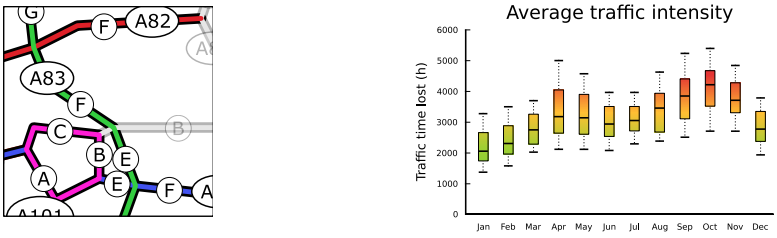
(a) (b)

**Figure 1.1** *An example of a complex road network is shown in (a), inspired by a real-world road network. The assignments of highways over the SPs is illustrated in (b), where each SP is identified by a unique colour.*

Figure 1.1a shows a complex network of highways (the assets). Although the network in this example is fictitious, it is modelled based upon the primary roads of a real-world traffic network. It is composed of $12$ different highways that have, in this example, been divided into $61$ arbitrarily chosen named segments. After performing a quality assessment, the asset manager has identified that next year the highways A80, A82, A83, A86 and A101 are most in need of service. Therefore the AM designs and procures five maintenance projects corresponding to the previously identified highways. For each project a "classical" contract is drafted in which the goal is to service all of the segments of a single highway, so that they meet the quality demands required by the AM, within the period of one year for a fixed price per segment. The SPs themselves are responsible for planning the work and they are rewarded each time they have successfully completed the maintenance of a segment. For uncompleted or unsatisfactory maintenance of segments, service providers will receive no payment. After a procurement phase with several rounds of negotiation and bidding, five service providers are elected as winners and each SP is made responsible for a single maintenance project. The assignment of highway projects that resulted from procurement is shown in Fig-

[6] *In reality continuation, reputation gain or 'getting a foot in the door' are also very valid reasons for a service provider to compete for a tender. The simplistic view taken here, which is oftentimes close to reality, is that they are mostly concerned with maximising their profits.*

ure 1.1b. Now the execution phase starts and the service providers need to plan and perform their maintenance operations.




(a) (b)

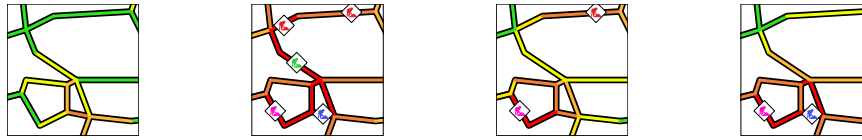
**Figure 1.2** *(a) The A101 region in which 4 service providers are active. The normal traffic intensity within this area is illustrated in (b), expressed as average sum of hours of traffic time lost per month.*

To clearly demonstrate the impact of goal misalignment on the outcome of the process, this example focuses on the planning and execution of maintenance work within a smaller area of this network. Consider the region around the A101 depicted in Figure 1.2a. In this region there are four highways due for maintenance, each serviced by a different SP. In order to quantify the impact that maintenance has on the network throughput, the asset manager monitors the *traffic time lost* for every month of the contract period. The traffic time lost, or *ttl* for short, captures the additional travel time (in hours) summed over all network users such as commuters, transport carriers and recreational traffic as an effect of the reduced capacity of the network compared to network at full capacity. That is, it operationalises the increase in traffic due to network disruptions caused by e.g. maintenance, accidents or major events.

For the A101 area, Figure 1.2b shows the 'regular' traffic time losses for this area based upon historical measurements, i.e. when the network is not full capacity due to for instance accidents or events. For instance, the graph of Figure 1.2b shows that under regular conditions, the traffic in March leads to a median traffic time lost of approximately 2,750 hours due to delays within this area but monthly traffic time losses of over 3,000 hours are not uncommon.[7] When performing maintenance operations these traffic time lost figures typically increase significantly. Many operations require at least a partial closure of segments, but occasionally all lanes in one or even both directions of a highway need to be closed entirely for service. Blocking roads is a necessary evil that has to be endured to improve the network quality but inherently paired with a (major) increase of ttl. Given that the *value of time* for commuters is estimated around 20 euro per hour lost [153], the economic impact of additional ttl due to maintenance can be considerable.

[7] *While the figures of this example are fictitious, they are not completely unrealistic. Rijkswaterstaat, the Dutch road authority and asset manager, has performed quarterly monitoring of traffic intensity on the Dutch road network between 2008 and 2012, and reported several segments where the monthly average ttl is of the magnitude of multiple thousand of hours. See for example Brandt [42].*

Even though ttl increases cannot be avoided, the negative impact of maintenance on the network throughput can be limited by the service providers through careful planning and applying less intrusive (but possibly more costly) maintenance approaches. An SP could for instance temporarily hire extra workers to decrease lead times, thereby reducing the duration of a road closure, or plan its work during 'quiet hours' in the night. However, even though the SP can limit the negative impact of its own work, it has no control over all maintenance operations. As in the example of Figure 1.2b it may be that multiple SPs act within the same area and therefore the planning decisions of other SPs influence the network throughput as well. In particular, given that the capacity of the network is limited, concurrent maintenance of two or more segments within a close proximity is likely to cause a super-linear increase of ttl, with potentially disastrous travel times for the network user. This is illustrated in the traffic density map of Figure 1.3b, with the regular network traffic shown in Figure 1.3a. On the other hand, in some cases it may not matter whether maintenance is performed concurrently (Figure 1.3c) or simultaneous maintenance, with heavy but brief hindrance, may be less intrusive than having two sequential road closures (Figure 1.3d).



(a) (b) (c) (d)

**Figure 1.3** *Various maintenance scenarios: (a) normal traffic conditions in the absence of maintenance, (b) many SPs working simultaneously, (c) concurrent work with minimal increase in ttl compared to sequential execution, and (d) concurrent maintenance of adjacent segments that only requires a single road closure.*

Obviously it is in the best interest of the road user – and therefore of the asset manager that represents it – that situations like that of Figure 1.3b are prevented while coordination of maintenance operations as in Figure 1.3d is encouraged. The problem is however that the actual planning and execution of work is done by the service providers that have a different goal than the AM. Because they are primarily focused on profits they will not actively seek to minimise their impact on ttl. Instead they try to minimise cost and maximise their revenue. Even stronger, the misalignment of their objectives is more likely to result in scenario (b) than (d). This can be a consequence of (shared) external factors such as holidays or weather conditions but also simply due to the fact that the coordination required to achieve the latter situation requires substantial (and costly) effort and is not in the interest of SPs. It is much easier for every SP to plan work individually without regard for other goals, quickly resulting in the situation of scenario (b). For any road maintenance planning approach to be successful, the asset manager first needs to resolve this misalignment.

The above example is an informal illustration of the MAINTENANCE PLANNING PROBLEM (MPP), one of the many decision-making problems in which the individual interests of agents do not align with the global goal. This is considered the key problem

of this thesis for it is both a highly relevant practical problem as well as an intuitive and characteristic example of a planning problem that is preferably optimised through self-regulation. The next section discusses a novel contracting approach to overcome the misalignment of interests in problems such as MPP through self-regulation, drawing ideas from multiple fields such as contracting, network management and mechanism design.

## 1.2 Implementing Self-regulation: Dynamic Contracting

Overcoming the misalignment of objectives between contracting parties while achieving maximum output of tenders has gained a substantial increase of attention in recent research. Particularly performance-based contracting with incentive schemes has become popular and understood quite well [41, 45, 218, 268]. Performance-based contracting offers many promising advantages over complete governance: increased flexibility, preservation of autonomy and authority, better use of expertise and skill, more innovation, higher level of performance of each individual participant as well as the group on the whole and, consequentially, better use of public funding [7, 141, 233, 47]. The concept of *self-regulation* however, i.e. self-enforcement of performance objectives within a group of contractors, is a relatively new and unexplored territory. The main obstacle that prevents applying self-regulation approaches in current contracts stems from this uncertainty with respect to the outcomes of such a procedure. Self-regulation involves a shift into unfamiliar roles and responsibilities for both the AM and the SPs, and their experience with this way of partnering is limited [112, 246]. Furthermore, there is no general guideline for the design of incentive structures that successfully implement the AM's objectives, such as minimising ttl and maximising quality, let alone ones that incite the SPs to perform maximally *as a group*. Even though some cautious applications are being seen in contracted work, like the congestion-based payment model used in the reconstruction of an intersection on a national road segment [43], they are handcrafted case-by-case [268]. A generic, theoretical framework to implement self-regulation within contracts is still missing, thus keeping its application tedious and limited to a handful of carefully designed projects [144, 205].

In previous work several researchers identified the aforementioned challenges in the particular domain of infrastructural maintenance planning and signalled the failure of traditional regulative approaches due to their inability to adapt to changing conditions, incorporate the network aspect and stimulate service providers to maximise their performance [11, 233, 235, 254]. Intent to overcome these challenges, Altamirano et al. proposed to employ an innovative long-term, performance-based contracting procedure to cope with especially the dynamic nature of such projects and the autonomy and associated opportunistic behaviour of agents [9]. Drawing upon these ideas, Volker et al. went a step further by proposing a two-phase framework for these innovative contracts, linking the planning of operations in the *execution phase* (simulated by the "Road Roles" game of Altamirano et al.) to the preliminary *procurement phase* in which

the operational boundaries are established in the form of a master agreement [254]. This *Dynamic Contracting* procedure is illustrated in Figure 1.4.

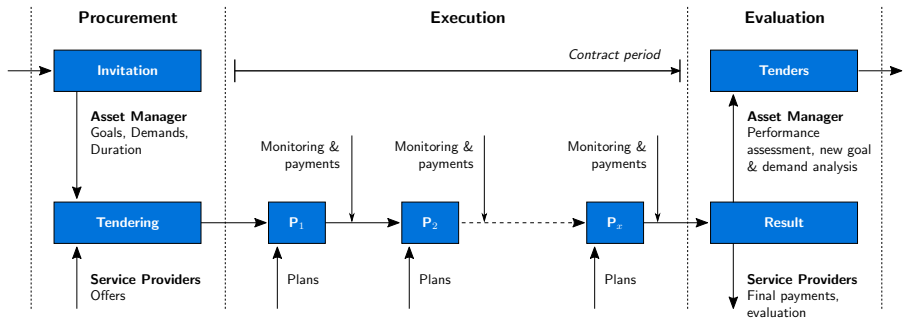



**Figure 1.4** *Schematic overview of the Dynamic Contracting procedure.*

The dynamic contracting procedure, at first glance, resembles traditional contracting. Projects are drafted and a request for tender is sent to service providers, resulting in contracted work for the SPs with the winning bids. At the end of the contract the outcomes and performance are evaluated. Traditional contracts however are typically fixed, bilateral agreements between the AM and SP in which the execution phase of a single project is specified exhaustively, following a typical waterfall-style of project management. Instead, dynamic contracting relies on establishing a master agreement for the entire group with a payment mechanism that aligns their objectives, while leaving the actual implementation of planning and performing maintenance in the execution phase to the service providers themselves. During the execution, SPs are monitored and rewarded based upon the observed performance in short cycles. If the payment mechanism is designed correctly, the service providers will maximise their profit exactly when they optimise the goals of the AM, e.g. minimise ttl. Under such a mechanism, the work of the SPs is coordinated *implicitly*: causing less ttl results in more profit to them, i.e. rewards are based upon their performance with respect to the contracted objectives. Moreover, as their revenue depends upon the ttl that is affected by other SPs as well, they will actively seek to collaborate and minimise ttl by coordinating their plans. Hence such a payment mechanism incites self-regulation. Finally, the existence of an incentive mechanism in combination with short planning and execution cycles allow for fast and dynamic interactions without the need for renegotiation, making it more suitable in dynamic environments than traditional contracting [83, 235, 236].

Indeed, dynamic contracts offer solutions to many issues of long-term group projects that cannot be adequately addressed in traditional contracts such as changing conditions within the contract period, rewards based upon group performance, decentralised coordination of work and shared responsibility [254]. The success of this approach, however, relies first upon the correct design of the contract framework with performance-based incentives that will make the self-interested SPs achieve the AM's goals in a self-regulating fashion. Secondly, for the incentives to achieve their desired effect, the

SPs must be able to coordinate their (interacting) decisions effectively and efficiently. A well-designed incentive scheme may result in unsatisfactory outcomes if SPs are unable or unwilling to coordinate their actions.

In related literature, much attention has gone towards network management strategies, i.e. how to establish a (public-private) partnership and structure its interactions [7, 233], trust in partnerships [150, 254] and individual performance-based incentives [9, 62, 157]. Many authors recognise the need for a standardised approach to implement performance-based incentives in contracts that lead to self-regulation and, implicitly, the successful achievement of contract goals [10, 144, 205, 218, 268]. Only a limited amount of work, however, addresses the link between *joint* strategy and outcome, let alone propose methods to ensure successful self-regulation in such partnerships. For example, both Bower et al. [41] and Rose and Manley [218] advocate the effectiveness of incentives within single-party contracts but conclude that it is yet unclear how to implement similar schemes in group contracts for equivalent results. Case studies as performed by Choi et al. [62], Kenley et al. [135], Nalbantian and Schotter [185] and Zenger and Marshall [267] show promising results, however the generalisation thereof into a generic design of dynamic contracts is not apparent. The closest to a generic framework for group contracts is the proposal of Volker et al. [254], but the model is only high level and does not offer guidelines on how to design incentives and coordinate decisions to maximise the team output in contracted projects.

In addressing this challenge, the literature of game theory and mechanism design offers promising techniques and tools to design team incentives within contracts that do provide guarantees on the quality of the outcome [67, 72, 168, 188]. In close relation to this work, Van der Krogt et al. [148] address the application of mechanism design in multi-agent planning to deal with the self-interested nature of the participating contractors. Practical examples can also be found in the literature. For instance, Gupta et al. [106] use mechanism design to define performance-based contracts for single highway maintenance projects that effectively stimulate the service provider optimise maintenance while preserving its autonomy. Hong et al. [117] employ mechanism design to determine the set of optimal service contracts for repair and maintenance, overcoming the asymmetrical information level between SP and customer, and encouraging customers to minimise the number of maintenance calls. Additional examples can be found in the literature [89, 91, 206, 245].

While these results seem encouraging, the effectiveness of mechanism design relies heavily on the structure of (private) information, known as the agent's *type* in mechanism design, and the ability to efficiently solve the underlying decision-coordination problem [72, 204]. As a consequence, existing work almost exclusively focuses on specific settings such as bilateral contracts, 'single-shot' decision problems or 'online' settings and these approaches do not scale well to the decision-space complexity of realistic group contracts. Only recently have researchers begun to focus specifically on mechanism design in the context of sequential decision-making problems where (private) information evolves over time, referred to as *dynamic mechanism design*. In close succession, Athey and Segal, Bergemann and Välimäki, and Segal and Toikka published several articles that together are considered the pioneering works in this field, extending core concepts of 'static' mechanism design to the dynamic setting. Athey and

Segal [16] and Bergemann and Välimäki [30], independently proposed dynamic variants of two of the most well-known mechanism classes, respectively d'Aspremont-Gérard-Varet (AGV) [15] and Vickrey-Clarke-Groves (VCG) [64, 101, 252], thus showing the potential of dynamic mechanism design. In parallel, Segal and Toikka [234] showed that the revelation principle can also be extended to the dynamic setting. As a corollary, many of the established theorems for mechanism design can – with some work – be transferred to the dynamic setting and be used to provide similar guarantees on the quality of the outcomes. Indeed, these fundamental papers have inspired many others to pursue dynamic mechanisms to overcome practical challenges [55, 131, 173, 237].

Dynamic mechanism design offers tools to design incentives for long-term group contracts such that the quality of outcomes is assured, even when multiple decisions need to be taken over time in a changing or uncertain environment. Nonetheless, approaches that combine realistic optimal sequential decision-making problems, like the MAINTENANCE PLANNING PROBLEM, with mechanism design are nearly non-existent [29, 204]. Current literature in the field of dynamic mechanism design is either mostly theoretical [16, 30, 202, 234], focuses on simpler type structures or decision-making problems [131, 173, 237], or relies upon approximation of the underlying decision-making problem [46, 111, 180]. Optimal multi-agent problem solving – required to determine exact payoffs and optimal strategies to maximise output – and complex type structures – required to represent the full richness of the decision problems of participants – have not received much attention in the dynamic mechanism setting. Nonetheless, recent work in this field by in particular Cavallo et al. show the potential of dynamic mechanism design in decision-making settings when individual decision problems can be modelled as a *Markov decision process* [53, 55, 56]. Although these works focus on maximising mechanism revenue and the redistribution thereof, the followed approach inspires research into the application of dynamic mechanisms in the context of decision making. In particular, these novel ideas merit further research into the dynamic mechanisms as a means to influence decision coordination and the link between mechanism design and the computational complexity of such decision-coordination problems.

Nonetheless, even though dynamic mechanism design seems a promising approach to design tenders, researchers in contracting management and (public) network management have raised several critical notes on the use of monetary incentives in contracting procedures to stimulate performance and achieve self-regulation. For instance, Bresnen and Marshall [45] state that there are limitations to the use of incentives as means of reinforcing collaboration and developing commitment and trust, and conclude from case studies that cognitive and social dimensions strongly affect the impact of incentives. A similar finding is also reported by Rose and Manley [218] who conclude financial incentives to be less important to motivation and performance than relationship enhancement initiatives. According to the survey by Turrini et al. [249], social integration is a structural characteristic that emerges as a major determinant of the effectiveness of contracted groups. Klijn et al. [140] and Volker et al. [255] insist that in particular trust between group members strongly affects the outcomes of the process. Hence, although dynamic mechanism designs offer the mathematical tools to design

the theoretically perfect incentive scheme, its success in a practical setting with human actors is certainly not guaranteed.

Summarising the current state of the literature on performance-based contracting of groups, there is a consensus that the use of innovative, performance-based contracts in partnerships benefit all stakeholders and there are promising developments in theoretical work on team incentive structures, while at the same time there is much uncertainty with regards to the design of incentives and their effectiveness to incite self-regulation in actual tenders. This impasse is strengthened by the absence of any substantial experimental evidence or real-world successes. Hence, this thesis set out to prove the concept of self-regulation in performance-based contracts, provide guidance on their design and contribute techniques to optimise the value of such tenders. The next section describes the main research question that needs to be answered to achieve this goal and describes a decomposition thereof into smaller challenges that are addressed in each of the chapters of this thesis.

## 1.3 The Challenges of Self-regulation in Contracts

This thesis aims to bring the theory of dynamic contracting closer to implementation in real-world contracts by combining the literature of contracting and network management with the tools and techniques of dynamic mechanism design. In particular, this research focuses on the computational aspect of implicit coordination of decisions through incentive design in the context of self-regulating contracts with multiple participants. Using the MAINTENANCE PLANNING PROBLEM as a characteristic representation of the problem, this thesis investigates the application of various algorithmic techniques for the coordination of joint decisions in planning problems with self-regulation and the design of incentive mechanisms to optimise the outcomes thereof. This is captured in the main research question:

**Main Research Question**

> Can algorithmic techniques be employed to efficiently coordinate planning in self-regulating contracts and ensure successful outcomes while preserving the autonomy and interests of the agents?

This research question is divided into two elements that are later brought together: efficiently coordinating the decisions of agents in self-regulating contracts and ensuring successful outcomes through performance-based incentives while preserving the self-interest and autonomy of agents in the decision-making process. The coordination aspect is addressed by applying the framework of *decision-theoretic planning* to model and solve the planning problem inherent to self-regulating contracts. The incentive design is tackled through *game theory* and, in particular, by employing *(dynamic) mechanism design* techniques to design performance-based payments that incite self-regulation while respecting the interests and autonomy of agents.

Both decision-theoretic planning and game theory stem from the same root (decision theory) and they are tightly related in the setting of planning with self-interested agents [35, 48, 261]. In addition, the relation between game theory and planning has

been exploited before to achieve cooperation of agents, e.g. by Cavallo et al. [55] and Van der Krogt et al. [148]. Indeed, the decision-theoretic model that is used to represent the multi-agent decision-making problem underlying self-regulating contracts corresponds directly to a representation of a game that describes the actions of agents and their (joint) outcomes. Nonetheless, while their models to represent decision-making problems are broadly similar, the respective areas of focus is on completely different parts of the decision-making process. Whereas decision-theoretic planning focuses on computing decision strategies that maximise a predefined reward function for a given decision-making problem, game theory typically considers decision-making strategies as a given and analyses the potential outcomes of a problem when these strategies are followed. Mechanism design, a specific strand of game theory sometimes also referred to as 'inverse game theory', adds to game theory the design of incentives to steer outcomes towards desired ones. This is illustrated by the following high-level problem formulation of *self-regulating planning*, the decision-theoretic formulation of the planning problem underlying self-regulating contracts.

**Definition 1.1 Self-regulating Planning (high-level)**

Given a model that captures the multi-agent decision-making process and a set of performance-based payments, the self-regulating planning problem is to find an optimal strategy that optimises the expected sum of agent rewards and performance-based payments, or

$$\text{optimal strategy} = \underset{\text{all strategies}}{\arg\max}\ \mathbb{E}\Big[\text{agent rewards} + \text{performance payments}\Big] \tag{1.1}$$

From the decision-theoretic planning perspective, the goal in self-regulating planning is equivalent to the formulation in Definition 1.1: find the optimal decision-making strategy that in expectation maximises the total sum of rewards, e.g. contractual payments or bonuses, and (potentially negative) performance payments. On the other hand, the objective in mechanism design is to construct the payments in such a way that rational, value-maximising strategies – the objective of decision-theoretic planning – always result in favourable outcomes. Translating this to dynamic contracting, the latter is fundamental for the design of the dynamic contracts while the former is necessary to successfully implement contracts. Even more strongly, as will be seen in later chapters, typically the success of mechanism payments depends on the ability of agents to produce optimal plans. Hence, optimal coordination may even be a requirement for the incentive structures.

The remainder of this chapter discusses both approaches and the respective sub-challenges that are to be tackled in order to produce an answer to the main research question. Section 1.3.1 presents a decision-theoretic formulation of the self-regulating planning problem and introduces three research questions regarding decision-theoretic planning. Section 1.3.2 discusses the modelling of self-regulating planning as a game and two research questions concerning the design of incentives to ensure desired outcomes of such a game while preserving the interests and autonomy of agents.

### 1.3.1 Coordinating Self-regulating Planning

The coordination element of the main research question is addressed through the framework of decision-theoretic planning. From the algorithmic point of view, decision-theoretical planning is the reasoning about strategies for agent decision making where the impact of actions can be expressed in terms of rewards or costs as a consequence of outcome. If actions involve risks that may result in different outcomes, with different gains or losses depending on the outcome, it is known as *stochastic* or *contingent decision-theoretic planning*, or stochastic planning for short. The MAINTENANCE PLANNING PROBLEM is an example of such a stochastic planning problem: decisions regarding the planning of maintenance lead to outcomes with measurable revenues and (traffic) costs, while it is also stochastic by nature due to the potential delay of maintenance. But the model applies to self-regulating planning problems in general.

This thesis uses the model of *Markov decision processes* (MDPs) to represent the decision-making problem of self-regulating planning in terms of *states*, *actions*, *transitions* and *rewards* [28]. Simply put, an agent has to make a decision (perform an action) in its current situation (its state) such that it maximises its gain or utility (the reward), after which the situation changes as a consequence of its own decision and stochastic external influences (it transitions to a new state). This process continues until a terminal condition has been satisfied, for instance the expiration of a finite amount of time. In terms of the MDP model, the goal of decision theory to find optimal decision-making strategies can be reformulated as developing *decision policies* (or *policies* for short) that prescribe an optimal action for every possible state of the world such that the sequence of transitions that results from following that policy maximises the expected reward of the agent. These policies can thus be regarded as contingent plans that describe how to act in each (foreseen) situation.

More formally, the MDP model represents respectively the states, actions, transition probabilities and transition rewards of the decision problem. In the case of MPP the states could represent the planning environment conditions such as the segments that have been serviced, the resources available to the agents and current time step. The actions correspond to the maintenance activities that need to be performed. The transition probability function captures the distribution over outcomes in terms of what state will result after a maintenance operation. For instance the situation in the next state may depend upon a (random) maintenance delay occurring or not. Finally, the reward function describes the value gained or lost as a result of a state/action transition. This reward function is an operationalisation of the objectives of the agent and describes for instance the price it is paid upon completing a maintenance action, the (state-dependent) cost of performing work, its resource costs, etc. In this reward formulation it is implicitly assumed that all objectives can be operationalised into a scalar value, later it is argued that this is a strong assumption. Given such a model of the decision problem, the main goal of the agent is to find a decision policy that for every possible state of the world specifies the action to perform that in expectation maximises its reward, taking into account transition probabilities. Such a policy that in expectation maximises the reward of the agent is referred to as an *optimal policy*.

When multiple agents participate in the decision-making process, the corresponding multi-agent MDP model (MMDP) becomes more complex due to the interactions

between agents. While the actions of agents are often isolated, they share the environment in which they act. Therefore the states, their transitions and associated rewards are usually not independent. In other words, the outcomes and rewards of their decisions depend also on the decisions of other agents. Such true multi-agent decision-making problems require explicit coordination of agent decisions in the form of a joint decision policy that in expectation maximises the *joint reward*, known as the optimal joint policy. While in self-regulating contracts the individual decision problems of participants are independent and could be optimised independently, they share a contracted goal that is enforced through performance-based payments. As these payments depend on the *joint actions* they introduce an inter-agent dependency in the reward function of the MDP model. These (performance-based) payments can either have positive or negative impact on agent rewards, intended to respectively stimulate or discourage certain actions. For instance, in MPP these payments charge the total cost of traffic time lost (ttl) as a result of joint maintenance, making the revenue of a single service provider dependent on actions of other service providers. That is, when multiple agents concurrently perform maintenance, they are each charged a portion of the total costs caused by their combined actions. The ultimate goal of such payments is to get the agents to consider not only their own interests, because their personal gain now depends on their performance with respect to a common goal or their impact on other agents. As a consequence, agents are better off coordinating their decisions with others, i.e. self-regulate their decision making.

The self-regulating planning problem, and in particular its MDP formulation, is the basis for developing optimal joint policies using automated planning techniques. In particular, the first research question is how to capitalise upon the significant body of decision-theoretical research already available, thereby taking advantage of existing tools and techniques to solve self-regulating planning problems:

**RQ1 Coordinating self-regulating planning with existing techniques**

Can the vast body of existing planning literature, with its tools and techniques, be employed to develop joint policies for self-regulating planning problems?

It will be demonstrated in Chapter 3 of this thesis that self-regulating planning problems may be encoded as multi-agent MDPs, which in turn may be flattened into single-agent MDPs. This allows the use of existing multi-agent and single-agent MDP solvers to optimise self-regulating planning problems, thus giving access to a substantial library of tools. Nonetheless, almost all methods are either general-purpose solvers, that aim to support the solving of all types of MDPs, or are dedicated to problems with distinct characteristics, such as specific sub-class of MDP. The former type of solver does not leverage the specific structure of self-regulating planning problems, as for instance the independence between agents in their decisions but not their rewards. The latter type of solver is not tailored towards self-regulated planning.

In the pursuit for leverage, several authors already identified the potential of exploiting the decision-independence of agents. For instance, Becker et al. [26], Nair et al. [184], Varakantham et al. [251], Witwicki and Durfee [264] all investigated de-

centralised MDP models that possess an inherent decoupling between agent actions but not rewards. In centralised models, like MMDP and therefore self-regulating planning, this is not universally true as agent rewards are potentially defined over the full state and action space [145]. This has led researchers to examine specific sub-classes of MMDP that do allow decoupling of actions [25, 179, 193, 251] or resort to approximating the reward function as an independently factored sum [103, 143, 194]. Given the need for optimal coordination of self-regulating planning in the context of designing incentives, the latter approach is not applicable to this domain. Hence, the challenge that remains is to develop more efficient automated planning techniques that build upon ideas of previous work but target the properties specific to self-regulating planning:

**RQ2 Leverage the structure of self-regulating planning**

Can the structural properties of self-regulating planning problems be leveraged to produce optimal joint policies significantly more efficient than currently available methods?

Another challenge that is encountered in realistic examples of self-regulating planning is the inherent multi-dimensional nature of the agent's objectives. So far it was implicitly assumed that all of the goals of the agents and the performance payments can be operationalised solely in terms of revenue and cost. Indeed this may be a realistic assumption in many cases, e.g. when all non-monetary objectives can be 'priced' as in for instance the work of Van Moffaert et al. [175], but in many other scenarios such an operationalisation is not directly possible [132, 139, 160]. It may be that such a function simply is impossible to define, in which case human decision-makers are typically responsible for trade-offs between objectives. However, more often the pricing criteria or preferences over the objectives are not known while developing the joint plan [215, 175] or it is unrealistic to expect that human planners can exactly quantify their preferences over all objectives on a $[0, 1]$ scale that is typically required by automated planners to find optimal solutions [187]. The approach taken in such settings, where the objective preferences or prices are not exactly known when developing policies, is typically to find all optimal trade-offs between objectives and present them as alternatives to human decision makers [65].

Mathematically speaking, the reward functions of the agents in a *multi-objective MDP* (or MOMDP) are represented through multi-dimensional objective functions. Examples of objectives in the domain of maintenance planning could be the cost of maintenance of course, but also traffic time lost, risk aversion, environmental impact, safety level and other similar goals that are hard to quantify exactly in financial terms but may be equally important in the decision-making process. To incorporate the multi-objective nature of decision problems, the MDP model of each agent's planning problem will have multi-dimensional rewards (and payments) such that each dimension of the reward function corresponds to a reward for a specific objective. The operationalisation of the objectives of the MOMDP, required for automated planning techniques, is then performed by applying a *scalarisation function* to the objective value vector paired with a set of *scalarisation weights* such that a scalar value can again be computed.

For instance, the scalarisation function may be a price-per-objective function where the weights define the unit price per objective, or a preference function over objectives such that the weights express the relative importance of each objective. In the maintenance planning problem, a scalarisation function is used to trade-off maintenance costs versus traffic time lost as in Chapter 5, such that decision makers may choose policies from a set of alternatives instead of specifying exact preference weights beforehand.

The scalarisation function provides a measure for the quality of the trade-offs that are being made and may therefore be used as optimisation criteria in an automated planner, even when the actual operationalisation function is unknown when developing joint policies. That is, the function itself is known beforehand but the weights that "instantiate" it are unknown at plan time. This does increase the complexity of the planning problem however. Whereas before a single joint decision policy was sought, now the purpose is to find a set of trade-off policies so that for each combination of weights this set contains an optimal decision policy. In part, this additional complexity can be reduced by restricting the scalarisation function to be a *linear* combination of trade-offs, i.e. weighted-sum functions. This is a very natural restriction in many multi-objective settings that captures typical trade-off functions like the aforementioned price-per-objective and relative importance of objectives, and introduces a structure into the optimisation function that can be leveraged during plan development [215]. Still, even under the assumption of linearity, computing all optimal linear trade-offs poses a substantial computational challenge for multi-objective planning approaches.

Currently several methods exists to compute the set of optimal policies for any given objective weight [263, 253, 257, 21]. Nevertheless, all of these approach rely on the size of the state space to be 'small enough' in order to allow exact solutions [215]. Whether these approaches scale in the context of self-regulating planning problems such as MPP is not known. Instead, the reinforcement-learning approach of [161, 162] that deals with large, continuous state spaces may be better equipped to solve instances of MPP, but such a learning approach is not able to guarantee optimality. Finding optimal policy sets for self-interested planning problems is therefore still an open question for research:

**RQ3 Self-regulating planning with multi-dimensional objectives**

Can multi-objective planning methods be applied to self-regulating planning problems with multi-dimensional objectives to efficiently find an optimal joint policy for every linear trade-off between objectives?

### 1.3.2 Designing Incentives for Self-regulation

The previous section focused on the coordination aspect and it was implicitly assumed that the agents participating in self-regulating planning are fully cooperative. That is, they are willing to share all their private information truthfully, comply with the global objective of optimising the sum of *all* agent rewards and mechanism payments, and accept that a centralised decision-making algorithm coordinates their actions. In competitive scenarios such as the setting of dynamic contracting however, agents are typically not so cooperative or truthful by nature. More commonly, agents strive only to maximise their personal gain without regard for the interests of other agents or

a global goal. More strongly, realistic agents will actively seek to exploit potential gaps in the contract or increase their gain by misreporting information regarding their decision-making process. For example, an agent may declare a much higher value for one of its actions just so that it will be included in the joint plan for sure, even though actions of other agents would have resulted in a much higher value *to the group* (and thus not to the 'lying' agent). Moreover, even if agents act truthfully, they are usually not eager to share their private information regarding profits, costs, resources, etc., in the context of a commercial contract, nor to completely surrender their autonomy to a central planner. Hence, in order for self-regulating contracts to ensure successful outcomes, it must provide strong incentives to the agent to effectuate truthfulness, sharing of private information and (partially) submission of autonomy.

Following the line of reasoning of authors in many related works [9, 106, 117, 135, 205, 254], the challenge of designing incentives to incite self-regulation is addressed using concepts from *game theory* and *mechanism design*. Game theory is a strand of decision theory inspired by economics that "aims to model situations in which multiple participants interact or affect each other's outcomes" [188]. In essence, the self-regulating planning problem underlying the dynamic contracting procedure (Definition 1.1) can be modelled as a multi-player game with actions and rewards, where players are autonomous entities that make their own planning decisions in a rational fashion. With such a game model, the decisions of agents and the impact thereof on the outcome of the game can be mathematically analysed. Hence it becomes possible to vary the rules of the game and investigate their impact on the outcomes of the game. Moreover, if a game-theoretical model is available, mechanism design techniques can be employed to construct payment structures such as the 'performance payments' of Equation 1.1 that guarantee desired outcomes of such a game (see e.g. the work by Conitzer and Sandholm [67], Dash et al. [72], Maskin [168] or Nisan et al. [188]). In effect, these payments reward or penalise agents for their contribution to a global goal so that it becomes in their personal interest to collaborate, implicitly leading to self-regulation.

The problem of truthfulness remains however, even if incentives stimulate self-regulation. Agents may deliberately misinform others to their personal benefit if they know the payment mechanism that is being used; typically public information in the context of contracts. To this end, recent studies have drawn inspiration from mechanism design approaches as taken by Cavallo et al. [55], Van der Krogt et al. [148] or Pathak [206] to design incentives that not only align interests but also stimulate truthfulness ("overcome the asymmetric distribution of information") in contracts [106, 117, 254]. In other words, these incentives can be used as the basis for the performance-based payments of Definition 1.1 to both stimulate performance and enforce truthfulness.

While performance-based rewards can preserve the personal interests of the agents and prevent opportunistic behaviour, it also introduces a new dilemma for the agents. On the one hand, their rewards depend on actions taken by or together with others and it is in their best interest to coordinate their joint decisions. On the other hand, agents often act in a competitive environment and therefore are reluctant to yield autonomy or exchange private information regarding e.g. their activities, resources and

costs with their competitors [11, 150]. This problem can be partially alleviated if the agents accept a neutral (third) party to coordinate the decisions on behalf of all agents, such as the asset manager in the domain of road maintenance [254, 228, 158]. This party, commonly referred to as the *centre*, collects the decision-making processes of all agents, combines them into a multi-agent model and determines the optimal joint decision policy.[8]

Still, although sharing sensitive business information with a neutral centre may be more acceptable, the submission of autonomy remains a prerequisite of this approach. If agents do not fully adhere to the joint plan developed by the centre, a successful (joint) outcome can no longer be assured. Nonetheless, the performance-based payments make it in the best interest of even these autonomous agents to coordinate their decisions with others, albeit it on the decision policy level only [115, 197]. In such scenarios, sharing and iteratively updating full decision policies may be a more acceptable alternative [127, 228]. Dealing with the both the self-interest as well as the autonomy of agents such that self-regulation between the agents is still encouraged is the first research question for incentive design:

**RQ4 Self-regulating planning with self-interested agents**

Can game-theoretical techniques be employed to guarantee optimal joint decision policies for self-regulating planning problems if agents are autonomous and self-interested?

In other words, the objective is to produce joint policies that optimise the *actual* reward of the team of agents and not their *reported* rewards. This can only be achieved if agents are honest about their interests. Hence the real challenge the design of incentives that stimulate agents to be truthful or, vice versa, to discourage agents from cheating by ensuring that reporting their true information yields them the highest reward.

The final challenge that is addressed in this thesis concerns the implementation of self-regulation incentives within realistic contracts. Indeed, game theory and mechanism design offer techniques to design incentives such that not only agents are encouraged to self-regulate but also willing to disclose private information, act truthfully and even surrender their autonomy. Nonetheless, the success of these approaches relies on the assumption that monetary incentives can effectively influence the decision-making process of agents and incite self-regulation within the group, which is a strong assumption in practice. Although the notion that collaboration could be 'engineered' through incentive systems was ventured by Bresnen and Marshall [44], the same authors surveyed that it has mostly led to a multitude of guidelines on partnering and alliances and "that there are limitations to the use of incentives as means of reinforcing collaboration" [45]. The same authors note that in particular the cognitive and social dimensions strongly affect the impact of incentives, a conclusion also drawn by authors in related studies [76, 140, 218, 249, 255]. Furthermore, the absence of empirical evidence makes it is unclear whether such perfectly engineered incentives achieve their

[8] *Although such a solution may still suffer from unfairness with respect to the reward of single agents, e.g. one agent suffers while the overall reward is maximised. This may be countered by compensating agents for their losses, as in the dynamic maintenance mechanism of Chapter 6.*

intended goals when confronted with real actors [254]. In this context, [218] and [249] raise a similar concern that despite an overall believe that incentive mechanisms improve value for money during procurement and project performance during execution, empirical research is scarce.

Therefore, validating the assumption that monetary incentives lead to the self-regulation – the key assumption underlying the approach proposed in this thesis – is key to a potential implementation in group contracts. The last challenge hence focuses on the decision-making process of human decision makers in the presence of monetary incentives and the emergence of self-regulation as an effect of such incentives to establish a first proof of concept of the potential for self-regulation in group contracts:

**RQ5 Confronting self-regulating planning with the real world**

Can the theoretical guarantees of self-regulating incentives be transferred to real-world group tenders to ensure successful outcomes if planning decision are made by human decision makers?

These challenges form the research questions for the research performed in this thesis. Throughout the chapters of this work, the research questions are addressed and contributions are made either in the form of potential solutions or new knowledge. The contributions per chapter are summarised in the next section.

## 1.4 Outline and Contributions

Summarising the introduction of this thesis, the main goal is to transfer the advantages of performance-based contracts into the settings where a team of agents is contracted. Through the use of monetary incentives, agents should be implicitly stimulated to contribute to the contracted goal as well as coordinate their joint decisions. That is, the team of agents becomes *self-regulating*.

As discussed in Section 1.3, self-regulation is addressed in two separate parts: the coordination of agent decisions and the design of incentives. The chapters in this work follow a similar ordering. First the Chapters 2 to 5 deal with the challenges of coordinating decisions in self-regulating planning problems. The subsequent chapters, Chapters 6 and 7, focus more on the design of incentives so that self-regulation may be successful when dealing with actual agents. Chapter 8 recapitulates on both parts and formulates an answer to the main research question of bringing self-regulating contracts such as the Dynamic Contracting Framework closer to reality. Below is a more detailed outline of each chapter and how it contributes to the goals of this research and the current state of the art in the areas corresponding to the subjects.

**Chapter 1** (this chapter) introduces the maintenance planning problem and the dynamic contracting approach that together form the basis for the work in this thesis. This chapter describes the problem, introduces the dynamic contracting procedure and presents current challenges of bringing dynamic contracting into real-world contracts. The dynamic contracting framework, its requirements and its design, was initially published by Volker et al. [254]. The combination of game theory, mechanism design and

decision theory as the basis for the operational environment was proposed by Scharpff et al. [228].

**Chapter 2** provides an overview of decision-theoretic planning literature that is required as background knowledge for the work in this thesis and discusses previous work that inspired the contributions of this thesis.

**Chapter 3** formalises MPP of Section 1.1 into a mathematical formulation and illustrates that it can be modelled as a Markov Decision Process (MDP). This modelling allows general MDP solvers to find optimal joint policies for MPP, thus making a first step towards surmounting RQ 1. Thereafter the chapter proposes also an improvement upon the first but naive encoding that exploits the structure of the MDP to achieve faster policy searches, hence addressing RQ 2 using *generic* MDP solvers. In addition, this chapter presents a first approximation algorithm for MPP based upon Monte-Carlo Tree Search. The contributions of this chapter have been published by Scharpff et al. [228] and Roijers et al. [216].

**Chapter 4** specifically targets the structure of MPP to produce optimal joint policies more efficiently. In particular, many self-regulating planning problems such as MPP exhibit a structure known as transition-independence, i.e. the state/action transitions of all agents are independent but their rewards are not, and this can be exploited in joint policy searches. This chapter describes the `Condition Return Policy Search` (`CoRe`) algorithm that uses a specific type of data structure called the *Conditional Return Graph* to compactly represent and re-use decision structures and corresponding rewards. The experimental analysis at the end of the chapter demonstrates that this novel algorithm is able to leverage the structure of MPP for more efficient searches and thus addressing RQ 2. The work in this chapter extends the first introduction of `CoRe` by Scharpff et al. [229].

**Chapter 5** considers the multi-dimensional nature of objectives, in light of RQ 3. Instead of methods that produce a single optimal joint policy, this chapter considers multi-objective planning approaches that find the *Convex Coverage Set* over all objectives when a specific scalarisation function is known but the scalarisation weights are not. That is, the aim is to find a set of policies such that for every weight this set will contain the policy that optimises the scalarised reward. Roijers et al. [217] proposed the `Optimistic Linear Support` method (`OLS`) that can produce such a set but it requires many optimal policy computations. The experiments of Chapters 3 and 4 show that one such a computation may require much effort, hence this approach is infeasible for all but the smallest problems. Instead, Chapter 5 proposes two new methods that approximate the convex coverage set through approximate policy computations, making it more feasible in practical settings such as the one considered in this thesis. The first algorithm is `Approximate Optimistic Linear Support`, an approximate version of `OLS` with bounded quality guarantees, and the second is `Scalarised Sample-based Iterative Improvement`, a method that is better when focusing on specific weight regions, both published Roijers et al. [216].

**Chapter 6** seeks to alleviate the assumption of cooperation that was implicitly made in all previous chapters to begin research at the planning problem itself. In this chapter agents are regarded as non-cooperative or selfish, and they will therefore do anything to increase their profits even when this hurts other agents. Furthermore, the chapter assumes a private values setting, i.e. all planning information is private to the agents, and optimal joint policies can only be developed based upon *reported* information that may be incomplete, untruthful or both. When agents are willing to reveal all their (possibly false) planning information to a centre, Chapter 6 shows that the payment mechanism can be designed as a *dynamic maintenance mechanism* that ensures that truthful reporting is actually in the best interest of agents, thus tackling RQ 4 and establishing the boundary conditions for self-regulation in contracts. If agents are reluctant to disclose their planning information or submit their autonomy entirely, it is impossible to guarantee jointly optimal policies. However, in such a setting the planning problem can still be modelled as a *stochastic planning congestion game* in which agents iteratively improve upon a joint plan in a best-response fashion. This game enables self-regulation even in a setting with autonomous agents. The two approaches were introduced by Scharpff et al. [228], this chapter extends that work with formal proofs that the maintenance mechanism is a dynamic mechanism and stochastic planning congestion games are a special class of congestion games with similar properties.

**Chapter 7** makes the first step towards bringing the theory of Chapters 3 to 6 into practice. Addressing the challenge of RQ 5, three research questions are presented that focus respectively on the effectiveness of incentives to change the decision-making of human planners, their potential to incite self-regulation and the role of social relationships in performance-based frameworks. These questions are studied through controlled experiments using a serious game that simulates the planning and execution of maintenance work similar to the MAINTENANCE PLANNING PROBLEM in the setting of a performance-based group contract. The experiments show that monetary incentives influence decision making, but their effect may be opposite to their intended aim and can lead to undesirable competition between group members. It was, however, also found that this competitiveness is not shown in groups where members are familiar with each other. This leads to the conclusion that penalty-based incentive mechanisms probably interfere with self-regulation and that the social dimension of contractor collaboration is paramount to the success of performance-based contracting in group tenders. Besides the validation, this work also contributes of a sandbox environment for dynamic contracting that can be used in related and further research without the enormous costs typically associated with real-world failures. The work on the design and application of the serious game is described by Scharpff et al. [232]. A detailed explanation of its modelling is given by Scharpff et al. [231].

**Chapter 8** discusses the implications of the work presented in all the chapters and reviews their contribution towards the research questions defined in this chapter.

### 1.4.1 Reading Guide

Naturally, all readers are encouraged to go over all chapters of this dissertation. However, for those with a specific background or focus of interest the following reading guide may help to quickly find the topics of interest. The summary, introduction and discussion chapters are relevant for all readers. They are found on page i and in Chapters 1 and 8 respectively.

Readers most interested in decision theory and stochastic multi-agent planning are referred to Chapters 3 to 5. These chapters address respectively maintenance planning as a multi-agent MDP model, a novel stochastic solving method that exploits the structure of particularly TI-MMDPs and solving the multi-objective version of maintenance planning. Chapter 2 and Appendix D may serve as a refresher on stochastic planning and complexity theory, although the reader knowledgeable in this area may focus on the more advanced Sections 2.6 and 2.7.

For readers with an emphasis on (algorithmic) game theory and mechanism design, it is advised to review Chapter 2 and Appendix E as background. Continue with the maintenance planning problem definition and its MDP formulation in Sections 3.1 to 3.3 before studying the dynamic mechanism proposed in Chapter 6. Furthermore, Chapter 7 may be of interest as it presents an empirical study into the rationale behind the human decision-making process in the context of team incentives.

Those mainly interested in self-regulation and contracting theory are advised to start with Chapter 3.1 to comprehend the decision complexity and challenges of self-regulating planning. A brief study of Appendix E and Chapter 6 provides the reader with an understanding of game theory and how mechanisms could be applied to incentivise agents, but also shows that using dynamic mechanisms to optimise performance and minimise costs are computationally very expensive. A more pragmatic approach is discussed in Chapter 7 where dynamic contracts with mechanism-design inspired payments are used in a simulation game to investigate their effect on human decision makers.

# Chapter 2

# Stochastic Planning using Markov Decision Processes

In this section all concepts related to self-regulating planning will be reviewed that are relevant for the work presented in this thesis. Starting from the basics, Section 2.1 will introduce the Markov decision process, that has proven one of the keystones in stochastic planning, and basic solution methods for this model are discussed in Section 2.2. From there on the model will be gradually extended to other settings: problems with states composed of variables (Section 2.3), problems with uncertainty about the current state (Section 2.4), multi-agent problems (Section 2.5), special cases of the latter two (Section 2.6) and problems with multiple objectives (Section 2.7). The background presented in this section has been derived from many sources but a few in particular that have been used many times are: the book on MDP by Puterman [208], the thesis of both Cassandra [51] and Kaelbling et al. [129], and the tutorial on multi-agent sequential decision making (MSDM) by Spaan et al. [243] presented at the International Conference on Automated Planning and Scheduling (ICAPS). For completeness, a brief overview of game theory – the background required in particular for Chapter 6 – can be found in Appendix E.

## 2.1 Markov Decision Processes

Over the course of time several frameworks have been proposed to model planning problems with uncertainty, such as the self-regulating planning problem of Section 1.3.1, but none has been studied and applied so widely as the Markov decision process (MDP) framework, introduced by Bellman [28]. Essentially, an MDP describes a Markovian ('memoryless') system of states[9], where the dynamics are defined through actions, transition probabilities and associated rewards. The most common definition of MDPs is given in Definition 2.1.

[9] *Although histories can be incorporated as part of the state to serve as memory.*

**Definition 2.1 Markov Decision Process (MDP) [28]**

A Markov decision process, or MDP, is defined as the tuple $\langle S, A, P, R \rangle$ such that

- $S$ is a finite set of states the process can be in, often referred to as the *state space* of the MDP,
- $A$ is a finite set of actions that can be performed, also known as the action space,
- $P$ is a transition probability function $P : S \times A \times S \mapsto [0, 1]$ where $P(s, a, \hat{s})$ denotes the probability that the system will end up in state $\hat{s}$ if action $a$ is taken in state $s$, and
- $R$ is a transition reward function $R : S \times A \times S \mapsto \mathbb{R}$ where $R(s, a, \hat{s})$ is the reward obtained when state $\hat{s}$ is reached after taking action $a$ in state $s$.

Many stochastic planning problems can be modelled quite naturally in the Markov decision process framework. One example of a domain in which MDP can be applied rather intuitively is path planning for robots, which is used in examples throughout this chapter. A few of the many others are maintenance planning [228], supply-chain management [88] and work-flow scheduling [266]. Note that here the most general definition of a reward function is presented. The reward can alternatively be written as a state-based or state/action-based function. Both notations can be trivially transformed into the full transition-based rewards presented here: $R(s) = \sum_{a \in A} R(s, a) = \sum_{a \in A} \sum_{\hat{s} \in S} P(s, a, \hat{s}) R(s, a, \hat{s})$. The transition function is sometimes denoted as a conditional probability $P(\hat{s}|s, a)$. However, to prevent confusion with probability functions this notation will not be used.

A solution to a stochastic planning problem modelled as an MDP is known as a contingent plan or, the term that will be used mostly, a *decision policy* or simply *policy*. Informally, a policy $\pi$ is a set of decision rules that for each state of the problem, and possibly the history of previously encountered states, prescribes an action to perform. After a policy has been developed for a stochastic planning problem it can be queried during the execution phase for the next action given the current state of execution. This results in a new system state, depending on the transition probabilities, and the process is repeated until some termination criterion has been achieved (reached the goal, ran out of time, etc.). Note that a policy needs to be developed *before* the execution is performed and this type of planning is therefore referred to as offline planning.

In general, a policy can depend on the entire history of actions and states encountered so far. If this is the case, the policy is said to be non-stationary. However, when only the current state is significant or a history can be compactly represented by every state, i.e. the state is Markovian, it suffices to consider only *stationary policies*. As all the problems discussed in this thesis fall in the latter category, the focus will be on stationary policies and this will no longer be mentioned explicitly when discussing policies in future sections. The most general definition of a (stationary) policy is given in Definition 2.2.

**Definition 2.2 Policy**

A (stationary) policy $\pi$ is a mapping of states and actions to a probability such that $\pi(s,a) \in [0,1]$ specifies the probability of taking action $a$ in state $s$, and $\sum_{a \in A} \pi(s,a) = 1$. If the policy for each state and action assigns 0 or 1 probabilities such that $\forall s \in S$, $\exists a \in A$: $\pi(s,a) = 1$ and $\pi(s,a') = 0$ for all $a' \neq a$, it is said to be *deterministic*, otherwise it is non-deterministic (multiple actions) or stochastic (distribution over actions).

This work will restrict itself to only deterministic policies, i.e. policies that always prescribe the same action for each of the states. Deterministic policies in this sense also have the Markovian property. In future sections, this will no longer be stated explicitly when discussing policies. In addition, the common notational shorthand $\pi(s) = a$ is used to denote $\pi(s,a) = 1$.

Typically in stochastic planning the interest is in finding a 'good quality' policy that is expected to yield a high reward or, in cost minimisation problems, a low cost. This quality is captured by the value of a policy, which is defined as the expected reward that is obtained when following the decisions specified by the policy. For problems with a *finite* planning horizon, i.e. a fixed maximal number of plan steps, the value of a policy $\pi$ is defined as:

$$\begin{aligned} V^{\pi}(s^0) &= \mathbb{E}\Big[\sum_{t=0}^{h-1} R(s^t, \pi(s^t), s^{t+1})\Big] \\ &= \begin{cases} \sum_{s^{t+1} \in S} P(s^t, \pi(s^t), s^{t+1})\Big(R(s^t, \pi(s^t), s^{t+1}) + V^{\pi}(s^{t+1})\Big), & otherwise \\ 0, & t \geq h \end{cases} \end{aligned} \tag{2.1}$$

where $s^0$ is the initial state of the planning problem and $h$ the planning horizon length. When $V^{\pi}$ is written the initial state is assumed implicitly, i.e. $V^{\pi} = V^{\pi}(s^0)$. In this formulation $s^t$ denotes the state at time $t$ and $s^{t+1}$ are possible successor states that, depending on the transition probability function, can result from taking the action $\pi(s^t)$ prescribed by the policy.[10] The value function can easily be adapted for problems with an *infinite* planning horizon ($h = \infty$) by adding a discount factor $\gamma$:

$$V^{\pi}(s^0) = \mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t R(s^t, \pi(s^t), s^{t+1})\Big], \gamma \in [0,1) \tag{2.2}$$

Discounting rewards that are further away in time makes it possible to compute the expected value of a policy even though the number of planning steps is unknown in advance. The key difference here is that eventually the expected value of an infinite horizon policy will converge to zero due to this discount factor and thereby making it

[10] *Accents are used to discern between states within the same time step, e.g. $s^2$ and $\hat{s}^2$ are two distinct states at time 2.*

possible to find policies for this type of planning problem. Moreover, as the discount factor decreases exponentially over time, policies that maximise reward early are likely favoured; however this depends greatly on the value of the discount factor. A value close to zero will result in a high preference for short-term reward (greedy-like behaviour) whereas a value close to one allows for more reasoning about future effects of current decisions. Keep in mind however that in most solution techniques a high value for the discount factor will have a negative effect on the computation time as more future states will still be interesting in terms of reward.

As mentioned before, the value of a policy expresses its quality in terms of expected future return. Since the realisation of the uncertainties is not known until the policy is executed, the best one can do is maximise the expected reward of a policy. A policy that does just this is known as an *optimal policy* $\pi^*$ and is defined for both finite and infinite problems as:

$$\pi^* = \arg\max_{\pi \in \Pi} V^{\pi}(s^0) \tag{2.3}$$

where $s^0$ is the initial state of the problem and $\Pi$ the set of all possible policies that exist for the problem. Note that although the expected optimal value $V^{\pi^*}$ is unique, there may exists multiple optimal policies that attain this value. Typically in such cases any of the optimal policies suffices and it is not important which one is chosen as both achieve the goal objective of maximising expected reward.

**Example 2.3** Robot path-planning as Markov decision process

To illustrate the Markov decision process model, this examples shows how robot path-planning can be represented as an MDP and describes an example policy for this problem. Figure 2.1a shows a stochastic path-planning example of a robot that needs to find its way to the exit, marked by the red symbol in the lower left corner of a 4 by 4 grid. This robot is deployed in a remote area and it is not possible to control it from afar (similar to e.g. a mars rover). Instead, the robot will be given a contingent routing strategy that is developed before its deployment such that by following this strategy it is able to respond to all (anticipated) eventualities that may occur. To develop such a contingent strategy, or policy in the context of stochastic planning, first a model of the environment in which the robot is acting must be constructed.

The robot can move horizontally or vertically but cannot move outside if the grid. When it has reached the exit, it will be awarded a bonus of $+100$. Every move North, East, South or West costs energy and therefore costs 1. Finally, with a probability of 10% ($p = 0.1$) the robot fails to perform a move, due to e.g. objects in its way or a brief malfunction. When the robot fails, it will remain at its current location but the cost of the move will still be charged. Figure 2.1b shows a solution to this problem in the form of a path, which requires at least four moves but may take more if failures occur.

This path-planning problem can be modelled as an MDP. A state in this problem describes the exact location of the robot, therefore the state space $S$ contains a state $s_{(i,j)}$ for every position $(i, j)$ on the grid. As the robot is in location $(3, 2)$ at the start, the initial state $s^0$ is $s_{(3,2)}$. The robot has four moves – move North, East, South or West – and the action set $A$ of the MDP has corresponding actions $\{\texttt{N}, \texttt{E}, \texttt{S}, \texttt{W}\}$. Transitions capture the move from one location to another, depending on the direction of the move as well as its success rate. For example, the transition probability for moving from state $s_{(2,4)}$ to state $s_{(2,3)}$ when performing action $\texttt{N}$ is captured by $P(s_{(2,4)}, \texttt{N}, s_{(2,3)})$ and is equal to

0.9, because it can fail with 0.1 probability. The possible failure is specified by the transition $P(s_{(2,4)}, \texttt{N}, s_{(2,4)}) = 0.1$. Together these two transitions are the only ones possible from a state $s_{(2,4)}$ when action N is taken (their probabilities sum to 1) and every other transition from $s_{(2,4)}$ with action N has probability 0, meaning that such a transition can never happen. All other action move transitions are defined analogously. Note that transitions for illegal moves are not in the transition probability function as there is no state outside the board that the MDP can transition to.

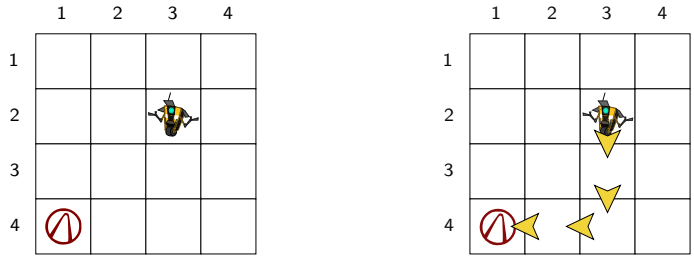



**Figure 2.1** *Example path-planning problem: (a) the robot needs to find its way to the exit in the lower left corner of the grid by moving North, East, South or West. An example of such a path is shown in (b). This path requires (at least) four moves and yields the robot a maximum reward of 100 - 4 = 96.*

The reward function corresponding to this example problem is easily defined: for every (failed) move the robot is penalised by 1 and when it reaches the goal state ($s_{(1,4)}$) it is rewarded 100. Thus for every transition from a state $s_{(i,j)} \in S$ for every action $a \in A$ to a new state $s_{(i',j')} \neq s_{(1,4)} \in S$ with non-zero probability, the reward is given by $R(s_{(i,j)}, a, s_{(i',j')}) = -1$. Observe that $s_{(i,j)}$ and $s_{(i',j')}$ may be the same state if the move fails. When the goal state is reached, a different reward is obtained. Every transition from a state $s_{(i,j)} \in S$ to $s_{(1,4)}$ through action $a \in A$ has a reward $R(s_{(i,j)}, a) = 99$ (100 - 1 to include the cost of the move).

The robot in the example has to plan a path towards the exit. Naturally, it wants to find the path that in expectation minimises its energy costs but ultimately reaches the goal (and its reward). As failures may occur during his moving towards the goal, it is not sufficient to simply specify a path like $\langle \texttt{S}, \texttt{S}, \texttt{W}, \texttt{W} \rangle$: if for instance either of its southward moves fails, it will end up in location $(1, 3)$ and never reach the goal. Instead, the robot requires a contingent path in the form of a policy that *given its current location* specifies the best way to go. One such a policy can be to first move South until the grid border is reached, then continue westward until the goal is reached. This can be expressed as a policy $\pi$ with two simple rules based on the current state $s_{(i,j)}$ the robot is in:

$$\pi(s_{(i,j)}) = \begin{cases} \texttt{S} & \text{, if } j < 4 \\ \texttt{W} & \text{, otherwise} \end{cases}$$

and of course, when the robot reaches the goal state, it is done and will no longer consult its policy what to do.

Now assume that the planning horizon $h$ is 5 or, in other words, the robot is given 5 moves to try and reach its goal. Following the previously described policy, the robot can end up in any of the states along the path displayed in Figure 2.1b, including its initial

state, depending on how many out of its five moves have failed. Figure 2.2 shows the state/action diagram that corresponds to executing this policy for 5 steps.

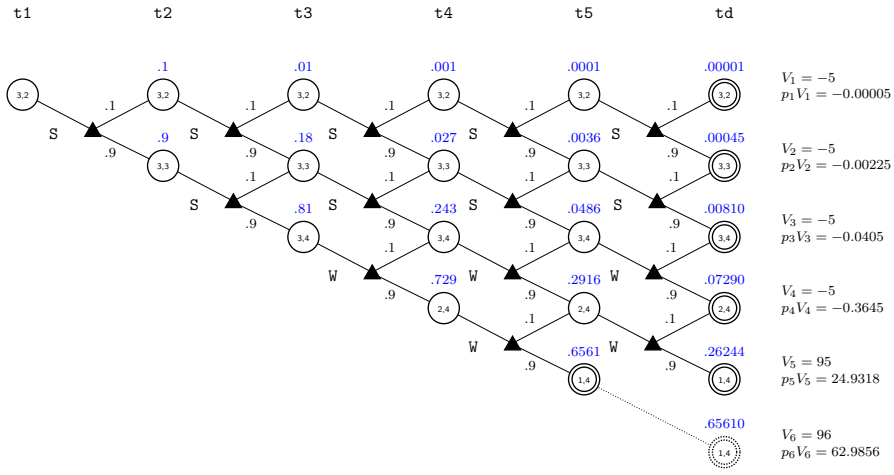



**Figure 2.2** *State/action diagram corresponding to the policy first South then West. States are represented by white circles. Transitions are depicted as a path of edges labelled by the action taken and the transition probability. The states are labelled above (blue) with the probability of reaching them.*

Initially, the robot starts in state $s_{(3,2)}$ and its only available action is the South move under the policy, due to its current state being not in the bottom row. The state/action diagram therefore contains an edge outwards from state $s_{(3,2)}$ labelled with action $S$. This action can have two outcomes: either the robot moves to location $(3, 3)$ or it remains at $(3, 2)$. This is illustrated by the chance node (black triangle) and the two edges after it, corresponding to both transitions, labelled with the transition probability. For example, the transition probability $P(s_{(3,2)}, \mathtt{S}, s_{(3,3)}) = 0.9$ is represented by the edges labelled S and $0.9$ between the state nodes labelled $(3, 2)$ and $(3, 3)$. The reward of each transition is shown in blue.

By following the policy the robot can end up in any of the terminal states, shown as double circles. In these states either the time is up or the goal has been reached, as in the bottom terminal states (or both in the case of the terminal state 5). Next to each of the terminal states the figure shows its value and its expected value. The first is obtained by multiplying transition probabilities of taken transitions, e.g. terminal state 2 can be reached from the top two states at time t5 and therefore has probability $0.0001 \times 0.9$ (successful move from the topmost state at time t5) plus $0.0036 \times 0.1$ (a failed move from the state below) which equals $0.00045$. Note that the probabilities of all transitions in a certain time step must sum to one. The same must be true for the set of terminal states.

The value of a terminal state is computed by summing over the transition rewards, in this example -1 times the number of moves and $+100$ if the robot is at goal state $s_{(1,4)}$. If the robot did not make it, the reward is always $-5$, otherwise it is $+96$ or $+95$, depending on whether it needed 4 or 5 steps to the goal respectively. Finally, the expected value of one terminal state is simply its value times the expectation of reaching it, i.e. the product of transition probabilities to get there. The expected value of the South-West

policy is $\sum_i p_i V_i = 87.5101$, which is actually the optimal expected value for this example (intuitively: no shorter path is possible).

## 2.2 Finding Optimal Policies



There has been a substantial body of research into solution techniques for MDPs, both exact and approximate. In this section only the two most influential techniques are presented that are the basis for many other solutions. For a more thorough study of available MDP solving methods the book by Puterman [208] is recommended. Here – and throughout the entire thesis – phrases like 'solving an MDP' are used to mean searching for the optimal policy for that MDP that has the highest expected value.

One of the earliest techniques to find optimal policies for general MDPs is due to Bellman and is known as `Value Iteration` [28]. This method is based upon the observation that the value of the optimal policy can be computed using a recursive formula, known as the Bellman equation:

$$V^*(s) = \max_{a \in A} \sum_{\hat{s} \in S} P(s, a, \hat{s}) \Big( R(s, a, \hat{s}) + \gamma V^*(\hat{s}) \Big) \tag{2.4}$$

Notice that the Bellman equation is almost equal to the previously discussed Equation 2.1, but for every state now the action that maximises the expected future reward is always taken. For an optimal policy $\pi^*$ that does exactly this, the two are equivalent. Essentially, the Bellman equation computes the optimal expected value of a MDP by recursively computing the optimal value from every new state that can be reached from the current state by trying all available actions and returning for each state the action that maximises the expected value, until the planning horizon has been reached. In other words, the optimal value computation for state $s$ is decoupled into $|S'|$ optimal value computations, where $S' = \{\hat{s} \in S \mid \exists a \in A \colon P(s, a, \hat{s}) > 0\}$.

The Bellman equation applies both to finite ($\gamma = 1$) as well as infinite horizon MDPs, although the former case requires an addition of $V^*(s) = 0$ for every state $s \in S$ with time $t \geq h$ (as in Equation 2.1). Once the optimal value has been computed, the optimal policy can be derived by choosing for each state the action that maximised the expected reward, i.e.:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{\hat{s} \in S} P(s, a, \hat{s}) V^*(\hat{s}) \tag{2.5}$$

The `Value Iteration` algorithm uses dynamic programming to successively improve an estimate of the Bellman equation (Equation 2.4) following Equation 2.6:

$$\begin{aligned} V^0(s) &= 0 \\ V^{k+1}(s) &= \max_{a \in A} \sum_{\hat{s} \in S} P(s, a, \hat{s}) \Big( R(s, a, \hat{s}) + \gamma V^k(\hat{s}) \Big) \end{aligned} \tag{2.6}$$

In Equation 2.4, $V^k$ denotes the value function of iteration $k$. In each iteration the value function will more closely resemble the optimal value function as the state

values will become increasingly accurate. As long as no state is excluded from the value computation, this algorithm is guaranteed to converge to the optimal value function $V^*$. Once the algorithm has converged, thus $V^{k+1} = V^k$ for all states, the optimal policy can easily be extracted from the value function by selecting for each state the action that maximises the state value (Equation 2.5). In practice often a convergence parameter $\epsilon$ is used such that the algorithm terminates when $\max_{s \in S} V^{k+1}(s) - V^k(s) \leq \epsilon$ and hence it can be used as an approximate algorithm that produces policies with a value of at most $\epsilon$ away from the optimal.

Another well-known solution method for MDPs is the `Policy Iteration` algorithm due to Howard [119]. `Policy Iteration` resembles value iteration in that it is also based on the Bellman equation, but instead of iteratively improving the state value estimate of Equation 2.6 it performs round-based improvement of an initial (random) policy. The algorithm is shown in Algorithm 2.4. In each iteration the algorithm evaluates the current policy, i.e. $V^\pi(s)$ is computed for all states and current policy $\pi$, and this value is used to check if there are states for which the policy can obtain a higher value by switching to another action. For all such states, the policy is changed such that these actions are now preferred and the process is continued until no more changes occur.

**Algorithm 2.4** `Policy iteration` [119]

1: $\pi^0 \leftarrow$ initial (random) policy
2: $k \leftarrow 0$
3: **repeat**
4: $\quad k \leftarrow k + 1$
5: $\quad V^k(s) = \sum_{\hat{s} \in S} P(s, \pi^{k-1}(s), \hat{s}) \Big( R(s, \pi^{k-1}(s), \hat{s}) + \gamma V^{k-1}(\hat{s}) \Big)$ ▷ $\forall s \in S$
6: $\quad \pi^k(s) = \arg\max_{a \in A} \sum_{\hat{s} \in S} P(s, a, \hat{s}) \Big( R(s, a, \hat{s}) + \gamma V^k(\hat{s}) \Big)$ ▷ $\forall s \in S$
7: **until** $V^{\pi^k} = V^{\pi^{k-1}}$
8: **return** $\pi^k$

The evaluation step in `Policy Iteration` (line 5) is commonly done by solving the set of linear equations that correspond to Equation 2.6 through linear programming [119]. As is the case for value iteration, this algorithm is guaranteed to converge as long as all states are considered during the search, but in practice it often tends to do so much faster than `Value Iteration` [208]. The `Modified Policy Iteration` algorithm, introduced by Puterman and Shin [209], replaces the exact linear programming step by a (small) number of successive approximations. This approximate policy evaluation is often good enough in practice and can be computed significantly faster.

Both `Value Iteration` and `Policy Iteration` are algorithms that run in polynomial time.[11] The first requires at most $|S|$ rounds of $|S| \times |A|$ (trivial) computations. For the latter algorithm a similar analysis can be made to obtain a worst-case computational complexity of $O(|S|^3)$, although in practice it might perform better. Still, finding optimal policies for real-world instances of Markov decision processes is often

[11] *A refresher on complexity theory can be found in Appendix D.*

very complex as the state space is typically exponential in the number of world features it models. For this reason, most techniques that are able to solve large MDPs either focus on approximation or exploit the state space structure.

## 2.3 Factored MDPs



One rather natural type of structure in Markov decision processes is captured in *factored* MDPs [40], where states are composed of several state features also known as factors or state variables. Most real-world stochastic planning problems exhibit this type of structure. In the path planning example from before the state of the robot could for instance consists of only one variable, its location, and the state space is defined by the values it can take on. Another example is construction planning where states consist of variables like "plumbing done" or "painting done" that take on values "yes" or "no", and the state space is formed by the Cartesian product of the variable domains. The formal definition of this factorisation, also referred to as a state factored MDP, is given in Definition 2.5:

**Definition 2.5 Factored Markov Decision Process**

A (state) factored MDP is a regular MDP $\langle S, A, P, R \rangle$ where the state space is composed of factors $\boldsymbol{X} = \{X_1, X_2, \ldots, X_{|\boldsymbol{X}|}\}$ such that $S = \mathrm{Dom}(X_1) \times \mathrm{Dom}(X_2) \times \ldots \times \mathrm{Dom}(X_{|\boldsymbol{X}|})$. Here $\mathrm{Dom}(X_i)$ denotes the domain of state feature $X_i$.

The factored MDP is almost the same as the regular MDP defined previously in Definition 2.1, however its state space is composed of state features, also referred to as (state) variables. Each state in a factored MDP corresponds exactly to a unique assignment of values to state features $\boldsymbol{X}$. At first glance this rather natural restriction on MDPs might not seem very powerful but it enables reasoning about the planning problem in a symbolic way similar to, for instance, STRIPS [87].

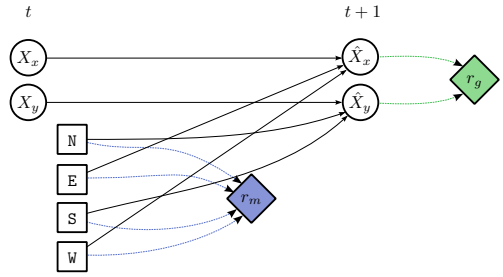
A first advantage of factored MDPs is that they often allow for compact formulations of the transition model and reward structure. By expressing them as functions of state variables or, in the case of transitions, as a conditional probability table (CPT), they concisely capture the same information that would otherwise be stored in an exhaustive matrix of size $|S|^2$. Secondly, the factored formulation can help to make (in)dependencies between state features explicit, and exploiting them can offer major reductions in the computation time required to find policies.

To better visualise and reason about these dependencies, factored MDPs are often illustrated as a *dynamic Bayesian network* [39] (DBN). The DBN provides a convenient graphical representation of interactions between state features, actions and rewards of the current state and those in the next state (see also Example 2.6 below).

**Example 2.6** Factored path planning

The states in the robot path planning example are naturally factored into a horizontal position $X_x$ and a vertical position $X_y$ such that every state $s_{(i,j)} \in S$ is an assignment

$X_x = i$ and $X_y = j$. The domains of both features correspond to the number of available horizontal respectively vertical tiles, e.g. the horizontal position has domain $\mathrm{Dom}(X_x) = [1, 4]$ (and similar for the vertical position). The corresponding DBN is shown in Figure 2.3a.




(a)

$\hat{X}_y$

| $X_x$ | N | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| | **1** | 0 | 0 | 0 | 0 |
| | **2** | .9 | .1 | 0 | 0 |
| | **3** | 0 | .9 | .1 | 0 |
| | **4** | 0 | 0 | .9 | .1 |

(b)

**Figure 2.3** *(a) Dynamic Bayesian network (DBN) of the robot example problem. The two state features are shown as circles, the four actions as squares and the reward components (move cost $r_m$ and goal reward $r_g$) as diamonds. Arrows between elements indicate a relation, either in transition (solid) or reward (dotted). For clarity, the reward interactions have been given a distinctive colour. (b) The conditional probability table (CPT) for action* `N`.

The DBN of the robot example provides insight into the interactions between state features, actions and rewards. For instance, the new horizontal position $\hat{X}_x$ of the robot depends on its previous horizontal position and whether action `E` or `W` was performed. Moreover, it shows that the new horizontal position is *independent* from the vertical position $X_y$ and actions `N` and `S`, which can be exploited when solving the planning problem. Another independence in this example is between the actions and the goal reward component $r_g$. Obtaining the goal reward does not depend on what move the robot performs, only its new horizontal and vertical position ($\hat{X}_x$ and $\hat{X}_y$ respectively) are relevant to determine whether or not it has reached its goal. On the other hand, move costs $r_m$ are independent from the actual position of the robot and depend only on the action that is taken. Note that a decomposition of rewards such as the one in this example is common in factored planning problems and the total reward can be computed simply by summing all the reward components.[12]

In factored MDPs, transitions can also be expressed over state features. An example thereof is given for the North action in Figure 2.3b. Here a conditional probability table is shown for the action that expressed the probability of setting $\hat{X}_y$ (columns) based on the value of $X_y$ in the current state (rows). Because only relevant state features are included, this CPT is typically much more compact than a complete transition table. The latter would have required $\mathrm{Dom}(X_x) \times \mathrm{Dom}(X_y) = 16$ rows and 16 columns for this example, and therefore a total of 256 entries whereas now only 16 are needed. Observe that the top row of the CPT contains only zeroes as no North move can be made when the robot is in the top row of the grid.

The rewards of the problem this DBN are represented as components $r_1$ and $r_2$, such that the total reward $R$ is given by $\sum_i r_i$, and they depend only on subsets of the state features (e.g. component $r_1$ only depends on features $X_1$ and $X_2$). Such a decomposition of rewards is often present in factored MDPs and is exploited in policy search.

In the literature of MDP, many researchers have reported the successful use of factors in MDP as a leverage in finding optimal policies. Boutilier et al. [40] introduced the `Structured Policy Iteration` algorithm that uses tree structures to compactly represent factored MDP policies and their values that is able to outperform `Modified Policy Iteration` (see Section 2.2) in many domains. Another approach that exploits factored MDPs is taken by Hoey et al. [116] where they introduce the `Stochastic Planner Using Decision Diagrams` (`SPUDD`). This planner uses algebraic decision diagrams (ADDs) to compactly represent the structure of transitions and their effects in terms of state features, leading to a very acceptable performance in practice. Other approaches include linear programming for factored MDP [103] that exploits locality in reward components to approximate the global reward (a distributed variant is given by Guestrin and Gordon [102]), hierarchical decomposition of state feature influences [126] and exploiting context-specific independence between state features, rewards and actions [37, 193].



## 2.4 Partial Observability

So far is has been assumed that, when dealing with stochastic planning problems, the only source of uncertainty lies in the execution of actions or more precisely: the transition that will occur to the next state. Hence when an action is taken in the current state it is not always known what the new state will be, but after the action is performed it can be observed with complete certainty. At any given time during the planning and execution process the current state is known for sure. This type of stochastic planning problems is only a special case of the more general *partially observable Markov decision process* or POMDP framework that includes uncertainty about the current state. This section will only introduce the basic framework and give a high-level description of finding policies for partially observable problems. For a more detailed study on POMDP and additional reading the reader is referred to Monahan [176] or Kaelbling et al. [129].

The definition of the POMDP framework is similar to that of MDPs except that it introduces an additional uncertainty regarding the current state of the system. Under this model the current state is not known with certainty. Instead, the system has to maintain a set of states in which it *could be* and the associated likelihood of being in each of those states, known as its belief state. Through feedback signals, referred to as *observations*, that are received after performing an action, the belief state can be updated to reflect the current belief about what state the system is in. These notions are formalised in this section, starting with the definition of partially observable MDPs:

**Definition 2.7 Partially Observable Markov Decision Process (POMDP) [241]**

A partially observable Markov decision process, or POMDP, is defined by the tuple $\langle S, A, P, \Omega, O, R \rangle$ such that:

[12] *Reward decomposition and independence is exploited in many approaches for factored MDPs, see Section 2.6.*

- $S$, $A$, $P$ and $R$ define the (possibly factored) states, actions, transitions probabilities and transition rewards as in a regular MDP (Definition 2.1),
- $\Omega$ is a finite set of possible observations, and
- $O$ defines the observation probability function $O : A \times \Omega \times S \in [0, 1]$ such that $O(a, o, \hat{s})$ (alternatively $O(o|a, \hat{s})$) specifies the probability of observing $o \in \Omega$ when taking action $a \in A$ such that state $\hat{s} \in S$ results. As with the transitions function it must be that $\forall a \in A, \hat{s} \in S$: $\sum_{o \in \Omega} O(a, o, \hat{s}) = 1$.

From this definition one can see that MDP can be modelled as a special case of POMDP where the observation for every state/action pair is certain. Put more formally, for all $\hat{s} \in S$ and $a \in A$ there exists exactly one $o \in \Omega$ such that $O(a, o, \hat{s}) = 1$, and hence zero for other observations by definition.

Similar to regular MDP, the solutions to POMDPs are in the form of policies that map each state to an action. However, in the context of state uncertainty it is not sufficient to base decisions upon the current state. Instead, a POMDP policy is a mapping from possible *observation histories* $\vec{o}^t = [o^0, o^1, \ldots, o^t]$ of length $t$ such that the policy $\pi(\vec{o}^t)$ returns the action that should be performed given the sequence of observations made, i.e. $\pi : \Omega^* \mapsto A$. For a deterministic, stationary policy this model is equivalent to the state-based policy as the current state of the system can be deduced from any sequence of observations.[13]

Finding optimal policies in a partially observable environment is generally much more complex as additional reasoning regarding the state of the system is needed. Although the (belief over) current state from any observation history can be deduced, it is not known what sequence of observations will be encountered in the future. Therefore a planner must reason about all such possible future sequences, along with resulting new states and the probability of their occurrence. An important notion to this end is that of a *belief state* that was mentioned briefly earlier:

**Definition 2.8 Belief State**

The belief state $b^t(s)$ for a state $s \in S$ in planning step $t$ summarises the probability of being in state $s$ at time $t$ when the initial state[14] was $s^0$:

$$b^t(s) = Pr(s^t = s|s^0, a^0, o^0, a^1, o^1, \ldots, a^{t-1}, o^{t-1}) \tag{2.7}$$

Simply put, the belief state is a summary of the (un)certainty about being in a state for every state of the system. During policy search the belief state is typically maintained for all states and only updated when new information becomes available (action and

[13] *If the policy is not deterministic, an action/observation history $\langle a_0, o_0, \ldots a_t, o_t \rangle$ is required to determine the current state with certainty. Note that such a sequence also suffices for non-stationary policies as it describes the entire execution history.*

[14] *If the initial state is uncertain the equation can be adapted to work with an initial belief state $b_0$.*

observation). For the update, Bayes' rule is applied:

$$b^{t+1}(a,o,\hat{s}) = Pr(s^{t+1}=\hat{s}|b^t,a,o) = \frac{Pr(o|a,\hat{s})Pr(\hat{s}|b^t,a)}{Pr(o|b^t,a)} = \frac{O(a,o,\hat{s})\sum_{s\in S} b^t(s)P(s,a,\hat{s})}{Pr(o|b^t,a)} \tag{2.8}$$

in which the probability $Pr(o|b^t,a)$ of observing $o$ when action $a$ is taken given our belief over the current state $b^t$ can be computed as

$$Pr(o|b^t,a) = \sum_{\hat{s}'\in S} Pr(o|b^t,a,\hat{s}')Pr(\hat{s}'|b^t,a) = \sum_{\hat{s}'\in S}\Big(O(a,o,\hat{s}')\sum_{s\in S} b^t(s)P(s,a,\hat{s}')\Big) \tag{2.9}$$

Informally, the update rule makes sure the belief about being in a certain state includes the latest observation and therefore multiplies the probability of that state by the likelihood of making an observation that leads to that state. The denominator in this rule normalises the probability to keep it within the $[0,1]$ range by dividing the observation probability by the sum of observation probabilities over all transitions that can yield the observation.

**Example 2.9** Partial observability in path-planning

The robot is still trying to find its way to the goal, however its positioning sensor has suffered some damage and it is having trouble trying to ascertain its position. It does know that it is in the vicinity of position $(3,2)$, however due to its malfunctioning sensor it is not sure whether it is exactly at $(3,2)$ or in one of its neighbouring grid positions. There is still some evidence that the sensor is working reasonably well and therefore the robot assigns a 60% chance of being at $(3,2)$ and 10% for every position that is adjacent to it. Its current belief state $b$ is thus given by $b(s_{(3,2)}) = 0.6$, $b(s_{(2,2)}) = b(s_{(4,2)}) = b(s_{(3,1)}) = b(s_{(3,3)}) = 0.1$ and $b(s) = 0$ for every other state. The belief state is visualised in Figure 2.4a.


**Figure 2.4** *Illustration of the belief state when the robot is not sure about its position on the grid: (a) possible positions of the robot with their probabilities based on its confidence in the location sensor and (b) the new belief state after performing one southward move and receiving observation $o_s$ that it succeeded.*

To get to the goal, the robot still uses the 'first South, then West' strategy as before. Previously this meant that the robot made its move and, based on the success probability

of the move, it would end up in a new state or remain in the same state, but it was certain what the state was after the action was performed. Now its sensor is broken and it cannot fully trust it to pinpoint its exact location. After the move, the location sensor notifies the robot whether his move was successful, and this notification is wrong in 20% of the times.

This can be modelled in the POMDP through a set of observations $\Omega = \{o_s, o_f\}$, denoting the move success and failure notifications respectively, such that for every action $a \in A$ the observation probability of receiving a success notification is $0.8$, while failure is thus observed with probability $0.2$. For instance, when moving East and observing success, the observation probability is given by $O(\mathtt{E}, o_s, s_{(i+1,j)}) = 0.8$ for every $i \in [1, 3]$ (and of course the corresponding $1 - p$ probability for false notifications).

Now assume the robot performs a South move and receives a success notification $o_s$ from its positioning sensor. The robot can now update its belief according to the formula presented in Equation 2.8. Its new belief $b''$ that is now at location $(3, 3)$ after performing S in $s_{(3,2)}$ is computed as:

$$\begin{aligned} b'(a, o, \hat{s}) &= \frac{O(a, o, \hat{s}) \sum_{s \in S} b^t(s) P(s, a, \hat{s})}{\sum_{s_{(3,3)} \in S} \left( O(a, o, s_{(3,3)}) \sum_{s \in S} b^t(s) P(s, a, s_{(3,3)}) \right)} \\ &= \frac{O(\mathtt{S}, o_s, \hat{s}) \times \left( b(s_{(3,2)}) P(s_{(3,2)}, \mathtt{S}, \hat{s}) + b(\hat{s}) P(\hat{s}, \mathtt{S}, \hat{s}) \right)}{\sum_{s_{(3,3)} \in S} \left( O(\mathtt{S}, o_s, s_{(3,3)}) \sum_{s \in S} b^t(s) P(s, \mathtt{S}, s_{(3,3)}) \right)} \end{aligned}$$

The new state $s_{(3,3)}$ can only result after a successful move from $s_{(3,2)}$ or a failed move from $s_{(3,3)}$ itself. For all other combinations of states and action S, either the belief or the transition probability is zero and therefore the parenthesised term in the numerator contains only two state/transition combinations. The denominator contains more terms as there are 8 state/transition combinations that could result in observing $o_s$ with non-zero probability. When ending in states $s_{(2,2)}$, $s_{(4,2)}$ or $s_{(3,1)}$, success can only be observed if a move has failed while in states $s_{(2,3)}$, $s_{(4,3)}$ and $s_{(3,4)}$ this is only possible when successfully moving southward. More involved are the states $s_{(3,2)}$ and $s_{(3,3)}$, of which the latter has been analysed already. In state $s_{(3,2)}$ it is possible to observe success when either a move has succeeded from $s_{(3,1)}$ or it failed to move. Together, this results in the following (omitting leading zeroes):

$$b'(s_{(3,3)}) = \frac{.6 \times .9 + .1 \times .1}{3 \times (.1 \times .1) + 3 \times (.1 \times .9) + (.1 \times .9 + .6 \times .1) + (.6 \times .9 + .1 \times .1)} = 0.55$$

where the observation probabilities $O(\mathtt{S}, o_s, s_{(i,j)})$ have been factored out because they are equal for all $s_{(i,j)} \in S$. The new belief state computation for all other states is performed analogously, varying only in the numerator of the fraction, and the final result is shown in Figure 2.4b. Note that the current position of the robot will be more uncertain after every move due to the uncertainty in it observing a successful move. In some partial-observable planning problems this is countered by including a special information-gathering action that can be used to refine the belief state.

The traditional exact approach to find an optimal policy for a POMDP is by solving a continuous-state *belief MDP* (see e.g. the work by Cassandra et al. [52]). This MDP has a continuous state space, i.e. there are an infinite number of states possible, such that each of the states equals exactly one belief state of the POMDP:

**Definition 2.10 (Continuous-state) Belief MDP**

The belief MDP for a POMDP $\langle S, A, P, \Omega, O, R \rangle$ is a continuous-state MDP such that the states correspond exactly to the belief states of the POMDP. It is defined as the tuple $\langle \mathcal{B}, A', P', R' \rangle$ with:

- $\mathcal{B}$ is the set of belief states defined as the space of all probability distributions over POMDP states $S$,
- $A'$ is the action set and is equal to $A$ of the original POMDP,
- $P' : \mathcal{B} \times A \times \mathcal{B} \mapsto [0, 1]$ is the belief transition probability function where $P'(b, a, b')$ denotes the probability of transitioning from belief state $b \in \mathcal{B}$ to a new belief state $b' \in \mathcal{B}$ when action $a \in A'$ is taken. The belief transition probability can be computed by summing the observation probabilities of the POMDP observations that lead to belief state $b'$, or:

$$P'(b, a, b') = \sum_{o \in \Omega | b' = b^{t+1}(a,o,b)} Pr(o|b, a) \tag{2.10}$$

  where $b' = b^{t+1}(b, a, o)$ denotes that belief state $b'$ results when taking action $a$ and receiving observation $o$ in the current belief state $b$, according to the belief update of Equation 2.8. The probability $Pr(o|b, a)$ of observing $o$ when performing $a$ in belief state $b$ is computed by Equation 2.9.

- $R' : \mathcal{B} \times a \times \mathcal{B} \mapsto \mathbb{R}$ is the belief reward function[15] and can be computed by summing over expected rewards obtained in the original POMDP when transitioning from belief state $b \in \mathcal{B}$ to $b' \in \mathcal{B}$ when taking action $a \in A'$:

$$R'(b, a, b') = \sum_{s \in S} \sum_{\hat{s} \in S} b(s) b'(\hat{s}) R(s, a, \hat{s}) \tag{2.11}$$

Although the belief MDP is a continuous-state MDP, the number of belief states is finite because there are only a finite number of actions and associated observations. As a result, there will be a finite amount (although possible very many) unique belief states in the state space of the belief MDP. Finding the optimal policy for the belief MDP, and therefore also for the original POMDP, is done using an adapted version of the `Value Iteration` algorithm that optimises over the finite amount of belief ranges that span the continuous belief state space, instead of discrete states. Several well-known such algorithms are `Value Vector Enumeration` [176, 241], `Linear Support` [60] and the `Witness Algorithm` [52]. Although the details of these algorithms are not relevant in the context of this thesis, the shared intuition they are based upon is closely related to work presented in Chapter 5 and therefore it is discussed here.

Simply put, because there is only a finite number of belief states, there must also be a finite number of unique optimal values that the expected value function

[15] *Here the most general reward notation is used to remain consistent with the previous sections, in almost all of the literature the reward is defined only over the current belief and the action taken.*

(e.g. Equation 2.1) can attain. The optimal expected value of a belief state $b$ its corresponding value vector $b \cdot V^*(b)$ are optimal in a region of the belief space. By taking the supremum of all the value vectors over the entire belief space, a Piece-Wise Linear Convex (PWLC) function is obtained that states the optimal value for every possible belief state of the POMDP.[16] By storing the actions and observations that optimise expected value in every belief state, the optimal policy can be derived (similar to Equation 2.5).

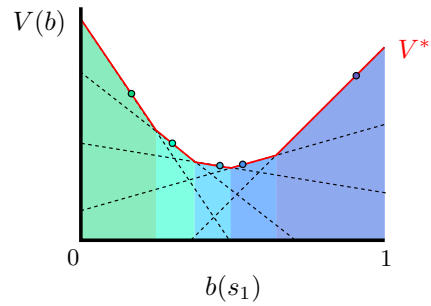



**Figure 2.5** *Example showing the expected policy value of a two-state POMDP where the belief of being in state $s^1$ is plotted on the horizontal axis. The optimal value vectors for every belief state are shown as dashed lines and the PWLC optimal value function $V^*$ is shown in red. The coloured dots on the PWLC function illustrate at what belief state the optimal value was found and the area below it is coloured according to the belief regions in which each of the optimal values dominates.*

A two-state POMDP example is shown in Figure 2.5. For a two-state POMDP, the belief state space can be visualised as a graph with the belief of being in state $s^1$ on the horizontal axis. Because the belief state is a probability distribution over the POMDP state space, the probability of being in the other state, state $s^2$, is given by $1 - b(s^1)$. In this example, there are five optimal value vectors that span the entire belief space. Each of these vectors corresponds to an optimal expected value at a belief state, visualised by a dot on the PWLC function $V^*$. As a consequence, the optimal policy will contain 5 unique action decisions (or less if multiple belief regions map to the same decision), one for each of the regions of the belief space in which that decision leads to the optimal expected value.

Including observational uncertainty in the MDP model is paired naturally with an increase in problem complexity. Finding the optimal policy for a finite horizon POMDP[17] has been shown to be PSPACE-complete whereas solving the problem without state uncertainty is in P [203]. One thing to remember here is that even though the problem is in PSPACE, i.e. the required memory is polynomial in the input size, the input size itself is typically exponential in the planning variables.

[16] *In the finite horizon setting, in the infinite horizon case this PWLC function can be approximated arbitrarily close [241].*

[17] *The infinite horizon setting has been shown to be undecidable in the work by Madani et al. [164].*

## 2.5 Planning with Multiple Agents

The framework discussed up until this point was restricted to a single agent. Although many applications exist for the single-agent model – for example in process controllers, expert systems or task planning – there are likely even more planning problems involving multiple agents. A great many scenarios can be found where coordination of actions between different actors is needed. Some examples of this are house construction planning that involves agents that have various skills (plumber, mason, etc.) [104], collaboration of emergency response robots that can only carry a person to safety when working together [138], and vehicle control in automated warehouses such that goods are transported with maximum efficiency and without vehicle collisions [159].



Multi-agent planning problems within the Markov decision process literature can be broadly divided into two categories: problems with free or instantaneous communication, and problems with no or costly communication. In the former category are problems in which agents have full knowledge about the states and decisions made by others during the planning and execution phase. Problems in the latter category are theoretically more complex as agents can no longer communicate their state or findings during the execution process, except through observations.[18] This requires agents to reason about possible states and decisions that other agents might visit or perform, and additionally they need to determine how to react to such before the plan is executed. For completeness it must be noted that there exists also a model in between the two categories that explicitly incorporates cost of communication. Solving problems that use this model requires making trade-offs between requesting information through communication or perform reasoning with the currently known information. This model is outside the scope of this thesis, nevertheless the interested reader can find more in the work by Goldman and Zilberstein [98]. The first category, fully observable planning problems with free communication, are modelled by multi-agent MDPs (MMDPs):

**Definition 2.11 Multi-agent Markov Decision Process [36]**

A (fully-observable) multi-agent Markov decision process, or MMDP, is defined as $\langle \boldsymbol{N}, S, \boldsymbol{A}, P, R \rangle$[19]:

- $\boldsymbol{N}$ is the finite set of agents, indexed as $1, 2, \ldots, n$,
- $S$ is the finite state space,
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{N}}$ is the collection of action sets where $A_i$ contains the actions only available to agent $i \in \boldsymbol{N}$,
- $P$ the transition probability function $P : S \times \boldsymbol{A} \times S \mapsto [0, 1]$,
- $R$ defines the joint reward for every transition, e.g. $R : S \times \boldsymbol{A} \times S \mapsto \mathbb{R}$.

[18] *Although in practice solving MMDP can be much harder because of the amount of information that is globally available, see Chapter 4.*

[19] *The difference in font weight is because $S$, $P$ and $R$ represent respectively the Cartesian products of states, the product of transition probabilities and the reward function while $\boldsymbol{N}$ and $\boldsymbol{A}$ are collections.*

Given such a model of a multi-agent planning problems, the optimal joint policy can be computed through a minor modification of the Bellman equation (Equation 2.4) that considers joint actions:

$$V^*(s) = \max_{\vec{a} \in \boldsymbol{A}} \sum_{\hat{s} \in S} P(s, \vec{a}, \hat{s}) \Big( R(s, \vec{a}, \hat{s}) + \gamma V^*(\hat{s}) \Big) \tag{2.12}$$

The most general extension of the MDP framework that captures both categories is known as the *decentralised Markov decision process* (Dec-MDP), sometimes also referred to as distributed MDP. Analogous to the single-agent setting there exists also a partially observable variant of this model. As Dec-MDP is a special case of the partially observable Dec-POMDP model (see Definition 2.13)[20], the latter is presented here in Definition 2.12:

**Definition 2.12 Decentralised POMDP [31]**

A decentralised, partially observable Markov decision process (Dec-POMDP) is defined by the tuple $\langle \boldsymbol{N}, S, \boldsymbol{A}, P, \boldsymbol{\Omega}, O, R \rangle$ where:

- $\boldsymbol{N}$ is the finite set of agents, indexed as $1, 2, \ldots, n$,
- $S$ is the finite state space,
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{N}}$ is action set collection where $A_i$ contains the actions only available to agent $i \in \boldsymbol{N}$,
- $P$ the transition probability function $P : S \times \boldsymbol{A} \times S \mapsto [0, 1]$,
- $\boldsymbol{\Omega} = \{\Omega_i\}_{i \in \boldsymbol{N}}$ is the collection of observation sets such that $\Omega_i$ is the set of observations that only agent $i \in \boldsymbol{N}$ can make,
- $O : \boldsymbol{A} \times \boldsymbol{\Omega} \times S \mapsto [0, 1]$ specifies the observation probabilities and
- $R$ defines the joint reward for every transition, e.g. $R : S \times \boldsymbol{A} \times S \mapsto \mathbb{R}$.

The definition of a Dec-POMDP does not differ much from that of the POMDP except that actions and observations are agent-specific in this model. All other elements are defined equivalently, although later Section 2.6 shows that the agent-specific aspect of these can be exploited to obtain more efficient solving procedures. The multi-agent case has joint actions $\vec{a} = \langle a_1, a_2, \ldots a_n \rangle \in \boldsymbol{A}$, where each $a_i$ denotes the action of agent $i$, joint observations $\vec{o} = \langle o_1, o_2, \ldots, o_n \rangle \in \boldsymbol{\Omega}$, with $o_i$ being the observation of agent $i$, and a is in the form of a joint policy $\boldsymbol{\pi} = \langle \pi_1, \pi_2, \ldots, \pi_n \rangle$ such that each policy $\pi_i$ is a mapping of local observation histories to a local action decision $\pi_i : \Omega_i^h \mapsto A_i$ (as in POMDP, see Section 2.4).

[20] *Which is in its turn a special case of partially observable stochastic games (POSG) with identical payoffs (i.e. a shared reward function for all agents), however this model is outside the scope of this thesis. More on POSGs can for instance be found in the work by Guo and Lesser [105] and Hansen et al. [108].*

As with the single-agent model, the states in a Dec-POMDP (and MMDP) can also be factored into state features or variables. Additionally, in the multi-agent setting the state space can also be factored per agent (see e.g. the work by Oliehoek et al. [193]). The state space in an *agent-factored* Dec-POMDP, occasionally written as fDec-POMDP, is divided into $n$ disjoint sets of states $S_1, S_2, \ldots, S_n$ that together they form the joint state space $\boldsymbol{S} = \times_{i \in n} S_i$. The models discussed in this thesis are all factorised, both state as well as agent-wise, unless explicitly stated that they are not.

All models presented so far are related to one another, with Dec-POMDP being the most general MDP framework. The Dec-POMDP model can capture all of the previously discussed models, it does not restrict the number of agents, type of observability, and availability of communication. Obviously, Dec-POMDP reduces to POMDP when there is only one agent present. When communication is freely available, the Dec-POMDP reduces to a (centralised) multi-agent POMDP or MPOMDP which is a special case of POMDP [129]. The reduction to other models follow from restrictions on the type of observations in the framework, which have been adopted from Goldman and Zilberstein [97].

The weakest observability restriction is that of joint full observability (JFO). Basically in a Dec-POMDP that is jointly fully observable the global state of the system can be determined with certainty by combining the observations of all agents:

**Definition 2.13 Joint Full Observability**

A Dec-POMDP is said to be jointly fully observable if there exists a mapping $J : \boldsymbol{\Omega} \mapsto S$ such that for every $\vec{o} \in \boldsymbol{\Omega}$ and $\vec{a} \in \boldsymbol{A}$ with $O(\vec{a}, \hat{s}, \vec{o}) > 0$ it holds that $J(\vec{o}) = \hat{s}$.

Thus, whenever there is a probability of ending in state $\hat{s}$ when taking joint action $\vec{a}$ and receiving joint observation $\vec{o}$, the next global state can be derived with complete certainty from the observations. This does not imply that the agents know their own state with complete certainty, only the global state is known. Therefore any Dec-POMDP that has the JFO property automatically reduces to a Dec-MDP. A stronger notion than JFO is that of full observability (FO), where each agent can individually establish the global state the system is in with complete certainty:

**Definition 2.14 Full Observability**

A Dec-POMDP is said to be fully observable (FO) if there exists a mapping $F_i : \Omega_i \mapsto S$ for each agent $i \in \boldsymbol{N}$ such that whenever $O(\vec{a}, \hat{s}, \vec{o}) > 0$ then $F_i(o_i) = \hat{s}$.

Agents in a fully observable Dec-POMDP can observe the global state with complete certainty from their own observation and therefore this model reduces to a multi-agent MDP. In between these two observation restrictions is that of local full observability (LFO). A Dec-POMDP is said to be LFO if the global state cannot be derived with certainty but the agents know their own state with perfect confidence:

**Definition 2.15 Local Full Observability**

An (agent-factored) Dec-POMDP is known as locally fully observable (LFO) if every state $s = \{s_1, s_2, \ldots, s_n\} \in \boldsymbol{S}$ is composed from individual agent states $s_i$ and there exists a mapping $L_i : \Omega_i \mapsto S_i$ for each agent $i \in \boldsymbol{N}$ such that whenever $O(\vec{a}, \hat{s}, \vec{o}) > 0$ then $\forall i \in \boldsymbol{N} : L_i(o_i) = \hat{s}_i$.

In this observation restriction it is assumed that the problem is agent-factored, i.e. the state space is defined as $\boldsymbol{S} = \times_{i \in n} S_i$, which is seldom not the case in multi-agent planning.

**Example 2.16** Multi-agent MDP

The robot has successfully repaired its sensor and is fully aware of its position again. On top of that, its sensors have detected the presence of another robot that is trying to find a way to its designated exit. The robots are not really interested in each other although they do want to avoid damage due to collisions and therefore have to coordinate their movements. When the robots do collide, both will receive a penalty of $-20$. Figure 2.6 illustrates the two-robot problem on the same grid as before, although now the exits have been coloured according to the robot that can use it.

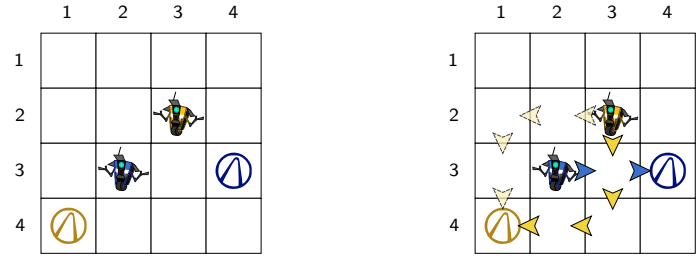



**Figure 2.6** *Two-agent example of the robot path planning problem: (a) there are two robots, each trying to reach their designated goal indicated by the colour of the symbol. When the robots collide, i.e. they are at the same location concurrently, they will be damaged and therefore this results in a negative reward. (b) Example of two shortest-path policies that, while individually optimal, can cause collisions when combined into a joint policy. When the yellow robot switches to the policy illustrated by the lighter path, no collisions can ever occur while its move costs do not increase.*

To avoid collisions and still optimise their path planning, the robots agree to share their location with one another and develop a jointly optimal policy. Communication between the robots is relatively cheap because they are both within the same small area and therefore this problem can be modelled as a (factored) jointly fully-observable Dec-POMDP with free communication, which is equivalent to a multi-agent MDP (MMDP).

The MMDP for this multi-agent problem resembles the MDP of Example 2.3 from each robot's point of view, but additionally it contains dependencies between the robots. The joint state space $\boldsymbol{S}$ is defined as the combination of two state spaces $S_1$ and $S_2$, for

the yellow and blue robot respectively, where each of the individual state spaces is defined by the combination of two features $X_x$ and $X_y$ as before. One single joint state is thus a combination of two grid positions at which the robots are located, for example the state $s = \{s_{(1,i,j)}, s_{(2,u,v)}\}$ says that the yellow robot (robot $1$) is at position $(i, j)$ and the blue robot (robot $2$) is at $(u, v)$.

The action space of the MMDP is defined as the collection of both individual action spaces, i.e. $\boldsymbol{A} = \{A_1, A_2\}$. A single element in this set is a joint action $\vec{a} = \langle a_1, a_2 \rangle$, with $a_1 \in A_i$ and $a_2 \in A_2$. For instance the joint action $\langle \mathtt{S}_1, \mathtt{E}_2 \rangle$ specifies that the yellow robot will perform a South move whereas the blue robot will move towards the East.

Transition probabilities in this problem can also be defined in a factored way. Although collisions may occur and therefore the reward is determined based on the position of both robots, the robot's new position only depends on its own movements. Therefore the transition probability for agent $1$ – the yellow robot – are in the form $P_1(s_{(1,i,j)}, a_1, s_{(1,i',j')})$ (and similarly for the blue robot) such that the joint transition probability is defined as $P(s, \vec{a}, \hat{s}) = \prod_{i \in \boldsymbol{N}} P_i(s_{(i,x,y)}, a_i, s_{(i,x',y')})$. This particular MMDP is actually transition-independent (formalised later in Definition 2.17) because the robots can not affect each other's state transition directly.

There is a dependency in the joint reward however (otherwise the problem could also have been solved as two individual single-agent MDPs) because of the collision penalty. In addition to the move costs and goal rewards, there is a joint penalty of $-40$ when the robots are at the same location at the same time. Continuing the component-wise specification from Example 2.6, there is an additional reward component $r_c$ for collisions that is defined as $r_c(s, \vec{a}, \hat{s}) = -40$ for every state $s \in \boldsymbol{S}$, joint action $\vec{a} \in \boldsymbol{A}$ and $\hat{s} = \{s_{(1,i,j)}, s_{(2,u,v)}\}$ in which $i = u$ and $j = v$.

From Figure 2.6b it is easy to see the need for coordination between the two robots: if the yellow robot used the SW-policy, as it did before, and the blue robot uses a similar shortest-path policy that continues moving East until the goal has been reached, the robots will crash into each other if both their moves succeed. Even though both policies were optimal from each agent's individual point of view, combining these into a joint policy results in a solution that is far from optimal. If either of the robots is willing to take a different route, the penalty of $-40$ can be prevented, resulting in a better outcome for both robots. Hence by coordinating their decisions while developing a joint policy they can both benefit. As the joint state is global knowledge in an MMDP, the yellow robot can actually detect that blue is in the position South(-West) of its current location and can therefore choose to perform a West move first whenever a global state of the form $\{s_{(1,i,j)}, s_{(2,u,v)}\}$ is encountered that has $i = u$ or $i = u + 1$ and $j = v - 1$. Alternatively, in this example, the yellow robot can switch from the SW-policy to a first West, then South policy without increasing its costs while avoiding collisions altogether, lifting the need for (further) coordination.

**Decentralised coordination** In the setting where communication is not freely available, the best these robots can do is coordinate their actions based on the *observations* they make regarding the location of the other robot, *after* execution of the joint action. While in the MMDP model it is possibly to observe the global state space, in a Dec-MDP the robots can only base decisions on their local state and the observed location of the other agent. A key difference is hence that coordination in a decentralised problem can only be performed based on the expectation about what the other robot is going to do, given the current state and last observation, whereas in MMDP it was possible to align actions based on the globally available joint state. Still, although action decisions during executions are based

on this local view of the world, it is possible to coordinate action decisions through a joint policy that is developed centrally before the execution is started. This will be demonstrated below for a decentralised version of the two-agent problem.

Consider again the two-agent robot path-planning example, but now the robots can only communicate their position after each (failed) move they have performed. Again the robots want to reach their goal, preferably without collisions. Each of the robots could maintain the last-known location of the other robot as part of its current state, however, as the other robot can be in any of the $4 \times 4 = 16$ locations, this would require $16 \times 16 = 144$ states to capture every combination of locations and leads to a joint state space of $144^2 = 20\,736$ states for this simple 4 by 4 grid example. Instead, each robot observes the position of the other robot relative to its current position. Each robot $i$ can make 9 observations $o^i_C, o^i_{SW}, o^i_N, \ldots$ ($C$ denoting the center) regarding the position of the other robot relative to its own in the $3 \times 3$ grid centered around the robot and $o^i_\neg$ if the other robot is not in the vicinity. This is shown in Figure 2.7a.


**Figure 2.7** *Two-robot example in the decentralised setting: (a) the observations each robot can make regarding the position of the other robot and (b) conditioning its action decision based on the previous observation prevents the yellow robot from performing a South move when blue is in either of the shown relative positions.*

The observation probabilities are defined such that every observation can be observed with equal probability ($\frac{1}{10}$) whenever the previous observation was that the robot is not in the vicinity, i.e. the previous observation of robot $i$ was $o^i_\neg$. When another observation was previously made, the probability depends on the observation and action. For example, when previously $o^i_{SW}$ was observed it can never observe $o^i_{NE}$ in the next state if it performs a North action.

An example of coordination through a joint policy is shown in Figure 2.7b. The yellow follows a modified version of the SW-policy such that it will prefer West over South when it has observed that the blue robot is in either one the two shown positions. During the search for a joint optimal policy it has been established that it will be jointly better if blue always moves eastward and yellow avoids possible collisions. After this joint policy has been developed, both robots can independently execute their own policy and be certain that no collision occurs because the policy of the yellow robot is conditioned on what it observes about blue and thus both robots have coordinated their path planning (implicitly).

The relation between the classes of MDPs discussed so far is summarised in Figure 2.8. Each class is illustrated as a filled oval and contain sub-classes when a reduction is possible. These reductions are shown by arrows and are labelled with their required property. The properties in this figure are: identical rewards, free communication (free comm.), joint full observability (JFO), full observability (FO) and single agent

(1Agent). Of course, any problem that is a subclass can be expressed and solved using the model and solving techniques of its containing super class.

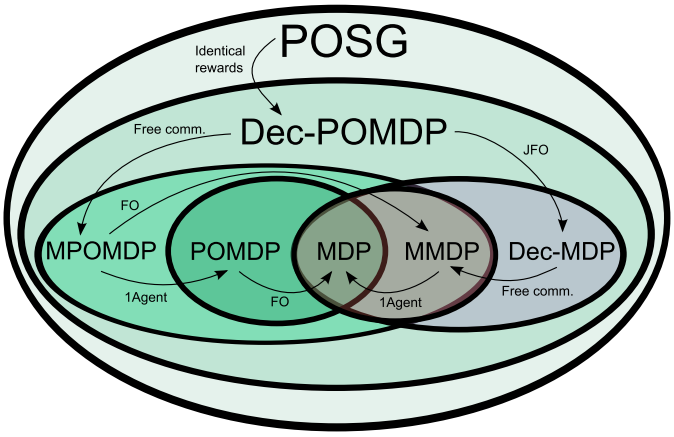



**Figure 2.8** *Overview of MDP classes discussed so far, all contained within the class of partially observable stochastic games (POSG). Each of the classes is shown as an ellipse. The arrows mark a relationship between classes and are labelled with the specific property necessary for the reduction to the sub-class.*

The decentralised POMDP model is more expressive than the previously discussed POMDP as it captures multi-agent problems and, consequentially, policies for the decentralised model are harder to find. Indeed, Bernstein and Givan [31] showed that solving Dec-POMDP is NEXP-complete (see Appendix D) and hence substantially more complex than its single-agent counterpart. An additional and remarkable result of Bernstein and Givan [31] is that restricting the individual observability does not affect the problem complexity. Finding the optimal policy for any Dec-MDP with at least 2 agents is NEXP-complete. When full observability is satisfied the problem can be reduced to an MMDP and, because every MMDP can be transformed into a single-agent MDP [36], P-complete to solve [203].

Even approximation turns out to be hard for general Dec-POMDP: Rabinovich et al. showed that even deciding whether a policy is arbitrarily close to the optimal one is a NEXP-hard problem [210]. This result implies that no tractable approximation with a bounded quality guaranty can exist because P is a strict subset of EXP which contains NEXP (again, see Appendix D).

These complexity results discourage practical application of the general Dec-POMDP model. Indeed many researchers have instead focused on special cases that exhibit particular structural properties. In the next section several of these special cases are presented with a brief explanation on how their structure can be exploited in policy search. For complete detail on the presented approaches, the reader is advised to read the works cited in the text. Here only the key ideas are presented that inspired this and similar research.

## 2.6 Gaining Traction on Dec-POMDPs

The computational complexity result presented in the previous section, solving general Dec-POMDPs is a NEXP-complete problem, gives little hope of ever finding efficient solution methods for this type of problem. To this end, many researchers have devoted their efforts into finding special cases that are (more) tractable to solve. Indeed, some of the approaches that are presented in this section have been able to solve much larger problems in terms of input size and/or number of agents.

Put coarsely, most approaches focus on the independence between agents in one way or another. Many authors have considered models where agents have limited or no influence on other agents such that agents do not have to reason about the impact of other agents' actions on their state or observation, e.g. Becker et al. [25] and Nair et al. [183]. Other methods specifically target the locality of interactions between agents instead of considering all agents at once, for instance Oliehoek et al. [193] and Varakantham et al. [251]. This section will outline several known MDP subclasses that use such restrictions on the parts of their model, allowing for more efficient policy searches. First however three general independence properties are presented that many of these subclasses make use of: transition, observation and reward independence. These properties all require that the Dec-POMDP is factored, hence its states, actions and observations are separated in disjoint, agent-specific sets. The transition independence property (TI) holds for any Dec-POMDP in which agents cannot affect the states of other agents through their actions. Put differently, this property states that the state transition of an agent only depends on its own current state and action:

**Definition 2.17 Transition Independence**

In a transition independent Dec-POMDP the global transition probability is the product of (independent) local transition probabilities: $P(s, \vec{a}, s) = \prod_{i \in \boldsymbol{N}} P_i(s_i, a_i, \hat{s}_i)$.

The main advantage of transition independent problems is that agents do not have to consider the states and actions of other agents when reasoning about the optimal transition from a certain state. This can potentially significantly reduce the size of the search space that needs to be considered in optimal policy search as agents only have to reason about the impact of their own actions, irrespective of what others may perform. Note that this property does not fully decouple the agents as they are typically still dependent through their observations and/or rewards. Similar to transition independence, agents can also be independent in their observations. This is the case when the observations an agent can make depends only on its own action.

**Definition 2.18 Observation Independence**

In an observation independent Dec-POMDP the global observation probability is the product of the (independent) local observations: $O(\vec{a}, s, \vec{o}) = \prod_{i \in \boldsymbol{N}} O_i(a_i, \hat{s}_i, o_i)$.

Again such an independence allows agents to focus only on the impact of their own decisions, albeit in terms of observations an agent can make. Lastly one can also identify an independence between agent rewards as in Definition 2.19:

**Definition 2.19 Reward Independence**

In a reward independent Dec-POMDP the global reward is the sum of (independent) local agent rewards: $R(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} R_i(s_i, a_i, \hat{s}_i)$.



In problems with reward independence, the reward of each agent only depends on its own state and decisions. Typically, reward dependencies occur when agents have sub-additive or super-additive value for performing certain actions or being in a specific combination of states concurrently. This is for instance the case in Chapter 4 where agents have a sub-additive valuation for performing certain combinations of actions concurrently. It might seem that when reward independence is present, the planning problems can be decoupled. Nonetheless, the state transition or observations may still depend on other agents and therefore, indirectly, the expected reward that an agent can obtain. Consequently, even when RI is present there has to be coordination between agents to optimise reward.

**Example 2.20** Independence in multi-agent problems

The (in)dependencies in the two-agent robot path-planning example can be easily visualised using again the Dynamic Bayesian Network (DBN, Section 2.3).

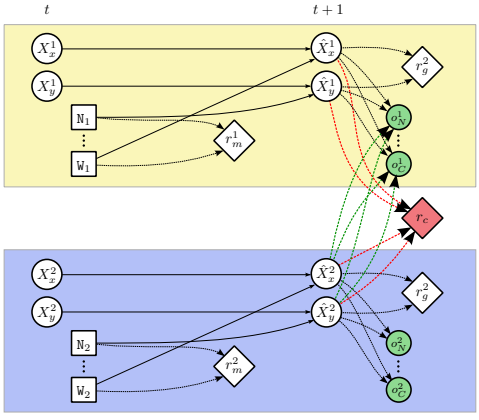



**Figure 2.9** *The DBN for the two-robot path-planning problem of Example 2.9, now with (some of) the observations that both agents can make. The state features, actions, observations and rewards are grouped per agent, except for the shared collision reward component $r_c$. Violation of independencies are shown by the green and red arcs for observation and reward independencies respectively. Only the observation dependencies of the blue robot on the yellow robot are displayed to preserve clarity.*

> The DBN now contains two agents and the elements corresponding to each of them have been grouped within the coloured areas, with the yellow robot at the top. First notice the addition of observations to the DBN (of which only two are shown) that depend on the new position of the robot as well as that of the other one. Because the probability of making each of these observations depends on the position of the other robot, this is a violation of observational independence. In this example there is also a violation of reward independence: the reward component for collisions, $r_c$, depends also on the new location of both robots. As the robots can never directly influence the position of another agent by any of its actions, this problem is transition independent. An example of a change to this problem that makes it transition dependent can be the addition of a physical collision effect such that when two robots end up in the same position after a move, they will be displaced slightly. In such an example, the choice of action in one robot can affect the state of another, thus violating transition independence.

An important complexity result regarding the independence properties is that when both transitions and observations are independent, the problem of finding an optimal policy becomes NP-complete [97]. If all three properties are present in a multi-agent planning problem, it can be decomposed into $|\boldsymbol{N}|$ independent single-agent problems that can be solved separately.

The remainder of this section will outline a number of subclasses of Dec-POMDP that use one or more of the independence properties just presented, or weaker variants thereof, to solve (restricted) planning problems more efficiently. This overview is by no means a complete overview of all existing subclasses and associated solving techniques, nor is it presented in full detail. Here the most significant concepts and intuitions of those approaches are included, both optimal and approximate, that are closely related to work presented in this thesis, in particular to that of Chapter 4. More details and references can be found in the papers cited.

### 2.6.1 Jointly Fully Observable Models

Many of the approaches for more tractable subclasses of decentralised POMDPs considered the jointly fully observable setting (JFO, Definition 2.13). As explained earlier in Section 2.4, any POMDP that is JFO can be reduced to an MDP, and a similar statement holds true for the decentralised case. Although the theoretical worst-case complexity of Dec-MDP is also NEXP-complete, it is often easier to solve than its partially observable counterpart.

One of the first models that was considered a more tractable subclass was the transition and observation independent decentralised MDP, or **TOI-Dec-MDP**. Becker et al. proposed the `Coverage Set Algorithm` (`CSA`) that can be used both as an optimal as well as an anytime algorithm for this class of problems when joint rewards can be structured such that interactions can be expressed through events. In the presence of events, the Dec-MDP can be separated into individual MDPs for every agent, each MDP with an augmented reward function that captures reward dependence with other agents. This allows each agent to find a compact set of best responses to all possible joint policies of all other agents and from these the optimal joint policy over all agents can be determined. As a result, the efficiency of this algorithm depends predominantly

on the 'locality' of the agent dependent rewards. The anytime aspect of this algorithm is due to the fact that it can return the current best joint policy at any time during the search. Allen et al. [8] developed a bilinear programming approach that speeds-up the policy search, and in particularly improves the anytime quality of the algorithm.

Another approach that targets TOI-Dec-MDPs is taken by Wu and Durfee [265] that introduces MILP-HCS, a combination of mixed integer linear programming and hill climbing search, to solve problems from this class. Their approximation algorithm is demonstrated empirically to produce high quality solutions in limited time for the collaborative task execution domain, and performs better than anytime CSA or JESP (the latter is explained later) in terms of quality versus runtime.



Dibangoye et al. [77] also considered TOI-Dec-MDPs. Their algorithm, termed Markov Policy Search (MPS), casts a Dec-MDP into a continuous MDP such that the states thereof correspond to probability distributions (called occupancy distributions) over the original Dec-MDP. This state space is then explored with a learning A* algorithm and subsequently policies are derived from the resulting state occupancy. With this approach, the authors have successfully scaled up solving to much larger problems of several previously studied domains.

An extension of the model of Becker et al. [26] was presented by the same authors. Becker et al. [25] considers the event-driven interaction MDP, or **EDI-Dec-MDP**, subclass and shows that the CSA algorithm is exponential in the number of event interactions present in the problem. This particular subclass does not assume full transition independence, instead it enforces that dependent transitions can be captured as mutually exclusive events. Although this is still a restriction on the type of transition interactions, it is less prohibitive than transition independence.

Beynier and Mouaddib [32] also use an approach that has a weakened form of transition independence but full observation independence. They introduced the opportunity cost Dec-MDP, or **OC-Dec-MDP**, that allows for temporal and precedence relations in transitions. The opportunity costs in this model capture the impact on reward due to agent dependencies and, using a modified Bellman equation (Equation 2.4) that includes them, they are exploited to find policies in a decentralised manner while still coordinating between agents. The authors provide an approximation algorithm that produces good policies on the problems they consider while taking only polynomial time in the state space size. Beynier and Mouaddib [33] proposes an improvement where the algorithm uses expected opportunity costs and an iterative improvement procedure to produce better joint policies, but both are without formal guarantees.

Closely related to this has been the work by Marecki and Tambe [165] on the continuous resource decentralised MDP, or **CR-Dec-MDP**, that was initially compared to the OC-Dec-MDP algorithm of Beynier and Mouaddib [32]. Under the CR-Dec-MDP formalism, agents are assigned sets of partially ordered methods that can be executed only once and at most one at a time. The actual length of execution is however unknown at planning time and is given by a probability function. This model adheres to the observational independence but allows for a restricted form of transition dependence in the form of temporal and precedence constraints between methods, possibly belonging to different agents. For this type of MDP problems, the authors developed the Value Function Propagation (VFP) algorithm. This algorithm is similar in structure to the

one for OC-Dec-MDP as described by Beynier and Mouaddib [32], but it propagates value functions instead of constant values. This seemingly minor adaption decreases the overestimation of opportunity costs in many cases. Moreover, passing entire value functions allows for better pruning during the policy search. Indeed VFP produces faster and better solutions for the (discrete) problems considered in both articles [165].

Later, Marecki and Tambe [167] targeted the continuous resource setting with their Multi-agent Dynamic Probability Function Propagation (M-DPFP) algorithm for the CR-Dec-MDP class. This algorithm is an extension of their earlier work on the single-agent setting [166] and alleviates the partial ordering restriction required in VFP. Although the continuous resource model is outside the scope of this thesis, this algorithm is worth mentioning because it has been benchmarked against the state-of-the-art SPIDER method (discussed later) and outperformed it in most of the domains used by Marecki and Tambe [167].

Based on both the event-driven interaction as well as the continuous resource model is the approach taken by Mostafa and Lesser [179]. They propose the event-driven interaction with complex reward model, or **EDI-CR-MDP**, a model that combines the events of the former with structured rewards of the latter in such a way that the two previous models are a subclass of EDI-CR-Dec-MDPs. Problems in this subclass can have both transition and reward independence, and therefore are more general, but can still be solved rather efficiently in practice. They extend the bilinear programming approach presented by Allen et al. [8] to deal also with transitional dependencies and show empirically that their method performs well on problems of this subclass. Nevertheless no formal guarantees can be given by their algorithm.

The last important subclass in the jointly fully observable setting that is discussed here is the decentralised MDP with sparse interactions, or **Dec-SIMDP**, proposed by Melo and Veloso [170] and extended from previous work by Spaan and Melo [242]. The intuition behind this model is that in many problems the interactions are limited to a few occasions and only decisions in such moments have to be coordinated. At all other times, agents may plan their actions completely independent from each other. Thus this model includes transition, observation and reward independence, but these independencies are context-specific (referred to as agent independence by Melo and Veloso [170]). This is also the basic intuition behind both the MPSI and LAPSI (respectively Myopic and Look-ahead Planning for Sparse Interactions) algorithms proposed in this work. Both methods use a mix of single-agent POMDP and multi-agent MDP solving for times when planning decisions respectively can and cannot be made without coordination. The MPSI algorithm considers all agents to be self-interested and therefore plans defensively whereas LAPSI assumes collaborative agents.

### 2.6.2 Partially Observable Models

All the subclasses so far reviewed all shared the jointly fully observability assumption, i.e. the global state could be determined with certainty. There have also been several approaches considering subclasses of the more general partially observable setting. One of the first of such methods was proposed by Nair et al. [183]. Although their approach is based on the multi-agent team decision processes model, this model can be translated

directly into observation-independent Dec-POMDPs, or **OI-Dec-POMDP**. For this class they developed a local search procedure known as `Joint Equilibrium Policy Search` (`JESP`). The essence of this algorithm is very simple: the algorithm fixes the policy for $n-1$ agents and find the policy of the $n$-th agent that optimises the expected value of the joint policy. This process is continued iteratively, alternating the fixing over the set of agents, until no agent can improve their policy without decreasing the expected value of the joint policy. This solution concept is known as a Nash equilibrium. Again as with most jointly fully observable approaches, `JESP` does not provide any quality guarantees but performs fairly well in practice.

This work was extended by Nair et al. [184] to network distributed POMDPs, or **ND-POMDP**. Network-distributed POMDPs also satisfy observation independence but require transition independence in addition.[21] When both these properties are satisfied, the reward function can be decomposed into local and shared components and their modified algorithm, `Locally Interacting Distributed JESP` (`LID-JESP`), uses distributed constraint optimisation to exploit this reward structure. As a result, the `LID-JESP` algorithm outperforms the previous `JESP` on instances of this subclass. Furthermore, this article presents the exact `Global Optimal Algorithm` (`GOA`) for this type of problems when the reward dependencies are binary. `GOA` constructs a tree from the reward dependencies between agents and performs a depth first search over this tree such that for each policy of a parent node, the child nodes recursively return an optimal best response. If each node recursively enumerates all possible policies exhaustively, the algorithm is ensured to return the optimal policy. The main benefit is that this solves sub-problems decoupled into sets of agents that are not (indirectly) reward dependent.

Varakantham et al. [250] also targeted the ND-POMDP framework and introduced the `Search for Policies in Distributed Environments`, or `SPIDER`, algorithm. This approach resembles `GOA`, in that they also build an agent tree based on reward dependencies to perform policy search on, but they apply a branch and bound search. This search uses the current best expected value as a lower bound and finds an upper bound for each branch in policy search through MDP approximation. By assuming full observability and a centralised problem, i.e. a standard MDP, the best possible expected value can be computed within reasonable time as MDP is much easier to solve. This approach immediately provides two benefits: firstly, the MDP approximation can be used to prune during the policy search when the upper bound is lower than the current best expected value. Secondly, because it has both a lower and an upper bound available, the algorithm is able to bound the quality of the produced policy in terms of error between the bounds. As a direct result, the algorithm can provide bounded approximations with only the minor adaptation of adding a tolerance parameter in its termination criterion.

Dibangoye et al. [78] considered also ND-POMDP problems and they applied their previous work of solving general Dec-POMDPs through continuous-state MDP to this class. Their `Feature Based Heuristic Search Value Iteration` (`FB-HSVI`) al-

[21] *ND-POMDP is hence equivalent to TOI-Dec-POMDP, but in the literature the former term is used predominantly.*

gorithm transforms Dec-POMDP instances into equivalent continuous-state MDPs such that the states of the latter correspond to belief states and histories of the former. [22] The algorithm exploits reward functions that can be represented compactly, which is the case in ND-POMDP as Dibangoyea et al. [79] show. In addition to a good performance in terms of runtime when reward structures are compact, FB-HSVI is able to bound the quality of the resulting joint policy and can be shown to converge to the optimal solution eventually.

Contrary to the previous approaches where significant dependency restrictions are required, there have also been approaches that target the sparsity of interactions. One such approach is taken by Varakantham et al. [251] where the authors propose the distributed POMDP with coordination locales, or **DPCL**. States in DPCL consist only of execution statuses (e.g. "done" and "not done") and it requires observation independence, however no other assumptions are necessary. The key idea behind problems in DPCL is that there are typically only a limited number of interactions that need to be coordinated, known as coordination locales, whereas the rest of the planning can be performed independently (similar to the work by Melo and Veloso [170] in the JFO setting). Their algorithm, known as the Team's Reshaping of Models for Rapid Execution (TREMOR), resembles the branch and bound method of the previous SPIDER algorithm. The branch and bound method is used to perform assignment of tasks over the agents after which a policy is found for the execution of these tasks. It approximates interactions between agents by reward shaping: for each agent a POMDP is solved where the reward function incorporates the impact of coordination locales.

Another approach that focuses mostly on the interactions itself as opposed to restricting dependencies between agents is that of influence abstraction proposed by Witwicki and Durfee [264]. Here the authors introduce the transition-decoupled POMDP, or **TD-POMDP**. Whereas most methods so far assumed one or more types of independence, TD-POMDP does not require any type of strict independence. Instead it is assumed that the states are factored and can furthermore be decomposed in uncontrollable, local(ly-controlled) and non-local(ly-controlled) features. Uncontrollable features are global features that are not directly influenced by any agent but can be observed by the agents. The other two types, local and non-local features, are variables that are controlled by the agent itself or by others respectively. In addition to this state decomposition, it is assumed that the global reward is monotonic in the individual reward functions so that any decrease in reward for any of the agents can not lead to an increase of global reward and vice versa. Using this model, it is possible to extract exactly the influences between agents and coordinate just these interactions. This is implemented in the Optimal Influence Space Search (OIS) algorithm that performs an exact joint policy search using influence summaries. As does the previous GOA, OIS generates an agent search tree and traverses this recursively to find the optimal joint policy, generating and propagating influence summaries downwards, returning best-response policies to each of these influences upwards.

The final subclass discussed in this section requires none of the independence criteria, only that the Dec-POMDP is factored as previously discussed in Section 2.5. The

[22] *An approach that had already been successfully applied to POMDP and MOMDP.*

**factored Dec-POMDP**, or **fDec-POMDP**, was considered as the framework of choice in Section 2.5, although factoring was implicitly assumed in all previous models as well. Oliehoek et al. [193] show how any factored Dec-POMDP can be solved as a series of collaborative graphical Bayesian games (CGBGs) and introduce the `Generalised Multi-agent A*` (`GMAA*`) that performs an exact policy search using this technique. By decomposing the independence over time into CGBGs, the algorithm tries to exploit the locality in agent rewards to perform a more efficient search. Their algorithm has been demonstrated to solve three-agent Dec-POMDP instances optimally, one of the few reported successes for general factored Dec-POMDPs.

This section has discussed many of the existing approaches that have been taken in order to solve at least problems that are modelled by a subclass of the decentralised partially observable Markov decision process. Table 2.1 summarises all the subclasses and corresponding solving techniques. This table gives an insight into the type of subclasses that have been proposed and what kind of restrictions they require on the input. Keep in mind that, as stated before, the table presented here includes a substantial part but not all of the models and approaches that have been dealt with in the literature. Only those that are (mostly) relevant to the work in Chapter 4 of this thesis have been included.

| Model | States | Transitions | Observations | Rewards | Solutions |
|---|---|---|---|---|---|
| TOI-Dec-MDP | Factored | Independent | JFO, indep. | Arbitrary[23] | `CSA` [26], `MPS` [77], `MILP-HCS` [265] |
| EDI-Dec-MDP | Factored, global events | Mutually exclusive events | JFO, indep. | Arbitrary | `CSA` [25] |
| OC-Dec-MDP | Factored | Temporal and precedence dependencies | JFO, indep. | Arbitrary | [32], [33], `VFP` [165] |
| CR-Dec-MDP | Factored | Temporal and precedence dependencies | JFO, indep. | Completion rewards | `VFP` [165], `M-DPFP` [166] |
| EDI-CR-MDP | Factored | Mutually exclusive events | JFO, indep. | Completion rewards | [179] |
| Dec-SIMDP | Factored | Sparse interactions | JFO, indep. | Sparse interactions | `MPSI & LAPSI` [170], `IDMG` [242] |
| OI-Dec-POMDP | Factored | Arbitrary | Independent | Arbitrary | `JESP` [183] |
| ND-POMDP | Factored | Independent | Independent | Arbitrary | `FB-HSVI` [79], `LID-JESP`, `GOA` [184], `SPIDER` [250] |
| DPCL | Factored, completion statuses only | Sparse interactions | Independent | Completion rewards | `TREMOR` [251] |
| TD-POMDP | Factored, decomposable | Sparse interactions | Sparse interactions | Monotonic global reward | `OIS` [264] |
| fDec-POMDP | Factored | Arbitrary | Arbitrary | Arbitrary | `GMAA*` [193] |

**Table 2.1** *Overview of (factored) Dec-POMDP subclasses. The columns of the table, from left to right, show: the model name, required restrictions on states, transitions, observations and rewards respectively, and the solution that was proposed in the literature.*

[23] *Although some structure is assumed in them but this follows from the combination of transition and observation independence, and does not pose any restriction on the model.*

## 2.7 Planning with Multiple Objectives

The final area of stochastic planning touched upon in this thesis is that of planning with multiple objectives. There exists many practical planning scenarios where it is more natural to try and optimise several, possibly conflicting, criteria at once. Some examples of this are allocation of several, distinct resources [132], energy and comfort management in buildings [139] and routing transport of hazardous materials [160]. As before, this section only outlines the concepts of multi-objective planning that are most relevant for the work presented in the thesis, in particular to Chapter 5. Most of these concepts have been adopted from the survey by Roijers et al. [215]. Furthermore, as approaches for the partial observable setting are not significantly different and are not within the scope of this thesis, a step back is taken in terms of generality. All significant multi-objective notions are presented only for the fully observable Markov decision process:

**Definition 2.21 Multi-objective Markov Decision Process (MOMDP)**

A multi-objective Markov decision process (MOMDP) is defined by a tuple $\langle S, A, P, \boldsymbol{R} \rangle$ such that:

- $S$, $A$, $P$ are the state space, action set and transition probability function as in Definition 2.1, and
- $\boldsymbol{R}$ is a collection of reward functions $\{R_1, R_2, \ldots, R_m\}$, where $m$ is the number of objectives, such that each $R_k(s, a, \hat{s})$ captures the reward for objective $k \in [1, m]$ when the current state is $s$, action $a$ is taken and new state $\hat{s}$ results.

Notice that this definition of multi-objective MDP is presented as a single-agent model. In Section 2.5 it was mentioned that any multi-agent MDP can be transformed into a single-agent MDP [36]. The same holds for any multi-objective MMDP and therefore the focus here is only on the latter model.

For an MOMDP, none of the methods presented in this chapter can be directly used because the reward can no longer be expressed as a single value. Instead, the expected value of any (stationary) policy $\pi$ given initial state $s^0$ is expressed as a vector over all objectives:

$$\boldsymbol{V}^{\pi}(s^0) = \mathbb{E}\Big[\sum_{t=0}^{h-1} \boldsymbol{R}(s^t, \pi(s^t), s^{t+1})\Big] \tag{2.13}$$

and the policy valuation can also be written as $\big\langle V_1^{\pi}(s^0), V_2^{\pi}(s^0), \ldots, V_m^{\pi}(s^0) \big\rangle$ where each $V_k^{\pi}$ is the valuation of policy $\pi$ for objective $k \in [1, m]$ as in Equation 2.1.

Because the value of a policy is now given by a vector, it is no longer possible to distinguish an optimal policy as before. With scalar rewards the value function can be used to induce a complete ordering of policies and select an optimal policy from it. However, if the valuation function has multiple dimensions, at most a partial ordering of policies can be defined. For example there might exist two policies such that one yields a higher value in the first objective but the other scores better in the second,

i.e. $\exists \pi, \pi' \in \Pi$: $V_1^{\pi}(s) > V_1^{\pi'}(s)$ and $V_2^{\pi}(s) < V_2^{\pi'}(s)$ for some $s \in S$, and therefore there is not one 'best' policy. At best, a point-wise comparison can be made between two different policies whereas previously all policies could be ranked based on their expected scalar reward. This lack of complete ordering is the reason why the methods discussed up to this section are not (directly) applicable to MOMDP solving.

Although MOMDPs have multi-dimensional valuations and it is therefore not possible to define a unique optimal policy, in the end any planning algorithm still needs to produce one single policy to execute. To this end, multi-objective approaches typically transform the vector value function into a scalar value using a so-called *scalarisation function*:



**Definition 2.22 Scalarisation Function**

A scalarisation function $f$ is a function that transforms a multi-dimensional value function $\boldsymbol{V}^{\pi}$ into a scalar reward for any policy $\pi$ and initial state $s^0$:

$$V^{\pi,\boldsymbol{w}}(s^0) = f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) \tag{2.14}$$

Here, $\boldsymbol{w}$ is a set of (normalised) objective weights $\{w_1, w_2, \ldots, w_m\} \in [0,1]^m$ that specify the relative importance of each objective.

Observe that if the objective weights are known during the planning process and the scalarisation function is not too hard to compute, the MOMDP can be transformed into a single-objective problem and solved through any of the existing MDP techniques (see Chapter 3). The need for specific multi-objective solving methods arises only when one of these two assumptions does not apply. Moreover, the type of (intermediate) result differs for both of the two conditions. When the scalarisation function is efficiently computable (discussed in more detail later) but the weights are unknown during planning, the MOMDP solving algorithm should return a set of policies for different weights so that the optimal policy can be retrieved from this set as soon as the weights become known. This occurs for instance in scenarios where objectives are weighted by their relative importance or as a price per unit. Examples of such problems are allocating miners to mines with unknown prices of ore types [217], resource gathering where trade-offs are made between time and resource value [175] or in decision support where the relative importance of objectives is chosen by human decision makers when presented a set of alternative solutions [187].

In this thesis, the focus is only on multi-objective problems where the objectives can be transformed into a scalar reward using a linear, non-decreasing scalarisation functions. Moreover, these weights are unknown during planning as otherwise the MOMDP can be transformed into and solved as a standard single-objective MDP, without the need for multi-objective approaches.[24] Formally, linear scalarisation functions are de-

[24] *For examples in which the scalarisation function is not linear or decreasing, the reader is referred to the survey of Roijers et al. [215].*

fined as the inner product of the weights and the component-wise valuation functions:

$$f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{V}^{\pi}(s^0) = \sum_{k \in [1,m]} w_k V_k^{\pi}(s^0) \tag{2.15}$$

The advantage of having a non-decreasing, linear scalarisation function, and the reason why most approaches assume one, is that it allows for more efficient multi-objective solution techniques for reasons similar to that of POMDP solving using belief MDPs (Section 2.4). Recall that the focus here is on MOMDPs where objective weights are not known a priori and hence a set of policies must be found so that the optimal policy can be retrieved from it later for *any weight vector*. Under an arbitrary function $f$ the scalarised value of a policy can vary enormously with every unique combination of weights. As the number of unique weight combinations is infinite, finding this policy set requires probing all (or, in practice, enough) weight vectors and is therefore likely computationally infeasible. In contrast, when the scalarisation function is linear and non-decreasing, it is possible to define value vectors $\boldsymbol{w} \cdot \boldsymbol{V}^{\pi}$ (assuming a common initial state $s^0$) and keep only those value vectors that maximise the expected scalarised reward (Equation 2.15) for at least one set of weights $\boldsymbol{w} \in [0,1]^m$. Because there is only a finite number of states and actions (and observations) in the MDP, there must also be a finite number of policies and thus a finite number of such value vectors. The supremum of the set of value vectors therefore defines a Piece-Wise Linear Convex (PWLC) polytope which constitutes a MOMDP solution, i.e. it contains an optimal policy for all weights.[25]

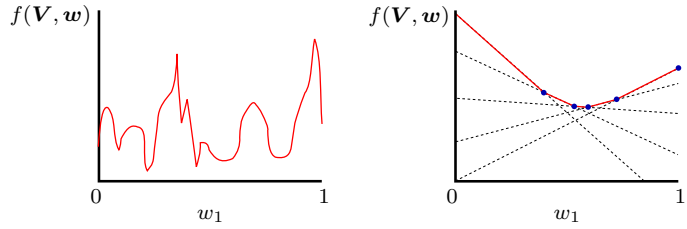



**Figure 2.10** *Illustation of policy values using an arbitrary scalarisation function (left) and a linear function (right). The horizontal axis displays the priority of weight $w_1$ and the other weight is given by $w_2 = 1 - w_1$.*

Figure 2.10 shows a typical two-objective example that illustrates an arbitrary scalarisation function on the left versus a non-decreasing, linear scalarisation function on the right. Observe that under this arbitrary scalarisation function there is no detectable structure in the scalar value as is almost always the case. On the other hand, in the linear case one can clearly see that all values are contained within the region bounded by the PWLC function $f(\boldsymbol{V}^{\pi^*}, \boldsymbol{w})$ in red and the graph axes. In this example, only five unique policies are sufficient to express all optimal policy values entire weight region $[0,1]$. Consider for instance the left-most blue dot, in the region starting from the

[25] *Indeed, solving MOMDP with a linear scalarisation function is closely related to POMDP solving through a belief MDP, as was underlined by White and Kim [263] where the MOMDP is transformed to a POMDP where the partial-observability models the objective weights.*

vertical axis until this intersection there exist (at least) one policy that achieves the highest possible scalarised value. At the intersection and moving to the right, another policy attains a higher expected scalarised value and therefore optimal for the next segment. This holds true for at least five policies, although more might exist that yield the same expected scalarised value over the same weight interval. In particular, the red line segments in the right graph of Figure 2.10 correspond to the scalar values of a set of policies known as the convex coverage set:



**Definition 2.23 Convex Coverage Set (CCS)**

The convex coverage set (CCS) for an MOMDP is a set of policies $\Psi \subseteq \Pi$ such that for every set of weights $\boldsymbol{w} \in [0,1]^m$ and a non-decreasing, linear scalarisation function $f$ the set $\Psi$ contains at least one policy $\pi$ that maximises the scalarised value $f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w})$, or

$$\Psi = \left\{ \pi \in \Pi \,\middle|\, \forall \boldsymbol{w} \in [0,1]^m, \forall \pi' \in \Pi, f, s^0 : f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) \geq f(\boldsymbol{V}^{\pi'}(s^0), \boldsymbol{w}) \right\} \quad (2.16)$$

Finding the CCS is a non-trivial problem and there have been several approaches in the literature of which some are briefly explained here. White and Kim [263] are arguably the first to target specifically the convex coverage set as a solution to MOMDP. Based on an observation made earlier by Sondik [241], they developed a procedure that transforms MOMDPs into a corresponding MDP where the partial observability models the uncertainty in objective weights. After this conversion finding the CCS becomes equivalent to finding the set of optimal policies over the belief space and standard POMDP techniques can be used – `Policy Iteration` in their work – to solve the problem.[26] Indeed observe the similarity between Figure 2.5, illustrating the optimal value over the belief space, and the right figure of Figure 2.10, illustrating the optimal CCS.

A different approach was taken by Barrett and Narayanan [21]. In this work, the authors propose a variant on `Value Iteration` (Section 2.2), called `Convex Hull Value Iteration` (`CHVI`), that extends the former algorithm by storing for each state/action pair also the linear valuation vectors. These valuations are maintained and propagated during the search to prevent an often substantial number of obsolete computations in the multi-objective setting. A similar algorithm but with better time and space bounds was developed by Lizotte et al. [161],[162].

Instead of designing a method dedicated to multi-objective planning problems, Roijers et al. [217] developed an algorithm that can harness the power of any single-objective MDP solver. Their `Optimistic Linear Support` (`OLS`) algorithm interleaves an algorithm that determines the minimal set of objective weights, called corner points or corner weights, with any existing single-objective MDP solver to compute the expected policy value at these corner points. The algorithm exploits a theorem due to Cheng [60] developed for POMDP solving. With a slight change of terminology from POMDP to MOMDP literature, this theorem states that the maximal error between

[26] *Although they must not require an initial belief state and can be inefficient on transformed multi-objective problems [215].*

the current known CCS and the optimal one can be found in the corner points. Using this theorem it can be shown that OLS decreases the error compared to the optimal CCS in each step and terminates when the optimal set is found. The details of the OLS algorithm can be found in Chapter 5.

**Example 2.24** Multi-objective path planning

The robot is given another mission: it is to extract precious minerals from the area it is located in. In addition to the exit, there are now also mineral deposits that the robot can reach for before leaving the area. By moving to such a deposit, the ore can be mined and the robot obtains additional reward for each ore that is mined. This situation is shown in Figure 2.11.

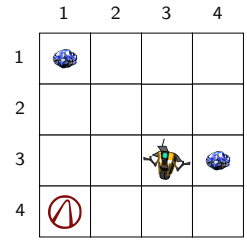



**Figure 2.11** *Multi-objective example of robot path planning where the robot may collect minerals for additional reward.*

Remember that it costs the robot energy to move around and therefore it is only interesting for the robot to collect a mineral if it expects to make a profit by obtaining it. This means that its expected revenue should outweigh the additional cost of getting to and from the mineral deposit, however the market prices for the mineral vary on a daily basis. It is possible to estimate the price and develop one single policy, but the revenue obtained under such a policy may be far from optimal if the estimated prices is not close to the real ore price. Instead, both objectives are combined using a weighting function such that its total reward when it has reached the goal is given by $100 - w_e \times \#moves - w_o \times \#ores$. In this reward, $w_e$ and $w_o$ are relative importance weights for the energy and ore objectives respectively. Both weights are normalised such that they are within the $[0, 1]$ range and, because they express relative importance, either weight can be expressed in terms of the other, i.e. $w_e = 1 - w_o$ and vice versa.

For ease of exposition it is now assumed that the moves of the robot will always succeed, later the impact of this assumption is discussed. Regardless of what weights are chosen, there are only three optimal paths that the robot can follow in this example, varying in the number of ores the robot will try to collect. These paths are shown in Figure 2.12a. When focusing primarily on minimisation of energy consumption, i.e. $w_e$ close to 1 and $w_o$ close to 0, the robot will choose the shortest path, coloured yellow in Figure 2.12a. This is the case when for example the mineral prices are very low. If both objectives are equally important, the robot will try to balance both and take the orange path. Finally, if obtaining the ore is preferred the robot will take the red path.

In Figure 2.12b, an example convex coverage set for this multi-objective problem is shown. Notice that the line segments of this function and the areas below them correspond to one of the three optimal paths. For instance, in the yellow region that corresponds to the

yellow path the value of the ore weight is relatively low. Because of the linear scalarisation, every optimal policy for the path-planning found in this region will have a scalarised value that lies on the blue line. In this example, the optimal policy for a low ore priority was found at approximately $w_o = 0.05$ but for every weight $w_o \leq 0.25$, where a corner point of the optimal scalarised value function $f(\boldsymbol{V}^{\pi^*}, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{V}^{\pi^*}$ occurs, that policy is optimal.

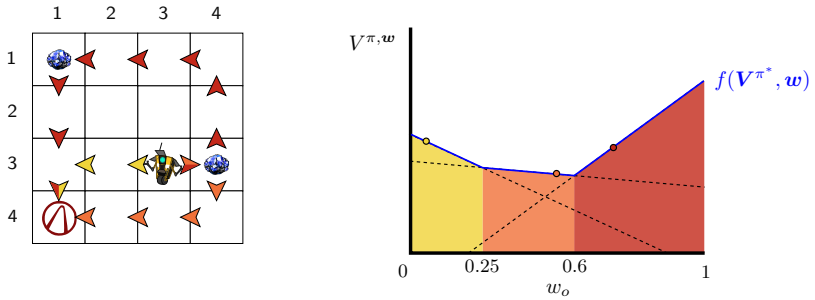



**Figure 2.12** *(a) Depending on the value of relative weights $w_e$ and $w_o$, the robot can follow one of the three optimal policies shown. (b) The scalarised value of the policies plotted for the ore weight $w_o$, each of the coloured areas corresponds to the (equally coloured) policy that is optimal for a range of ore weight values. The dots indicate the weights used to find the optimal policy for the scalarised version of the MOMDP. The upper surface of all value vectors, coloured blue in this figure, defines the convex coverage set for this problem.*

Because of the assumption that moves cannot fail, there are only three unique paths in this example. When moves can fail again, there exist a large number of different policies for each of these paths. For instance, for a certain ore weight value the robot might try to get to the ore to the right of it for an $x$ number of times. Thereafter it is no longer profitable to obtain the mineral and therefore it will instead move to the exit via the shortest path. Thus, Figure 2.12b will be similar in structure but with many more line regions and associated value vectors corresponding to all possible 'intermediate' policies.

# Chapter 3

# Solving the Maintenance Planning Problem

The first chapter of this thesis introduces the class of self-regulating planning problems, a decision-making problem that is encountered in the planning and execution phase of innovative contract forms such as the Dynamic Contracting approach of Volker et al. [254]. In particular, Chapter 1 presents a real-world example of such a multi-agent decision-making problem from the domain of infrastructural maintenance, the MAINTENANCE PLANNING PROBLEM (MPP). In Section 3.1 of this chapter, the MAINTENANCE PLANNING PROBLEM is formalised as a mathematical optimisation problem of finding a contingent plan that in expectation maximises the total reward of all agents, or service providers in the case of maintenance planning. Moreover, several approaches to solve this problem are presented, both optimal as well as approximate, laying the foundation for the work in subsequent chapters.

In a first attempt to solve MPP, Section 3.2 introduces a dynamic programming approach that optimises a recursive formulation of the optimisation function of MPP. Although this algorithm performs an exhaustive search over the (exponential) decision space, it produces optimal policies and functions as a baseline implementation that MPP solvers can be validated with and compared against. From there on the next step is to make use of the vast body of stochastic planning literature and the many tools and techniques that are readily available in the field, as outlined by Chapter 2. This is possible by modelling the problem as a *Markov Decision Process* (MDP), the most widely used model to represent stochastic decision-making problems in decision-theoretic planning and acknowledged by many solvers as a standardised format to express planning problems.

Section 3.3 starts by proposing a multi-agent Markov Decision Process (MMDP) formulation for MPP, thus enabling the use of any existing MMDP solver to produce optimal decision policies that correspond to solutions to the problem. Nonetheless, solving MMDPs still requires a specific type of solver that is equipped to handle multi-agent decision making and the typically exponential fully observable state space, of which only a limited number exist [40, 196, 224]. Hence Section 3.3 continues with

techniques to 'flatten' the multi-agent MDP into a single-agent MDP that represents the decision problem of all agents together. In essence, this flattened MDP 'encodes' the joint actions of all agents as if there is only one agent that makes the decision for all agents. Indeed a naive way to obtain this MDP is to simply enumerate the exponential number of combinations of action sets and state spaces, which is exactly what the enumeration encoding entails. However, more elaborate encodings are possible to avoid an exponential blow-up in *both* the state and action space. More specifically, MDP solvers are typically well-equipped to deal with huge state spaces; it is the action set size that commonly causes scalability issues in terms of computational effort. Section 3.3 therefore concludes with two encoding techniques, agent chaining and activity grouping, that limit the action set size by transferring some of the decision-making complexity into the state space.

In addition to the aforementioned techniques to find optimal solutions for MPP, Section 3.4 presents an approximate algorithm for scenarios in which optimality is not a requirement. An example of such a setting is decision-support scenario in which an approximate algorithm supports human decision makers by quickly showing estimates of the impact of the decisions they make, allowing for a fast comparison of alternatives. Waiting for an MDP solver to produce an optimal policy is typically not desirable in such settings, especially since this may take anywhere from several minutes up to multiple hours or even days for realistically sized instances. The latter is supported by the results of empirical evaluation of Section 3.5 that investigates the scalability of all MDP-based approaches presented in this chapter.

**Contributions** In this chapter a mathematical definition is presented for the MAINTENANCE PLANNING PROBLEM (MPP) that was first introduced by Altamirano et al. [9] and formalised as part of a dynamic mechanism design solution by Scharpff et al. [228]. Given the formal definition of MPP, this chapter presents two approaches to develop optimal contingent plans, i.e. joint decision policies that maximise the expected value. The first is based upon dynamic programming and optimises a recursive formulation of the policy value function. This approach is straightforward and provides an excellent benchmark for more sophisticated methods. The second method encodes the problem as an MDP so that it can be solved using one of the very many the existing MDP solvers. Additionally, this chapter shows that by carefully encoding the problem, the efficiency of policy finding is increased significantly. This is demonstrated using the state-of-the-art SPUDD [116] solver. The optimal solving algorithms and MDP encodings have been published by Scharpff et al. [228].

Besides optimal solving, this chapter also proposes approximation of joint policies for MPP based upon Monte-Carlo Tree Search (MTCS). This approach uses an exploration versus exploitation approach to quickly find joint policies of high quality but it can only guarantee optimality in the limit, i.e. when all parts of the search tree have been explored. Nevertheless, the experiments in the end of this chapter demonstrate that such an approach often produces near-optimal policies quickly and finds the optimal policy on many occasions, therefore making it a viable candidate for many cases unless optimal solutions are explicitly required. The MCTS approach was published as part of the work by Roijers et al. [216] to find approximate convex coverage sets.

## 3.1 The Maintenance Planning Problem

In this section the maintenance planning problem is formally defined as a computational problem and a first solution based on dynamic programming is proposed. The maintenance planning problem is one specific instance of multi-agent problems with time-dependent action rewards, and has been the main motivation for this research. It serves as a good example of such a planning problem that can be found in a realistic setting and it is characteristic for the tension between individual and global goal.

In the maintenance planning problem there is a group of contracted service providers, the agents $\boldsymbol{N} = \{1, 2, \ldots, n\}$, responsible for the maintenance of a network consisting of roads $E$ over a period of $h$ discrete time steps. For convenience, the set of time steps is denoted $T = \{1, 2, \ldots, h\}$. Recall from Section 1.1 that the dynamic contracting procedure consists of three phases: *procurement*, *planning* and *execution*. In the preliminary procurement phase each agent $i$ is assigned a disjoint subset of roads $E_i \subseteq E$ that it has to service. Furthermore, possible maintenance activities for these roads are identified in the procurement phase, as they are typically part of a contractor's bid. The maintenance activities for each road $e_k$ are given as a set $\mathcal{A}_i$. The set of activities for which an agent $i \in \boldsymbol{N}$ is responsible, is given by the union of all activities for each of the roads it is assigned to plus an additional do-nothing activity. This *no-operation* or simply *no-op*, denoted by $\circ_i$, allows the agent to be idle for one unit of time. Put together, the activity set for agent $i$ is given by $\mathcal{A}_i = \bigcup_{k|e_k \in E_i} \mathcal{A}_k \cup \{\circ_i\}$ and the collection thereof over all agents is denoted by $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_i\}_{i \in \boldsymbol{N}}$. As the planning of maintenance activities occurs in the planning phase of the dynamic contracting procedure, the assignment of roads and the set of agent activities are given as input.

Formally, activities are maintenance operations that can be performed and are defined as a tuple $\langle w, d, p, d' \rangle$. Here $w \in \mathbb{R}$ is a constant specifying the reward that is given to the agent upon completion of the activity, $d \in \mathbb{Z}^+$ is the number of consecutive time steps minimally required to complete the activity, $p \in [0, 1]$ is the probability of the activity being delayed, due to e.g. accidents, unforeseen conditions, incorrect assessment of road quality, etc. If an activity is delayed, the additional number of consecutive time steps needed to complete the activity is given by $d' \in \mathbb{Z}^+$. Whether an activity is delayed or not becomes known after it has been started, as this can only be assessed correctly once the maintenance work has begun. The no-op is modelled as an activity $\circ_i = \langle 0, 1, 0, 0 \rangle$ – it yields zero revenue, has unit time duration and zero probability of delay – and can be repeated.

**Outcomes and Histories** Delay realisations are known as outcomes. For every activity $\mathsf{a}_k$, the possible outcomes $O(\mathsf{a}_k)$ are $\{o_k^+, o_k^-\}$, denoting that $\mathsf{a}_k$ did not or did delay respectively, and the probability of both outcomes is given by $Pr(o_k^-) = p_k$ and $Pr(o_k^+) = 1 - p_k$. For the no-op activity, only the not-delayed outcome is available and its probability is 1. The realised duration of an activity $\mathsf{a}_k$, given its outcome $o_k$, can be computed using:

$$\hat{d}(o_k) = \begin{cases} d_k, & \text{if } o_k = o_k^+ \\ d_k + d'_k, & \text{if } o_k = o_k^- \end{cases} \tag{3.1}$$

In addition to a single activity and its corresponding outcome, it is also convenient to define similar notions for joint activities and possible outcomes thereof. For a joint activity $\vec{\mathtt{a}} = \langle \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n \rangle$, the set of potential outcomes is given by the Cartesian product of all individual activity outcomes, i.e. $\boldsymbol{O}(\vec{\mathtt{a}}) = O(\mathtt{a}_1) \times O(\mathtt{a}_2) \times \ldots \times O(\mathtt{a}_n)$. The probability of one such a joint outcome $\vec{o} = \langle o_1, o_2, \ldots, o_n \rangle \in O(\vec{\mathtt{a}})$ is defined as $Pr(\vec{o}) = \prod_{i \in \boldsymbol{N}} Pr(o_i)$, such that the outcome probability $Pr(o_i)$ for a single activity is independent of all other activities. Note that the sum of probabilities over all joint outcomes of a joint activity must always be equal to one.

To keep track of executed activities and their outcomes, a history is maintained during the execution of maintenance plans. For a specific time point $t$, the history is given by $H^t \colon \boldsymbol{\mathcal{A}} \mapsto T \times \boldsymbol{O}$ such that for every activity $\mathtt{a}$ that has been started before or at time $t$, $H^t(\mathtt{a})$ returns the start time and delay realisation (outcome) as a pair $\langle t_\mathtt{a}, o_\mathtt{a} \rangle$. The notation $\mathtt{a} \in H^t$ denotes that the history contains an entry for activity $\mathtt{a}$, i.e. it conveniently denotes $\mathtt{a} \in \mathrm{Dom}(H^t)$, and it has thus been started in this history. Correspondingly, $\mathtt{a} \notin H^t$ means that the activity has not started yet.[27] When referring to either just the start time or the outcome of an activity, the notations $H^t_s(\mathtt{a})$ and $H^t_o(\mathtt{a})$ will be used respectively. Furthermore, $H^t_{\mathcal{A}}(t')$ is used to denote the set of activities that was being executed at time $t'$ according to history $H^t$, i.e.:

$$H^t_{\mathcal{A}}(t') = \left\{ \mathtt{a} \in \mathcal{A} \mid \mathtt{a} \in H^t, H^t_s(\mathtt{a}) \leq t', H^t_s(\mathtt{a}) + \hat{d}(H^t_o(\mathtt{a})) - 1 \geq t' \right\} \tag{3.2}$$

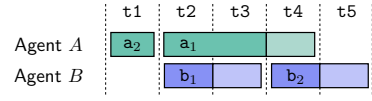
Finally, the probability of a certain history occurring can be expressed in terms of the activities and outcomes that it contains. The probability of any history $H^t$ is defined as the product over the probabilities of individual outcomes, or

$$Pr(H^t) = \prod_{\mathtt{a} \in H^t} Pr(H^t_o(\mathtt{a})) \tag{3.3}$$

**Example 3.1** History

$H^t(\mathtt{a}_1) = \langle \mathtt{t2}, o^-_{\mathtt{a}_1} \rangle \quad H^t(\mathtt{b}_1) = \langle \mathtt{t2}, o^-_{\mathtt{b}_1} \rangle$
$H^t(\mathtt{a}_2) = \langle \mathtt{t1}, o^+_{\mathtt{a}_2} \rangle \quad H^t(\mathtt{b}_2) = \langle \mathtt{t4}, o^-_{\mathtt{b}_2} \rangle$
$H^t(\mathtt{a}_3) = \ ?$

(a)




(b)

**Figure 3.1** *Example history for a two-agent planning problem. The history is listed as the start times and outcomes of activities in (a) and visualised in (b). The rectangles symbolise the activities and the size thereof matches their durations, where the delay duration is coloured slightly lighter. In this example,* $\mathtt{a}_3$ *has not been started yet. Keep in mind that although activity* $\mathtt{a}_2$ *does not delay in this example, in another history it may be delayed (and vice versa for the others).*

[27] *For an activity that has not been started one could for instance return a pair* $\langle -1, ? \rangle$*. The definitions and algorithms in this thesis assume that membership is tested before querying the history.*

> In Figure 3.1 an example history is illustrated for a two-agent problem, the left figure shows its textual form and its graphical interpretation is shown on the right. Observe that the sets $H^t_{\mathcal{A}}(\texttt{t1})$ to $H^t_{\mathcal{A}}(\texttt{t5})$ correspond exactly to the time slots t1 to t5 of Figure 3.1b.

A last remark concerning histories is that the notation used here enables a very compact representation. Alternatively, one could use the state/action equivalent that is used in many stochastic planning articles, referred to as *execution sequences* in this thesis (see Chapter 4). In this form, the history is written as the sequence of states, actions and result states that were successively encountered during execution up to time $t$, e.g. $H^t = [s^0, \mathsf{a}^0, s^1, \mathsf{a}^1, \ldots, s^t]$ such that each pair of $s^x$ and $\mathsf{a}^x$ denote respectively the state that was encountered and the action that was taken in that state. Naturally, this sequence must always end in the current (last) state $s^t$.



**Value of Maintenance** The cost of performing maintenance is known to each agent through a time-dependent cost function $c_i \colon \mathcal{A}_i \times T \mapsto \mathbb{R}$. Incorporating time into the cost function allows for example modelling of costs based on the varying number of resources available to an agent over time, different labour costs during day and night time, or changing market prices for raw materials. The revenue an agent $i$ obtains for completing a single activity $\mathsf{a}_k$, assuming it starts at time $t' \in T$ and has outcome $o_k$, is therefore given by $w_k - \sum_{t=t'}^{t'+\hat{d}(o_k)-1} c_i(\mathsf{a}_k, t)$ and its total profit can be computed by

$$\sum_{\mathsf{a}_k \in \mathcal{A}_i} \left( w_k - \sum_{t=t'}^{t'+\hat{d}(o_k)-1} c_i(\mathsf{a}_k, t) \right) \tag{3.4}$$

In this thesis it is assumed that no-ops do not incur any maintenance costs and therefore $c_i(\circ_i, t) = 0$ for every agent $i$ and time $t$. However, in general one could use this function to model the cost of an agent being idle. Finally, the total cost of maintenance for a joint activity $\vec{\mathsf{a}} \in \boldsymbol{\mathcal{A}}$ at time $t$ is

$$c(\vec{\mathsf{a}}, t) = \sum_{i \in \boldsymbol{N}} c_i(\mathsf{a}_i, t) \tag{3.5}$$

where $\mathsf{a}_i$ is the activity of agent $i$ in joint activity $\vec{\mathsf{a}}$. Notice that it can always be assumed that a joint activity contains $n$ elements: idle agents are 'performing' a no-op action.

Besides individual cost functions, there is a network cost function $\ell$ such that $\ell(\vec{\mathsf{a}}, t) \in \mathbb{R}$ describes the (monetary) impact on the network throughput when maintenance operations $\vec{\mathsf{a}}$ are concurrently performed at time $t$. Maintenance requires at least a partial closure of roads, causing a reduction in network throughput and consequential delays to the network users. This delay is expressed in terms of hours of *traffic time lost* (ttl) and is typically computed as the time lost due to hindrance in the current situation minus the time that is required under average network conditions. Essentially, ttl expresses the marginal contribution of network maintenance to the network delays and the economic impact thereof is computed using a standardised 'value of time'.[28]

[28] *The value of time for most traffic in the Netherlands can be found in the report by Bates [24], in which also ttl is explained more thoroughly and models for its computation are presented.*

The network cost function can have any source: it could be a system of heuristic rules, extracted from historical data or derived through simulation.

The throughput of the road network depends greatly on the type of maintenance that is performed but also on the combination of activities. When multiple maintenance activities are performed concurrently, requiring several parts of the network to be closed, their impact on the network throughput is typically much worse than when they would have been performed sequentially. In such cases the network cost function is super-additive, i.e. there is at least one time step $t$ in which $\ell(\langle \mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n \rangle, t) > \ell(\langle \mathsf{a}_1 \rangle, t) + \ell(\langle \mathsf{a}_2 \rangle, t) + \ldots + \ell(\langle \mathsf{a}_n \rangle, t)$. In general, both super and sub-additive network rewards are possible.

To compute the total network cost of performing maintenance, the network cost is summed over all time steps $T$. Given a complete execution history $H^h$ for a planning horizon of length $h$, the total network cost that was incurred can be computed by

$$\sum_{t \in T} \ell(H^h_{\mathcal{A}}(t), t) \tag{3.6}$$

where $H^h_{\mathcal{A}}(t)$ is the set of activities that is concurrently active at time $t$, as defined by Equation 3.2. In the future when maintenance or network costs for the joint activity contained in history $H^t$ at time $t$ are meant, the notational shorthands $c(H^t)$ and $\ell(H^t)$ denote respectively $c(H^t_{\mathcal{A}}(t), t)$ and $\ell(H^t_{\mathcal{A}}(t), t)$.

The goal in the maintenance planning problem is to plan maintenance activities in such a way that the overall profit, i.e. the sum of agent revenues minus the network costs, is maximised. However, in the presence of uncertain durations it is not sufficient to develop a (static) optimal plan, e.g. a one-time assignment of start times to all the activities. Instead the goal is to find a *contingent plan* that specifies optimal planning decisions for every possible realisation of activity delays:

**Definition 3.2 Contingent Plan**

A contingent plan $\mathcal{P} : \mathcal{H} \mapsto \boldsymbol{\mathcal{A}}$ is a plan that for every history $H^t \in \mathcal{H}$ at time $t \in T$ returns the joint activity $\vec{\mathsf{a}} \in \boldsymbol{\mathcal{A}}$ that should be started at time $t$.

A contingent plan can be viewed as a reactive plan that can adapt to *foreseen* eventualities, such as a maintenance activity possibly delaying, in a predefined way. Informally, it specifies a set of rules that dictate what activities to start given the current history of activities and their realised outcomes. To produce an (optimal) contingent plan it is typically necessary to study all possible histories $\mathcal{H}^h$ of length $h$ that can be encountered, which makes it more time consuming to develop than a single plan. However, as this is done before its actual use in the execution phase of the dynamic contracting procedure, this is not an issue here. Moreover, the possibility of reacting to eventualities always results in outcomes at least as good but typically better that single-plan outcomes. Thus when 'sufficient' time is available, it is worth to invest it in finding a contingent plan.

The quality of a contingent plan is expressed in terms of the reward that can be expected from following it. The expected reward $V^{\mathcal{P}}$ over all possible histories $H^h \in$

$\mathcal{H}^h$ of length $h$ can be specified by combining Equations 3.5 and 3.6:

$$V^{\mathcal{P}} = \mathbb{E}\Big[ \sum_{\mathtt{a}_k \in H^h} w_k - \sum_{t \in T} \Big( c(H^t) + \ell(H^t) \Big) \;\Big|\; H^t = H^{t-1} \oplus \mathcal{P}(H^{t-1}, t) \Big]$$
$$= \sum_{H^h \in \mathcal{H}^h | \mathcal{P}} Pr(H^h) \Big( \sum_{\mathtt{a}_k \in H^h} w_k - \sum_{t \in T} \big( c(H^t) + \ell(H^t) \big) \Big) \tag{3.7}$$

where $\mathcal{H}^h|\mathcal{P}$ denotes the set of all possible histories of length $h$ reachable by following contingent plan $\mathcal{P}$, $Pr(H^h)$ is computed using Equation 3.3 and $H^{-1} = \emptyset$. Recall that $c(H^t)$ is short for $c(H^t_{\mathcal{A}}(t), t)$ (and a similar shorthand is used for $\ell$). The optimal contingent plan is defined as one that maximises this expected reward, or $\mathcal{P}^* = \arg\max_{\mathcal{P}} V^{\mathcal{P}}$. In the maintenance planning problem the focus is typically on finding an optimal contingent plan, unless explicitly mentioned otherwise.

Notice that in the expected reward of Equation 3.7, the network cost function $\ell$ implicitly specifies the degree of coupling between agents and is the main reason why coordination is required. If there would be no network costs, the expected plan reward can be factored per agent and, as a consequence, optimised individually. On the other hand, when every activity of every agent can interact with all other activities, the agents are fully coupled. And, as finding a contingent plan that optimises the expected reward requires considering these interactions, coordination of a fully coupled network is typically much harder. Also, the ratio between individual and network costs indicates the importance of using a coordinated approach: when network costs are relatively high, a jointly coordinated approach is much more preferred, whereas comparatively low network costs may make the (computational) effort typically required by such an approach not worthwhile.

There are a few restrictions on the set of contingent plans. First, an agent can only start a maintenance activity if it has enough time left to complete it within the contracted duration, even if it is delayed. This means that an activity $\mathtt{a}_k$ has to be started before $h - (d_k + d'_k)$, whether it actually delays or not.[29] Furthermore, maintenance activities are non-preemptive and thus have to be executed in their entirety once they have been started. As maintenance of major road networks involves a lot of preparation time and heavy equipment, reallocation of the agent's resources to another job site is generally considered too costly. And finally, agents are allowed to perform at most one maintenance activity at a time due to the resources available to an agent.[30] The set of feasible contingent plans that comply to these restrictions is denoted by $\mathtt{P}$. Throughout the remainder of the thesis it will always be assumed implicitly that only contingent plans from this set will be considered.

Together, all of the aforementioned elements can be summarised into a formal definition of the maintenance planning problem:

[29] *Alternatively, one could soften this restriction using late fees.*

[30] *This latter restriction can be circumvented in the case where concurrent maintenance by the same agent is required. To do so, one can split the agent into several sub-contractors or maintenance teams, each responsible for a subset of its maintenance activities.*

**Definition 3.3 Maintenance Planning Problem (MPP)**

An instance of the Maintenance Planning Problem is given by a tuple $\langle \boldsymbol{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{c}, \ell, T \rangle$ where

- $\boldsymbol{N} = \{1, 2, \ldots, n\}$ is the set of agents.
- $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_i\}_{i \in \boldsymbol{N}}$ is the collection of activity sets in which each $\mathcal{A}_i$ is the set of activities available to agent $i \in \boldsymbol{N}$ including a no-op $\circ_i$. Every activity $\mathsf{a}_k$ specifies a revenue $w_k$, integer duration $d_k$, probability of delay $p_k$ and integer duration of delay $d'_k$. Activities can have two outcomes, delayed ($o_k^-$) or not-delayed ($o_k^+$), with probabilities $p_k$ and $1 - p_k$, and the duration of an activity given its outcome $o_k$ is given by $\hat{d}(o_k)$ (Equation 3.1).
- $\boldsymbol{c} = \{c_i\}_{i \in \boldsymbol{N}}$ is a collection of cost functions in which each $c_i \colon \mathcal{A}_i \times T \mapsto \mathbb{R}$ is the cost function of agent $i \in \boldsymbol{N}$.
- $\ell \colon \boldsymbol{\mathcal{A}} \times T \mapsto \mathbb{R}$ is a network cost function such that for each combination of activities $\vec{\mathsf{a}} \in \boldsymbol{\mathcal{A}}$ the network cost at time $t \in T$ is given by $\ell(\vec{\mathsf{a}}, t)$.
- $T = \{1, 2, \ldots, h\}$ is a set of discrete time steps that constitute the planning period.

Given such an instance $\langle \boldsymbol{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{c}, \ell, T \rangle$, the MAINTENANCE PLANNING PROBLEM is to find an optimal contingent plan $\mathcal{P} \colon \mathcal{H} \mapsto \boldsymbol{\mathcal{A}}$ from the set of feasible contingent plans P that maximises $V^{\mathcal{P}}$ (Equation 3.7) over all possible histories $\mathcal{H}^h$ of length $h$.

Observe that in Definition 3.3 the set of network edges $E$ is not included. This is because all of the network information required is included implicitly in the activities, the agent cost functions and the network cost. From an algorithmic point of view it is irrelevant how the network is composed, only the effects and costs of maintenance are necessary when developing maintenance plans. Furthermore, it can be shown that any instance of the MAINTENANCE PLANNING PROBLEM is in fact a special case of self-regulating planning that was defined in Section 1.3. This proof is rather involved and makes use of concepts introduced in subsequent chapters, however the interested reader is referred to Section A.1.

Further in this chapter several approaches to solve the maintenance planning problem using existing techniques will be discussed. In the context of the dynamic contracting approach of Section 1.2, it is realistic to assume that once the agents have been contracted for the procured maintenance operations they are given a substantial amount of time to plan and coordinate their operations. This makes it feasible, in terms of effort versus expected gain, to develop optimal contingent plans. In contexts where the time available for planning is limited, approximate methods might be preferred. One such a method is briefly reviewed later in Section 3.4.

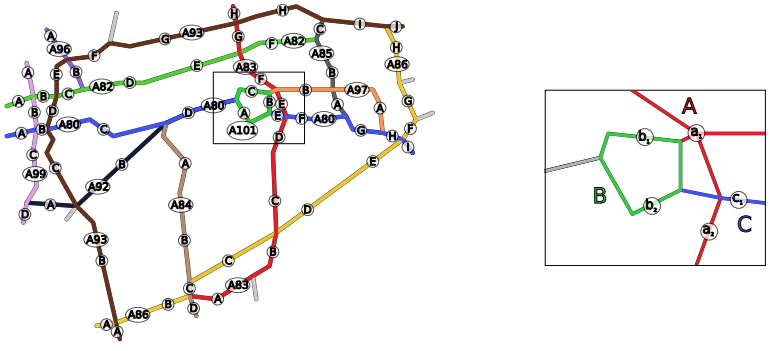
**Remark 3.4 Quality as an Objective**

All of the approaches that will be presented in this thesis consider the maintenance planning problem as formalised in Definition 3.3. However, it is also possible to extend the problem with quality demands and costs, which was done by Scharpff et al. [228] in order to more closely resemble the real-world problem. This extension to MPP emphasises even further on the trade-offs that service providers typically need to make in planning decisions. Furthermore, it can serve as an interesting additional objective in the multi-objective setting that is studied in Chapter 5. For these reasons, adding the quality objective is briefly outlined here.

All roads $e_k \in E$ have an associated quality level $q_k \in [0, 1]$ that expresses the state of the road and a quality degradation function $\Delta q_k : q \times T \mapsto q$ that expresses the degradation of a road given its current state and time. The dependency on current quality and time in this function enables modeling of many natural aspects, for instance new roads might degrade less than already damaged roads (or more), summer and winter seasons cause for a more significant deterioration of roads, or degradation varies according to amount of network traffic which in turn also varies over time (e.g. holidays). In addition to a maintenance cost function, each agent is also given a quality cost function $Q_i : q \times T \mapsto \mathbb{R}$ that states the cost of having a certain quality level for each time step and the sum of all quality cost functions is included in the contingent plan value of Equation 3.7 in a way similar to the sum of maintenance cost functions.

**Example 3.5** An example problem

During the remainder of this chapter, a guiding example problem is used to aid the explanation of all the presented approaches. The example network, shown in full in Figure 3.2a, is derived from the German 'Ruhrgebiet' and was initially used as the network in the serious game of Chapter 7. In order to keep the example simple but illustrative, only the small area inside the box will be considered.




(a) (b)

**Figure 3.2** *Example road network, shown in full in (a). This example focuses on the area contained within the rectangle, shown enlarged in (b) on the right. The highway codes and segments are now coloured corresponding to the agent that is responsible for their maintenance, and the required maintenance operations are shown as labels on the segments.*

Within this area, enlarged in Figure 3.2b, three contractors – $A$, $B$ and $C$ – are hired to perform maintenance within a period of 5 weeks and each time point in $T = \{\texttt{t1}, \texttt{t2}, \ldots, \texttt{t5}\}$ spans one week. Each set of road segments belonging to a contractor is coloured to match the agent's colour, e.g. agent $B$ is responsible for the road segments of the A101 of the network on the left. The road segments are labelled with maintenance activities that need to be performed on them. For instance, agent $A$ has to service a junction ($\texttt{a}_1$) and a road segment ($\texttt{a}_2$) and its set of activities is hence defined as $\mathcal{A}_A = \{\texttt{a}_1, \texttt{a}_2\}$. Table 3.1 specifies the activity sets for every agent as well as the characteristics of the activities:

| Agent | Activity | Revenue | Duration (w) | Delay chance | Delay dur. (w) |
|---|---|---|---|---|---|
| $A$ | $\texttt{a}_1$ | 140 | 2 | 25 % | 1 |
| $A$ | $\texttt{a}_2$ | 60 | 1 | 0 % | - |
| $B$ | $\texttt{b}_1$ | 17 | 2 | 30 % | 1 |
| $B$ | $\texttt{b}_2$ | 90 | 1 | 80 % | 1 |
| $C$ | $\texttt{c}_1$ | 205 | 3 | 15 % | 1 |

**Table 3.1** *The activity sets and the properties of each activity, delay durations are in weeks.*

A few remarks regarding this table can be made. First of all, activity $\texttt{a}_2$ has no probability of delay and can therefore never require more time than its regular duration specifies. Agent $B$'s activities are very tight: if both his activities delay, all five time steps are required in order to perform all maintenance.

In addition to the activities, the agents all have maintenance cost functions – $c_A$, $c_B$ and $c_C$ respectively – that define the cost of maintenance for each of their activities and every time slot. These functions are presented in a tabular form in Table 3.2 with the activities as rows and weeks as columns. Notice that in this cost matrix, agent $B$ is indifferent when its activity $\texttt{b}_1$ is performed as its maintenance costs do not vary over time. For its other activity, the costs are increasing over time hence $\texttt{b}_2$ is preferably performed as early as possible.

| Costs | t1 | t2 | t3 | t4 | t5 |
|---|---|---|---|---|---|
| $\texttt{a}_1$ | 18 | 15 | 13 | 14 | 13 |
| $\texttt{a}_2$ | 23 | 18 | 20 | 40 | 25 |
| $\texttt{b}_1$ | 11 | 11 | 11 | 11 | 11 |
| $\texttt{b}_2$ | 7 | 8 | 10 | 14 | 19 |
| $\texttt{c}_1$ | 30 | 25 | 27 | 31 | 24 |

**Table 3.2** *Activity maintenance costs per week in matrix form.*

The network costs of this example problem are specified using a binary, factor-based model that specifies the impact on traffic relative to the 'normal' traffic distribution. This model expresses the individual effect of each of the maintenance activities on the traffic and the interaction between pairs of activities performed concurrently through pair-wise coefficients (e.g. based on proximity). In later sections, it will be shown that realistic models can be approximated using this model (Section 7.1.3) while it can be compactly encoded in for instance MDP or RDDL (as in the experiments of Section 3.5). Therefore this is the network cost model of choice in this example.

The matrix in Table 3.3a shows the regular traffic time lost (ttl) summed over the area that is affected by each activity. On the right, Table 3.3b shows the additional cost

factor matrix for every pair of activities that is executed concurrently. In this model, every unit of ttl has an economic impact of 1 unit. The network cost of a single time step can be computed by summing the normal ttl times the factor for every pair of concurrently performed activities, or $\ell(\langle \mathtt{a}_i, \mathtt{a}_j \rangle, t) = \big(ttl(\mathtt{a}_i, t) + ttl(\mathtt{a}_j, t)\big) \cdot factor(\mathtt{a}_i, \mathtt{a}_j)$ (see below for an example). In this model, the normal ttl peaks in time t3 for all roads (the ttl is correlated) and is the quietest at time t5. The factor for activity pairs depends on the proximity of the activities, for instance the factor for activities $\mathtt{a}_1$ and $\mathtt{b}_2$ is relatively high (1.3) whereas $\mathtt{a}_2$ and $\mathtt{b}_1$ have only limited interaction (0.2).

| TTL | t1 | t2 | t3 | t4 | t5 |
|---|---|---|---|---|---|
| $\mathtt{a}_1$ | 20 | 21 | 19 | 16 | 13 |
| $\mathtt{a}_2$ | 16 | 16 | 15 | 14 | 10 |
| $\mathtt{b}_1$ | 32 | 35 | 33 | 28 | 21 |
| $\mathtt{b}_2$ | 26 | 28 | 26 | 23 | 18 |
| $\mathtt{c}_1$ | 17 | 18 | 16 | 14 | 11 |

(a)

| Factor | $\mathtt{a}_1$ | $\mathtt{a}_2$ | $\mathtt{b}_1$ | $\mathtt{b}_2$ | $\mathtt{c}_1$ |
|---|---|---|---|---|---|
| $\mathtt{a}_1$ | - | - | 1.3 | 0.8 | 0.9 |
| $\mathtt{a}_2$ | - | - | 0.2 | 0.7 | 0.7 |
| $\mathtt{b}_1$ | 1.3 | 0.2 | - | - | 0.6 |
| $\mathtt{b}_2$ | 0.8 | 0.7 | - | - | 1.0 |
| $\mathtt{c}_1$ | 0.9 | 0.7 | 0.6 | 1.0 | - |

(b)

**Table 3.3** *Factor-based network cost model with (a) the 'idle' conditions and (b) the factor by which concurrent maintenance increases/decreases the ttl.*

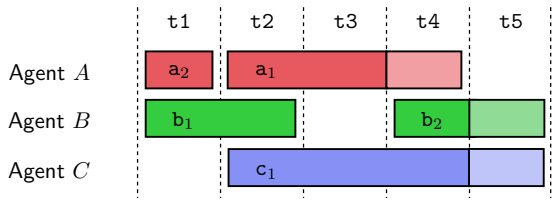



**Figure 3.3** *Illustration of a possible history after executing a contingent plan.*

To illustrate the computation of revenue, maintenance costs and network costs, an example is shown in Figure 3.3. The figure shows an example history resulted from the execution of a contingent plan, i.e. the activities have been executed and the delay realisations are known. In this example, the maintenance cost of agent $A$'s activities are $15+13+14 = 42$ and 23 for respectively activity $\mathtt{a}_1$ and $\mathtt{a}_2$. For agent $B$ the total maintenance costs are $11 + 11 = 22$ and $14 + 19 = 33$ and agent $C$ has a total of $25 + 27 + 31 + 24 = 107$. Every agent has performed all of its activities and therefore the revenues are respectively 200, 265 and 205, leading to a profit of 135, 210 and 93. At time t1, activities $\mathtt{a}_2$ and $\mathtt{b}_1$ are performed concurrently and the network cost thereof is $\ell(\mathtt{a}_2, \mathtt{b}_1, \mathtt{t1}) = (16 + 32) \times 0.2 = 9.6$. In time step t2, three activities are performed at the same time ($\mathtt{a}_1$, $\mathtt{b}_1$ and $\mathtt{c}_1$) and the network cost is summed over every pair. This results in $\ell(\mathtt{a}_1, \mathtt{b}_1, \mathtt{t2}) = (21 + 35) \times 1.3 = 72.8$, $\ell(\mathtt{a}_1, \mathtt{c}_1, \mathtt{t2}) = (21 + 18) \times 0.9 = 35.1$ and $\ell(\mathtt{b}_1, \mathtt{c}_1, \mathtt{t2}) = (35 + 18) \times 0.6 = 31.8$. The remainder of the computations can be done likewise to obtain the network costs for every time step (9.6, 139.7, 31.5, 95.2 and 29 respectively), summing to a total of 305 for the entire history. The total value of this history is therefore $135 + 210 + 93 - 305 = 133$.

The expected value of this history can be easily computed as the probability of this history occurring times its value. Recall from Equation 3.3 that the probability of a history

is given by the product of individual activity delay probabilities. Given the outcomes of this history, its probability is

$$\begin{aligned} &Pr(o^-_{\mathtt{a}_1}) \times Pr(o^+_{\mathtt{a}_2}) \times Pr(o^+_{\mathtt{b}_1}) \times Pr(o^-_{\mathtt{b}_2}) \times Pr(o^-_{\mathtt{c}_1}) \\ &= p_{\mathtt{a}_1} \times (1 - p_{\mathtt{a}_2}) \times (1 - p_{\mathtt{b}_1}) \times p_{\mathtt{b}_2} \times p_{\mathtt{c}_1} \\ &= 0.25 \times (1 - 0) \times (1 - 0.3) \times 0.8 \times 0.15 = 0.021 \end{aligned}$$

or 2.1%. As a consequence, the expected value of this particular history is $0.021 \times 133 = 2.793$. The expected value for this history is low but indeed it is not very likely to be encountered: both activities $\mathtt{a}_1$ and $\mathtt{c}_1$ are delayed, even though their delay probabilities are relatively low.

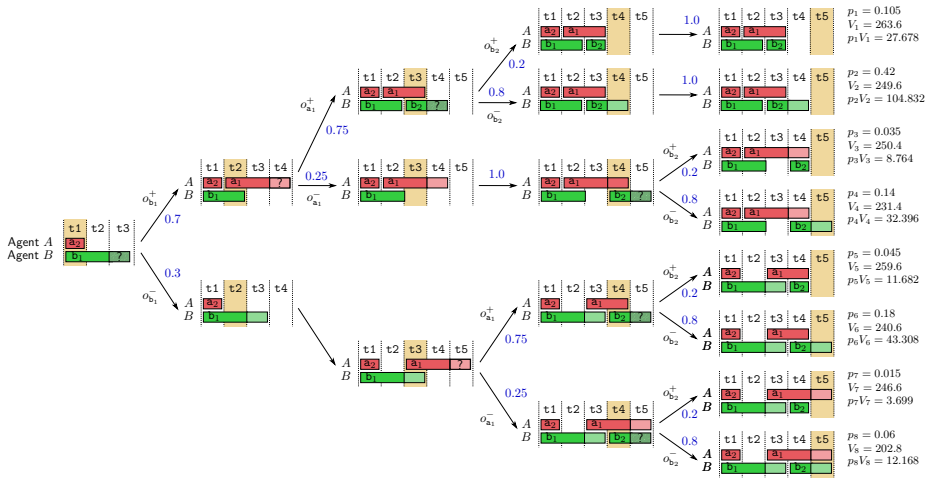



**Figure 3.4** *Contingent plan for the activities of agents $A$ and $B$. Every possible history that can be encountered during execution is shown with the corresponding decisions (joint activities, highlighted in each history) and the outcomes thereof (labelled on the arrows). The transition probabilities are displayed in blue. A total of 8 unique histories of length 5 can be reached, which are shown on the right with their corresponding probability $p_i$, value $V_i$ and expected value $p_iV_i$.*

Finally, in Figure 3.4 an example contingent plan is visualised for the activities of agents $A$ and $B$. It shows the possible unique histories that can be encountered while following the decisions of the contingent plan. On the far left, the joint activity is shown that is taken when given the (initially) empty history at time $\mathtt{t1}$, which is $\langle \mathtt{a}_2, \mathtt{b}_1 \rangle$ in this example. At the time of starting a joint activity, it is not yet known whether activities will be delayed or not, and this is illustrated by the question mark in the delay duration part of $\mathtt{b}_1$. For this joint activity two outcomes are possible, one in which $\mathtt{b}_1$ is not delayed and one in which it is, resulting in two unique new histories for the next decision step. Observe that in the next step, the decision depends on history: if $\mathtt{b}_1$ does not delay, agent $A$ will start its activity $\mathtt{a}_1$, but if $\mathtt{b}_1$ does delay, it will not start any activity (it performs a no-op).

The unique histories for every time step are shown column-wise in this figure. Transitions between them are shown as one or more arrows, depending on the number of

outcomes the joint activity can have. Notice that in this example every history leads to at most two new histories, based upon the outcome of the action started. This does not have to be true in general: when multiple activities with delays are started, the number of outcomes is (exponentially) larger. The probabilities of the outcomes are displayed as labels on the arrows, e.g. the first transition has outcomes $o^+_{\mathrm{b}_1}$ and $o^-_{\mathrm{b}_1}$ with probabilities $0.7$ and $0.3$ respectively. On the right, one can find all full-length histories that are reachable under this example contingent plan, i.e. these 8 histories are the unique results of following the contingent plan. This number is equal to 8 because there are 3 activities with stochastic outcomes ($\mathrm{a}_2$ cannot delay), and there are $2^3 = 8$ unique combinations of delay realisations. For every full-length history, the probability, value and expected value are shown in this figure and the total expected value of this contingent plan is equal to $\sum_i p_i V_i = 244.527$. The value of every individual history is computed as demonstrated previously for the example history of Figure 3.3.



## 3.2 Solving MPP with Dynamic Programming

An initial attempt to find optimal maintenance plans is to formulate a recursive version of the expected plan reward (Equation 3.7) that can be solved using dynamic programming. This approach serves to get an insight into the characteristics that make MPP a non-trivial problem and gives a result in terms of number of evaluations that such an algorithm needs to do before finding the value of the optimal contingent maintenance plan, and therefore provides an intuition of its scalability. The dynamic programming approach is similar to the well-known value iteration procedure for MDPs by Bellman [28] (Section 2.2). As with value iteration, the recursive formula only computes the expected reward of a contingent plan and does not return the contingent plan. However, the latter can be easily constructed using the former by concurrently maintaining the joint activity that maximises the expected reward during search. This step is trivially implemented and is not included in the algorithm presented here.

Essentially, the dynamic programming algorithm sums over the expected value of all possible histories $H^h \in \mathcal{H}^h$ within the planning period length $h$. In every step of the recursion, the current history is extended by all available joint activities and associated outcomes, written as $H^{t-1} \oplus \langle \vec{\mathrm{a}}, \vec{o} \rangle$, and the expected value of every such a possible extension is computed recursively until the planning horizon has been reached. The set of available joint activities is denoted by $\boldsymbol{\mathcal{A}}|H^{t-1}$, which can be interpreted as the set of joint activities that are still available given execution history $H^{t-1}$, and as such excludes activities that have previously started or cannot be completed anymore in the remaining time. As a result of each recursion, the algorithm returns the maximum expected value that can be obtained *from the current history* forward, i.e. the value of the history extended with the joint activity $\vec{\mathrm{a}}$ that maximises the sum of expected reward over all its possible outcomes $\boldsymbol{O}(\vec{\mathrm{a}})$. Eventually, all recursions will return a value and thus, at the root of the computation, the maximum expected reward that can be

obtained is known. This is captured by the recursive formulation:

$$
\begin{aligned}
V^{\mathcal{P}*} &= V(\emptyset, 0) \\
V(H^{t-1}, t) &= \begin{cases} 0, & \text{if } t = h \\ \max\limits_{\vec{\mathsf{a}} \in \boldsymbol{\mathcal{A}} | H^{t-1}} \sum\limits_{\vec{o} \in \boldsymbol{O}(\vec{\mathsf{a}})} Pr(\vec{o}) \cdot V'(H^{t-1} \oplus \langle \vec{\mathsf{a}}, \vec{o} \rangle, \vec{\mathsf{a}}, t), & \text{otherwise} \end{cases} \\
&\text{s.t.} \\
V'(H^t, \vec{\mathsf{a}}, t) &= \sum_{\mathsf{a}_k \in \vec{\mathsf{a}}} w_k + c(H^t_{\mathcal{A}}(t), t) + \ell(H^t_{\mathcal{A}}(t), t) + V(H^t, t+1)
\end{aligned}
\tag{3.8}
$$

The formula includes the base case, $V^{\mathcal{P}^*} = V(\emptyset, 0)$, and the recursive steps $V$ and $V'$. $V$ returns the maximum of all possible joint activity extensions by summing the expected reward over all possible joint outcomes (recall that joint activities may include no-ops at all times). The actual value evaluation is performed in $V'$, that computes the immediate value of the joint activity that is being evaluated and adds to it the expected future reward of this extension ($V(H^t, t+1)$). Note that the revenue $w_k$ is summed over the joint activity, i.e. the activities that start now, so that it is awarded only once. The immediate costs, however, are computed over the set of activities that are concurrently active at time step $t$, given by $H^t_{\mathcal{A}}(t)$, because activities that started in an earlier time step might still be in progress (e.g. as in $t_3$ of Figure 3.1b where $\mathsf{b}_2$ is started while $\mathsf{a}_1$ is being executed).

From Equation 3.8 it is possible to derive a first insight into the runtime required to solve the maintenance planning problem based upon the number of evaluations that need to be performed before the optimal expected reward is found. In the worst-case every agent has $\alpha = \max_{i \in \boldsymbol{N}} |\mathcal{A}_i|$ activities and hence there can be $\alpha^n$ joint activities that need to be evaluated. As activities may have two possible outcomes, the number of evaluations – and hence the number of recursions – of the dynamic programming algorithm is bounded by $O(2^{\alpha^n} \cdot h)$.[31]

## 3.3 Maintenance Planning as a Markov Decision Process

Arguably the most used framework for stochastic planning problems is that of Markov Decision Processes (MDPs, see Chapter 2). On the one hand, modelling a planning problem as an MDP is typically conceptually very simple while on the other hand very complex problems can be modelled through its states, actions and transitions. This has made the MDP model a very attractive choice for stochastic planning problems, which is confirmed by the enormous body of literature considering MDPs. As a consequence of

[31] *A tighter bound can be made on the number of evaluations by considering also the possibility that all activities have been completed. Additionally, in the case that all activities must be completed once, it can be tightened even more by checking if there is enough time left to complete all remaining activities. Nonetheless, even if both are included still a doubly-exponential number of evaluations is necessary.*

the amount of attention this model has received over a long period of time – MDP was already introduced in 1957 [28] – there are highly-optimised solvers readily available for problems formulated as MDPs. In this section, an MDP encoding of the maintenance planning problem is discussed so that any of existing state-of-the-art MDP techniques can be used to solve it.

First the single-agent MDP model is discussed from which the multi-agent MDP (MMDP) can be trivially constructed. Unless using specialised algorithms tailored to MMDPs (e.g. Chapter 4), solving an MMDP can be done by 'flattening' it into a single-agent *joint MDP* that models joint states, actions, probabilities and rewards as a single-agent decision-making process, which in its turn can be solved by any existing MDP technique. The way in which the MMDP is flattened does however impact the solving efficiency and hence care must be taken in encoding the problem, as is discussed in this section and demonstrated through empirical evaluation in Section 3.5. First, however, the construction of a single-agent MDP is described. Later this will be extended to the multi-agent setting.



The MDP $M_i$ for a single-agent is defined by the tuple $\langle S_i, A_i, P_i, R_i \rangle$. The state of an agent only contains the current time and information regarding the completion of its maintenance activities. For each of its activities two variables are required to incorporate the necessary knowledge: only its start and end time need to be recorded. Alternatively, the combination of start time and the delay realisation of the activity convey the same information and can be used in the encoding at the same cost in terms of state space size but, as will become clear later, it is more convenient to store the end time of an activity.

Once an agent decides to perform an activity $\mathtt{a}_k$, the corresponding start time is set to the current time; the end time is set probabilistically to $d_k$ or $d_k + d'_k$, based on delay probability $p_k$. The start time of an activity can take on at most $h$ unique values in the worst case and the (correlated) end time can take on two, resulting in an upper bound of $2h$ unique value assignments for the activity time variables. As this is true for every activity and each of these assignments can, in the worst case, occur in every time step, the state space of a single agent is bounded by $h \times \prod_{\mathtt{a} \in \mathcal{A}} 2h = O(2h^{|\mathcal{A}|})$ states. When taking into account that activities can be performed only once, this bound can be tightened to $O(h \times \binom{h}{h-d})$ such that $d = \sum_{\mathtt{a}_k \in \mathcal{A}} d_k$ is the minimally required time for all activities.

**Remark 3.6 Actions and activities**

In the thesis both terms are used in the context of actions but they are used consistently to denote distinct things. The term action only denotes actions of an MDP, i.e. elements of the set $A$. Activities refer to high-level, complex tasks that can be encoded in an MDP through one or more actions. They can always be discerned by their notation: activities $\mathtt{a} \in \mathcal{A}$ are denoted using a monotype and caligraphic font, while MDP actions $a \in A$ are in regular fonts.

The action set of a single-agent MDP contains exactly one $\mathtt{start}_k$ action for each of its maintenance activities $\mathtt{a}_k \in \mathcal{A}_i$. This action signals the start of the activity and sets the corresponding start time variable to the current time. Depending on its stochastic outcome, the end time for its activity is set to either the current time plus

its regular duration $d_k$ (not delayed) or to the current time plus both its regular as well as its delay duration $d'_k$ (delayed). The probability of these outcomes is defined by its the activities delay probability $p_l$, as before. To prevent activities from being performed more than once or when they cannot be completed anymore, the agent is given a reward of negative infinity when its start time is already set or its end time may be past the planning horizon, respectively. The action set also includes the $\texttt{noop}_i$ action that allows the agent to wait one time step without doing any maintenance. In summary, the MDP action set of each agent consists of $|\mathcal{A}_i| + 1$ actions.

The transition function $P_i$ of the agent MDP includes basically three high-level transitions, which can be easily specified in a factored MDP. When an agent performs a start action $\texttt{start}_k$, a transition is made to the state where the start time of $\mathtt{a}_k$ is set to the current state time $t$ and the new state time is set to $t + d_k$. From this state only a delay action can be taken that result in a probabilistic transition to either the state where the activity is completed and the end time is set to $t$, or the state with the current time and end time both set to $t + d'$, depending on delay probability $p$. The last transition is that of $\texttt{noop}_i$, in this case a simple state transition occurs to the state with $t + 1$. In all the transitions described it is assumed that only the explicitly mentioned variables change value, all other state variables will have the same value in the new state that results from the transition.

Lastly, the reward function of the agent MDP includes the revenue and maintenance costs of the agent. Upon starting an activity $\mathtt{a}_k$ the agent receives a reward

$$r_i(\mathtt{a}_k t_s^k, t_e^k) = w_k - \sum_{t=t_s^k}^{t_e^k} \big(c_i(\mathtt{a}_k, t) - \ell(\mathtt{a}_k, t)\big) \tag{3.9}$$

where $t_s^k$ and $t_e^k$ are respectively the start and end time variables of the activity. This reward is typically assigned to the agent after it becomes known whether or not his activity is delayed, i.e. for a full transition $(s, a, s')$, or at the end using Equation 3.7 when all start and end times are known. All summarised, the planning problem of a single agent is modelled as:

**Definition 3.7 Single-agent MDP formulation of the maintenance planning problem**

A single-agent instance $M = \langle \{i\}, \{\mathcal{A}_i\}, \{c_i\}, \ell, T \rangle$ of the MAINTENANCE PLANNING PROBLEM is represented through a *Markov Decision Process* such that the planning problem of agent $i$ is modelled by an MDP $M_i = \langle S_i, A_i, P_i, R_i \rangle$ where:

- $S_i$ is the set of states storing the current time and for each of its activities $\mathtt{a}_k \in \mathcal{A}_i$ the start and end time of that activity,
- $A_i$ the action with for each activity $\mathtt{a}_k \in \mathcal{A}_i$ an action $\texttt{start}_k$ and one no-op action $\texttt{noop}_i$,

- $P_i$ the state transition probability function defined such that for every possible transition $(s, a, \hat{s}) \in S_i \times A_i \times S_i$:

$$P_i(s,a,\hat{s}) = \begin{cases} 0, & t(\hat{s}) = h \\ p_k, & a = \texttt{start}_k,\ \mathtt{a}_k \notin s,\ t(\hat{s}) = t(s) + d_k + d'_k \text{ and } o_k^- \in \hat{s} \\ 1 - p_k, & a = \texttt{start}_k,\ \mathtt{a}_k \notin s,\ t(\hat{s}) = t(s) + d_k \text{ and } o_k^+ \in \hat{s} \\ 1, & a = \texttt{noop}_i \text{ and } t(\hat{s}) = t(s) + 1 \\ 0, & \text{otherwise} \end{cases}$$

  such that for a state $s \in S_i$, $t(s)$ represents the time in that state, $\mathtt{a}_k \notin s$ denotes that activity $\mathtt{a}_k$ has not yet been performed in the state, and $o_k^-, o^+ \in s$ respectively express that activity $\mathtt{a}_k$ has or has not delayed in the state,

- $R_i$ the reward function such that for every transition $(s, a, \hat{s})$ the reward is:

$$R_i(s,a,\hat{s}) = \begin{cases} r_i(\mathtt{a}_k, t(s), t(\hat{s})), & \text{if } a = \texttt{start}_k \\ 0, & \text{otherwise,} \end{cases}$$

  where $r_i$ is defined as in Equation 3.9. Notice that the (delayed) activity duration is implicit in the transition from $s$ to $\hat{s}$.

**Example 3.8** Example Maintenance MDP

Figure 3.5 shows a graph that illustrates the states and actions of a single-agent MDP for agent $A$ of Example 3.5. There are three types of nodes in this graph, intermediate state nodes (single circles), terminal state nodes (double circle) and chance nodes (black triangles), and they are grouped column-wise per time step. For compactness start actions $\texttt{start}_k$ are labelled with their associated activity $\mathtt{a}_k$ and no-ops by $\circ_A$.

The state nodes correspond to all the unique states in the state space of the agent. In this illustration, they have been labelled with the start and end times of activities that are completed. A state is terminal, i.e. the planning problem has been solved completely, if there are no more activities to perform. Chance nodes are required for actions with stochastic outcomes (start $\mathtt{a}_1$ in this example), depending on the realisation of the delay one of the connected state nodes is reached. For instance, if activity $\mathtt{a}_1$ is chosen in the initial state the top state (at time t3) after the chance node is reached if it does not delay – the outcome is $o^+_{\mathtt{a}_1}$ – and the bottom state (at time t4) otherwise. Notice that one additional time step td is required in this graph to allow activities to end in the last time slot, e.g. when starting $\mathtt{a}_2$ in time t5.

In this example, all transition probabilities equal one except those including a chance node. Recall from Example 3.5 that activity $\mathtt{a}_1$ has a delay probability of $0.3$ and therefore the transition probability for every transition of the form $(s, \texttt{start}_1, s')$, where activity $\mathtt{a}_1$ is not delayed in the new state $s'$, is 0.7. Similarly, when $\mathtt{a}_1$ is delayed in $s'$ the transition probability $P(s, \texttt{start}_1, s')$ is $= 0.3$. Finally, although it is possible to perform no-ops in terminal states where the time is less than $t_{done}$, they do not contribute to the reward in maintenance planning and such transitions are therefore omitted.

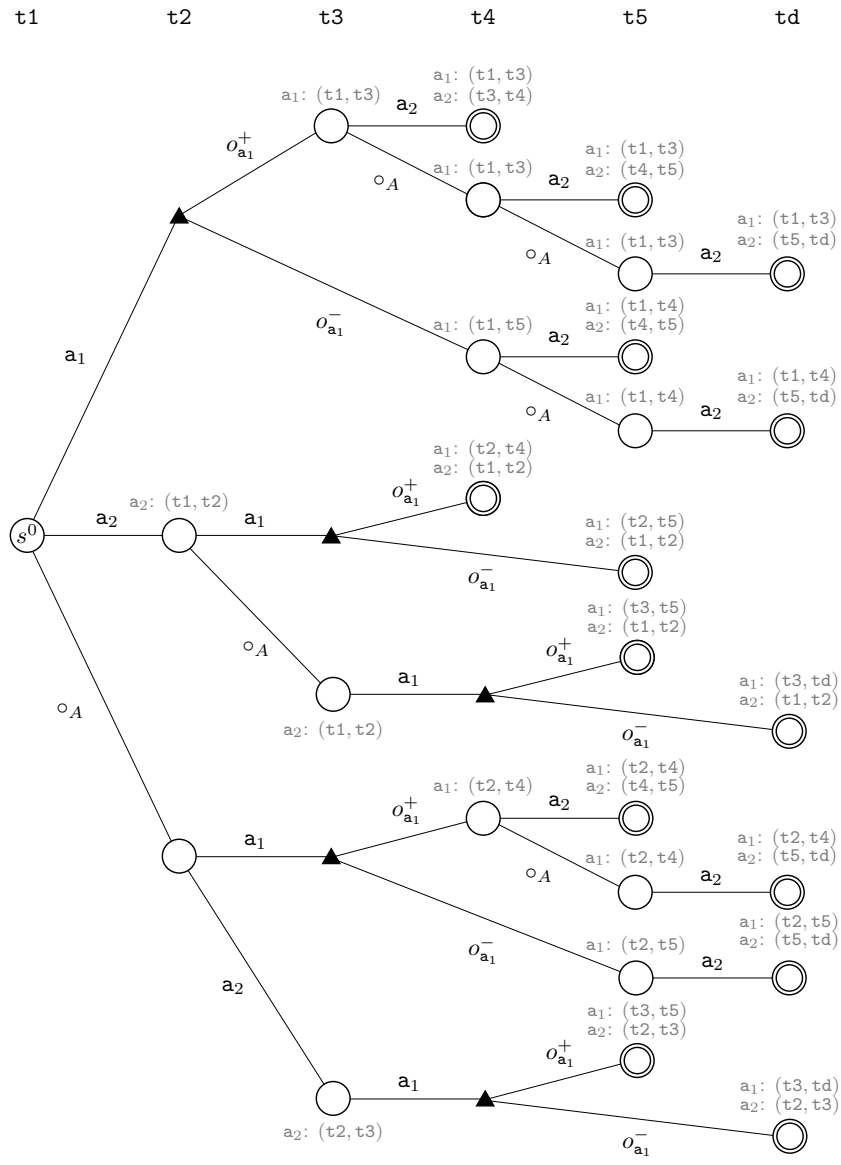



**Figure 3.5** *Graph illustrating a single-agent MDP, states are shown as circles and the actions as edges. Stochastic transitions are depicted using chance nodes (black triangles) and depend on the outcome of the action. Terminal states are shown as double circles.*

Reward functions are illustrated for a part of the graph in Figure 3.6. The reward of each transition $(s, a, s')$ is displayed on top of state $s'$. For instance, the transition reward from performing activity $\mathsf{a}_1$ in the initial state leading to state $s'$ in which $\mathsf{a}_1$ is not delayed, is given by its revenue minus its maintenance costs, or $w_1 - \sum_{t=\mathsf{t1}}^{\mathsf{t2}} c(\mathsf{a}_1, t)$, which is equal to

109.[32] If activity $\mathtt{a}_2$ is started afterwards, the transition reward $R(s, \mathtt{start}_2, s') = 40$ and the total reward obtained for this execution sequence from initial state to terminal state is 149.

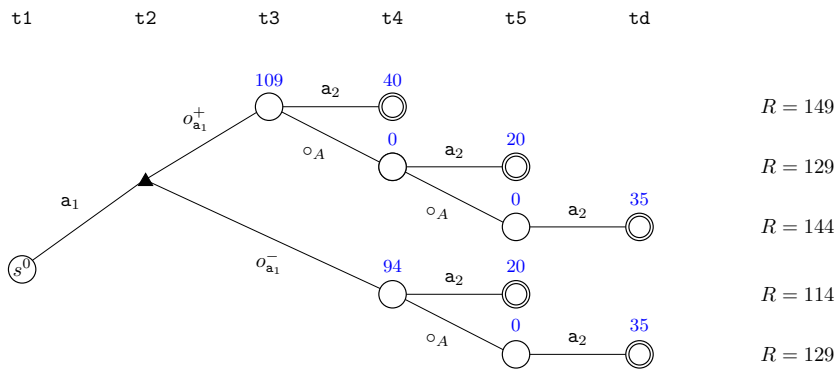



**Figure 3.6** *Illustration of transition rewards for a part of the MDP of Figure 3.5. Every state is labelled with the transition reward for transferring to it and the total reward obtained in every execution sequence is shown on the right.*

Using single-agent MDPs it is possible to obtain a joint policy $\boldsymbol{\pi} = \{\pi_i\}_{i \in \boldsymbol{N}}$ by solving them individually and combining the individually optimal policies. But although each policy $\pi_i$ is optimal for agent $i$, this approach completely ignores dependencies between agents and therefore the resulting joint policy might be far from optimal. Agents interact on the network level, which is not considered when agents develop their policies individually. When network cost are relatively low compared to the individual revenues and maintenance cost of every agent, such an approach may be a feasible approximation that leads to acceptable results, otherwise a coordination approach is required that develops contingent plans for all agents simultaneously. A solution is to model the problem as a multi-agent MDP that includes the reward dependency:

**Definition 3.9 maintenance planning problem as multi-agent MDP**

An instance $\langle \boldsymbol{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{c}, \ell, T \rangle$ of the MAINTENANCE PLANNING PROBLEM can be modelled as a multi-agent MDP $M = \langle \boldsymbol{N}, S, \boldsymbol{A}, P, R \rangle$ by constructing for each agent the single-agent MDP $M_i$ according to Definition 3.7 and combining them such that

- $S = \times_{i \in n} S_i$ is the set of joint states,
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{N}}$ the set of joint actions,
- $P(s, \vec{a}, \hat{s}) = \prod_{i \in \boldsymbol{N}} P_i(s, \vec{a}, \hat{s})$ is the joint transition probability for every transition $(s, a, \hat{s}) \in S \times \boldsymbol{A} \times S$,

[32] *Here no network costs occur because of the binary model used in the example. In general these can also be defined for a single activity.*

- $R(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} R_i(s, a_i, \hat{s}) + \ell(s, \vec{a}, s)$ is the joint reward function that now separates the 'local' reward from the inter-agent ttl penalties. The local reward is (re)defined as

$$R_i(s, a_i, \hat{s}) = \begin{cases} w_k - \sum_{t=t(s,i)}^{t(\hat{s},i)} c_i(\mathtt{a}_k, t), & \text{if } a_i = \mathtt{start}_k \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

  in which $t(s, i)$ is the value of the time variable of agent $i$ encoded in state $s$. This reward thus no longer includes the $\ell$ component. Furthermore it depends on its own action $a_i \in \vec{a}$, while the joint reward $\ell$ depends on the joint transition.

Note that the function $\ell(s, \vec{a}, \hat{s})$ that computes the network costs, now over the shared state of all agents, can be implemented in multiple ways. The simplest way is to sum over all network costs as defined in Equation 3.6 only in every transition that leads to a 'terminal' state, i.e. one in which the time in the state of each agent is equal to the planning horizon $h$:

$$\ell(s, \vec{a}, \hat{s}) = \begin{cases} \sum_{t \in T} \ell(H^h_{\mathcal{A}}(t), t), & \forall i \in \boldsymbol{N} : t(\hat{s}, i) = h \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

and observe that $H^h_{\mathcal{A}}$ can be derived from the activity start and end variables in the terminal state $s$.

For MDP solvers that use backwards iterations such as SPUDD [116], this formulation is likely beneficial to the policy search because the network costs are computed immediately and value optimisation starts from there. Solvers that start from an initial state could potentially do a lot of unnecessary work before concluding that the network costs, not computed until hitting a terminal state, make the policy infeasible.

Although the above formulation enables the use of MDP solvers, MMDPs such as this one can only be solved by specialised multi-agent MDP methods such as the one presented in the next chapter. To make use of any of the many generic MDP solvers, the MMDP must be 'flattened' into a single-agent joint MDP representation of the MMDP but much of the solving efficiency relies on the way the MMDP is flattened. To this end, the next sections will discuss several ways to obtain a joint MDP, starting with a straightforward but inefficient enumerative approach working up to encodings that can greatly reduce MDP size and, consequentially, solving time.

### 3.3.1 Enumerative Joint MDP

A first naive but illustrative approach to model the MMDP as a joint MDP is obtained through enumerating all combinations of states and actions of the agent MDPs $M_i$. Such an enumeration however requires a modification of the individual MDPs. While previously it was no problem to have actions that can take multiple time steps to complete in the single-agent case, when dealing with multiple agents this is no longer trivial. Either all agents must share the same time variable, requiring a completely specified joint action in each time step (as in Figure 3.7), or each agent must maintain

its own time variable, leading to very complex joint states and an even larger state space (see below). The first approach is taken in the joint MDP encoding discussed here, for the main reason that later more efficient MDP encodings are possible only when sharing a global time variable.

The state space of the joint MDP can be obtained by making their time variable part of a global set of states $S_g$ (which hence has exactly $h$ states) and effectively enumerating all combinations of individual agent states and the global state, effectively 'flattening' the state space. This leads to the state space $S = S_g \times S_1 \times \ldots \times S_n$. In the maintenance planning problem, $S_g$ only contains states that correspond to the current global time, but in general it can represent various pieces of shared knowledge. Recall from the previous section that the size of the state space of each agent $i$ is in the order of $O(2h^{|\mathcal{A}_i|})$, thus the size of the joint state space is worst-case bounded by $\prod_{i \in \boldsymbol{N}} 2h^{|\mathcal{A}_i|} \times |S_g|$, which is of order $O(2h^{\alpha^n})$ for $\alpha = \max_{i \in \boldsymbol{N}} |\mathcal{A}_i|$, and therefore exponential in both the activity set size and the number of agents.

Defining the joint action set is more cumbersome in the multi-agent setting. Agents can start activities jointly, but they may be of various durations and therefore transition to different times, which is not possible when using a shared global time. In order to overcome this problem, maintenance operations are encoded using sequences of unit time actions as $\texttt{start}_k$, $\texttt{cont}_k^1, \texttt{cont}_k^2, \ldots$ until all $\hat{d}(o_k)$ steps have been performed. The start action $\texttt{start}_k$, if included in a joint action, sets the start and end time of activity $\mathrm{a}_k$ as before. In every following state with a global time $t$ less than the end time of the activity, the continue action $\texttt{cont}_k$ must be executed by its associated agent, otherwise again a penalty of negative infinite is incurred. Vice versa, no agent can perform a continue action if no activity is being performed (all end time values are lower than the global timer). The agent will always perform $\hat{d}(o_k) - 1$ continue operations sequentially after the start action to match the regular duration and possible delay duration, depending on the outcome $o_k$. Note that the continue action $\texttt{cont}_k$ is action specific because of activity rewards (discussed later). An example of activity decomposition in illustrated in Figure 3.7.

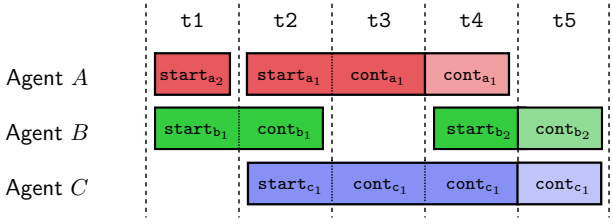



**Figure 3.7** *Decomposition of activities into unit-time actions shown for the example history of Figure 3.3.*

Summarising the previous discussion, the action set $A_i$ of each agent contains for each activity $\mathrm{a}_k \in \mathcal{A}$ a start action $\texttt{start}_k$ and continue action $\texttt{cont}_k$ and again a no-op action $\texttt{noop}_i$. The set of all available joint actions is defined by the Cartesian

product of all individual action sets: $A = \times_{i \in n} A_i$. Let $\alpha = \max_{i \in \boldsymbol{N}} |\mathcal{A}_i|$, then the joint action set size is bounded by $(2\alpha + 1)^n = \Omega(2^n \cdot \alpha^n)$.

The transition probabilities can be easily defined when a factored MDP is assumed as before. Transitions are expressed in terms of their impact on state features that is caused by each joint action. In this encoding, each joint action always increases the state time variable by one. Other features change depending on the actions in the joint action. Every start action sets the start time and end time of the activity it corresponds to, other actions do not cause a change in state features. When an activity $\mathtt{a}_k$ can delay, its start action can have two outcomes that result in a different end time for the activity in the resulting state. Each of these start actions necessarily specifies two possible transitions and therefore a joint action with $x$ start actions has $2^x$ result states, each with a unique assignment of end times. The probability of such a transition is equal to the product of individual delay probabilities. For all other transitions, without start actions, the probability is always 1 (which is of course equal to $2^0$).

Rewards are specified in a similar way as the transition probabilities in that they can be easily expressed over joint actions. As the joint action exactly specifies the set $\mathcal{A}^t$ of activities that are being performed at time $t$, the *cost* of each joint action is given by the sum of individual cost functions $c_i(\mathtt{a}_i, t)$ and the network reward $\ell(\mathcal{A}^t, t)$ of Equation 3.6. The current activity $\mathtt{a}_i$ of agent $i$ can be determined from the joint action: an agent is performing activity $\mathtt{a}_k \in \mathcal{A}_i$ if the joint action contains any of the actions $\mathtt{start}_k$ or $\mathtt{cont}_k$ (notice that this requires the continue action to be unique for the activity). The cost of a no-op action is assumed to be zero, as before. The revenue of an activity should be awarded once, which is done only when a start action is performed. Hence the revenue of a joint action is given by the summing over all activity revenues $w_k$ of every activity $\mathtt{a}_k$ that has a start action $\mathtt{start}_k$ in the joint action. As in the joint MDP joint action decisions are made through single actions, the reward of every single action in the joint MDP is set exactly to that of the joint action as just described.

**Remark 3.10 Complexity of the network cost model**

Although the reward can be defined as just discussed, the choice of reward function can greatly influence computational complexity. In general the MDP model does not pose any restrictions on the type of reward function that is used. Encoding reward functions can nevertheless be far from trivial and can have a significant impact on the computational effort required to solve the MDP. Encoding a non-linear reward may require an exponential number of data nodes to represent or one massive value table of exponential size and hence can make an otherwise easy MDP intractable to solve. For this reason network costs will be restricted to only binary, linear relations, e.g. combinations of activities $\mathtt{a}_i$ and $\mathtt{a}_j$ for which there is at least one time step $t$ where $\ell(\langle \mathtt{a}_i, \mathtt{a}_j \rangle, t) \neq \ell(\mathtt{a}_i, t) + \ell(\mathtt{a}_j, t)$; a restriction that for instance the factor-based models of Example 3.5 satisfies. The encoding of binary relations is polynomial in size and as a result also tractable to compute.

### 3.3.2 Avoiding Exponentially Sized Action Spaces

In the previous section a first attempt at constructing the joint MDP for maintenance planning problems was proposed. However this leads to an MDP that has both an

exponentially sized state space as well as action set. Most current-day MDP solvers are well-tailored to handle exponentially sized state spaces but cannot deal adequately with exponentially sized action sets. To overcome this problem, two alternative encodings are presented in this section that both result in an action set of polynomial size. Both encodings rely on moving part of the information currently encoded as part of the actions into the state space using a two-state activity selection process: first, every agent determines the activity it will start or continue and, when all agents have determined their activity, the joint activity is executed. Although both encodings use this two-stage idea, they differ in how activity selection is performed.

The main problem of the enumerative encoding is that in each time step a joint activity must be decided on for *all agents concurrently* and the number of such joint activities is exponential in the number of agents. Intuitively it seems better to decouple the activity selection process of the agents into $n$ separate decisions and, after these decisions have been made, jointly execute these decisions for the current time step. If the activity selection per agent can be done more compactly than exhaustive enumeration, the action set size of the resulting MDP can be substantially smaller.



To achieve such a two-stage encoding, it is first necessary to enforce an agent-by-agent action selection. Moreover, as the joint activity is not 'executed' until for all agents the activity to perform is selected, additional bookkeeping is required to keep track of activities chosen. The ordering of agent action selection is enforced using a global state variable that keeps track of the agent that should decide on its activity, from this point on referred to as the *agent token*. In order to make sure that actions can only be started when its associated agent has the agent token, a penalty of negative infinity is given if it does not have the token. To keep track of the chosen activity of each agent, a current activity variable is appended to each of the local state spaces $S_i$. The two-stage approach is illustrated in Figure 3.8: agents take the token, decide on their activity and pass it on. Eventually the joint activity is 'executed', meaning that the effects of the joint activity are applied on the state space, e.g. setting a start and end time of an activity. The execute action can thus have $2^x$ different (stochastic) outcomes, where $x$ is the number of activities that can delay and starting now ($\texttt{start}_k$).

Now, the key to improving the efficiency of the MDP encoding lies with the activity-selection procedure. Two approaches are introduced that can effectively reduce the action set size through smarter encoding: agent chaining and activity grouping.[33] Agent chaining is based upon the observation that agents sequentially decide on their activity to start or continue, and therefore it is not necessary to enumerate joint activities. Instead, for every agent only $|\mathcal{A}_i|$ start actions have to be generated in order to start an activity. Continuation of the current activity can be done using an agent-specific continue action $\texttt{cont}_i$. The resulting encoding resembles Figure 3.8, albeit that each $\texttt{a}_x$ (analogously for b and c of course) is replaced by a start action $\texttt{start}_x$ and every agent $i$ has additional actions $\texttt{cont}_i$ and $\texttt{noop}_i$. In addition to setting the start time, the start actions sets the current activity variable of the agent. When an action is selected for all agents, the execute action exec applies the effects of the chosen activities,

[33] *In work preceding this thesis, Scharpff et al. [228] termed these encodings respectively action chaining and activity chaining.*

increments the state time and resets the agent token to the first agent. If an activity takes multiple time steps and it is not yet completed in the next time step, its agent can only select the continue action. The continue action only passes the token to the next agent; its current activity variable remaining unchanged in the state.

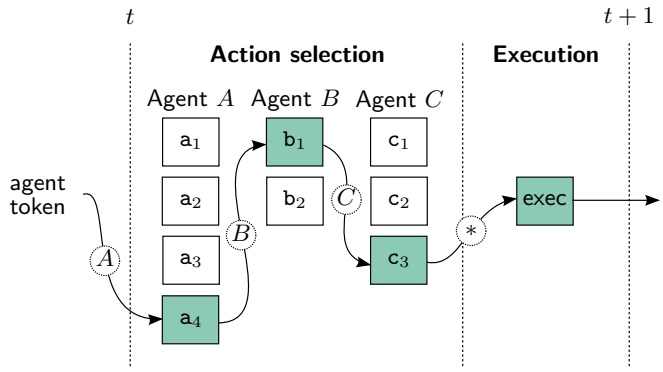



**Figure 3.8** *Illustration of the two-stage encoding approach using the agent chain encoding on the example problem. Every time an agent has the token it determines the activity to perform and then passes the token to the next agent. When all agents have determined their activity, the token is set to done (*) and the joint activity* $\langle \mathtt{a}_4, \mathtt{b}_1, \mathtt{c}_3 \rangle$ *is executed (highlighted in the figure). Both stages are contained within a single time step.*

The size of the action set that results from the agent chaining encoding is much smaller than that of the aforementioned enumerative encoding. Each agent has $|\mathcal{A}_i|$ start actions, one no-op action and one continue action, and there is one global execute action. The number of actions is therefore given by $\sum_{i \in \boldsymbol{N}} (|\mathcal{A}_i|+2)+1 = |\mathcal{A}|+2n+1$. As typically the number of activities is much larger than the number of agents, the action set size is of order $O(|\mathcal{A}|)$. Using only $n + 1$ additional variables – the current activity of each agent and global agent token – the action set size can be reduced from exponential to polynomial in the number of activities, at the cost of a linear increase in the (exponential) state space size. Nevertheless, it is possible to achieve an even compacter encoding by grouping activities.

In agent chaining, each agent sequentially decides on starting or continuing an activity and to this end $|\mathcal{A}|$ start actions need to be enumerated. But, since there is already an agent token present in the encoding, activities can be grouped by index into a single start action that sets the current activity based on the value of the agent token. Instead of generating a start action for every activity, only $\max_{i \in \boldsymbol{N}} |\mathcal{A}_i|$ start actions are required to cover all activities. Formally, activity groups are defined as $\mathcal{A}^j = \{\mathtt{a} \mid \exists i \colon \mathcal{A}_i[j] = \mathtt{a}\}$ for every $j = 1, 2, \ldots \max_{i \in \boldsymbol{N}} |\mathcal{A}_i|$. For all of the activity groups $\mathcal{A}^j$, a start action $\mathtt{start}^j$ is encoded that sets the current activity for agent $i$ to $\mathcal{A}_i[j]$ when the agent token value is $i$. After setting the current activity, the agent token is passed on as before.

An example of activity grouping is shown in Figure 3.9 for the same problem as before. To better relate with agent chaining, the activities are shown as squares with dashed borders but they are not the actual actions in this encoding. Observe that

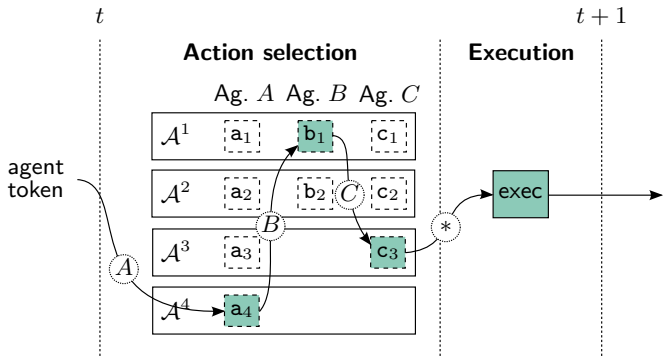



**Figure 3.9** *Illustration of the activity grouping encoding. In this example the same joint activity* $\langle \mathtt{a}_4, \mathtt{b}_1, \mathtt{c}_3 \rangle$ *as before is executed (again highlighted) although through different actions.*



the activity group encoding results in only 5 actions whereas the previous needed 10. In general, because of the construction of the activity groups, this encoding requires $\max_{i \in \boldsymbol{N}} |\mathcal{A}_i| + 2$ actions where the two extra actions are a no-op group and the execute action. Therefore the action set size is of order $O(\max_{i \in \boldsymbol{N}} |\mathcal{A}_i|)$, which is typically much smaller than the previous $O(|\mathcal{A}|)$ achieved by agent chaining. Also interesting to note is that the activity group encoding completely decouples the action set size from the number of agents in the problem. Of course, increasing the number of agents is not without cost. Although the number of actions does not increase with the number of agents, the complexity of each action does as more activities are grouped into it. Still, existing solvers scale better in terms of runtime due to the very compact action space compared to the enumerative or even agent-chaining encodings.

## 3.4 Approximation of Maintenance Plans

From both the dynamic programming formulation of Section 3.2 and the more involved, but exponential state MDP formulations of Section 3.3.2, it is clear that finding an optimal maintenance plan is a computationally-demanding task. Indeed the experimental evaluation of 3.5 later in this chapter clearly illustrates that finding an optimal solution for instances of all but the smallest sizes is quickly considered intractable. For some applications, finding an optimal maintenance plan is possible because either the number of agents and actions is manageable, or simply because there is enough time available to develop the plan. In other settings, for example the decision support setting of Chapter 5 where many problem instances need to be solved, it may be acceptable to trade some of the solution quality for a (substantial) decrease in runtime.

This section proposes an approximation approach for MPP, or multi-agent MDPs in general, based upon `Monte-Carlo tree search (MCTS)` (see e.g. the work by Coulom [70]). In `MCTS` the search space is modelled as a tree, that is gradually being built by the algorithm. In the case of MMDPs, the nodes in this tree correspond to the states $S$ and the edges to the transitions $P$ that result from taking one of the actions $A$,

similar to the graph of Figure 3.5. In each iteration of the algorithm, a node will be selected, a new action will be tried and the corresponding transitions will be explored to construct a new part of the search tree. The node representing the state will then store the expected rewards as a result of function $R$ and keep track of the action that in expectation maximises that reward. Observe that if this process would continue indefinitely, the full policy search tree would be constructed and the optimal policy is easily extracted from the tree. The key idea however is that at any given time during the construction of the tree, the `MCTS` algorithm can return an approximation of the optimal policy by returning the best transitions for every node *known so far*. In other words, it functions as an anytime-approximation algorithm that with every iteration gets closer to an optimal solution. The generic procedure is shown in Algorithm 3.11.

**Algorithm 3.11** `Generic Monte-Carlo Tree Search`

**Require:** Search tree $\tau$, time limit $t_{lim}$

**function** $\mathrm{MCTS}(\tau,\, t_{lim})$
  **while** `time` $< t_{lim}$ and $\exists n, a \colon \tilde{V}(n,a) = unknown$ **do**
    $n \leftarrow$ `SelectNode`$(\tau)$ ▷ Determines next node to evaluate
    $a \leftarrow$ `SelectAction`$(\tau, n)$ ▷ Select the action to take from node $n$
    $n' \leftarrow$ `ExpandSearchTree`$(\tau, n, a)$ ▷ Expand $n$ with $a$, leading to new node $n'$
    $H^h \leftarrow$ `SimulateRollOut`$(n')$ ▷ Simulate rollout from node $n'$
    $\tilde{V}(n, a, n') \leftarrow$ `BackPropagateValues`$(\tau, H^h)$ ▷ Estimate values of rollout $H^h$
  **end while**
  **return** $T$
**end function**

Implementations of the `MCTS` algorithm vary in how each of the steps `SelectNode`, `SelectAction`, `ExpandSearchTree`, `SimulateRollOut` and `BackPropagateValues` are performed. The first two steps are heuristic functions to determine the tree node that is evaluated next and the action that is taken from that node. The step `ExpandSearchTree` uses this node $n$ and action $a$ to expand the search tree by a single new node $n'$ that represents a possible outcome of taking the action $a$ from node $n$. Unless $n'$ is already a terminal state, the rollout step `SimulateRollOut` samples *one* possible execution sequence from node $n'$ until the planning horizon is reached by realising outcomes, yielding one possible full-length history $H^h$. Finally, the backwards-propagation step `BackPropagateValues` sets a value estimate (as only the values in sampled areas are known) in all of the nodes of the current search tree that are present in the rollout $H^h$ using a predetermined heuristic. This is an important difference with for example typical dynamic programming approaches that evaluate all possible futures to determine the value of a node; here only an estimate is made and it is gradually improved as more and more samples are taken from the search tree. In the limit, when the search tree has been fully expanded, these estimates will equal the exact rewards. The main benefits of performing `MCTS` over the exhaustive approach are that often a small number of samples can be enough to approximate these values closely and, as the estimated value improves with every new sample, it allows an any-time approximation of the policy.

**Example 3.12** `Monte-Carlo Tree Search`

In Figure 3.10a an example search tree for a one-agent, two-activity MDP instance is shown, in which 5 samples have been performed.

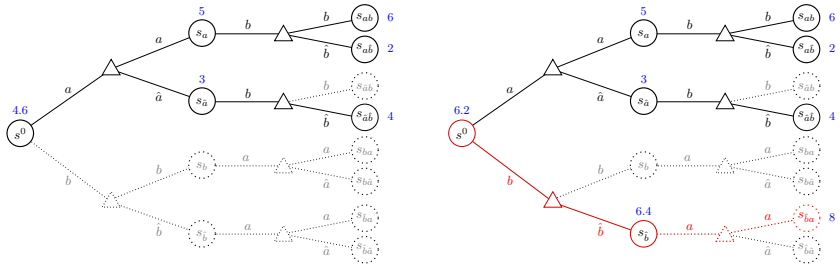



**Figure 3.10** *Illustration of Monte-Carlo tree search. The decision nodes are displayed as circles, with triangles illustrating chance nodes with multiple possible outcomes. The branches descending from a decision node are labelled by the action whereas the branches after each chance node are labelled by the possible not-delayed or delayed outcomes (e.g. $a$ and $\hat{a}$). The nodes and edges expanded during the tree search are displayed solid, the remaining unexplored parts are depicted dotted and greyed and (estimated) rewards are shown in blue. In (a) five samples have been taken, leading to the expansion of the tree by nodes $s_a$, $s_{\hat{a}}$, $s_{ab}$, $s_{a\hat{b}}$ and $s_{\hat{a}\hat{b}}$, labelled by their corresponding rewards. In (b) an example next evaluation is shown where action $b$ is selected at node $s^0$ and realised by outcome $\hat{b}$.*

In Figure 3.10a both outcomes for action $a$ from initial state $s_0$ (the root node) have been evaluated, leading to new nodes $s_a$ and $s_{\hat{a}}$, as well as both outcomes of action $b$ from state $s_a$ and only outcome $\hat{b}$ from state $s_{\hat{a}}$. The greyed parts of the search tree correspond to the remaining transitions in this simples, two-step example that have not yet been visited by the algorithm. Figure 3.10b shows a subsequent sample being performed (in red) from the initial state node $S_0$ when activity $b$ is chosen. Here its outcome is randomly determined as $\hat{b}$ and hence the search tree is expanded with a node for the newly found state $s_{\hat{b}}$. The parts in red that are dashed illustrate a simulated rollout that is being performed from node $s_{\hat{b}}$ to sample an outcome of the planning problem and get an estimate of its reward. The rewards that were found during this and previous simulations are shown as labels at the nodes. The new simulated rollout led to a first reward estimate for state $s_{\hat{b}}$ but also to an update of the estimate for the root node $s_0$ during the backward propagation as a new part of the search tree has been explored that was reachable from it. Note that in this simple example the value set in backwards propagation corresponds to the expected value over the currently known outcomes. In general, any backwards propagation function can be used but its choice is crucial to the theoretical guarantees and practical performance of the tree search algorithm and may vary per domain.

The node and action selection methods determine where and how sampling is performed in the search tree. As `MCTS` is typically used as an anytime approximation, it is preferable to expand the most interesting parts of the search space as soon as possible e.g. the areas that are likely to yield the most reward. Many heuristics exist for selection but

one particular successful variant of MCTS is the Upper Confidence bounds applied to Trees (UCT) algorithm due to Kocsis and Szepesvári [142]. UCT uses the UCB1 algorithm, originally proposed for stochastic bandit problems by Auer et al. [17], as its heuristic to guide MCTS. For a given Monte-Carlo search tree $\langle V, E \rangle$, UCB1 suggests expanding the node $n^*$ such that

$$n^* = \arg\max_{n \in V} \; B \cdot \sqrt{\frac{\log C^k(parent(n))}{C^k(n)}} + \tilde{R}^k(n) \tag{3.12}$$

in which $k$ is the number of rollouts performed so far, $B^k$ is a bias, set to the estimated expected policy value in UCT, $C^k(n)$ is the number of times node $n$ has been visited in $k$ trials and $\tilde{V}(n)$ is an expected reward estimation for node $n$ after the first $k$ trials.

In practice, UCB1 adequately balances between exploration infrequently visited states and exploiting states with high reward, making UCT an effective anytime MCTS algorithm. Indeed, Keller and Eyerich [133] show that their implementation of UCT for domain independent stochastic planning is very successful. Their PROST toolkit, with their own implementation of UCT, did very well in the International Probabilistic Planning Competition of 2011.[34] As a follow-up, Keller and Helmert [134] proposed an improved algorithm that combines UCT with heuristic and/or search (AO*), known as UCT*. This algorithm uses the MCTS approach and UCB1 heuristic of the former, but uses the greedy rollout simulation with a variable search depth of the latter to focus more on the short-term rewards following a decision. In this way, node expansion is still balanced between exploration and exploitation but, instead of performing complete depth-first rollouts until the planning horizon, UCT* performs rollouts of limited length. As a consequence the tree is built in a way that more resembles breadth-first search.

Still, UCT and UCT* suffer from two main drawbacks: it cannot determine whether the current state is solved optimally unless all of its child nodes have been sampled completely and its UCB1 heuristic does not offer any search space pruning [134]. Moreover, while MCTS offers a good anytime approximation, it is not guaranteed to find a fully specified policy for a stochastic planning problem. Consider the example of Figure 3.10b, if the algorithm would terminate at this point during the search, the policy will not contain any action for the state $s_b$ as none of its transitions have been expanded during the search. Such a policy that is not fully specified is referred to as a *partial policy*. Partial policies can still be usable in practice, although some additional planning must be performed when a non-terminal state is reached for which no transition is known. This can be acceptable in cases where the partial policy covers the states that are often encountered during execution or when there is limited time for offline planning but more time is available during execution. In addition to having to perform additional planning, it is not possible to directly compute the expected value of a partial policy because the values for unexpanded states and transitions are not known. When the goal is to find an approximate policy for the problem, but partial policies are unacceptable, the latter can be augmented to obtain a fully specified policy. For instance by appending do-nothing operations until the planning horizon is reached or setting actions randomly.

[34] *See the IPPC 2011 website at* `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/`.

## 3.5 Empirical Evaluation of MPP

This chapter introduces the MAINTENANCE PLANNING PROBLEM and first approaches to solve it, that is, to find a policy that maximises the sum of expected rewards. Section 3.3 proposes an MMDP model and various encoding techniques to transform it into a joint MDP that can solved by any existing MDP solver. Furthermore, Section 3.4 introduces an approximation for the MMDP model based upon Monte-Carlo tree search for settings where efficient policy computation is preferred over maximising the reward. In this section, both approaches are evaluated empirically on instances of MPP. In particular, several experiments are conducted that

1. compare the scaling of the three proposed MDP encodings – enumeration, agent chaining and activity grouping – in terms of runtime and memory usage,
2. investigate the influence of the activity set size and planning horizon on the scaling of the problem complexity, and
3. compare the approximation approach against the best performing optimal MDP method in terms of scalability in the size of the problem instance.



To find optimal policies for the MDP encoding the `Stochastic Planning Using Decision Diagrams` algorithm (SPUDD) of Hoey et al. [116] is used. This algorithm is a variant of the `Structured Policy Iteration` algorithm of Boutilier et al. [38] that uses algebraic decision diagrams [19] to efficiently represent rewards as graphs. This representation allows for a compact modelling of problem rewards as well as using state factors to group states that result in similar rewards. Therefore it is particularly well-equipped to deal with problems that have large state spaces and highly structured rewards, both true for MPP.[35] Finding approximate policies is performed using the `UCT*` algorithm of the `PROST` toolkit[36] by Keller and Eyerich [133] and run on the MMDP modelling of MPP instances. The instances themselves are encoded according to Definition 3.3 in the relational dynamic influence diagram language (RDDL) [226], a descriptive language for multi-agent planning problems. Finally, `UCT*` is an anytime-approximate algorithm and it does not provide any bounds on the error of the produced policy. In order to compare the quality of the approximations of the algorithm against the optimal policy values, a slightly modified version of the algorithm is used. This algorithm, $\epsilon$-`UCT*`, iteratively runs the `UCT*` algorithm with increasingly more runtime until a policy has been found with a maximal error of $\epsilon$ compared to the optimal policy value. For the experiments themselves, three different sets of instances are used:

- `actsize`: Test set of 45 instances to determine the influence of activity set sizes. This set includes 15 2-agent instances with $|T| = 46$ and activity set sizes increasing from 1 to 15. For the 3, 4 and 5-agent instances, only activity set sizes 1 to 10 were considered (again $|T| = 46$).

[35] *This is discussed in more detail in Chapter 4. Indeed the algebraic decision diagrams and the resulting reward trees largely resemble the concept of Conditional Return Graphs.*

[36] *Although Keller and Eyerich [133] refer to the* `PROST` *planner, the* `PROST` *toolkit currently includes many other algorithms for probabilistic planning and therefore here referred to as such.*

- `horizon`: Test set of 70 instances to determine the influence of the planning period length. For each agent set size, different activity set sizes are used to account for the complexity increase of more agents. The following activity set sizes are used: 10 and 15 for 2 agents, 5 and 10 for 3 agents, 3 and 10 for 4 agents and 10 for 5 agents. The latter two are only solved by the activity group encoding and therefore the set sizes of 10 are determined to further investigate the scaling of that particular encoding. For all 7 types of instances, 10 actual instances are generated with increasing planning horizon length $|T| \in [1, 6, 11, \ldots, 46]$.

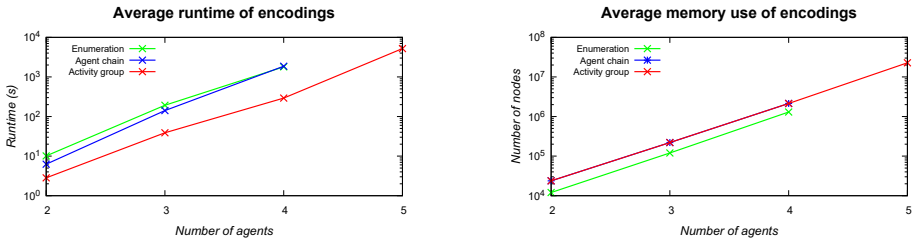
- `random`: Test set of 81 2-agent and 27 3-agent instances to compare the optimal algorithm versus approximate methods. For the 2-agent problems 9 random instances are generated with $|\mathcal{A}_i| \in [1, 3]$ and $T \in [6, 8, 10]$, while in the case of 3-agent problems 9 random instances are generated for only $|\mathcal{A}_i| = 3$ and the same planning period length.

Furthermore, all activities of the test sets have a non-zero probability of delaying and the traffic cost model is fully connected, i.e. every combination of activities results in a super-linear cost. In a sense, these instances are the "hardest" MPP problems.

The experimental results presented in this section were obtained by running them on a system with an Intel i7 Quad Core processor, each with a clock speed of 1.60 GHz, and 12GB of internal memory (DDR3). The time limit used in these experiments is 3 hours for the experiments with the optimal solver. For the comparison with approximate techniques a time limit of 10 minutes was set. Instances not solved within the time limit are not considered in the results and this is noted in the presentation of the results. Note that in these experiments the goal is not to establish exact runtimes or compare against other approaches. This evaluation is primarily meant to provide an insight into the scaling of techniques and the influence that problem characteristics may have.

The first experiment investigates the scalability of the three MDP encodings presented in Section 3.3, namely enumeration, agent chaining and activity grouping. To this end, the 45 instances of the `actsize` and the 70 instances of the `horizon` test set are encoded using all three methods and subsequently solved by the SPUDD solver. After every solving run, SPUDD reports the runtime in seconds and memory usage in terms of nodes required to represent the state space and reward structure of the MDP. The average runtime and memory usage of these results are shown in Figure 3.11 where they are grouped per agent set size.

A first observation is that in both figures only the activity group encoding is plotted for the 5-agent instances. Only using this encoding SPUDD was able to solve all 5-agent instances within the time limit of 3 hours (with an average runtime of approximately 87 minutes). Indeed, Figure 3.11a illustrates that in terms of runtime, the activity group encoding typically produces an optimal policy an order of magnitude faster than the other two encodings. Although these results are not sufficient to generalise with respect to the scalability, they do suggest that all techniques scale exponentially in the agent set size and, indirectly, the action and state space of the problem, and that of these three encodings the activity group contributes most to the scaling of MDP techniques.



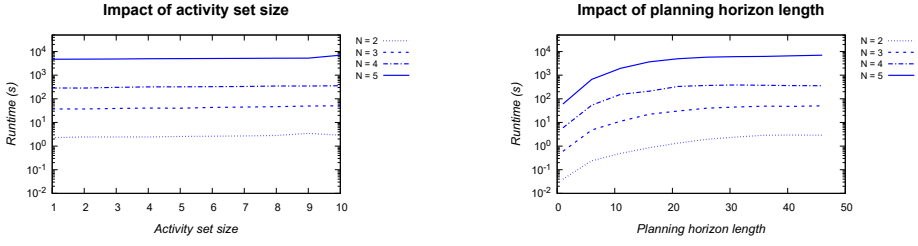

(a) (b)

**Figure 3.11** *Runtime and memory usage as reported by* SPUDD *for the 115 instances of the* `actsize` *and* `horizon` *test sets for the three MDP encodings enumeration, agent chaining and activity grouping. Both graphs use a logarithmic scale for the Y-axis.*



In terms of memory usage, shown in Figure 3.11b, the SPUDD algorithm performs comparably on the three encodings, with a slightly lower memory consumption when the enumeration encoding is used. This result is to be expected: both the agent chaining and the activity grouping transfer the complexity of action sets into the state space, therefore requiring more memory to represent all the states of the problem. Observe that the agent chaining and activity grouping encodings exhibit a similar pattern in their memory usage. This is because both techniques have a similar additional overhead in the state space. Also, SPUDD does not report the exact amount of memory required but only the number of nodes, which in turn are pre-allocated in chunks as is typical in data structures such as lists and arrays. The memory usage only provides insight into the order of magnitude. An exact comparison of memory usage is not possible.

Having established that the activity group encoding is the most effective technique to scale the SPUDD solver, further analysis focuses on identifying the characteristics of MPP that make the problem hard to solve. From the previous experiments it was already clear that the agent set size largely influences the complexity, however the role of the other parameters is unclear. Therefore the results are also plotted by per test set, `actsize` and `horizon`, that respectively vary in activity set size and planning horizon. The results are shown in the graphs of Figure 3.12.




(a) (b)

**Figure 3.12** *Runtime as a function of (a) activity set size and (b) planning horizon length. Again both graphs use a logarithmic scale for the Y-axis.*

Figure 3.12a clearly demonstrates the effectiveness of the activity grouping in transferring the action set space complexity into the state space, which in turn is more manageable for MDP solvers such as SPUDD. While the impact of the agent set size is clearly noticeable, an increase of activity set size only marginally increases the runtime required by the solver. On the other hand, Figure 3.12b shows that the planning horizon does substantially influence the solving time required. Especially in the lower-value regions, e.g. a horizon of 20 or below, the planning length has a lot of impact on the scalability. Instances with low planning horizon are expected to be relatively easy as there are only a limited number of possible plans. Increasing the planning horizon introduces an exponential number of new plans, therefore the computation time increases rapidly. As the horizon reaches higher values, the number of possible new plans continues to decrease as the activity set size is fixed. Instead the activities just take more time and/or more no-ops are included, leading to only minor increases in runtime.

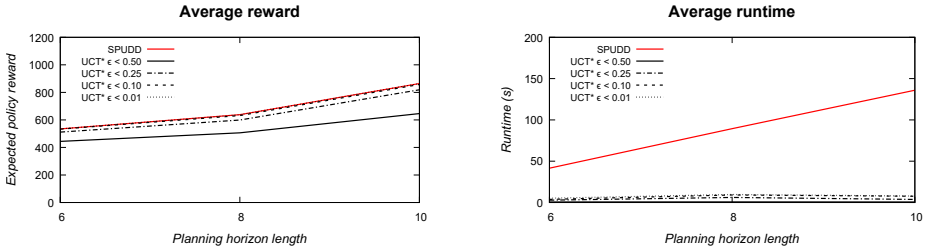
The last experiment compares the scalability of the optimal SPUDD algorithm against the approximate UCT* Monte-Carlo tree search algorithm. In particular, the bounded-error variant $\epsilon$-UCT* is used and the quality of the produced is compared to that found by the optimal algorithm in terms of expected value that is obtained. On the 108 instances of the `random` test set, the SPUDD algorithm is run as well as the $\epsilon$-UCT* algorithm with four different values for $\epsilon$: $0.5$, $0.25$, $0.1$ and $0.01$. In other words, the expected value of the approximated policy must respectively have an expected value of at least $50\%$, $75\%$, $90\%$ and $99\%$ of the value of the optimal policy. To diminish the impact of chance in the Mote-Carlo tree search, each data point for $\epsilon$-UCT* is an average over 3 solving runs. To measure the quality of the policy $\boldsymbol{\pi}$ produced by the approximation algorithms relative to any optimal policy $\boldsymbol{\pi}^*$, the error $\varepsilon = 1 - V^{\boldsymbol{\pi}}/V^{\boldsymbol{\pi}^*}$ is computed where $V^{\boldsymbol{\pi}}$ and $V^{\boldsymbol{\pi}^*}$ are the expected reward of respectively the approximate and optimal joint policies.

| | $\lvert\boldsymbol{N}\rvert = 2$ | | | $\lvert\boldsymbol{N}\rvert = 3$ | | |
|---|---|---|---|---|---|---|
| **Method** | $V^{\pi}$ | Time (s) | $\varepsilon$ | $V^{\pi}$ | Time (s) | $\varepsilon$ |
| CoRe | 671.425 | 87.005 | - | 724.702 | 152.872 | - |
| 0.50-UCT* | 527.295 | 0.703 | 0.195 | 525.954 | 0.951 | 0.274 |
| 0.25-UCT* | 635.765 | 4.260 | 0.048 | 685.264 | 1.504 | 0.056 |
| 0.10-UCT* | 666.240 | 6.808 | 0.007 | 705.450 | 1.706 | 0.022 |
| 0.01-UCT* | 671.224 | 7.572 | 0.000 | 723.969 | 2.497 | 0.001 |

**Table 3.4** *Comparison of average runtimes and expected values of the optimal (SPUDD) and approximation algorithms ($\epsilon$-UCT*) for various values of $\epsilon$, summarised over 2 and 3 agent instances. The column $\varepsilon$ contains the approximation error relative to the optimal value.*

The results of the experiment are summarised in Table 3.4 for the 2 and 3 agent instances. It can be seen immediately that the quality of the approximated policies are relatively close to the that of the optimal policy, whereas the runtime required by approximation is orders of magnitude less. Even a high value of $\epsilon$, e.g. $0.5$, the obtained quality is at most $20\%$ and $28\%$ lower than that of optimal algoritm for respectively the 2 and 3 agent instances, with a speedup of almost $120$ times for the 2-agents instances

and 160 times for 3 agents. Furthermore if a quality with an error of at most 1% is desired, the $\epsilon$-UCT* algorithm produces such near-optimal policies within a fraction of the time required by SPUDD to produce an optimal policy that is only slightly better.[37]




(a) (b)

**Figure 3.13** *Average expected policy value and runtime as a function of planning horizon length for the optimal* SPUDD *and approximate* $\epsilon$-UCT* *methods, with desired quality* $\epsilon \in [0.5, 0.25, 0.1, 0.01]$ *and* $|\boldsymbol{N}| = 2$.

Finally, to get an insight into the scaling of the optimal and approximate algorithms, Figure 3.13 depict the runtime and expected policy value of all techniques as a function of the planning horizon for the 2-agent instances of `random`. Figure 3.13a shows the average expected values obtained by the policies of both techniques, whereas Figure 3.13b illustrates the average runtime. Although the 3-agent instances are not shown here, the figures are highly similar to those of the 2-agent instances albeit that the runtime required by the SPUDD algorithm in particular increases more rapidly as the horizon increases. Both graphs correspond to the expectations set by the figures in Table 3.4: the expected value of approximate policies is close to the optimal policy, and approaches the optimal policy for smaller $\epsilon$, whereas the scaling of the approximate algorithms in terms of runtime is much better than that of the optimal algorithm. Note that with respect to the attained policy value, Figure 3.13a shows that for harder problems the 'gap' between optimal and approximate policy values widens.

## 3.6 Further Discussion

In this chapter the MAINTENANCE PLANNING PROBLEM, an example self-regulating planning problem inspired by a real-world use case, is presented as a mathematical optimisation problem. As a first method to find solutions to the optimisation problem a recursive formulation is defined that can be solved through dynamic programming. Thereafter, a multi-agent Markov Decision Process (MMDP) formulation of MPP is presented, enabling the use of off-the-shelve MMDP solvers to produce solutions to the optimisation problem. Furthermore, a 'flattening' of the MMDP into the more widely studied MDP model is proposed via three distinct encodings: enumeration, agent chaining and activity grouping. Whereas the former is a naive (exponential) enumeration

[37] *Although optimality is in some cases necessary, e.g. for the dynamic mechanism of Chapter 6.*

of all joint activity combinations, the latter two are more advanced encodings that produce action set sizes linear in the number of activities. This is achieved by transferring some of the complexity into the state space, which is typically better handled by MDP algorithms. Next, an approximate UCT* algorithm based on Monte-Carlo tree search is discussed for those settings where computational efficiency is more important than optimality. Finally, the effectiveness of the encodings and the scaling of all the presented (MDP) techniques are evaluated, showing that indeed MPP is a challenging problem even for state-of-the-art solvers.

With respect to the empirical evaluation, only the SPUDD and UCT* algorithms have been tested on MDP-encoded instances of MPP. The reward structure of SPUDD and Monte-Carlo search tree of UCT* make them suitable candidates for highly structured problems such as MPP. Nonetheless it would be interesting to solve the instances using other techniques like the ones presented by Boutilier et al. [40], Dai [71], Oliehoek et al. [196], Plutowski [207], Ruiz and Hernández [224] or even the point-based POMDP algorithm of Walraven and Spaan [259], and to compare them to the methods used here. Furthermore, other approximation techniques such as heuristic-based action selection or reactive planning may also be suitable to produce (near-)optimal policies fast.

Regarding the MDP encoding of MPP also some improvement can be implemented in future work. First of all, the states of the problem can be generalised in such a way that many paths of the decision policies converge to the same states. Consider the encoding of states as described in Section 3.3.2, every state contains a global time variable as well as a start and end time for every activity. However, for the decision making in future states it is sufficient to know an agent is currently idle and that a certain activity is no longer available. Moreover, the start time and end time of each activity can be derived from the actions $\texttt{start}_k$ and cont prescribed by the joint policy. In other words, all states with information $\langle t_{global}, t_k^s, t_k^e \rangle$ can be summarised into four states $\langle t_{global}, \neg k, idle \rangle$, $\langle t - global, \neg k, \neg idle \rangle$, $\langle t_{global}, k, idle \rangle$ and $\langle t_{global}, k, \neg idle \rangle$ such that $\neg k$ and $k$ respectively denote that activity $k$ is not or is completed in the state, and similarly $idle$ and $\neg idle$ denote that the agent is idle or not in that state. This abstraction of states may be beneficial to reduce the state-space complexity, in particular in problems with a large horizon where the number of possible combinations of start and end times may be high. In a similar line of work, Walraven and Spaan [258] introduces the concept of scenarios that may be very relevant in the context here to reason about commonly reoccurring sequences of states.

For the SPUDD solver an interesting enhancement is the addition of preconditions to MDP actions. Currently the MDP encodings use minus infinite value penalties to enforce domain rules regarding feasible outcomes. The downside thereof is that the solver may perform a lot of computations and evaluations before finding out that a certain action choice was infeasible to begin with. By adding preconditions, that are to be evaluated before the action-selection phase, the MDP encoding can help the solver to avoid infeasible areas of the solution space all together.

A final note here is that the focus of this chapter has been primarily on MPP. Although the ideas and concepts transfer to other (MDP-modelled) planning problems, and in particular self-regulating planning problems, generalising the approach taken here into guidelines or a standard methodology is left to future work.

# Chapter 4

# Maintenance Planning with Multiple Agents

In the previous chapter, the maintenance planning problem (MPP) was introduced. This problem, abstracted from a realistic challenge in need of solutions, poses a complex task for (automated) decision making. The previous chapter proposed several rudimentary methods to solve the problem, relying on existing approaches and techniques to develop both exact and approximate solutions. In particular, through efficiently 'flattening' the planning problem into a single-agent Markov decision process, it was demonstrated that optimal policies for MPP can be found using the SPUDD solver, albeit for small instances. The line of optimal solving is continued in this chapter, but the focus is shifted from effectively applying available methods to the design of a novel approach that targets the *structure* inherent to MPP. and similar problems that can be modelled as a multi-agent MDP (MMDP).

Recall from the previous chapter that the agent group and activity chain encodings of Section 3.3 successfully reduce the action set size from an exponential to a polynomial size. Nonetheless, when searching for the optimal policy, a typical MDP solver still has to evaluate all possible action decision sequences of every possible joint action to compute the joint reward, even when agents' decisions do not interact. In the maintenance planning domain, for example, revenues and costs depend only on the action of the agent that executes it. *Reward interaction* between agents only arises when network costs are involved. Ideally, reward computation should thus be decoupled for all decisions but those involving network costs in the MPP domain, or any multi-agent reward in general. In fully-observable, multi-agent MDPs, however, the value function cannot be factored into additive local components without loss of optimality as these components depend on non-local transition probabilities [145].

This chapter introduces an alternative approach to compute the expected value of a policy based on *returns*, defined as the cumulative reward obtained from a sequence of transitions. In other words, the return of a series of joint states and actions is the total reward that an agent receives, without the discounting by state/action transition probabilities typical to the expected value. Exactly because returns do not rely on the

joint transition probability, they can be decomposed without loss of optimality when the agents have independent transitions even though the optimal value function cannot.

In this chapter it is shown that return computations can be factored during policy search to compute the expected value of a policy more efficiently. In particular, in Section 4.2 a novel data structure called the *conditional return graph* is presented, a directed acyclic graph that compactly represents the returns local to an agent. The key idea is that when inter-agent rewards – referred to as *interaction rewards* – are sparse or limited in scope, they can be expressed compactly in local reward graphs which in turn are used to compute and bound joint transition rewards in a factored way. Moreover, when independence between CRGs is detected, (subsets of) of local components can be optimised independently, effectively decoupling agents during optimal policy search.

Consider the following example. There are two agents, numbered $1$ and $2$, with actions $A_1 = \{a_1\}$ and $A_2 = \{a_2, b_2, c_2\}$ respectively. Furthermore, the (joint) MMDP reward function $R$ is composed of two individual reward functions $\bar{R}_1$ and $\bar{R}_2$, one for each agent, and a network cost function $\bar{R}_{1,2} = \ell(\langle a_1, a_2 \rangle, t)$. The latter yields $0$ for all transitions but the ones involving joint action $\langle a_1, a_2 \rangle$, because they take place within close proximity of each other. Figure 4.1 visualises all state transitions available from a joint state $\left\{s_1^0, s_2^4\right\}$. Each arc is labelled by the transition and the reward functions required to compute the resulting reward of the transition.[38]

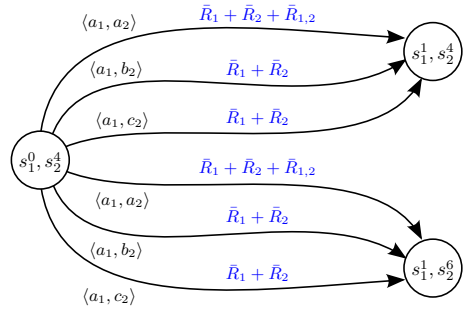



**Figure 4.1** *State-transition graph showing all available transitions, i.e. with non-zero probability, from a joint state $\left\{s_1^0, s_2^4\right\}$. Each arc represents a joint action (labelled in black) and the reward functions required to compute the transition reward (blue).*

First of all, observe that the interaction reward function $\bar{R}_{1,2}$ needs to be computed in only two transitions: the ones involving $\langle a_1, a_2 \rangle$. In addition, all transition reward computations require the individual reward functions $\bar{R}_1$ and $\bar{R}_2$, but they only depend on the local state transition, e.g. to compute $\bar{R}_1$ only knowledge regarding the transition of agent $1$ is necessary. Obviously, a better idea would be to decompose at least the individual rewards of both agents into two completely isolated reward graphs. The interaction reward $\bar{R}_{1,2}$ must nevertheless still be accounted for. To this end the following two graphs are constructed: one for agent $1$ that represents the transition reward given $\bar{R}_1$ and $\bar{R}_{1,2}$ (assigned arbitrarily to agent $1$) and another for agent $2$

[38] *Note that for a single transition the terms reward and return are interchangeable.*

with only the individual reward $\bar{R}_2$. The latter is similar to a typical single-agent MDP transition graph (and therefore not illustrated), the former is shown in Figure 4.2a and includes all of the transitions of Figure 4.1.

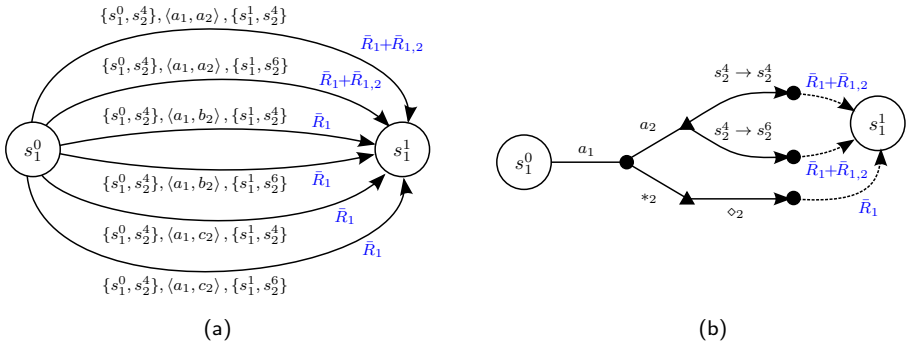



**Figure 4.2** *Two state-transition graphs for agent 1 that represent all rewards obtainable given functions $\bar{R}_1$ and $\bar{R}_{1,2}$. The graph of (a) includes all possible joint transitions and their rewards, whereas (b) illustrates the conditional return graph that compactly represents the same rewards by grouping transitions. Here, $*_2$ and $\diamond_2$ are wildcards representing the (remaining) actions and state transitions of agent 2 respectively.*

Note that the nodes of this graph are local states only but the reward $\bar{R}_{1,2}$ can still be computed because the arcs represent joint transitions. Moreover, the total joint reward is found by summing the transition rewards of both agents' transition graphs (again, the graph of agent 2 is not shown). Another observation reveals more structure that can be exploited: many of the transitions the graph of Figure 4.2a do not require knowledge about agent 2's transition and can be grouped. Only transitions with joint action $\langle a_1, a_2 \rangle$ need this information, the others are *independent*, which is made explicit in the conditional return graph of Figure 4.2b.

The above example demonstrates the main intuition behind the effectiveness of conditional return graphs: when inter-agent reward interactions are limited in number, they can be efficiently represented through local structures, simultaneously decoupling returns of agents where possible. Additionally, although returns cannot be directly used to compute the expected value of a joint policy, local bounds on the minimum and maximum return can be used to bound the local expected value. Moreover, summing these local bounds yields an upper and lower limit of the expected value of the joint policy, thus enabling a branch-and-bound style policy search. Finally, in MPP and many similar problems like machine scheduling, finite-resource production planning or rostering of personnel, actions can be performed only a single or limited number of times, after which they are no longer available. Therefore any reward interaction that involves such an action can also no longer occur, e.g. $\bar{R}_{1,2}$ after completing $a_1$ or $a_2$. When such *conditional reward independence* is detected in a state, agents can be decoupled from the state onward for the remainder of the policy search.

Figure 4.3 visualises this *conditional reward independence* for a 3-agent example where all agents have three actions: $a_i$, $b_i$ and $c_i$. Notice that after such a decoupling as

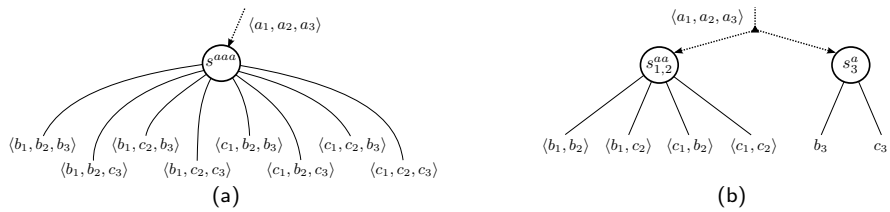



**Figure 4.3** *Agent decoupling during policy search: (a) part of a typical joint policy search tree with joint states and actions and (b) the same graph when agent sets $\{1, 2\}$ and $\{3\}$ can be decoupled after joint action $\langle a_1, a_2, a_3 \rangle$. After the decoupling, agents $1$ and $2$ no longer have to consider transitions of agent $3$ (and vice versa) when searching for the optimal joint actions for the remainder of the planning problem.*

shown in the figure, agent $3$ needs only its own CRG with local states and actions to find the transitions that optimise its expected reward. The CRGs in this example reduce the number of transitions from $2^3 = 8$ to $2^2 + 2^1 = 6$. Considering that a naive policy search algorithm requires $O(|S_1|^2 |A_1| \times |S_2|^2 |A_2| \times |S_3|^2 |A_3|)$ further evaluations in the worst case, the example decoupling would reduce this to $O(|S_1|^2 |A_1| \times |S_2|^2 |A_2| + |S_3|^2 |A_3|)$.

**Contributions** In this chapter a new optimal joint policy search algorithm is presented that uses CRGs as its main data structure, termed `Conditional Return Policy Search` (`CoRe`). The algorithm resembles a typical branch-and-bound policy search, however it computes rewards using CRGs and decouples policy search whenever agents become independent as a result of no more future reward interactions. The experiments at the end of this chapter demonstrate that `CoRe` finds optimal policies typically faster than the SPUDD approach of the previous chapter and scales better in terms of horizon and number of agents for sparse-reward MPP problems. The `CoRe` algorithm was first introduced by Scharpff et al. [229], this thesis contributes to it a strengthened definition of the conditional return graph and its elements (Definitions 4.3 to 4.5) and an elaborate example that illustrates the impact thereof (Example 4.14). The source code can be found at `https://github.com/AlgTUDelft/core-solver`.

The focus of this chapter is on *transition-independent* MMDPs, or TI-MMDPs, a sub-class of MMDPs in which agents depend on others only through their rewards. The decoupled reward structure that is characteristic to this set of MMDPs has been the inspiration for the theory underlying the `CoRe` algorithm. As a result, `CoRe` can typically exploit the TI-MMDP reward structure really well. Nevertheless, transition independence is not a necessary prerequisite for the approach and in Section 4.5 it is discussed how the algorithm can be extended to general MMDPs.

## 4.1 Returns in MDP

As mentioned in the introduction, the policy search method presented in this chapter and its data structure are based on returns. Before going into the concept of returns,

however, it is first necessary to clarify precisely the transition-independent MMDP (TI-MMDP) model and to introduce a general partitioning of rewards that will be the basis for the approach presented in this chapter. Although transition-independent MMDPs have been discussed in Chapter 2 (Definitions 2.11 and 2.17), here the model is restated in its entirety to ease further exposition:

**Definition 4.1 Transition-independent Multi-agent MDP (TI-MMDP)**

A (factored, fully-observable) transition-independent, multi-agent Markov decision process, or TI-MMDP, is defined by the tuple $\langle \boldsymbol{N}, \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R} \rangle$ where:

- $\boldsymbol{N}$ is the finite set of agents, indexed as $1, 2, \ldots, n$,
- $\boldsymbol{S} = \times_{i \in n} S_i$ is the factored, finite state space,
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{N}}$ is the collection of action sets where $A_i$ contains the actions only available to agent $i \in \boldsymbol{N}$,
- $\boldsymbol{P} = \{P_i\}_{i \in \boldsymbol{N}}$ is the collection of transition probability functions such that each $P_i : S_i \times A_i \times S_i \mapsto [0, 1]$ is the transition probability of agent $i$, based on its *own* states and actions,
- $\boldsymbol{R}$ defines the joint reward for every transition, e.g. $\boldsymbol{R} : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \mapsto \mathbb{R}$.

The MMDP is transition independent when $\boldsymbol{P}(s, \vec{a}, \hat{s}) = \prod_{i \in \boldsymbol{N}} P_i(s_i, a_i, \hat{s}_i)$, such that $s_i, \hat{s}_i \in S_i$ are respectively the state and next state of agent $i$, and $a_i \in A_i$ the action that agent $i$ takes in joint action $\vec{a}$.

Given that the TI-MMDP is factored and transition independent, it is possible to decompose the reward function into a set of locally scoped reward functions $\boldsymbol{R} = \left\{ \bar{R}_e \,|\, e \subseteq \boldsymbol{N} \right\}$, such that $\forall i \in \boldsymbol{N}$: $\bar{R}_i$ is the *local reward* (function) of agent $i$ and any other function $\bar{R}_e$ with $|e| > 1$ is called an *interaction reward* (function) with agents $e$ in its scope. This decomposition is without loss of generality as the original reward function $R$ can be represented through a single interaction reward $\bar{R}_{\boldsymbol{N}}$ with all agents in its scope.

An important advantage of such a partitioning is that to compute the reward due to each of these functions, only the states and actions of the agents in its scope are required. Therefore, the reward from any function $\bar{R}_e$ with agent(s) $e$ in its scope is given by $\bar{R}_e(\{s_i\}_{i \in e}, \langle a_i \rangle_{i \in e}, \{\hat{s}_i\}_{i \in e})$, conveniently written $\bar{R}_e(s_e, \vec{a}_e, \hat{s}_e)$ such that $s_e, \hat{s}_e \in \times_{i \in e} S_i$ and $\vec{a}_e \in \times_{i \in e} A_i$. Moreover, the total reward obtained from a joint transition $(s, \vec{a}, \hat{s})$, when the rewards are decomposed as described above, is computed by

$$\boldsymbol{R}(s, \vec{a}, \hat{s}) = \sum_{\bar{R}_e \in \boldsymbol{R}} \bar{R}_e(s_e, \vec{a}_e, \hat{s}_e) = \sum_{\bar{R}_e \in \boldsymbol{R}} \bar{R}_e(\{s_i\}_{i \in e}, \langle a_i \rangle_{i \in e}, \{\hat{s}_i\}_{i \in e}) \tag{4.1}$$

where the symbol $\boldsymbol{R}$ is slightly abused to denote both the set of all reward functions as well as the function to compute the joint reward for a given transition.

Let $\bar{\boldsymbol{R}}$ denote the set of only interaction rewards, i.e. $\bar{\boldsymbol{R}} = \{\bar{R}_e \,|\, e \subseteq \boldsymbol{N}, |e| > 1\}$. Agents $\boldsymbol{N}' \subseteq \boldsymbol{N}$ are said to be *dependent* if they are all within the scope of at least one such a function, i.e. $\exists \bar{R}_e \in \bar{\boldsymbol{R}} \colon \boldsymbol{N}' \subseteq e$. Furthermore, an action $a_i$ of agent $i$ is said to be a dependent action if there is at least one interaction reward function $\bar{R}_e \in \bar{\boldsymbol{R}}$ with $\bar{R}_e(s_e, \vec{a}_e, \hat{s}_e) \neq 0$ for at least one combination of (joint) states $s_e, \hat{s}_e$ and joint action $\vec{a}_e$ that includes action $a_i$. For dependent actions, $\bar{R}_e$, $a_i \in \mathrm{Dom}(\bar{R}_e)$ is used to denote that action $a_i$ belongs to the input of that function, i.e. it depends on another agent *through* interaction reward $\bar{R}_e$. Of course, a single action can have several interaction rewards. Note that when $\bar{R}_e$ is mentioned, it is implicitly assumed that it is an interaction reward and not a local reward, thus $\bar{R}_e \in \bar{\boldsymbol{R}}$, unless stated otherwise.

Recall from Section 2.2 that the expected value of an optimal policy for a (finite-horizon) MDP is computed by the Bellman equation, which can be rewritten to accommodate for partitioned rewards:

$$\begin{aligned} V^*(s^t) &= \max_{\vec{a}^{\,t} \in \boldsymbol{A}} \sum_{s^{t+1} \in \boldsymbol{S}} \boldsymbol{P}(s^t, \vec{a}^{\,t}, s^{t+1}) \Big( \boldsymbol{R}(s^t, \vec{a}^{\,t}, s^{t+1}) + V^*(s^{t+1}) \Big) \\ &= \max_{\vec{a}^{\,t} \in \boldsymbol{A}} \sum_{s^{t+1} \in \boldsymbol{S}} \boldsymbol{P}(s^t, \vec{a}^{\,t}, s^{t+1}) \Big( \sum_{\bar{R}_e \in \boldsymbol{R}} \bar{R}_e(s^t, \vec{a}^{\,t}, s^{t+1}) + V^*(s^{t+1}) \Big) \end{aligned} \tag{4.2}$$

where $V^*(s^{t+1}) = 0$ for every state $s^t$ with $t > h$ due to the finite horizon. A key observation is that, although the expected value $V^*$ can be computed through a series of maximisations over the planning period, it cannot be factored into locally solvable components without losing global optimality, unless they are independent in both transitions and rewards [145]. In contrast, the cumulative reward obtained from following a state/action sequence *can* be factored into local components that can be leveraged in optimal policy search. This cumulative reward is known as the *return* and is computed over *execution sequences*, i.e. the history of transitions, first introduced in Section 3.1. Briefly reiterating, a history $H^t$ from time 0 to $t$ contains the state/action transitions $(s^0, \vec{a}^{\,0}, s^1), \ldots, (s^{t-2}, \vec{a}^{\,t-2}, s^{t-1}), (s^{t-1}, \vec{a}^{\,t-1}, s^t)$ that lead to state $s^t$. In its (equivalent) compact form, a history is typically written as a state-action sequence: $\theta^t = [s^0, \vec{a}^{\,0}, s^1, \vec{a}^{\,1}, \ldots, s^{t-1}, \vec{a}^{\,t-1}, s^t]$. The return of such a sequence is defined as:

**Definition 4.2 Return**

The return of an execution sequence is the accumulated reward from the transitions it represents[39], or

$$Z(\theta^t) = \sum_{x=0}^{t-1} \boldsymbol{R}(s^x, \vec{a}^{\,x}, s^{x+1}) \tag{4.3}$$

in which $s^x, s^{x+1}$ and $\vec{a}^{\,x}$ are respectively the (joint) states and actions at time $x$ in sequence $\theta^t$.

Similar to the reward partitioning introduced earlier, it is possible to decompose the return of Equation 4.3 into localised components, such that the return is alternatively

[39] *And hence the notions of return and reward are equivalent for a single transition.*

represented as

$$Z(\theta^t) = \sum_{x=0}^{t-1} \sum_{\bar{R}_e \in \boldsymbol{R}} \bar{R}_e(s_e^x, \vec{a}_e^{\,x}, s_e^{x+1}) \tag{4.4}$$

as long as the states and actions of all agents in every set $e$ are included in $\theta^t$.

The formulation of Equation 4.4 may seem trivial, but it opens the way for a factored policy evaluation method and – demonstrated later – decoupling of agents during policy search. Recall that the expected value of a policy is the product of the reward and probability of each history, i.e. execution sequence, that may be realised under the policy. Therefore, it is possible to express the expected value of policy $\pi$ as

$$V^\pi(s^0) = \sum_{\theta^h|\pi,s^0} Pr(\theta^h) Z(\theta^h) = \sum_{\theta^h|\pi,s^0} Z(\theta^h) \prod_{t=0}^{h-1} \boldsymbol{P}(s^t, \vec{a}^{\,t}, s^{t+1}) \tag{4.5}$$

such that $\theta^h|\pi, s^0$ denotes all execution sequences (histories) up to horizon $h$ reachable under policy $\pi$ from initial state $s^0$ (analogous to Equation 3.7). Although Equation 4.5 does not directly express the optimal expected value, it does allow for a factored computation of the expected value of any given policy $\pi$.



## 4.2 Conditional Return Graphs

The definition of return – the sum of rewards as a result from the transitions taken – and the knowledge that the joint transition reward in an MMDP can be expressed as a sum of locally scoped reward functions $\bar{R}_e$, gives rise to a novel reward data structure. This section formally introduces the *conditional return graph (CRG)*, a data structure that represents all MDP returns effectively in a local transition graph when the reward interactions between agents are sparse. Here, sparse means that the interaction rewards are non-zero for a relatively small subset of joint transitions and/or that there are at most a few agents in the scope of such rewards (formalised in Theorem 4.7 presented later). The term conditional is due to the decision-graph structure of the CRG in the sense that the return is conditional on the execution sequence observed, represented by paths through the CRG nodes and edges.

The main intuition behind the effectiveness of CRGs is that by partitioning the joint reward function, according to the aforementioned reward partitioning, and grouping them into a smaller and coherent data-structure, they represent locally computable reward components that are easily combined when policy evaluation demands it. This enables both a more compact representation of all potential rewards of the MMDP as well as caching, and later re-use, of these rewards to facilitate computation of expected value when probabilities become known during policy search.

The partitioned rewards can be grouped in many ways, but the most natural grouping of return components is at the agent level, which is also logical because each agent is at least involved in its private reward function $\bar{R}_i$. Note that the choice of partitioning does not change the outcome, it only influences the efficiency of the search procedure. Hence the rewards $\boldsymbol{R}$ are partitioned into disjoint sets $\boldsymbol{R}_i = \{\bar{R}_i\} \cup \bar{\boldsymbol{R}}_i$

such that $\bar{\boldsymbol{R}}_i \subseteq \bar{\boldsymbol{R}}$ is the set of interaction rewards assigned to agent $i$ and together these sets form the joint reward, i.e. $\boldsymbol{R} = \bigcup_{i \in \boldsymbol{N}} \boldsymbol{R}_i$. Given this factoring of rewards the computation of Equation 4.1 still holds, although slightly altered:

$$\boldsymbol{R}(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} \boldsymbol{R}_i(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} \sum_{\bar{R}_e \in \boldsymbol{R}_i} \bar{R}_e(s_e, \vec{a}_e, \hat{s}_e) \tag{4.6}$$

and notice that only the scope $e_i \subseteq \boldsymbol{N}$ such that $e_i = \left\{ j \in \boldsymbol{N} \,|\, \exists \bar{R}_e \in \boldsymbol{R}_i : j \in e \right\}$ is required to compute the rewards over sets $\boldsymbol{R}_i$. This is the key function of a CRG: it represents only the rewards $\boldsymbol{R}_i$ of the joint transitions of the agents $e_i$.

Decoupling of reward is a first step towards a more effective policy search method because only interacting states and actions are considered instead of the complete joint state and action spaces. However, the assumption of transition independence introduces an additional structure in the rewards that can be exploited, especially when interactions are sparse, allowing grouping of transitions that lead to equivalent states and rewards. For this it is necessary to define the *available* (joint) transitions:

$$\boldsymbol{\tau}_e = \{(s_e, \vec{a}_e, \hat{s}_e) \,|\, s_e, \hat{s}_e \in S_e,\, \vec{a}_e \in A_e,\, P_e(s_e, \vec{a}_e, \hat{s}_e) > 0\} \tag{4.7}$$

where $S_e = \times_{i \in e} S_i$ and $A_e = \times_{i \in e} A_i$ denote the joint states and actions of agents $e$, a notation that was introduced in the previous section. Thus, $\tau_e \in \boldsymbol{\tau}_e$ denotes a joint transition $(s_e, \vec{a}_e, \hat{s}_e)$ of agents $e$, whereas $\tau_j \in \boldsymbol{\tau}_j$ implies a single-agent or local transition $(s_j, a_j, \hat{s}_j)$. A local transition $\tau_i = (s_i, a_i, \hat{s}_i)$ is said to be contained in $\tau_e$, denoted $\tau_i \in \tau_e$, if $s_i \in s_e$, $a_i \in \vec{a}_e$ and $\hat{s}_i \in \hat{s}_e$. In addition, a (joint) transition of all the agents in $e$ without transition $\tau_j$ of agent $j$ is conveniently written as $\tau_e \setminus \{\tau_j\}$ when $\tau_j \in \tau_e$, i.e. the transition of agent $j$ is part of the joint transition $\tau_e$.

With the notion of available transitions, it is possible to capture the set of actions of other agents that potentially influence the rewards $\boldsymbol{R}_i$ assigned to agent $i$. An action $a_j$ of agent $j \neq i$ is said to be *dependent* with respect to local transition $\tau_i$ of agent $i$ if (a) the action $a_j$ occurs in one of the available joint transitions $\tau_e$ that contains $\tau_i$, (b) its presence as part of a transition $\tau_j = (s_j, a_j, \hat{s}_j)$ influences the interaction reward and (c) there is at least one transition with another action $a'_j$ of agent $j$ that does *not* cause the same interaction reward. The last condition is included to prevent marking all actions as dependent when actually the interaction reward depends on the state and/or state transition of agent $j$ (captured by the transition influence, Definition 4.4). This leads to the definition of *dependent actions*:

**Definition 4.3 Dependent Actions**

The set of *dependent actions* of an agent $j \in \boldsymbol{N}$ that may reward-interact with agent $i \neq j$ when agent $i$'s transition is $\tau_i = (s_i, a_i, \hat{s}_i)$ is given by:

$$\begin{aligned} D(\tau_i, j) = \{a_j \in A_j \,|\, \exists\, \bar{R}_e \in \boldsymbol{R}_i,\, \exists\, \tau_e \in \boldsymbol{\tau}_e, \exists\, a'_j \neq a_j \in A_j : \\ \tau_i \in \tau_e \wedge a_j \in \vec{a}_e \qquad (4.8) \\ \wedge\, \bar{R}_e(\tau_e) \neq \bar{R}_e(\tau_e \setminus \{\tau_j\}), \quad \text{s.t. } \tau_j \in \tau_e \qquad (4.9) \\ \wedge\, \bar{R}_e(\tau_e) \neq \bar{R}_e(s_e, \vec{a}_e \setminus \{a_j\} \cup \{a'_j\}, \hat{s}_e)\} \qquad (4.10) \end{aligned}$$

All actions of an agent $j \neq i$ that are not dependent with respect to a transition $\tau_i$, i.e. the actions $A_j \setminus D(\tau_i, j)$, are grouped in the CRG for agent $i$ via a 'wildcard' $*_j$. Note that conditions (a) through (c) are represented in this definition by respectively the terms 4.8, 4.9 and 4.10.

Although the aforementioned dependent actions influence the rewards $\boldsymbol{R}_i$, they may not do so for every possible transition of the corresponding agent. For instance, joint action $\langle a_i, a_j \rangle$ may only cause a reward interaction when agent $j$ transitions from a state $s_j^x$ to $s_j^y$, but will not for any other transition. Furthermore, when the interaction reward is not action-based but state-based, i.e. when condition (4.10) is not met for any action of agent $j$ but the interaction reward *is* influenced by the agent's transition, it is often still the case that only a subset of the (state pairs in) transitions will actually influence the interaction reward. This is captured by the *(transition) influence*.

The definition of transition influence is rather similar to that of dependent actions. For a state transition from $s_j$ to $\hat{s}_j$ of an agent $j \neq i$ to be considered an influence with respect to local transition $\tau_i$, both states must (a) be part of a transition $\tau_j$ such that both $\tau_i$ and $\tau_j$ are contained in a joint transition $\tau_e$, (c) such a transition $\tau_j$ must have an impact on at least one interaction reward and (c) there must exist at least one other transition of agent $j$ that does not have the same interaction reward impact. The latter condition is, alike before, to prevent all state transitions being marked an influence whereas the interaction reward depends solely on the action. Formally, the influence is defined as

**Definition 4.4 Transition Influence**

The set of state pairs of an agent $j$ that may lead to a reward interaction when agent $i \neq j$ follows transition $\tau_i = (s_i, a_i, \hat{s}_i)$ and agent $j$ performs action $a_j \in A_j$ concurrently, known as the *(transition) influence*, is defined as

$$
\begin{aligned}
I(\tau_i, a_j) = \{(s_j, \hat{s}_j) \in S_j^2 \mid \exists\, \bar{R}_e \in \boldsymbol{R}_i, \exists\, \tau_e \in \boldsymbol{\tau}_e, \exists\, (s'_j, \hat{s}'_j) \neq (s_j, \hat{s}_j) \in S_j^2 : \\
\tau_i \in \tau_e \wedge \tau_j = (s_j, a_j, \hat{s}_j) \in \tau_e \qquad (4.11) \\
\wedge\, \bar{R}_e(\tau_e) \neq \bar{R}_e(\tau_e \setminus \{\tau_j\}), \quad \text{s.t. } \tau_j \in \tau_e \qquad (4.12) \\
\wedge\, \bar{R}_e(\tau_e) \neq \bar{R}_e(s_e \setminus \{s_j\} \cup \{s'_j\}, \vec{a}_e, \hat{s}_e \setminus \{\hat{s}_j\} \cup \{\hat{s}'_j\})\} \qquad (4.13)
\end{aligned}
$$

Finally, for any set of actions $A_j$ of an agent $j$, the transition influence of that set with respect to local transition $\tau_i$ of agent $i$ is defined as the union of all influences, or

$$
I(\tau_i, A_j) = \bigcup_{a_j \in A_j} I(\tau_i, a_j) \qquad (4.14)
$$

This last definition is useful to capture the influence of a wildcard set $*_j$. Again, non-influencing state transitions can be grouped. The symbol $\diamond_j$ is used to denote the set of all non-influencing state pairs of agent $j$, given a local transition $\tau_i$ and action $a_j$ (or wildcard $*_j$).

Together, the definitions of dependent actions and transition influences capture all actions and state transitions of other agents that need to be considered to represent all (joint) transition rewards $\boldsymbol{R}_i$. This is captured by a *Conditional Return Graph*:

**Definition 4.5 Conditional Return Graph**

Given a disjoint, complete partitioning $\boldsymbol{R} = \bigcup_{i \in \boldsymbol{N}} \boldsymbol{R}_i$ of rewards over agents $i \in \boldsymbol{N}$, the *Conditional Return Graph (CRG)* $\phi_i$ is a *directed acyclic graph* with for every stage $t$ of the decision process a node for every reachable local state $s_i \in S_i$ and for every available local transition $\tau_i = (s_i, a_i, \hat{s}_i)$ a tree compactly representing all transitions $\tau_e = (s_e, \vec{a}_e, \hat{s}_e)$ of the agents $e \subseteq \boldsymbol{N}$ in the scope of $\boldsymbol{R}_i$, or $e = \{i \in \boldsymbol{N} \mid \exists \bar{R}_e \in \boldsymbol{R} \colon i \in e\}$.

The tree consists of two parts: an *action tree* that specifies all dependent actions and an *influence tree* that contains the relevant local state transitions. For every action $a_i \in A_i$ of agent $i$, the state $s_i$ is connected to the root node $v_{a_i}$ of an action tree by an arc labelled with action $a_i$. For every root node $v_{a_i}$, let $v = v_{a_i}$ be the root of an action tree such that it is defined recursively over the remaining $\boldsymbol{N}' = e \setminus \{i\}$ agents as follows:

A1 If $\boldsymbol{N}' \neq \emptyset$ take some $j \in \boldsymbol{N}'$, otherwise stop.

A2 For every $a_j \in D(\tau_i, j)$, create an internal node $v_{a_j}$ connected from $v$ by an arc labelled with the action $a_j$.

A3 Create one internal node $v_{*_j}$ to represent all actions of agent $j$ not in $D(\tau_i, j)$ (if any), connected by an arc labelled by the 'other action' wildcard $*_j$ from the root node $v$.

A4 For each leaf node $v_{a_j}$ (or $v_{*_j}$) that results from the previous steps, create a sub-tree with $N' = N' \setminus \{j\}$ and $v = v_{a_j}$ as its root using the same procedure.

When all action arcs have been created, each leaf node $v_{a_x}$ of the action tree is the root node $u$ of an influence tree, where $a_x$ is either the last dependent action or wildcard $*_x$ of the agent $x$ that is visited in the last recursion. Starting again from $\boldsymbol{N}' = e \setminus \{i\}$:

B1 If $\boldsymbol{N}' \neq \emptyset$ take some $j \in \boldsymbol{N}'$, otherwise stop.

B2 If the path from $s_i$ to node $u$ contains an arc labelled with action $a_j \in D(\tau_i, j)$, create child nodes $u_{s_j \to \hat{s}_j}$ to represent all local pairs of state transitions $(s_j, \hat{s}_j)$ of agent $j$ in the action influence $I(\tau_i, a_j)$, connected to node $u$ by arcs labelled $s_j \to \hat{s}_j$.

**else**

The path from $s_i$ to node $u$ contains the wildcard $*_j$. Create child nodes $u_{s_j \to \hat{s}_j}$ for all pairs of local state transitions $(s_j, \hat{s}_j) \in I(\tau_i, *_j)$, i.e. the influence of the set of actions represented by $*_j$ (all $a_j \notin D(\tau_i, j)$), and connect them to $u$ with arcs labelled $s_j \to \hat{s}_j$.

B3 If there remains any pair of local states $(s_j, \hat{s}_j) \in S_j^2$ with $P(s_j, a_j, \hat{s}_j) > 0$ that is not in $I(\tau_i, a_j)$ or a pair with $\sum_{a_j \in *_j} P(s_j, a_j, \hat{s}_j) > 0$ that is not in $I(\tau_i, *_j)$, depending on the action of agent $j$ on the path to node $u$, create another child node $u_{\diamond_j}$ connected by an arc labelled by the 'other state pairs' wildcard $\diamond_j$.

B4 For each leaf node $u_{s_j \to \hat{s}_j}$ (or $u_{\diamond_j}$) that results from the previous step, create a sub-tree with $\boldsymbol{N}' = \boldsymbol{N}' \setminus \{j\}$ and root $u = u_{s_j \to \hat{s}_j}$ (resp. $u = u_{\diamond_j}$) repeating the same procedure.

Finally, each leaf node $u_{s_x \to \hat{s}_x}$ ($x$ again being the last agent) of every influence tree is connected to the new local state node $\hat{s}_i$ by an arc labelled with the transition reward $\boldsymbol{R}_i(s_e, \vec{a}_e, \hat{s}_e)$ that corresponds to the actions and state pairs on the path from $s_i$ to $\hat{s}_i$.

---

From the above definition it is trivial to implement a CRG construction algorithm that produces direct, acyclic reward graphs, alike the one shown in Figure 4.4b of Example 4.6. Observe that any path from the root node to the new local state sufficiently specifies a joint transition of the agents in the scope of the CRG. In other words, each path contains all the information required to compute $\boldsymbol{R}_i(s_e, \vec{a}_e, \hat{s}_e)$ of any local transition of the agents $e$ in the scope of the CRG $\phi_i$. Whenever a wildcard action or state transition occurs, any of the state pairs and actions represented by that wildcard can be used to compute the transition rewards because they all result in the same reward.

Recall that a sequence of transitions up to time $t$ is captured by an execution sequence $\theta^t$ and that the *return* of such a sequence is the total reward obtained from its execution. Also, observe that paths through a CRG $\phi_i$ to local state nodes $s_i^t$ correspond to *local* execution sequences $\theta_{e_i}^t$, where $t$ is the stage in the planning horizon and $e_i$ is the set of agents in the scope of $\phi_i$. Each of these sequences contain only those transitions $\tau_{e_i} = (s_{e_i}, \vec{a}_{e_i}, \hat{s}_{e_i})$ local to the CRG of agent $i$. Moreover, due to the decomposition of rewards into disjoint sets $\boldsymbol{R}_i$ it is possible to decompose the returns over local execution sequences $\theta_{e_i}^t$ as

$$Z(\theta^t) = \sum_{i \in \boldsymbol{N}} Z(\theta_{e_i}^t) \tag{4.15}$$

**Example 4.6** CRG

Figure 4.4 again shows the two local reward graphs for the rewards $\boldsymbol{R}_1 = \{\bar{R}_1, \bar{R}_{1,2}\}$ of agent 1 from the introduction. Figure 4.4a illustrates the full transition graph whereas Figure 4.4b illustrates the CRG that is constructed from following Definition 4.5. As agent 1 has only action $a_1$, the action tree of the CRG for the transition $\tau_1 = (s_1^0, a_1, s_1^1)$ requires only a single arc from the root to node $v_{a_1}$, followed by an arc for the dependent action $a_2$, which is (the only) member of $D(\tau_1, 2)$, and one for the remaining actions $*_2 = \{b_2, c_2\}$. From both new leaves, $v_{a_2}$ and $v_{*_2}$, an influence tree is generated. For the leaf $v_{*_2}$ that results from $*_2$ no interaction reward follows as $I(\tau_1, *_2) = \emptyset$ and an arc labelled $\diamond_2$ is generated that represents every possible state transition of agent 2. The influence tree starting from node $v_{a_2}$, containing $a_2$ in its path, is slightly more complex because there is reward interaction. Here the influence is $I(\tau_1, a_2) = \{(s_2^4, s_2^4), (s_2^4, s_2^6)\}$, thus two influence arcs. Note that if agent 2 would have had more state transitions available than the two already contained in the influence, a wildcard influence $\diamond_2$ arc would also have been added from node $v_{a_2}$.

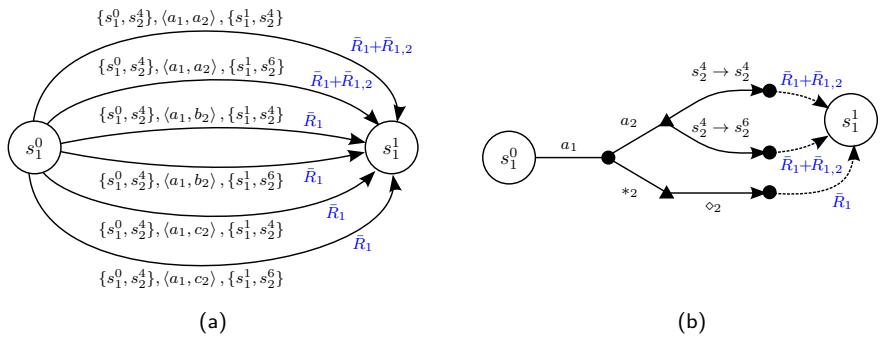



**Figure 4.4** *The two state-transition graphs for the rewards $\boldsymbol{R}_i$ agent 1 from the introduction that represent all rewards obtainable given functions $\bar{R}_1$ and $\bar{R}_{1,2}$. The graph of (a) includes all possible joint transitions and their rewards, whereas (b) illustrates the conditional return graph that compactly represents the same rewards by grouping actions and influences.*

### 4.2.1 CRG Size

CRGs provide a compact representation of rewards when reward interactions are sparse or have a limited number of agents in their scope. In order to formalise this compactness, a bound can be made on the maximum number of nodes in the CRG. Let the maximal size of any of the local state spaces be denoted by $\left|\bar{S}\right| = \max_{i \in \boldsymbol{N}} |S_i|$ and that of the local action spaces by $\left|\bar{A}\right| = \max_{i \in \boldsymbol{N}} |A_i|$. Furthermore, $\alpha = \max_{i,j \in \boldsymbol{N}} \max_{\tau_i \in \boldsymbol{\tau}_i} |D(\tau_i, j)|)$ and $\beta = \max_{i,j \in \boldsymbol{N}} \max_{\tau_i \in \boldsymbol{\tau}_i} \max_{a_j \in A_j} |I(\tau_i, a_j)|$ are respectively the sizes of the largest dependent action set and transition influence set. When no dependent actions or transition influences occur respectively $\alpha$ or $\beta$ (or both) include at least the wildcard arc, thus both are guaranteed equal to or larger than 1. It is not necessary to include $*_j$ when computing $\beta$, because the size of the influence of $*_j$ is determined whenever any action $a_j \in *_j$ is evaluated. Lastly, $w = \max_{\bar{R}_e \in \bar{\boldsymbol{R}}} |e| - 1$ is the maximum number of agents on which an agent can depend.

**Theorem 4.7 The CRG size is bounded in the number of dependencies**

The size of a CRG is bounded by the number of dependent actions and transition influences

$$O\Big( h \cdot \left|\bar{A}\right| \left|\bar{S}\right|^2 \cdot (\alpha\beta)^w \Big) \tag{4.16}$$

*Proof.* A CRG has as many layers as the planning horizon $h$. In the worst case, in every stage there are $\left|\bar{S}\right|$ local state nodes, each connected to at most $\left|\bar{S}\right|$ next-stage local state nodes via multiple arcs. The number of action arcs between two local state nodes $s^i$ and $\hat{s}^i$ is at most $\left|A^i\right|$ times the maximal number of dependent actions, which is bounded by $\alpha^w$. Finally, the number of influence arcs is bounded by $\beta^w$. □

Putting Theorem 4.7 in perspective, the size of a full policy search space tree with $n = |\boldsymbol{N}|$ agents is bounded by $\Theta(h \cdot \left|\bar{S}\right|^{2n} \cdot \left|\bar{A}\right|^n)$. In theory all actions can be dependent, in

which case the worst-case size of all CRGs together is of order $O(nh \cdot |\bar{S}|^{2+2w} \cdot |\bar{A}|^{1+w})$, $2w$ because there are at most $|\bar{S}|^2$ state pairs in an influence set, which is typically still much more compact unless $w \approx n$. In many planning problems, however, the interaction rewards are more sparse such that $\alpha^w \ll |\bar{A}|^w$ and $\beta^w \ll |\bar{S}|^w$. Moreover, (4.16) upper bounds the general CRG size, for a specific CRG $\phi_i$ this bound is often more tight as for many agents the number of dependent actions and/or influences is smaller than respectively $\alpha$ or $\beta$.

**Remark 4.8 State features**

Definition 4.5 states the general definition of CRGs with dependent actions and pairs of states that fully specify local transitions. If states are factored, i.e. composed from individual features $\boldsymbol{X}$ (see Definition 2.5), the influence is sufficiently expressed by (combinations of) features instead of states. For example, for an interaction reward that is only non-zero when some feature $X_j^x$ of agent $j$ is set from 0 to 1, the influence just contains feature transition $\{X_j^x = 0\} \rightarrow \{X_j^x = 1\}$ instead of all state pairs $s_j \rightarrow \hat{s}_j$ that correspond to $\{X_j^x = 0\} \in s_j$ and $\{X_j^x = 1\} \in \hat{s}_j$, leading to a much smaller $\beta$ in Theorem 4.7. The use of features is illustrated in Example 4.14.

4

### 4.2.2 Bounding the Returns

Although CRGs allow a compact representation of returns given the reward functions $\boldsymbol{R}_i$ assigned to the CRG $\phi_i$ of agent $i$, recall that it is not possible to directly compute the expected value of the local execution sequences represented by the CRG. This is because the rewards are known for every possible local transition, but the actual transition probabilities (of other agents) are not. Only when these probabilities become known during policy search, returns can be used to compute the expected value (Equation 4.5). Nevertheless, the returns represented by the CRGs may be used to *bound* the expected value of local execution sequences and, as a corollary of Equation 4.15, the expected value of the globally joint policy. In particular, the maximal and minimal obtainable returns from a local state onward directly bound the expected value that can be achieved from that state. By summing the return bounds of all local states, it is possible to bound the global return of all (remaining) execution sequences, which also directly bounds the expected value of the (optimal) joint policy. In addition, these bounds depend only on local states and transitions, and can therefore be determined and stored while generating the CRG for efficient global bound computation in policy search.

Let $\phi_i(s_i)$ denote the set of local transitions $(s_{e_i}, \vec{a}_{e_i}, \hat{s}_{e_i})$ available from state $s_i$ (with $s_i \in s_{e_i}$), captured in CRG $\phi_i$, i.e. $\phi_i(s_i) = \{\tau_{e_i} \in \boldsymbol{\tau}_{e_i} \mid (s_i, a_i, \hat{s}_i) \in \tau_{e_i}\}$ where $e_i$ is the scope of $\boldsymbol{R}_i$. The upper bound on the remaining return from a state $s_i \in S_i$ is then defined recursively as

$$U(s_i) = \max_{(s_{e_i}, \vec{a}_{e_i}, \hat{s}_{e_i}) \in \phi_i(s_i)} \left( \boldsymbol{R}_i(s_{e_i}, \vec{a}_{e_i}, \hat{s}_{e_i}) + U(\hat{s}_i) \right) \tag{4.17}$$

such that $\hat{s}_i \in \hat{s}_{e_i}$ and naturally $U(s_i) = 0$ for every terminal state of the planning problem, e.g. when $t = h$ in the finite horizon setting. This upper bound can be applied to a joint state $s_e$ involving agents $e$ simply by summing the upper bound

of the individual local states, or $U(s_e) = \sum_{i \in e} U(s_i)$. For a specific joint transition $(s, \vec{a}, \hat{s})$, the maximal future return that may be obtained is

$$U(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} \left( \boldsymbol{R}_i(s_{e_i}, \vec{a}_{e_i}, \hat{s}_{e_i}) + U(s_i) \right) \tag{4.18}$$

Naturally, all of the above formulations of upper bounds are easily modified into lower bounds $L(s_i)$ and $L(s, \vec{a}, \hat{s})$ by replacing the $\max$ operator by a $\min$ operator. It can be shown that both bounds are admissible with respect to the expected value that can be obtained from a given state, i.e. they never underestimate (upper bound) or overestimate (lower bound) the expected value.

**Lemma 4.9 Admissible heuristics**

The bounding heuristics $L(s)$ and $U(s)$ are admissible with respect to the expected value $V(s)$ obtained from a joint state $s \in \boldsymbol{S}$ onward.

*Proof.* See Appendix A.2. □

The upper and lower bound are computed for every state of every CRG after the initial generation step that implements Definition 4.5. Using a recursive depth-first procedure based on Equations 4.17 and 4.18 the bounds are determined and annotated in every state of the CRG.

### 4.2.3 Conditional Reward Independence

So far it was shown that conditional return graphs offer a compact representation of all local returns – the accumulated reward for every sequence of local transitions – and enable the bounding and storage of the minimum and maximum obtainable return from each state. In domains where interactions become unavailable as a result of past decisions, CRGs can help to exploit the consequential agent independence. This occurs for instance when actions can be executed only once or when transitions are unavailable when a state feature takes on a certain (irreversible) value ('task is done', 'out of fuel', 'money $<$ 400', etc.). Especially problems with transient state spaces, i.e. where every state is visited at most once, typically exhibit a lot of such dynamic independencies.

Task-modelling MMDPs, such as MPP, are a good example of a problem with such a structure. Maintenance activities may be performed only once, after which they are no longer available to the agent. Hence, once the activity is completed, no more reward interactions (the network costs in MPP) involving that activity will occur. The intuition is then that when no more reward interactions will occur between (sets of) agents, it is possible to factor the remaining computation of expected value into independently-computable components. In other words, when such independence is encountered, policy search can be decoupled into sub-problems that can be solved separately, and their solutions may be combined to obtain the joint policy without losing global optimality. This is formalised by the notion of conditional reward independence:

**Definition 4.10 Conditional Reward Independence**

Given an execution sequence $\theta^t$, two agents $i, j \in \boldsymbol{N}$ $(i \neq j)$ are *conditionally reward independent*, denoted $CRI(i, j, \theta^t)$, if for all reachable future states $s^t, s^{t+1} \in \boldsymbol{S}$ and every joint action $\vec{a}^{\,t} \in \boldsymbol{A}$ still available from the state at time $t$ that results from execution history $\theta^t$:

$$\forall \bar{R}_e \in \boldsymbol{R} \text{ s.t. } i, j \in e\colon \bar{R}_e(s^t, \vec{a}^{\,t}, s^{t+1}) = 0 \tag{4.19}$$

Furthermore two sets of agents $A, B \subseteq \boldsymbol{N}$ such that $A \cap B = \emptyset$ are conditionally reward independent, denoted $CRI(A, B, \theta^t)$, if as a result of an execution sequence $\theta^t$:

$$\forall i \in A,\ \forall j \in B\colon\ CRI(i, j, \theta^t) \tag{4.20}$$

---

If for an agent $i$ and sequence $\theta^t$ it holds that $CRI(\{i\}, \boldsymbol{N} \setminus \{i\}, \theta^t)$ it has no more future reward interactions and is said to be *locally* conditional reward independent. This special case of CRI is typically easily detected from the local state of the agent during the generation of the CRG. For each such a state $s_i^t$, the CRG generation is completed using an optimal policy search that finds $\pi_i^*(s_i^t)$ (and only the local transitions $(s_i^t, a_i^t, s_i^{t+1})$ of that policy are included in the remaining part of the CRG).

**Lemma 4.11 CRI decouples returns**

Given an execution history $\theta^t = [s^0, \vec{a}^{\,0}, \ldots, s^u, \ldots, s^t]$ up to time $t$ that can be partitioned into two histories, $\theta^u = [s^0, \ldots, s^u]$ and $\theta^{u'} = [s^u, \ldots, s^t]$, and a disjoint partitioning of agent sets $\boldsymbol{N} = N_1 \cup N_2 \cup \ldots \cup N_k$ such that for every pair $N_a, N_b \in \boldsymbol{N}$ it holds that $CRI(N_a, N_b, \theta^u)$ when $a \neq b$, then the return can be decoupled as:

$$Z(\theta^t) = Z(\theta^u) + \sum_{i=1}^{k} Z_{N_i}(\theta_{N_i}^{u'}) \tag{4.21}$$

Here, $\theta_{N_i}^{u'}$ is the execution history of the agents in the set $N_i \subseteq \boldsymbol{N}$, starting from time $u$.

*Proof.* See Appendix A.3. □

---

A concluding remark to this section is that conditional reward independence is usually but not always easily detected from the current state. For instance, in MPP the state describes what tasks have been completed and this is enough to immediately determine CRI. Although this and many other domains allow the formulation of similar reward independence conditions, in general detecting CRI may need to iterate over all combinations of future states and actions to conclude that indeed no more reward interactions occur. Such a check would require $O((\left|\bar{S}\right|^2 \left|\bar{A}\right|)^n)$ in the very worst case ($\left|\bar{S}\right|$ and $\left|\bar{A}\right|$ being the maximum state and action set size as before) and may be computationally prohibitive for some domains.

## 4.3 Policy Search based on Returns

The conditional return graphs and related concepts together form the theoretical foundation of a new policy search algorithm, termed `Conditional Return Policy Search` (`CoRe`). The `CoRe` algorithm resembles a branch-and-bound style search over the joint policy space to find optimal policies for TI-MMDPs, relying on CRGs for efficient reward computation, bound-based pruning and decoupling of agents during the search. The algorithm searches through a DAG of joint states and actions that models the TI-MMDP, while maintaining a current state in each of the CRGs such that together they correspond to the global joint state. 'Performing' a joint transition $(s, \vec{a}, \hat{s})$ in this graph then comes down to selecting and following the paths $(s_e, \vec{a}_e, \hat{s}_e)$ in the CRGs that (locally) correspond to the joint transition. This enables `CoRe` to quickly retrieve cached rewards, stored on the paths, and bounds, stored at the CRG local state nodes. First, however, the CRGs need to be generated.

The very first step is to partition the reward functions over the agents such that $\boldsymbol{R} = \sum_{i \in \boldsymbol{N}} \boldsymbol{R}_i$, where each $\boldsymbol{R}_i$ is the set of reward functions assigned to agent $i$ as before, but the distribution of rewards impacts performance. Intuitively, assigning the rewards to agents such that their sizes are approximately equal would result in the most balanced CRGs and therefore a good distribution of complexity and dependence. Although this approach works well for MPP, the impact of reward function distribution on the performance of `CoRe` depends greatly on the problem domain and needs to be studied on a per-domain basis. Having established the preferred partitioning heuristic, each CRG $\phi_i$ for rewards $\boldsymbol{R}_i$ is generated according to Definition 4.5 with the addition of a backward pass to compute return bounds. Moreover, when local conditional reward independence is detected, the CRG is completed using a simple value-iteration algorithm that determines the optimal transitions $\tau_i$ and includes only those.

The `CoRe` main routine is shown in Algorithm 4.12. As with `Value Iteration` (Equation 2.6) and `Policy Iteration` (Algorithm 2.4) this algorithm computes the expected value of the optimal policy, adapting the algorithm to construct the optimal joint policy is trivial. The algorithm performs a recursive branch-and-bound search on the (sub-)set of agents $\boldsymbol{N}$. When CRI occurs between sub-sets of agents, they are decoupled during expected value computation. This is detected in line 5 using a connected component algorithm on a graph with nodes $\boldsymbol{N}$ and an edge $(i, j)$ for every pair of agents $i, j \in \boldsymbol{N}$ that are still dependent given the current execution sequence $\theta^t_{\boldsymbol{N}}$, or $\neg CRI(i, j, \theta^t_{\boldsymbol{N}})$, maintained during policy search. The connected components that are returned are exactly those sets $e$ of agents that are independently solvable (passed as $\boldsymbol{N}$ in the next recursion on line 13).

Now for every such a subset $e$, `CoRe` determines which joint action $\vec{a}^{\,t}_e$ from the set of available joint actions $A_e = \times_{i \in e} A_i$ for agents $e$ maximises the expected policy value (which includes the single-agent case). To enable pruning sub-optimal actions, however, first the maximal return of each such action (line 7) and the best minimal return (line 9) are computed. These are subsequently used in line 10 to discard any joint action that does not at least obtain the minimal return $L_{\max}$. Lines 11 to 15 compute the expected value of the joint action by summing over the expected values of all possible outcome states, which are found by recursing further until the planning

**Algorithm 4.12** `Conditional Return Policy Search (CoRe)`

**Require:** CRGs $\phi$, agent scope $\boldsymbol{N}$, current execution sequence $\theta^t_{\boldsymbol{N}}$, planning horizon $h$

1: **function** CoRe($\phi, \boldsymbol{N}, \theta^t_{\boldsymbol{N}}, h$)
2: **if** $t = h$ **then return** $0$
3: **end if**
4: $V^* = 0$
5: **for all** conditional reward independent subsets $e \subseteq \boldsymbol{N}$ given $\theta^t_{\boldsymbol{N}}$ **do**
6: **for all** $\vec{a}^{\,t}_e \in A_e$ **do**
7: $U(s^t_e, \vec{a}^{\,t}_e) = \sum_{s^{t+1}_e \in S_e} P(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e) U(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e)$ ▷ Comp. upper bounds
8: **end for**
9: $L_{\max} = \max_{\vec{a}^{\,t}_e \in A_e} \sum_{s^{t+1}_e \in S_e} P(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e) L(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e)$ ▷ Best lower bound
10: **for all** $\vec{a}^{\,t}_e \in A_e$ with $U(s^t_e, \vec{a}^{\,t}_e) \geq L_{\max}$ **do**
11: $V_{\vec{a}^{\,t}_e} = 0$, $V_{\max} = 0$
12: **for all** $s^{t+1}_e \in S_e$ with $P(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e) > 0$ **do**
13: $V_{\vec{a}^{\,t}_e} += P(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e) \boldsymbol{R}_e(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e)$ ▷ Eval. next state
$\quad + \; P(s^t_e, \vec{a}^{\,t}_e, s^{t+1}_e)$CoRe$(\phi, e, \theta^t_e + [\vec{a}^{\,t}_e, s^{t+1}_e], h)$
14: **end for**
15: $V_{\max} = \max(V_{\max}, V_{\vec{a}^{\,t}_e})$
16: $L_{\max} = \max(L_{\max}, V_{\vec{a}^{\,t}_e})$ ▷ Tighten lower bound
17: **end for**
18: $V^* += V_{\max}$
19: **end for**
20: **return** $V^*$
21: **end function**

horizon has been reached. Line 16 is included to update the lower bound with the expected value of the joint action that was just evaluated, if higher, and potentially prune even more of the joint-action iterations done in line 10. Finally, the optimal value for the subset $e$ is added to the known optimal values, which is returned ultimately.

**Theorem 4.13** `CoRe` **correctness**

Given TI-MMDP $M = \langle \boldsymbol{N}, \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R} \rangle$ with (implicit) initial state $s^0$, `CoRe` always returns the optimal expected value $V^*(s^0)$.

*Proof.* Given Lemmas 4.9 and 4.11 the proof becomes rather straightforward. First it is shown that agent partitioning based on CRI does not harm optimality when computing the expected policy value. Then it is show that the branch-and-bound procedure of `CoRe` is correct.

Lemma 4.11 shows that conditional reward independence allows for a decoupling of returns. Additionally, for any disjoint partitioning of agents $N_1 \cup N_2 \cup \ldots, N_k = \boldsymbol{N}$ with for every pair of agents $N_a, N_b \in \boldsymbol{N}$: $CRI(N_a, N_b, \theta^t)$ $(a \neq b)$ it can be shown that the expected value of a policy $\pi$ decouples as:

$$V^\pi(s^t) = \sum_{i=1}^{k} V^\pi_{N_i}(s^t_{N_i}) \tag{4.22}$$

From Equation Equation 4.5, recall that the return $Z(\theta^{u'})$ from timestep $u$ onwards is equal to $\sum_{i=1}^{k} Z_{N_i}(\theta^{u'}_{N_i})$ (Lemma 4.11) and that, because of transition independence, each set $N_i$ of agents has independent probability distributions over future execution histories $\theta^{u'}$. This leads to the following series of equalities

$$
\begin{aligned}
V^{\pi}(s^t) &= \sum_{\theta^{u'}|\pi,\theta^t} Pr(\theta^{u'}) Z(\theta^{u'}) = \sum_{\theta^{u'}|\pi,\theta^t} Pr(\theta^{u'}) \sum_{i=1}^{k} Z_{N_i}(\theta^{u'}_{N_i}) \\
&= \sum_{\theta^{u'}|\pi,\theta^t} \sum_{i=1}^{k} Pr(\theta^{u'}) Z_{N_i}(\theta^{u'}_{N_i}) = \sum_{i=1}^{k} \sum_{\theta^{u'}|\pi,\theta^t} Pr(\theta^{u'}) Z_{N_i}(\theta^{u'}_{N_i}) \\
&= \sum_{i=1}^{k} \sum_{\theta^{u'}_{N_i}|\pi,\theta^t} Pr(\theta^{u'}_{N_i}) Z_{N_i}(\theta^{u'}_{N_i}) = \sum_{i=1}^{k} V^{\pi}_{N_i}(s^t_{N_i})
\end{aligned}
$$

Furthermore, Lemma 4.9 shows that both the lower and upper bound return heuristics used in CoRe are admissible with respect to the optimal expected value of a policy. It remains to show that its branch-and-bound search does not prune optimal joint actions from the policy search space. Let $V(s^t, \vec{a}^{\,t})$ denote the expected value of joint action $\vec{a}^{\,t}$ from state $s^t$ (often referred to as the Q-value in MDP literature), e.g.

$$
V^{\pi}(s^t, \vec{a}^{\,t}) = \sum_{s^{t+1} \in \boldsymbol{S}} P(s^t, \vec{a}^{\,t}, s^{t+1}) \big( \boldsymbol{R}(s, \vec{a}, \hat{s}) + V^{\pi}(s^{t+1}) \big) \tag{4.23}
$$

and the bounds on this action value are given by

$$
B(s^t, \vec{a}^{\,t}) = \sum_{s^{t+1} \in \boldsymbol{S}} P(s^t, \vec{a}^{\,t}, s^{t+1}) \big( \boldsymbol{R}(s, \vec{a}, \hat{s}) + B(s^{t+1}) \big) \tag{4.24}
$$

where $B$ is either the lower bound $L$ or upper bound $U$. Following standard branch and bound, CoRe may discard a joint action $\vec{a}$ when there exists at least one other joint action $\vec{a}\,'$ such that $U(s^t, \vec{a}) < L(s^t, \vec{a}\,')$ because both bounds are admissible. Because CoRe evaluates every combination of joint states and actions, unless pruned by branch and bound, the search will eventually return the optimal policy value $\pi^*(s^0)$ from initial state $s^0$. □

Because of Theorem 4.13, the CoRe algorithm always finds the optimal policy for any given TI-MMDP. Moreover, as there is only a finite number of execution histories, it is also guaranteed to terminate in a finite number of recursions. The working of the algorithm is illustrated in detail in the elaborated example following next.

**Example 4.14** `Conditional Return Policy Search`

All of the concepts of CoRe are illustrated on a two-agent example MPP problem with horizon $h = 2$. Let $\boldsymbol{N} = \{1, 2\}$ and the activities be unit-time actions $A_1 = \{a_1, b_1, c_1\}$ and $A_2 = \{a_2, b_2, c_2\}$. For ease of exposition only action $c_2$ is stochastic, with outcomes $c$

and $\bar{c}$ and probabilities $Pr(c) = 0.75$ and $Pr(\bar{c}) = 0.25$ (both still taking unit time). Local reward functions of agents are typical private maintenance cost functions $c_1$ and $c_2$, and in this example there are two network cost functions $\ell$ that lead to interaction rewards.

For the purpose of illustrating the potential of CoRe in this example, here a deviation is made from the standard network cost functions of MPP. The first network cost function $\bar{R}^{aa}_{1,2}$ penalises both agents if actions $a_1$ and $a_2$ are performed concurrently (at any time), but the cost depends on whether or not agent 1 is blocking the road from all traffic. This is modelled through feature $X_1$ of agent 1 that can assume values $X_1^+$ (not blocked) or $X_1^-$ (blocked). Furthermore, actions $a_1$ and $c_2$ also cause hindrance, but only when action $c_2$ results in outcome $\bar{c}$, e.g. because of the additional work required. The latter network cost is captured by $\bar{R}^{ac}_{1,2}$. Both interaction rewards are assigned to agent 2, thus $\boldsymbol{R}_1 = \bar{R}_1$ and $\boldsymbol{R}_2 = \bar{R}_2 \cup \bar{R}^{aa}_{1,2} \cup \bar{R}^{ac}_{1,2}$.

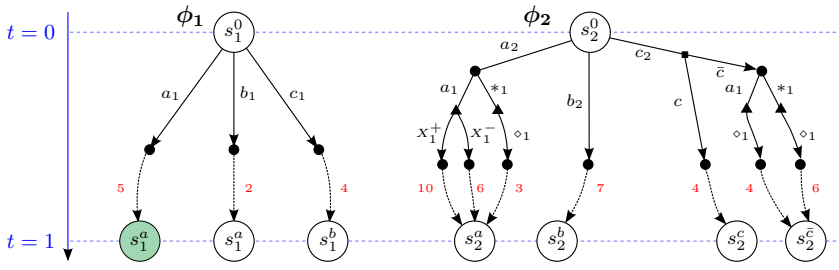



**Figure 4.5** *The CRGs of agents 1 (left) and 2 (right). The local states $s_i^t$ are grouped vertically by the planning stage $t$. $\phi_1$ only models the local rewards (red), and hence just local transitions, of agent $i$ ($*_2$ and $\diamond_2$ are both omitted if no alternative is available). The CRG $\phi_2$ of agent 2 includes dependent actions and transition influences. The state $s_1^a$ is highlighted green because it is locally conditional reward independent.*

Figure 4.5 shows the CRGs for both agents for the first time step. The CRG $\phi_1$ of agent 1 is rather straightforward because it represents just the local reward function $\bar{R}_1$. As a result, $\phi_1$ is a standard state/action transition diagram with the addition of the reward arcs; the $\diamond_2$ branches have been omitted as no transition influence occurs. Observe that state $s_1^a$ is locally conditional reward independent because after execution of $a_1$ it cannot perform $a_1$ again and, as $\bar{R}^{aa}_{1,2}$ and $\bar{R}^{ac}_{1,2}$ both cause network costs only when action $a_1$ is performed, no future reward interactions can occur. Observe also that in this specific example all actions lead to a unique new state, according to the MPP MDP formulation of the previous chapter. In general an arbitrary number of actions may lead to the same result state (resulting in even more compact CRGs).

Agents 2 however has a much more complex CRG. For action $b_2$ a simple transition again suffices, but actions $a_2$ and $c_2$ may cause reward interaction and thus induce more CRG arcs. First consider action $a_2$ that interacts with action $a_1$, i.e. $D((s_2^0, a_2, s_2^a), 1) = \{a_1\}$. The action tree part for this transition contains thus an arc for action $a_1$ and all remaining actions $\{b_1, c_1\}$, represented by wildcard $*_1$. Recall that when joint action $\langle a_1, a_2 \rangle$ is performed, the reward depends on whether the road is blocked or not. Therefore the transition influence $I((s_2^0, a_2, s_2^a), a_1)$ contains the state pairs $\{(X_1^+ \to X_1^-), (X_1^- \to X_1^+)\}$ that model the feature value change – assuming it cannot remain the same for sake of simplicity – and they are shown in the CRG labelled by their *new* state value, e.g. $X_1^-$ in the CRG denotes $(X_1^+ \to X_1^-)$, followed by the reward arcs for re-

wards $\boldsymbol{R}_2(s_2^0 \cup X_1^-, \langle a_1, a_2 \rangle, s_2^a \cup X_1^+) = \bar{R}_2(s_2^0, a_2, s_2^a) + \bar{R}_{1,2}^{aa}(s_2^0 \cup X_1^-, \langle a_1, a_2 \rangle, s_2^a \cup X_1^+) = 10$ and $\bar{R}_2(s_2^0, a_2, s_2^a) + \bar{R}_{1,2}^{aa}(s_2^0 \cup X_1^+, \langle a_1, a_2 \rangle, s_2^a \cup X_1^-) = 6$. From the 'other action' branch $*$ no dependency occurs, hence it is followed bt the $\diamond$ influence arc and finally the reward $\bar{R}_2(s_2^0, a_2, s_2^a) = 3$.

In this example, $c_2$ is the only stochastic action and has two possible outcomes: $c$ or $\bar{c}$. For clarity, each outcome is included as an additional arc after action $c_2$ (after the square node) but note that this is actually implied the result state of each transition, here $s_2^c$ and $s_2^{\bar{c}}$. Recall that $c_2$ in $\bar{R}_{1,2}^{ac}$ interacts with just action $a_1$ and then only when outcome $\bar{c}$ is realised. Therefore, the $c$ path is simple whereas the path with $\bar{c}$ contains an action tree part similar to that of $a_2$, with all leaves followed by the 'any influence' $\diamond_1$ branch and the rewards $\boldsymbol{R}_2(s_2^0, \langle a_1, a_2 \rangle, s_2^c) = 4$ and $\boldsymbol{R}_2(s_2^0, \langle a_1, a_2 \rangle, s_2^{\bar{c}}) = 6$.

The process for CRG creation is continued for $t = 2$ and the result is shown in Figure 4.6, where some parts of CRG $\phi_2$ are omitted for clarity (greyed branches with their local actions). In addition to the previous figure, this one shows the upper and lower return bounds as pairs $[L, U]$ annotated at local states and intermediate CRG nodes when they are non-trivial.

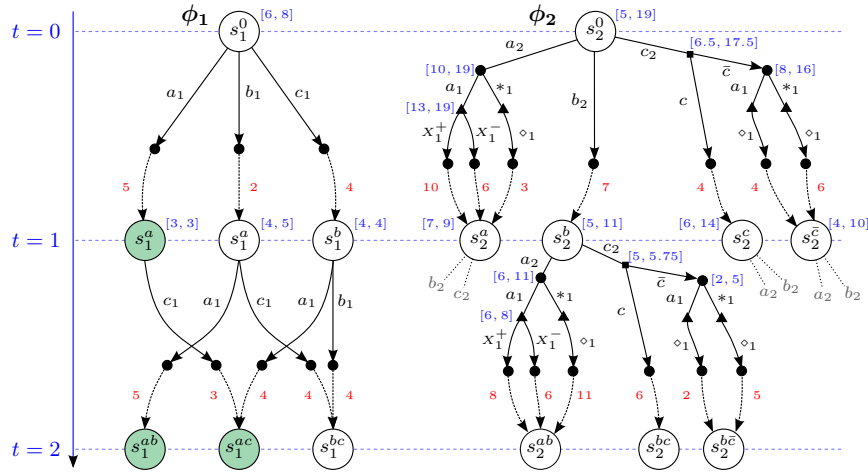



**Figure 4.6** *The CRGs from the previous example expanded for the second timestep. To preserve readability only the transitions from state $s_2^b$ are shown in $\phi_2$. Return bounds are shown next to nodes as $[L, U]$ pairs.*

A few interesting aspects are highlighted here. First note that return of stochastic action $c_2$ is an expected return, thus also its bounds are expected. Consider the state $s_2^b$ and the action $c_2$ directly after it, the bounds at the stochastic node (black square) are expected values over both outcomes, e.g. $[5, 5.75] = Pr(c) \times [6, 6] + Pr(\bar{c}) \times [2, 5]$; all other bounds are simply the minimum and maximum remaining return. Also interesting is that after state $s_1^a$ in $\phi_1$ only one transition is included. Due to the local independence in $s_1^a$ it is possible to compute locally optimal policy $\pi^*(s_1^a)$ from which just the optimal remaining transition(s) are added to the CRG (chosen randomly as $c_1$ in this example). Observe that of course the next state $s_1^{ac}$ is locally independent but also the state $s_1^{ab}$ as action $a_1$ is completed. Finally, notice that $\phi_1$ illustrates multiple transitions leading to the same result state in the CRG.

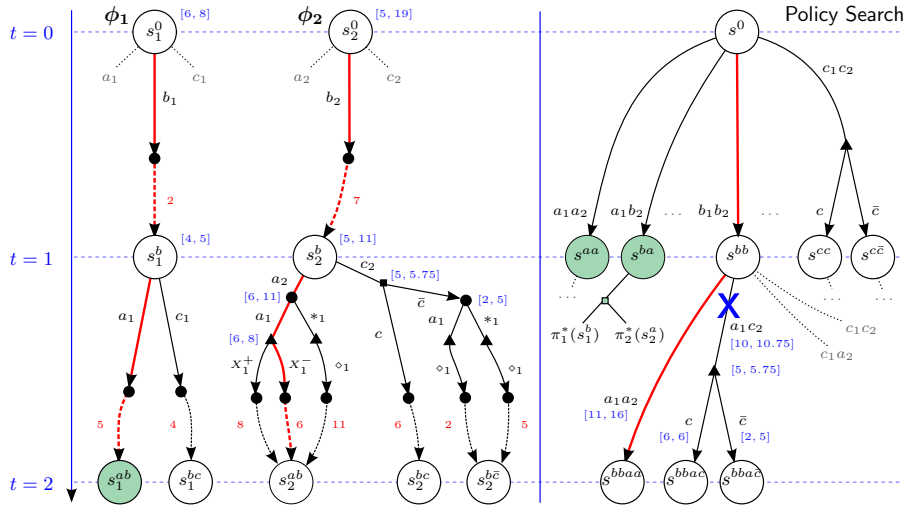



**Figure 4.7** CoRe *policy search in progress. On the left the CRGs are shown with highlighted in red the execution sequences that correspond to the current joint execution sequence, shown in the policy search tree on the right. The branch marked with the blue cross can be pruned due to branch and bound. The green highlighted states in the search tree result in the two agents becoming independent.*

An example of the CoRe algorithm at work is given in Figure 4.7. On the left are the previous CRGs, minimised to show only the elements relevant for the current policy search evaluation, displayed on the right in the policy search tree. The tree shows a handful of the available joint actions at time $t = 1$, with their result states, and only two of the four possible joint actions for $t = 2$ from *joint* state $s^{bb}$ that results after execution of joint action $\langle b_1, b_2 \rangle$ at $t = 1$. An example execution sequence $\theta^2 = [s^0, \langle b_1, b_2 \rangle, s^{bb}, \langle a_1, a_2 \rangle, s^{bbaa}]$ is highlighted red in the policy search tree, as are the corresponding local sequences $\theta_1^2 = [s_1^0, b_1, s_1^b, a_1, s_1^{ab}]$ and $\theta_2^2 = [s_2^0 \cup X_1^-, \langle b_1, b_2 \rangle, s_2^b \cup X_1^+, \langle a_1, a_2 \rangle, s_2^{ab} \cup X_1^-]$, such that $X_1^- \in s_1^0$ in this example. Combining the rewards on both CRG paths, the total return for this particular sequence is $(2 + 7) + (7 + 6) = 22$.

Through branch-and-bound pruning, joint action $\langle a_1, c_2 \rangle$ from state $s^{bb}$ at time $t = 1$ is discarded because the upper bound on its return is $10.75$ (summed from the CRGs: $(5) + (6 \times 0.75 + 5 \times 0.25)$) whereas action $\langle a_1, a_2 \rangle$ is guaranteed to yield at least $11$. Conditional reward independence occurs in the policy search tree at the green states, for example in the joint state $s^{ab}$. Note that previously state $s_1^a$ was identified locally independent but state $s_2^b$ could not have been. Only when it has become known that agent $1$ completed $a_1$, it is sure that no further reward interaction occur, which agent $2$ could not have detected locally. In this two-agent example, the remaining optimal transitions are found through individual optimisation (possibly with a more efficient single-agent algorithm), resulting in optimal policies $\pi_1^*(s_1^a)$ and $\pi_2^*(s_2^b)$ for agents $1$ and $2$ respectively, and they are combined to produce all optimal joint actions. Had there been more agents, e.g. a third that is only interacting with agent $2$, the search would be decoupled into an individual optimisation of agent $1$ and a continued joint policy search with agents $2$ and $3$.

## 4.4 Experimental Evaluation of CoRe

The potential of CoRe is analysed through an experimental evaluation on instances of MPP. In a series of tests, the performance of the algorithm is investigated in terms of the leverage it provides, by means of joint actions pruned, and how it compares to the optimal SPUDD-based method of the previous chapter, in regard to both runtime and scalability in the planning horizon and number of agents. Four experiments are outlined here that study respectively

1. the number of evaluated joint actions to quantify the impact of branch and bound and conditional reward independence on the (reduction of) policy search space,
2. the solving coverage of CoRe versus its alternatives when given random MPP problems to verify that it enables more problems to be solved,
3. the runtime required to solve random MPP problems to compare CoRe against its alternatives, and
4. the solving coverage when given problems that exhibit a CoRe-favourable structure to demonstrate the scalability of the algorithm.

For the experiments four algorithms are considered. The first is the simple depth-first policy search described in Section 3.2, abbreviated DP, that functions as a baseline against which all other algorithms can compare. SPUDD represents the exact MDP-based solution from the previous chapter that generates an efficient MDP encoding of the problem, solved by the SPUDD solver (see Section 3.5). CoRe is the conditional return policy search algorithm as presented in this chapter. Finally, mainly intended to study the impact of branch-and-bound pruning, an additional variant of CoRe with pruning disabled is included (CRG-PS). Note that the latter variant still detects and exploits conditional reward independence to decouple policy search.

The benchmark set for the first three experiments, random, is a set of 2, 3 and 4 agent MPP instances (400 each) in which agents have 3 maintenance activities, the planning horizon varies from 5 to 10, the delay probabilities are random and the reward interactions are all binary. For the last series of tests, aimed at demonstrating the scalability of the CoRe algorithm, another test set is used. This set, referred to as pyra, contains instances with a number of agents varying from 2 up to 10, where each agent has again 3 activities but now the interactions are structured in a way that is favourable to CoRe. For the first experiments, however, the random set is used.

The first experiment studies the number of joint actions that need to be evaluated before reporting that the optimal policy has been found, to investigate the impact of branch and bound and conditional reward independence. The dynamic programming algorithm DP evaluates exactly all sequences of joint actions that lead to a terminal state, a state in which all activities are completed, and hence functions as a baseline in this experiment. CRG-PS is expected to reduce this number because it decouples action decisions whenever agents become independent due to CRI. CoRe should reduce this number even more with its branch-and-bound pruning. SPUDD does not report this information and is therefore excluded from this test.

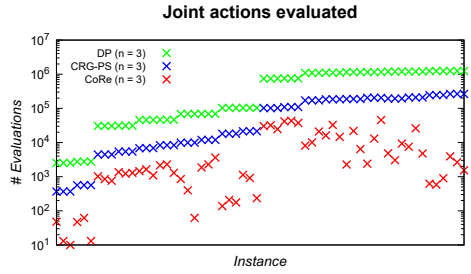
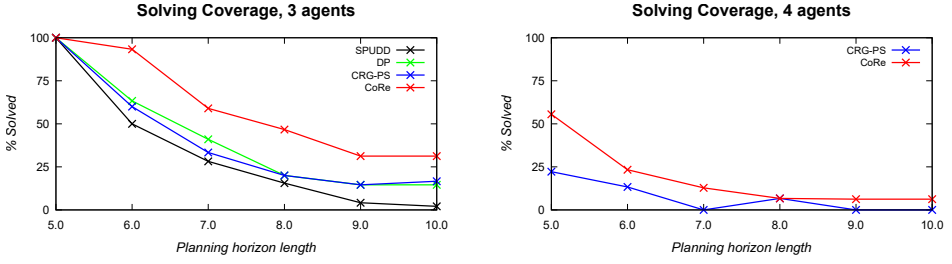



**Figure 4.8** *Number of joint actions evaluated in 3-agent instances from* `random` *by* DP, CRG-PS *and* CoRe *(log scale, ordered by the number of* DP *evaluations).*

The results of the experiment are shown in Figure 4.8, depicting the number of joint actions that were evaluated per algorithm. The outcomes have been ordered based on the amount of evaluations that DP needed. First note that the vertical axis uses a log scale and that CRG-PS thus decreases the amount of joint-action evaluations by an order of magnitude over DP only through exploiting conditional reward independence. When branch and bound is also included, i.e. through the CoRe algorithm, this reduction can be much better. Although its effect varies per instance, the number of evaluations is always reduced and for many instances this decrease is of a factor between $10^2$ and $10^3$.




(a) (b)

**Figure 4.9** *Relative solving coverage (y axis) of all algorithms of instances from the* `random` *benchmark set versus the planning horizon length (x axis), for 3 agents (a) and 4 agents (b).*

Having observed the policy search space reduction achieved by the CRG-enabled algorithm, the next experiment focuses on the scalability of the algorithms in terms of number of agents and planning horizon. Because CoRe exploits the structure of the problem, and therefore the hypothesis is that more problems will be solved with respect to the other approaches. Figure 4.9 shows the solving coverage, the percentage of problems solved within a 30-minute time limit, of all algorithms with respect to the `random` test set.[40] When considering the 3-agent instances (Figure 4.9a) it can be

[40] *All two-agent instances were solved by all algorithms within the time limit and are hence omitted.*

seen that all algorithm scale comparably, with CoRe having a slight advantage over the others. For the 4-agent instances (Figure 4.9b), however, the picture changes. Only the CRG-enabled methods were able to solve instances of this size, where naturally CoRe outperforms the "branch-and-bound-less" variant CRG-PS. The main reason for this difference is that CRGs successfully exploit the conditional action independence that decouples the agents *for most of the planning decisions*. Only when reward interactions may occur actions are coordinated, whereas SPUDD coordinates all joint decisions.

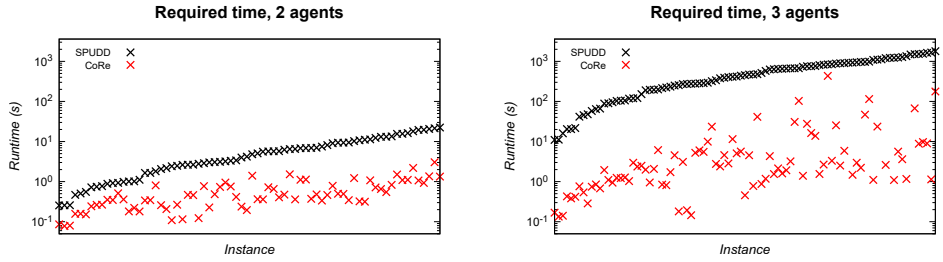



**Figure 4.10** *Average runtime (y axis, log scale) of* SPUDD *and* CoRe *to solve* random *instances for 2-agent (a) and 3-agent (b) instances solved by both (ordered by* SPUDD *runtime).*

Given the fact that CoRe solves more problems than the other approaches, it is also expected to be faster overall. Figure 4.10 visualises the runtimes of SPUDD and CoRe taken in the previous experiment, sorted on their SPUDD runtime. As CoRe achieves a greater coverage than SPUDD, outcomes that have not been solved by SPUDD are excluded. CoRe solves all but the most trivial instances faster than SPUDD both with 2 and 3 agents, and has a greater solving coverage: CoRe failed to solve 3.4% of the instances plotted, whereas SPUDD failed 63.9% of the instances that CoRe solved (which are thus not in this figure).

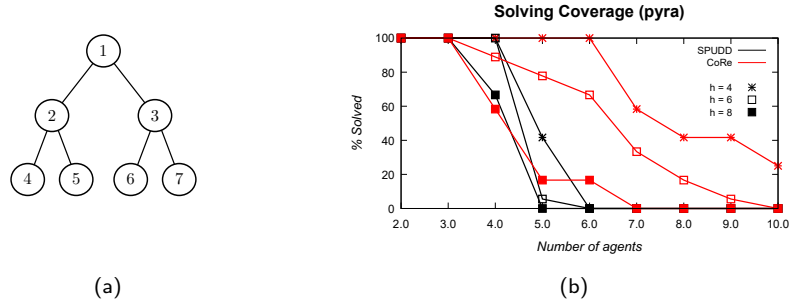



**Figure 4.11** *The* pyra *benchmark tests: (a) the pyramid-like reward-interaction graph with edges between agents (nodes) that are reward dependent, illustrated for 7 agents, and (b) the solving coverage (y) of* SPUDD *and* CoRe *versus the number of agents (x axis) of instances of the* CoRe*-favourable* pyra *instances.*

As a final experiment, CoRe is tested against the set pyra a set of problems that is structured in such a way that it exhibits a lot of conditional reward independence. This experiment is mainly to demonstrate that if indeed the planning problem possess a structure that is targeted by CoRe, the algorithm scales really well in comparison to other approaches. In instances of this set, every first activity of the $k$-th agent depends on the first action of agents $2k$ and $2k+1$, resulting in the pyramid-like dependency structure of Figure 4.11a. As a result, performing a dependent action causes the set of agents to be partitioned into two sets, at best halving the policy search space. Indeed, Figure 4.11b shows that this favourable structure allows the CoRe algorithm to scale up to solving instances with 10 agents (with $h = 4$), whereas SPUDD can do no better than solving 5-agent instances.

## 4.5 Further Discussion

This chapter presents a novel optimal policy search algorithm for multi-agent MDPs that are transition-independent (TI-MMDPs) based on a data structure known as Conditional Return Graphs (CRGs). These graphs provide efficient storage, bounding and decoupling of rewards when inter-agent reward interactions are sparse and/or limited. The focus of this chapter is on transition-independent MMDPs because problems that fit this model typically possess a structure that CoRe can exploit well. The need for transition independence, however, can be alleviated by two alterations.



First of all, the CRGs represent state transitions that are local to its corresponding agent. Due to the transition independence property actions of other agents only influence the reward of a transition, not the state that the agent ends up in. Therefore the definitions of dependent actions (Definition 4.3) and transition influence (Definition 4.4) need to be extended to also include those actions and transitions that do not necessarily impact reward but at least influence the local state. Note that in general these actions and influences may belong to any of the agents outside the scope of the CRG rewards $\boldsymbol{R}_i$.

The second change required to drop the transition independence requirement lies with the decoupling of agents. Without the presence of transition independence, conditional reward independence is no longer a sufficient condition to decouple agents because, although their returns can be decoupled, the expected value function that includes transition probabilities cannot. The correctness proof of Theorem 4.13, and in particular the part that establishes Equation 4.22, relies on the independence of transition probabilities to decouple execution sequences. This can be resolved by incorporating the transition independence condition explicitly in the definition of CRI. In essence, two disjoint sets of agents $A$ and $B$ can be decoupled as a result of an execution sequence if they are CRI and for all future reachable states and every remaining future joint action the transition probabilities between sets $A$ and $B$ are independent, i.e. $P(s^t_{AB}, \vec{a}^{\,t}_{AB}, s^{t+1}_{AB}) = P(s^t_A, \vec{a}^{\,t}_A, s^{t+1}_A) \times P(s^t_B, \vec{a}^{\,t}_B, s^{t+1}_B)$. This additional condition guarantees that no more reward dependencies as well as transition dependencies can occur between (groups of) agents. Still it must be remarked that, although it is possible to extend the method to general MMDPs, it will likely be effective only on instances where reward and transition interactions are very sparse/limited. The addition

of transition dependence will lead to an enormous blow-up of CRG sizes for multi-agent problems with many inter-agent interactions.

Asides the generalisation of the CoRe method, a number of other improvements can be made to the implementation presented in this chapter, relevant for both the TI-MMDP and MMDP case. Although it has not been discussed earlier, the execution sequences can be used to prune states, actions and/or entire transitions during the generation of the CRGs. This technique is shown in Example 4.14 where for instance action $a_1$ is no longer included as a branch from state $s_1^a$ because it has been completed previously, detectable from the execution sequence $\theta_1 = [s_1^0, a_1, s_1^a]$ leading to the state. This sequence-pruning is possible from the MDP model. For example, any action $a_i$ for which no transition $(s_i, a_i, \hat{s}_i)$ exists such that $P(s_i, a_i, \hat{s}_i) > 0$ can be omitted during CRG creation, but more intelligent procedures may be developed to exclude more arcs and nodes from the generation to minimise the size of the CRGs.

The CoRe main loop of Algorithm 4.12 can be strengthened by interleaving joint action selection with agent set decoupling. Now the algorithm always generates and evaluates joint actions of size $|e|$ for the sets $e \subseteq \boldsymbol{N}$ that result from the decoupling in line 5 of the algorithm. More efficient could be selecting actions one by one and decouple agents immediately when the action choice results in no more future interactions *after* completing the action, thus interleaving action selection and CRI decoupling. For instance, consider 5 agents that depend all on the action $a$ of their neighbour, i.e. $a_1$ depends on $a_2$ whereas $a_3$ depends on $a_2$ and $a_4$ and so on, but also have additional non-interacting actions. When agent $3$ is the first agent for which an action is selected, and the selected action is $a_3$, the agent set can immediately be decoupled in sets $\{1, 2\}, \{3\}$ and $\{4, 5\}$. Not only does this result in small, easily solvable problems *for all joint actions involving* $a_3$, it also leads to a quick global upper bound on the reward. How to order agents and do this agent selection to maximise the benefit of decoupling remains open for future work. Other techniques proposed in the literature to decouple agents are also promising, e.g. the work by de Nijs et al. on decoupling agents using marginal utility costs [189] or constraint decoupling [191].

Besides the optimal algorithm presented here, CRGs can be employed to approximate TI-MMDPs in several ways. First, the reward structure of the problem itself may be approximated. For instance, the reward-function approximation of Koller and Parr [145] can be applied to increase reward sparsity, or CRG paths with relatively small reward differences may be grouped, trading off a (bounded) reward loss for compactness. Secondly, the CRG bounds directly lead to a bounded-approximation variant of CoRe, usable in e.g. the approximate multi-objective methods presented in the next chapter. Lastly, the CRG structure can be implemented in any (approximate) TI-MMDP algorithm or, vice versa, any existing approximation scheme for MMDP that preserves TI can be used within CoRe.

It must be noted that although the experimental evaluation of Section 4.4 illustrates that the performance of CoRe is outstanding, it is unclear how it would relate to other (TI-)MMDP methods and perform on other domains. At the point of writing this, there is no clear TI-MMDP alternative to compare against. Still, the results here show a significant improvement compared to the highly-optimised, state-of-the-art MDP solver SPUDD. Furthermore, only the MPP domain has been considered in these experiments.

Although MPP is a perfect example of a problem that fits the TI-MMDP model that typically has sparse reward interactions, further research should try and transfer the results onto other domains.

Finally, although this work has been inspired by approaches from decentralised (PO)MDP methods, the fully-observable MMDP model targets a fundamentally different class of problems. In the latter all agents observe the global state and therefore policies are conditioned on the entire state. In decentralised models, the joint policy is a combination of local policies that, although possibly developed centrally, only contain decisions based on the local state of one agent. The key difference is that, because of full observability in TI-MMDP, the value function cannot be factorised as a sum of localised components as in Theorem 4.13. Additionally, Dec-(PO)MDPs exploit observational independence that is not present in MMDP. Arguably Dec-MDP methods can be applied to *approximate* MMDP policies, but will at best equal the expected value of the latter and may be arbitrarily worse in domains where a high level of joint coordination is required (e.g. responding immediately to activity delays). For an experimental confirmation thereof see the original paper [229].

# Chapter 5

# Maintenance Planning with Multiple Objectives

So far several approaches have been presented to solve the maintenance planning problem. Optimal policies can be found using the SPUDD MDP encoding of Chapter 3 or the Conditional Return Policy Solver (CoRe) that more efficiently exploits the structure of the problem. Additionally, in Section 3.4 an approximation method for MPP was presented that is more suitable if efficiency is required. Although these approaches vary in their quality guarantees and, inevitably, their run-time complexity, they have in common that they all seek to maximise 'the value' of the joint policy. Typically, this value expresses a single monetary quantity that is associated with the policy, such as the expected total revenue or costs. Other criteria may be incorporated in this value if they can be operationalised through a scalar value, e.g. expressed in terms of money.

In many real-world problems the quality of a policy can be measured in many – possibly non-monetary – ways. For example the increase in asset quality, the hours of traffic time lost that are expected to be incurred/prevented by the policy, the number of concurrent resources required to perform the maintenance, the quality level or even the minimum level of safety of the maintenance work. Although many of such *objectives* can be operationalised in terms of money, this is not always possible the moment the policy is developed. For instance, the cost of an hour of ttl may vary over time and hence it is impossible to know during planning what the actual cost will be when the policy is executed. Moreover, every agent may have its own preference over these objectives that can change as the planning and/or execution of maintenance progresses or might be hard to quantify in the first place. In such scenarios it is not possible to produce a single optimal joint policy that optimises over all objectives before its execution. The typical approach in such settings is to find a set of policies that defines an optimal policy for every unique combination of preferences, known as *weights*. Then, when these weights become known just before the execution phase, the optimal policy for that moment can simply be selected from the solution set.

Policies in a multi-objective setting yield a multi-dimensional expected value, therefore a measure to express the overall policy quality as a single value is still necessary

to produce such a solution set. For this purpose typically a *scalarisation function* is considered in multi-objective planning. The scalarisation function operationalises the policy quality over all objectives based on scalarisation weights that express the value-per-unit or (relative) preference for each objective. Given such a vector of weights, the scalarisation function transforms the multi-dimensional (expected) policy value – e.g. a vector containing one weight value for every objective – into a scalar value that represents the total policy value *for the given weight vector*. Essentially, with a scalarisation function and a given weight vector, a multi-objective planning problem can be transformed to a single-objective equivalent and, consequentially, be solved using any existing single-objective planning method (formalised later in Equation 5.2).

Recall, however, that in the previously described scenarios the weights were not known during the planning phase and hence it is impossible to develop a single, optimal policy beforehand. Instead, a solution for this type of problem must specify an optimal policy for every possible combination of weights. A naive solution approach to find a set of 'good' alternatives would thus be to enumerate all possible weight combinations and solve their corresponding single-objective problems, returning only the dominating policies as the solution. Here a policy dominates another policy if its scalarised expected value is higher. However this approach is infeasible for most planning problems as the number of weight combinations is typically very large or even (near-)infinite.

Instead, by restricting the scalarisation function to be a *linear, non-decreasing* function, the solution set, i.e. the set of optimal policies for every combination of weights, equals a *Convex Coverage Set (CCS)* that can be computed much more efficiently. The main intuition is that when the scalarisation function is linear and non-decreasing, policies will be optimal for a range of weights and the CCS can easily be constructed by combining the optimal policies such that the entire weight region is covered. More specifically, for every optimal policy $\pi^*$ a *value vector* $\boldsymbol{w} \cdot \boldsymbol{V}^{\pi^*}$ can be constructed. Then, the CCS is simply the piece-wise linear convex (PWLC) function that is the supremum over all value vectors such that the entire weight range is covered. Figure 5.1 illustrates this intuition through a two-objective example.

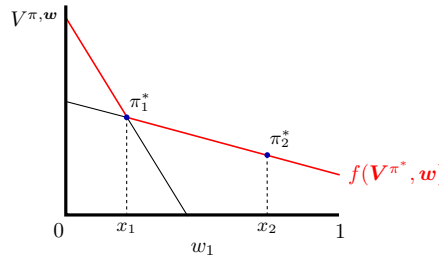



**Figure 5.1** *Example of a Convex Coverage Set for a two-weight problem that is optimal for all weights $w_1 \in [0, 1]$ such that $w_2 = 1 - w_1$. Due to the linearity and non-decreasing properties of $f$, the policy $\pi_1^*$ is optimal for $w_1 \in [0, x_1]$ whereas $\pi_2^*$ is optimal for the range $[x_1, 1]$.*

The linearity and non-decreasing assumptions restrict the types of problems that can be modelled, but such scalarisation functions capture many of the most natural

multi-objective operationalisations such as a revenue/cost per resource unit, relative preference over objectives or priority trade-offs. Note that this model also captures problems with non-increasing scalarisation function through a simple negation of the scalarisation function. Examples that do not fit the linearity constraints are problems with combinatorial rewards (e.g. a 'bonus' only when a set of items is obtained), random reward signals and similar more complex reward structures. The maintenance planning problem fits this model perfectly however: although the MDP model defined in Section 3.3 has a single-objective reward function, this reward is more realistically expressed linearly over two objectives, profits and ttl, e.g.:

$$\begin{aligned} &\boldsymbol{R}(s,\vec{a},\hat{s}) = \left\langle R_{profit}(s,\vec{a},\hat{s}), R_{ttl}(s,\vec{a},\hat{s})\right\rangle \text{ s.t.} \\ &R_{profit}(s,\vec{a},\hat{s}) = \sum_{\mathsf{a}_k\in\vec{a}} w_k - \sum_{t\in T} c(\vec{a}) \\ &R_{ttl}(s,\vec{a},\hat{s}) = \sum_{t\in T} \ell(\vec{a}) \end{aligned} \tag{5.1}$$

such that the profit objective value is computed by $R_{profit}(s,\vec{a},\hat{s})$ and that of the ttl objective by $R_{ttl}(s,\vec{a},\hat{s})$. Then the importance of each objective can be specified with weights $w_{profit}$ and $w_{ttl}$ such that a scalar reward for any transition $\tau = (s,\vec{a},\hat{s})$ is given by $w_{profit} \times R_{profit}(\tau) + w_{ttl} \times R_{ttl}(\tau)$.

In Section 5.1 an existing method to find the optimal CCS for any multi-objective MDP (MOMDP) is presented, which is called `Optimistic Linear Support (OLS)` [217]. Given an optimal MDP solver, `OLS` is able to find the CCS with a provably minimal number of policy computations. Therefore, using any of the methods from the previous chapters, `OLS` can solve the multi-objective variant of MPP optimally. However, the previous chapters also show that solving only one single-objective MDP of MPP is already a very time-consuming process. Computing the optimal CCS for MPP[41] is hence only feasible when enough time and resources are available beforehand.

This chapter focuses on multi-objective planning scenarios where the latter is *not* the case. Consider as an example a decision-support system for MPP that lets human planners collaboratively develop a joint policy through a series of iterative planning rounds (as in Section 6.2 or the game of Chapter 7). In each round, planners are presented the current, but not yet agreed upon, joint policy that resulted from all previous rounds together and they each are requested to submit a new policy for their own maintenance tasks until all parties are satisfied with the joint policy. An automated planner that provides policy suggestions would greatly assist human decision makers in this task, but `OLS` is not suitable in this scenario. Although `OLS` is able to incorporate user preferences (e.g. $w_{profit}$ and $w_{ttl}$) that may vary during the planning process, its time-demanding policy search makes it inadequate to provide many planning suggestions quickly. Finding the optimal CCS is not exactly a requirement in these scenarios. As long as the quality of the produced suggestions is 'good enough', it is acceptable to human decision makers in this and many similar settings.

[41] *As multi-objective planning is the subject of this chapter, it is implicitly assumed that by* MPP *the multi-objective variant of* MPP *is meant in this chapter, i.e. with the reward function of Equation 5.1.*

**Contributions** In this chapter, two methods are presented to approximate the CCS of multi-objective planning problems with varying weights: `Approximate Optimistic Linear Support` (`AOLS`) and `Scalarised Sample-based Iterative Improvement` (`SSII`), presented originally by Roijers et al. [216]. Both algorithms produce an approximate CCS, but differ in their approach. The former method, described in Section 5.2, is an approximate version of the `OLS` algorithm. Essentially, the theorem underlying `OLS` can be adapted to show that, given an $\epsilon$-approximate MDP solver, `AOLS` produces an $\epsilon$-approximate CCS with a minimal number of policy computations. Section 5.3 presents the latter approach, `SSII`, that approximates the CCS by sampling policies over (a part of) the weight range and iteratively improving these samples by allowing the incorporated single-objective MDP solver increasingly more time per sample. Both methods have their strengths and weaknesses, demonstrated by the experiments of Section 5.4. The main selling points of both algorithms are the bounded-approximation guarantee of `AOLS` versus the typically higher quality approximation of `SSII` over a known prior, i.e. a weight region of focus, given equal time.

The work presented in this chapter has been published and presented at the international conference on automated planning and scheduling (ICAPS) [216]. Because this has been a joined research effort, parts of this chapter – in particular regarding `OLS` and `AOLS` – are also discussed in the thesis of Roijers, titled "Multi-Objective Decision-Theoretic Planning" [214]. Also, Section 5.1 consists entirely of previous work by (mainly) Roijers et al. [217], but a thorough understanding of `OLS` is necessary for the presentation of the `AOLS` algorithm. The empirical evaluation of Roijers et al. [216] has been expanded in Section 5.4 to include the `CoRe` solver of Chapter 4.

## 5.1 Multi-objective Planning with Unknown Weights

Before presenting the two new algorithms mentioned in the introduction, this section reiterates the relevant concepts from Section 2.7 to formalise the setting that is considered in this chapter and discusses the previous state-of-the-art algorithm, `Optimistic Linear Support` [217], from which `AOLS` is derived. In multi-objective planning problems, the goal is to find all policies $\pi$ that optimise the vectored expected value $\boldsymbol{V}^{\pi}$. If the vectored expected value can be transformed into a scalar value through a scalarisation function $f$, that takes a vector of objective weights as its input, a single 'total' value over all objectives can be computed. Then, when the weight vector become known, the multi-objective problem can be transformed into an equivalent single-objective problem. More formally, given a linear scalarisation function $f$, a vector of scalarisation weights $\boldsymbol{w}$ and an $m$-dimensional value $\boldsymbol{V}^{\pi}$ for a policy $\pi$ with initial state $s^0$, the scalar value of a policy is given by

$$V^{\pi,\boldsymbol{w}}(s^0) = f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{V}^{\pi}(s^0) = \sum_{k \in [1,m]} w_k V_k^{\pi}(s^0) \tag{5.2}$$

Observe that when the values of these weights are known before planning starts, there is no need for multi-objective approaches. The setting considered here is there-

fore one in which weights specify for example the relative importance of each of the objectives, but the value of these weights are unknown until the execution of starts. In other words, an optimal policy for every combination of weights must be readily available when execution is initiated. When the scalarisation function is linear, the set of optimal policies for the entire weight range is captured by the *Convex Coverage Set* (Definition 2.23):

$$\Psi = \left\{ \pi \in \Pi \,\middle|\, \forall \boldsymbol{w} \in [0,1]^m, \forall \pi' \in \Pi, f, s^0 \colon f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) \geq f(\boldsymbol{V}^{\pi'}(s^0), \boldsymbol{w}) \right\} \quad (5.3)$$

Without loss of generality it is assumed that all objectives are desirable, thus they always assume positive values. To model non-desirable objectives, objective values can be negated. Moreover, as only relative importance matters when computing optimal policies, the weights are assumed to lie within the $[0, 1]$ range and sum to 1.

In previous work, Roijers et al. presented an algorithm to find the CCS for multi-objective planning problems with unknown weights [217]. Their `Optimistic Linear Support` (`OLS`) approach exploits the linearity property of the scalarisation function to minimise the number of policy searches required to find the CCS. In essence, `OLS` iteratively adds policies $\pi$ with corresponding value vectors $\boldsymbol{w} \cdot \boldsymbol{V}^{\pi}$ for specifically chosen weights to a partial CCS until the supremum over all its value vectors exactly matches the CCS value. It determines the weights to search based on a theorem that was originally presented in the context of POMDPs [60].[42] This theorem, due to Cheng, states that when comparing the value of an incrementally constructed partial CCS with that of the CCS, the largest difference will always be at a vertex of the former, i.e. a point at which the piece-wise linear convex function that is the supremum of all value vectors changes slope (see the work by Roijers [214] for their formal definition). These vertices are known as corner points or *corner weights*[43] and are exactly the weight vectors considered by the `OLS` algorithm in its iterations.

Put more formally, let $\Psi$ denote the CCS and $\psi$ the partial CCS that is being constructed by an algorithm such as `OLS`. Furthermore, let the value of a (partial) CCS $\Psi$ at a given weights vector $\boldsymbol{w}$ (with implicit initial state $s^0$) be defined as

$$V_{\Psi}(\boldsymbol{w}) = \max_{\pi \in \Psi} \boldsymbol{w} \cdot \boldsymbol{V}^{\pi} \quad (5.4)$$

then the following theorem holds:

**Theorem 5.1 Maximal CCS error**

Let $\Psi \subseteq \Pi$ denote a CCS for a multi-objective MDP. Then for any partial Convex Coverage Set $\psi \subseteq \Pi$, the weight vector $\boldsymbol{w}$ that maximises $V_{\Psi}(\boldsymbol{w}) - V_{\psi}(\boldsymbol{w})$ is a corner weight of $\psi$.

*Proof.* Given by Cheng [60]. □

[42] *Using the term 'parsimonious representation' for the Convex Coverage Set.*

[43] *Although the singular noun is used, a corner weight is actually a weight vector. The term is adopted here to remain consistent with previous work.*

An immediate corollary of this theorem is that, when searching for the CCS, only the corner weights of the partial CCS need to be considered, thus greatly limiting the size of the search space. In the worst case there are exactly as many corner weights as there are unique optimal policies for the MOMDP plus one, which is a major reduction compared to the infinite number of weight vectors in $[0, 1]^m$. However, the impact of Theorem 5.1 becomes really apparent when it is combined with the linearity property of the scalarisation function. Observe that if the scalarisation function $f$ is linear and all weight vectors $\boldsymbol{w}$ contain only zero or positive entries, the corresponding value $\max_{\pi \in \Psi} \boldsymbol{w} \cdot f(\boldsymbol{V}^\pi, \boldsymbol{w})$ of any (partial) CCS must be both linear as well as convex. Consequentially, the maximal possible CCS error can be expressed with respect to the best potential value, captured by the hypothetical *optimistic Convex Coverage Set*. Below the OCCS is defined formally, followed by a visualisation in Example 5.3.

**Definition 5.2 Optimistic Convex Coverage Set (OCCS) [217]**

An optimistic Convex Coverage Set (OCCS) $\bar{\psi}$ for a partial CCS $\psi$ is a set of value vectors $\boldsymbol{v}$ that yields the highest possible scalarised value for all $\boldsymbol{w} \in [0, 1]^m$, consistent with the value vectors $\boldsymbol{w} \cdot \boldsymbol{V}^\pi$ associated with policies $\pi \in \psi$.

**Example 5.3** Optimistic CCS

Figure 5.2a shows a partial CCS $\psi$, for a two-objective problem as before, that consists of two value vectors (red lines) for two (values of) previously found policies (red dots). The optimistic CCS, shown in green, is the maximum expected value any policy in between the two previous point can attain. Due to the linearity and convexity, no policy value can exceed the optimistic CCS. Moreover, the maximum error is bounded by the distance $\Delta_{\max}$ between the two PWLC functions.

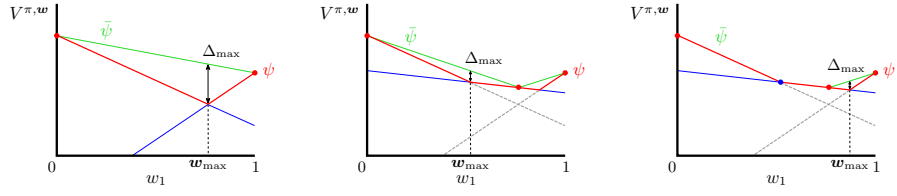



**Figure 5.2** *Two-objective examples showing the relation between the partial CCS $\psi$ (red) and optimistic CCS $\bar{\psi}$ (green) when (a) the partial CCS only contains two policies (dots) and value vectors (blue lines), (b) a third policy is added to the partial CCS that was found at $\boldsymbol{w}_{\max}$ and improves the partial CCS, and (c) a fourth, non-improving policy is found and added. The maximal possible error $\Delta_{\max}$ is always at a corner weight $\boldsymbol{w}_{\max}$ of $\psi$.*

Figure 5.2b shows an updated situation where a policy $\pi$ is added that is optimal at the weight vector $\boldsymbol{w}_{\max}$ of the left figure. This new policy improves the partial CCS, as its PWLC value function now contains a bigger area. Moreover, the optimistic CCS must be

updated to remain consistent the new with value vector $\boldsymbol{w} \cdot \boldsymbol{V}^{\pi}$. In Figure 5.2c a new policy (blue dot) is found but its expected value is equal to that of the value vector at $\boldsymbol{w}_{\max}$ and therefore the partial CCS is not improved. This policy is preserved in the search history but will not be included in the final CCS (and hence marked blue instead of red). Still, from this policy it follows that the OCCS value must exactly equal the partial CCS value for all weight vectors in the region up to the third policy from the left, found at $\boldsymbol{w}_{\max}$ of Figure 5.2a. Note that by definition of the partial CCS and the optimistic CCS, the value function of the CCS itself must lie on or in between the two.

Now, let $V_{\bar{\psi}}(\boldsymbol{w}) = \max_{\boldsymbol{v} \in \bar{\psi}} \boldsymbol{w} \cdot \boldsymbol{v}$ define the value of the optimistic CCS at weights $\boldsymbol{w}$, i.e. the best possible value at each $\boldsymbol{w}$. Then, the maximal possible error of a partial CCS $\psi$ at $\boldsymbol{w}$ with respect to its optimistic CCS $\bar{\psi}$ is given by:

$$\Delta_{\psi}(\boldsymbol{w}) = V_{\bar{\psi}}(\boldsymbol{w}) - V_{\psi}(\boldsymbol{w}) \tag{5.5}$$

As a corollary of Theorem 5.1 and Equation 5.5 it must be that the maximal error with respect to both the CCS as well as the optimistic CCS is at a corner weight of the partial CCS. Moreover, if the maximal error at all corner weights of the partial CCS is zero with respect to the optimistic CCS, the partial CCS must be a CCS for the MOMDP. Even more so, Equation 5.5 forms the basis of an anytime algorithm that iteratively decreases the maximal error with respect to the optimistic CCS, by prioritising over the corner weight that currently maximises $\Delta_{\psi}(\boldsymbol{w})$. This leads to `OLS`, shown in Algorithm 5.4.

**Algorithm 5.4** `Optimistic Linear Support (OLS)`

**Require:** MOMDP $M$, scalarisation function $f$ and single-objective MDP solver $\mathbb{M}$.

**function** `OLS`$(M, f, \mathbb{M})$
  $\psi = \emptyset$ ▷ Partial CCS
  $WV = \emptyset$ ▷ Searched weight vectors and found values
  $Q_{\boldsymbol{w}} = \{\langle \mathbf{e}_k, \infty \rangle \mid \forall k \in [1, m]\}$ ▷ $\langle \boldsymbol{w}, \Delta_{\psi}(\boldsymbol{w}) \rangle$-queue, initially unit vectors $\mathbf{e}_k$
  $\langle \boldsymbol{w}_{\max}, \Delta_{\max} \rangle = Q_{\boldsymbol{w}}$.`pop`()
  **while** $\Delta_{\max} > 0$ **do**
    $\langle \pi^*, \boldsymbol{V}^* \rangle = \mathbb{M}($`scalarise`$(M, f, \boldsymbol{w}_{\max}))$ ▷ Solve scalarised MDP at $\boldsymbol{w}_{\max}$
    $WV = WV \cup \{\langle \boldsymbol{w}_{\max}, \boldsymbol{V}^* \rangle\}$
    **if** $\boldsymbol{w}_{\max} \cdot \boldsymbol{V}^* > V_{\psi}(\boldsymbol{w}_{\max})$ **then** ▷ Add policy if CCS improves at $\boldsymbol{w}_{\max}$
      $\psi = \psi \cup \{\pi^*\}$
      $W =$ `cornerWeights`$(\psi)$ ▷ Recompute corner weight(s) after adding $\pi^*$
      $\bar{\psi} =$ `calcOCCS`$(WV, W)$ ▷ Compute optimistic CCS for $\psi$
      **for each** $\boldsymbol{w} \in W \setminus WV$ **do**
        $Q_{\boldsymbol{w}}$.`add`$( \left\langle \boldsymbol{w}, V_{\bar{\psi}}(\boldsymbol{w}) - V_{\psi}\boldsymbol{w} \right\rangle )$ ▷ Add new vector with error $\Delta_{\psi}(\boldsymbol{w})$
      **end for**
    **end if**
    $\langle \boldsymbol{w}_{\max}, \Delta_{\max} \rangle = Q_{\boldsymbol{w}}$.`pop`() ▷ Get next weight vector that maximises $\Delta_{\psi}$
  **end while**
  **return** $\psi$
**end function**

The full description of the OLS algorithm can be found in the work by Roijers et al. [217], here it is only outlined briefly. Put simply, in every iteration the algorithm first computes the optimal policy for the single-objective MDP that is *scalarised* given the scalarisation function $f$ and weight vector $\boldsymbol{w}_{\max}$ that maximises error $\Delta_{\max} = \Delta_{\psi}(\boldsymbol{w}_{\max})$ using solver $\mathbb{M}$ (line 7). Note that it is assumed that the solver returns the expected value per objective, i.e. as a value vector $\boldsymbol{V}^*$. This is achieved by either altering the solver, if possible, or by including a separate, subsequent multi-objective policy evaluation step that is not listed here. Next, lines 9 to 12, check whether the partial CCS is improved by policy $\pi^*$ and, if so, add it to the partial CCS, determine the set of (new) corner weights and the corresponding optimistic CCS. The latter two are done via simple linear programs. `cornerWeights` computes all intersections of value vectors within the $[0, 1]^m$ domain. The OCCS is determined by for every corner weight $\boldsymbol{w} \in W$ running the following linear program:

$$\begin{aligned} &\max \quad \boldsymbol{w} \cdot \boldsymbol{v} \\ &\text{subject to} \quad \forall \langle \boldsymbol{w}_{\pi}, \boldsymbol{V}^{\pi} \rangle \in WV : \ \boldsymbol{w}_{\pi} \cdot \boldsymbol{v} \leq \boldsymbol{w} \cdot \boldsymbol{V}^{\pi} \end{aligned} \tag{5.6}$$

This program returns the value vector that maximises the potential CCS value at $\boldsymbol{w}$, from which in turn the maximal CCS error can be computed using Equation 5.5. Finally, all new corner weights (slightly abusing notation to remove the previously searched weight vectors $WV$) and their corresponding maximal error with respect to the optimistic CCS are added to the queue (lines 13-15) and the process is repeated. When the maximal error $\Delta_{\max}$ becomes equal to zero, the value of the partial CCS $\psi$ must equal that of the CCS over the entire weight range and the algorithm terminates and returns the CCS.

For completeness, it must be remarked that Algorithm 5.4 lists OLS in its simplest form. The algorithm can be optimised by iteratively recomputing the new corner weights, instead of a full computation every iteration, and be made anytime approximate by incorporating a time limit and/or a bounded approximation algorithm by terminating when $\Delta_{\max}$ falls below a specified threshold [217].

## 5.2 Approximate Optimistic Linear Support

The OLS algorithm of the previous section produces a Convex Coverage Set for any multi-objective MDP within a limited number of iterations. Still, every iteration requires solving a typically complex single-objective MDP optimally, which in itself may be very computationally demanding, and hence this approach is not feasible for many MOMDPs. This section introduces the Approximate Optimistic Linear Support (AOLS) algorithm, a bounded-approximation variant of the previously discussed OLS. The AOLS algorithm is nearly the same as the OLS method, however instead of an optimal single-objective MDP solver it takes an $\epsilon$-approximate MDP solver as its argument and produces an $\epsilon$-CCS, i.e. a Convex Coverage Set with a bounded error of $\epsilon$ with respect to the optimal CCS value.[44] At its core is a modified version of Theorem 5.1,

[44] *The term optimal is actually implied by Convex Coverage Set and thus superfluous. It is used, however, to explicit the difference between the CCS and the approximate $\epsilon$-CCS.*

adapted to the approximate setting, such that again it is possible to guarantee that the algorithm terminates within a finite number of iterations. Additionally, even though AOLS builds its CCS using approximate policies that may cause corner weights to shift each round, the approximate CCS (not to be confused with partial CCS) value can be shown to converge through the modified theorem. Further in this section the AOLS algorithm is presented, followed by its correctness proof based on the maximal approximate CCS error of Theorem 5.8. First, it is necessary to formalise the bounded-approximation solver:

**Definition 5.5** **$\epsilon$-approximate MDP solver**

An $\epsilon$-approximate MDP solver that produces a policy $\pi$ with an expected value $V^{\pi}$ such that for any $\epsilon \geq 0$: $V^{\pi} \geq (1-\epsilon)V^{\pi^*}$. Here, $V^{\pi^*}$ is the expected value of optimal policy $\pi^*$.

and the corresponding $\epsilon$-approximate CCS:

**Definition 5.6** **$\epsilon$-approximate Convex Coverage Set ($\epsilon$-CCS)**

The $\epsilon$-approximate Convex Coverage Set ($\epsilon$-CCS) for an MOMDP is a set of policies $\tilde{\Psi} \subseteq \Pi$ such that for every weight vector $\boldsymbol{w} \in [0,1]^m$, a linear scalarisation function $f$ and $\epsilon \geq 0$ the set $\tilde{\Psi}$ contains at least one policy $\pi$ with a scalarised value at most a factor $1-\epsilon$ lower than the CCS value $V_{\Psi}(\boldsymbol{w})$ at $\boldsymbol{w}$, or

$$\tilde{\Psi} = \{\pi \in \Pi \mid \forall \boldsymbol{w} \in [0,1]^m,\ \forall \pi' \in \Pi,\ f, s^0 :\ f(\boldsymbol{V}^{\pi}(s^0), \boldsymbol{w}) \geq (1-\epsilon) f(\boldsymbol{V}^{\pi'}(s^0), \boldsymbol{w})\} \tag{5.7}$$

and note that the near-optimality condition of Definition 5.6 can be formulated more briefly as $\forall \boldsymbol{w} \in [0,1]^m : V_{\tilde{\Psi}}(\boldsymbol{w}) \geq (1-\epsilon)V_{\Psi}(\boldsymbol{w})$, with implicit $f$ and $s^0$.

Although Definition 5.6 formally bounds the approximate CCS to have a value at most $(1-\epsilon)$ away from the optimal CCS, an algorithm to produce the approximate CCS must be able to guarantee this bound without knowing the actual CCS. Therefore again the optimistic CCS is used, albeit with a minor adaptation of the CCS error. Recall that the CCS error $\Delta_{\psi}(\boldsymbol{w})$ of Equation 5.5 defines an absolute error of a partial CCS $\psi$ with respect to its corresponding OCCS $\bar{\psi}$ at weights $\boldsymbol{w}$. This is easily transformed into a *relative* maximal CCS error:

$$\Delta^r_{\psi}(\boldsymbol{w}) = 1 - \frac{V_{\psi}(\boldsymbol{w})}{V_{\bar{\psi}}(\boldsymbol{w})} \tag{5.8}$$

such that the partial CCS error is bounded relative to the OCCS and, indirectly, to the optimal CCS. Altogether, the above lead to the AOLS algorithm of Algorithm 5.7.

The first thing to notice is that the AOLS algorithm is highly similar to the previously presented OLS algorithm, but has some subtle differences. First notice that throughout the algorithm the *relative* error $\Delta^r_{\psi}$ is used, instead of the absolute error $\Delta_{\psi}$, so that the algorithm may terminate when the partial CCS $\psi$ value is within at most a factor $\epsilon$

**Algorithm 5.7** `Approximate Optimistic Linear Support (AOLS)`

**Require:** MOMDP $M$, scalarisation function $f$, error bound $\epsilon$ and $\epsilon$-approximate, single-objective MDP solver $\bar{\mathbb{M}}$.

1: **function** AOLS($M, f, \epsilon, \bar{\mathbb{M}}$)
2: $\quad\psi = \emptyset, WV = \emptyset$ ▷ Partial CCS, searched weights and values
3: $\quad Q_{\boldsymbol{w}} = \{\langle \mathbf{e}_k, \infty\rangle \mid \forall k \in [1, m]\}$ ▷ $\left\langle \boldsymbol{w}, \Delta^r_\psi(\boldsymbol{w}) \right\rangle$-queue, intially unit vectors $\mathbf{e}_k$
4: $\quad\langle \boldsymbol{w}_{\max}, \Delta^r_{\max}\rangle = Q_{\boldsymbol{w}}$.pop()
5: $\quad$**while** $\Delta^r_{\max} > \epsilon$ **do**
6: $\quad\quad\langle \pi, \boldsymbol{V}\rangle = \bar{\mathbb{M}}(\texttt{scalarise}(M, f, \boldsymbol{w}_{\max}))$ ▷ Approximate scalarised MDP at $\boldsymbol{w}_{\max}$
7: $\quad\quad WV = WV \cup \{\langle \boldsymbol{w}_{\max}, \boldsymbol{V}\rangle\}$
8: $\quad\quad$**if** $\boldsymbol{w}_{\max} \cdot \boldsymbol{V} > V_\psi(\boldsymbol{w}_{\max})$ **then** ▷ Add policy if CCS value improves
9: $\quad\quad\quad\psi = \psi \cup \{\pi\}$
10: $\quad\quad\quad\psi = \texttt{prune}(\psi)$ ▷ Prune dominated policies
11: $\quad\quad\quad W = \texttt{cornerWeights}(\psi)$ ▷ Recompute corner weight(s) after adding $\pi$
12: $\quad\quad\quad\bar{\psi} = \texttt{calcOCCS}(WV, W, \epsilon)$ ▷ Compute new OCCS
13: $\quad\quad\quad$**for each** $\boldsymbol{w} \in W \setminus WV$ **do**
14: $\quad\quad\quad\quad Q_{\boldsymbol{w}}$.add( $\left\langle \boldsymbol{w}, 1 - V_\psi(\boldsymbol{w})/V_{\bar{\psi}}(\boldsymbol{w}) \right\rangle$ ) ▷ Add $\boldsymbol{w}$ with relative error $\Delta^r_\psi(\boldsymbol{w})$
15: $\quad\quad\quad$**end for**
16: $\quad\quad\quad WV = W$
17: $\quad\quad$**end if**
18: $\quad\quad\langle \boldsymbol{w}_{\max}, \Delta^r_{\max}\rangle = Q_{\boldsymbol{w}}$.pop() ▷ Get next weight vector that maximises $\Delta_\psi$
19: $\quad$**end while**
20: $\quad$**return** $\psi$
21: **end function**

away from the value of the CCS (line 5). Furthermore, because AOLS uses approximate solver $\bar{\mathbb{M}}$, policies may become dominated in the partial CCS. Therefore AOLS includes a pruning step each time the partial CCS is expanded (line 10) that removes any policy $\pi$ with value $\boldsymbol{V}^\pi$ if $\forall \boldsymbol{w} \in [0,1]^m, \exists \pi' \in \psi : \boldsymbol{w} \cdot \boldsymbol{V}^\pi \leq \boldsymbol{w} \cdot \boldsymbol{V}^{\pi'}$ with $\pi \neq \pi'$. Finally, the calcOCCS program is adapted slightly to incorporate the possibility of being $\epsilon$ wrong at every corner weight $\boldsymbol{w} \in W$:

$$\begin{aligned} &\max \quad \boldsymbol{w} \cdot \boldsymbol{v} \\ &\text{subject to} \quad \forall \langle \boldsymbol{w}_\pi, \boldsymbol{V}^\pi\rangle \in WV : \ \boldsymbol{w}_\pi \cdot \boldsymbol{v} \leq \frac{\boldsymbol{w} \cdot \boldsymbol{V}^\pi}{1-\epsilon} \end{aligned} \tag{5.9}$$

In other words, the optimistic CCS overestimates the value of the CCS by a factor $1/(1-\epsilon)$ so that is always an upper bound on the (partial) CCS value. Note that the AOLS algorithm may be terminated at any point to produce a partial CCS with a value of at least $(1 - \Delta^r_{\max})V_\Psi(\boldsymbol{w})$ at all weight vectors $\boldsymbol{w} \in [0,1]^m$. The algorithm can therefore be employed as an anytime approximation method, in a similar fashion as OLS, but with a bounded error at the point of termination. In addition, as was assumed by OLS, AOLS expects the single-objective solver $\bar{\mathbb{M}}$ to return a value vector, but in the approximate setting this may introduce a dominating component to the runtime. Recall that the number of states in a policy may be exponential and therefore

a naive policy evaluation routine could introduce an exponential time requirement. To combat this, more efficient or approximate evaluation methods can be used, although the latter might affect the bound on the maximal CCS error.

Now it remains to show that the AOLS algorithm indeed produces an $\epsilon$-CCS within a finite number of iterations. First it is shown that AOLS also needs to consider only the corner weights of its partial CCS to maximise the CCS improvement in each iteration of the algorithm.

**Theorem 5.8 Maximal approximate CCS error**

Let $\Psi \subseteq \Pi$ denote a CCS for a multi-objective MDP. Then for any approximate partial Convex Coverage Set $\psi \subseteq \Pi$, the weight vector $\boldsymbol{w}$ that maximises $V_\Psi(\boldsymbol{w}) - V_\psi(\boldsymbol{w})$ is a corner weight of $\psi$.

*Proof.* Recall that $\Delta_\psi(\boldsymbol{w}) = V_\Psi(\boldsymbol{w}) - V_\psi(\boldsymbol{w})$ (Equation 5.5) and that this function hence must also be PWLC because it is the difference between two PWLC functions. Therefore the weight vector at that which $\Delta_\psi(\boldsymbol{w})$ is maximised must be in one of the four cases illustrated by Figure 5.3.

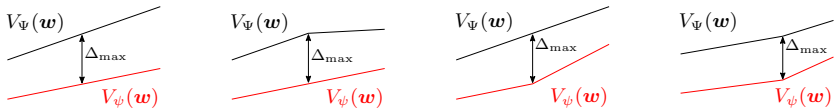



**Figure 5.3** *Four possible cases for the weights at which $\Delta_\psi(\boldsymbol{w})$ is maximal: at a weights $\boldsymbol{w}$ that is (a) neither a corner point of $\Psi$ or $\psi$, (b) a corner point of $\Psi$ but not of $\psi$, (c) a corner point of $\psi$ but not of $\Psi$ or (d) a corner point of both.*

Case (a) maximises $\Delta_\psi(\boldsymbol{w})$ if and only if the slope of $\Delta_\psi(\boldsymbol{w})$ is equal to 0, in which case its value is never higher than the value at the adjacent corner point(s) where its slope changes. Case (b) can never occur because if the weight vector $\boldsymbol{w}$ that maximises $\Delta_\psi(\boldsymbol{w})$ is a corner point of $\Psi$ but not of $\psi$ and $\Delta_\psi$ is maximised then the value $V(\boldsymbol{w})$ must also be at maximum at $\boldsymbol{w}$. However, if $V(\boldsymbol{w})$ is maximal at $\boldsymbol{w}$ then it is not a PWLC function, which contradicts its definition. Only cases (c) and (d) remain and in both cases the maximum error resides in a corner point of the partial CCS $\psi$, proving the theorem. □

Due to Theorem 5.8, the AOLS algorithm is guaranteed to make the best possible improvement to the partial CCS (value) in each iteration. In addition, for every weight vector that is checked, the partial CCS value will either increase or remain the same if no improvement is found (the condition of line 5). As a corollary, the maximum relative error also decreases or remains the same and thus ultimately converges when all corner weights have been checked. Then it remains to show that this convergence is within a finite number of iterations, i.e. there are a finite number of such corner weights considered by AOLS.

**Theorem 5.9** `AOLS` **correctness and convergence**

Given an $\epsilon$-approximate MDP solver $\bar{\mathbb{M}}$, such that $\epsilon \geq 0$, `AOLS` produces an $\epsilon$-CCS within a finite number of iterations.

*Proof.* The algorithm iterates over a priority queue of tuples, consisting of corner weights $\boldsymbol{w}$ and their corresponding relative CCS error $\Delta^r_\psi(\boldsymbol{w})$, until the first relative CCS error lower or equal to $\epsilon$ is encountered. In this case, all remaining corner weights in the queue must have a relative error less than $\epsilon$ due to the queue sorting. For new corner weights, $\bar{\mathbb{M}}$ always produces a policy with an expected value of at least $1-\epsilon$ times the expected value of the optimal policy and hence $V_\psi(\boldsymbol{w}) \geq (1-\epsilon)V_\Psi(\boldsymbol{w})$. Moreover, by adding a new policy to the partial CCS, the value of the partial CCS must increase and, consequentially, the maximum relative CCS error in the newly established corner weights must decrease – and ultimately converge – correspondingly. Finally, because every corner weight evaluated by `AOLS` is added to a searched-weights list, therefore considered only once, and the number of possible corner weights is finite, the number of corner weights considered by `AOLS` is finite.

At termination, the maximum relative error at all corner weights $\boldsymbol{w}$ must be below $\epsilon$ and, as a corollary of Theorem 5.8, this must hold for all possible weight vectors $\boldsymbol{w} \in [0,1]^m$. Consequentially, the produced partial CCS satisfies Equation 5.7 and is hence an $\epsilon$-approximate CCS. □

**Remark 5.10** `AOLS` **with an unbounded MDP Solver**

It is possible to run `AOLS` with an unbounded approximate MDP solver and the algorithm will still converge on an approximate CCS, however no bound can be derived on the quality of the resulting approximate CCS. Moreover, in this case it is not possible to prioritise over relative CCS errors. Instead `AOLS` must continue checking – randomly selected – corner weights until no more improvement is made or there are no more corner weights to the partial CCS.

## 5.3 Scalarised Sample-based Iterative Improvement

The `Approximate Optimistic Linear Search` method of the previous section provides a more time-efficient method to produce a high-quality CCS that the previous `OLS` algorithm. Moreover, if an approximate, single-objective MDP solve with bounded error $\epsilon$ is used, the value of the approximate CCS is guaranteed at most a factor $\epsilon$ lower than that of the optimal CCS. When time is limited, however, `AOLS` may 'waste' much of its time on areas of the weight range that could actually be of little interest to the planner. There are many situations in which a planner has some preliminary knowledge regarding the most interesting areas of the weight range, but not on specific weight values. For instance, if one weight expresses the relative cost of a resource unit, the planner may not be interested in any solution where the cost weight exceeds 50%. In such a scenario, the planner prefers a method that tries to maximise the CCS value in

the range where $w_{cost} \in [0, 0.5]$, given the little time available. `AOLS`, however, cannot distribute its planning effort in such a way; it always considers the entire range $[0, 1]^m$.

In such settings, where a prior distribution over the weights (or *prior* for short) is known, the `Scalarised Sample-based Iterative Improvement` (`SSII`) algorithm typically produces a better CCS value *in the specified weight range*. The essence of this approach is very straightforward: given a prior distribution $pr$ over the weight simplex, sample weight vectors from that prior and spend the most time on finding high-quality policies for these vectors. `SSII`, shown in Algorithm 5.11, does this in an iterative fashion to make the best use of the time it is given. It continually improves its partial CCS until there is no more time, at which point the current partial CCS is returned.

**Algorithm 5.11** `Scalarised Sample-based Iterative Improvement (SSII)`

**Require:** MOMDP $M$, scalarisation function $f$, anytime single-objective MDP solver $\bar{\mathbb{M}}$, weight sample vectors $\tilde{W}$, sample runtime funcion $T$ and time limit $t_{limit}$.

1: **function** `SSII`$(M, f, \bar{\mathbb{M}}, \tilde{W}, T, t_{limit})$
2: $\quad \psi = \emptyset$
3: $\quad WV = \emptyset$
4: $\quad$ **for each** $\boldsymbol{w} \in \tilde{W}$ **do**
5: $\quad\quad t_{\boldsymbol{w}} = T(-1)$ ▷ Set initial sample runtime
6: $\quad\quad WV[\boldsymbol{w}] = \bar{\mathbb{M}}(M, f, \boldsymbol{w}, t_{\boldsymbol{w}})$ ▷ Store tuple $\langle \pi, \boldsymbol{V}^{\pi} \rangle$ found at weight vector $\boldsymbol{w}$
7: $\quad$ **end for**
8: $\quad$ **while** time used $< t_{limit}$ **do**
9: $\quad\quad \bar{\psi} = \texttt{calcOCCS}(WV, \tilde{W} \cup \mathbf{e})$ ▷ Comp. OCCS from samples $\tilde{W}$ and extrema $\mathbf{e}$
10: $\quad\quad \boldsymbol{w}_{\max} = \texttt{nextSample}(\bar{\psi}, WV, \tilde{W}, \boldsymbol{t})$ ▷ Determine next sample weight $\boldsymbol{w}_{\max}$
11: $\quad\quad t_{\boldsymbol{w}_{\max}} = T(t_{\boldsymbol{w}_{\max}})$ ▷ Increase allowed runtime for the sample
12: $\quad\quad \langle \pi, \boldsymbol{V}^{\pi} \rangle = \bar{\mathbb{M}}(M, f, \boldsymbol{w}_{\max}, t_{\boldsymbol{w}_{\max}})$ ▷ Compute new policy and value
13: $\quad\quad$ **if** $\boldsymbol{w}_{\max} \cdot \boldsymbol{V}^{\pi} > \boldsymbol{w}_{\max} \cdot WV[\boldsymbol{w}_{\max}]$ **then**
14: $\quad\quad\quad WV[\boldsymbol{w}_{\max}] = \langle \pi, \boldsymbol{V}^{\pi} \rangle$ ▷ Update policy at sample vector $\boldsymbol{w}_{\max}$
15: $\quad\quad$ **end if**
16: $\quad$ **end while**
17: $\quad \psi = \{ \pi \in WV \mid \exists \boldsymbol{w} \in [0,1]^m, \forall \pi' \neq \pi \in WV : \boldsymbol{w} \cdot \boldsymbol{V}^{\pi} > \boldsymbol{w} \cdot \boldsymbol{V}^{\pi'} \}$ ▷ Comp. new CCS
18: $\quad$ **return** $\psi$
19: **end function**

Besides the MOMDP $M$ and (linear) scalarisation function $f$ as before, the `SSII` algorithm requires an anytime, single-objective, MDP approximation algorithm $\bar{\mathbb{M}}$, a set of weight vectors $\tilde{W}$ sampled from the prior distribution $pr$, an increasing list of allowed runtimes $T$ and a time limit $t_{limit}$. The anytime solver, in combination with the runtime list, make it possible for `SSII` to iteratively improve on its samples until the time limit is reached. Initially, 'a quick-and-dirty' partial CCS is determined by running $\bar{\mathbb{M}}$ solver at every sample weight vector with very little time (lines 4 to 7). Then, as long as there is time available, the algorithm tries to improve the policies at each of the samples by allowing the MDP solver increasingly more time. For this, it first computes the optimistic CCS (line 9) using the weight sample vectors in $\tilde{W}$, appended with the weight simplex extrema $\mathbf{e} = \bigcup_{k \in m} \mathbf{e}_k$ that do not necessarily need

to be contained in prior $pr$. The OCCS is then fed, alongside the current results $WV$, prior sample vectors $\tilde{W}$ and currently used runtime for each weight vector $\boldsymbol{t}$, into the `nextSample` routine that determines the sample weight vector to improve next (line 10). This function can be arbitrarily complex and may use any combination of the aforementioned parameters to determine the next sample weight. In the experiments of Section 5.4, the sample vector $\boldsymbol{w}$ that maximises $\Delta_\psi(\boldsymbol{w})$ is selected.

Having determined the weight sample vector to improve next, the allowed runtime for that vector is increased in line 11 by function $T$. This function returns a new maximal allowed runtime for sample vector $\boldsymbol{w}_{\max}$, given that its last attempt was given $t_{\boldsymbol{w}_{\max}}$ time (such that $T(-1)$ is assumed to return the initially allowed runtime). Therefore, this function is typically asymptotic and (exponentially) increasing in its argument. Given the sample weight vector and the allowed runtime, the anytime solver is ran once more on the scalarised MDP to compute a new policy and associated value. Note that because an anytime, approximate solver is used, the policy value is expected but does not necessarily have to improve. For this reason, line 13 check whether the new policy should replace the previous one. After the policy is added or discarded, the process is repeated for a new (possibly the same) sample weight vector, until there is no more time left. When time runs out, the algorithm returns its partial CCS determined from all the policies at the sample weight vectors (line 17). Note that only the non-dominated policies are returned, with the highest scalarised value in at least one weight vector $\boldsymbol{w} \in [0,1]^m$.

**Example 5.12** `Scalarised Sample-based Iterative Improvement`

Once more, a two-objective problem is considered with objectives $w_1 \in [0,1]$ and $w_2 = 1 - w_1$. Furthermore, in this example $w_1$ represents a per-unit selling price of a resource and the planner knows that only the area $w_1 \in [0.5, 0.9]$ is of interest. A value lower than $0.5$ would lead to losses whereas a relative price above $0.9$ makes the resource too expensive. This information can be transformed in a prior distribution over weights such that $\sum_{w_1=0.5}^{0.9} pr(\langle w_1, 1-w_1\rangle) = 1$ and, correspondingly, $pr(\langle w_1, 1-\boldsymbol{w}\rangle) = 0$ for all $w_1 \in [0, 0.5) \cup (0.9, 1]$ In addition, the algorithm is given a sample runtime function that doubles the previous runtime, i.e. $T(x) = 2x$ with initial runtime $T(-1) = 1$ second.

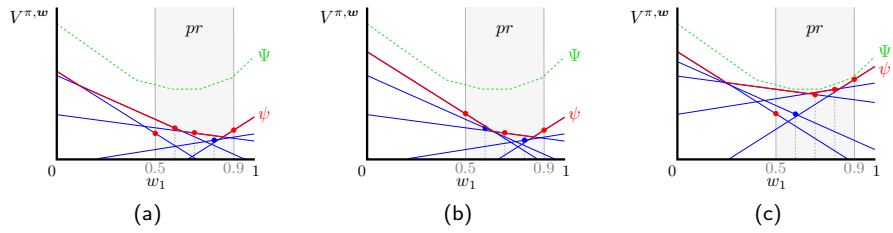



**Figure 5.4** *Illustration of the* `Scalarised Sample-based Iterative Improvement` *algorithm in three subsequent stages: (a) after computation of the initial sample set, (b) after finding a higher-value policy at weight $w_1 = 0.5$ and (c) upon termination of the algorithm.*

Figure 5.4 visualises the execution of the SSII algorithm on this problem in consecutive stages of the planning process. Each of the figures illustrate the partial CCS (red line segments), composed from the value vectors (blue lines) corresponding to the (values of the ) policies, illustrated by the dots. Red dots represent policies that are included in the partial CCS, whereas blue dots indicate policies that are (currently) dominated. The dotted green line visualises the CCS for this example problem. Note that this is purely to illustrate typical SSII results, this information is not actually available to the algorithm. The prior of interest, i.e. the range $w_1 \in [0.5, 0.9]$, is shown as a gray area and in this example $5$ sample weight vectors are used, uniformly distributed over the prior with interval $0.1$.

The first figure, Figure 5.4a, shows the result of the prior weight sample initialisation performed by lines 4 to 7 of Algorithm 5.11. This initialisation produces a fast (max 1 second per sample) but inaccurate first partial CCS, with a value much below that of the optimal CCS. However, from this partial CCS an optimistic CCS is computed that is employed to guide the sample improvement algorithm. The OCCS itself is not shown in this figure, because it nearly overlaps the partial CCS. Only in the regions $[0, 0.5]$, $[0.6, 0.7]$ and $[0.8, 0.9]$ its value is slightly higher than the partial CCS. Recall that SSII does neither know the optimal CCS value nor any bound on the approximation error. Therefore, the best it can do is to determine the OCCS based on the sample vectors $\tilde{W}$ and the weight simplex extrema, i.e. the unit vectors $\mathbf{e}_k$ for $k \in [1, m]$. Notice also that there is one policy shown blue in this figure because its value vector is dominated by the others for all weight vectors.

In this example, SSII is given a next-sample heuristic that prioritises samples based on the OCCS error $\Delta_\psi(\boldsymbol{w})$. This leads to the selection of $\boldsymbol{w}_{\max} = \langle 0.5, 0.5 \rangle$ as the sample to improve next and now it is given $T(1) = 2$ seconds to find a policy. Figure 5.4b shows the outcome with regard to the value of the partial CCS, which is increased by the newly sampled policy (value). As a side effect of this improvement, the policy at sample $\langle 0.6, 0.4 \rangle$ becomes dominated and would not be in the partial CCS if the search terminated now.

Figure 5.4c shows the eventual situation upon termination of the algorithm, when the time limit was reached. After multiple iterations the partial CCS has increased much in value, especially in the region of the prior. Within this area the value of the partial CCS is very close to the optimal CCS value. Outside this area, however, the partial CCS value is much lower than that of the CCS. This is typical for the behaviour of SSII: whereas AOLS does its best on finding a 'good-quality' approximation overall, SSII focuses on maximising the partial CCS value in a specific range. Note that the quality of the SSII approximation can be increased by extending its time limit, leading to better samples, and/or using more samples in the prior, leading to a closer approximation of the shape of the CCS value function. Of course, both measures have their impact on the algorithms anytime performance.

As opposed to AOLS the SSII algorithm does not produce an approximate CCS with bounded error, but produces a high-quality approximation of the optimal CCS in its prior. The former is simply because it focuses on a specific area of the weight range and provides no guarantees outside this range. Note that it is possible to provide a quality bound in the prior if the anytime approximate MDP solver $\bar{\mathbb{M}}$ provides a bound on the value of the policies it produces. Furthermore, while it is not designed as such, the algorithm can be modified to continue until a specified quality is achieved (instead of running until time is out) if the single-objective solver provides a quality bound.

## 5.4 Comparison of Multi-objective Algorithms

This section analyses the strengths and weaknesses of the previously described AOLS and SSII algorithm through empirical evaluation. Both algorithms are tested on a set of MPP instances with two objectives: maximising profits and minimising the traffic time lost. Similar to the introduction, the (relative) weights corresponding to these objective are denoted by $w_{profit}$ and $w_{ttl}$ respectively and the reward function of Equation 5.1 is used with the simple linear scalarisation function $\boldsymbol{w} \cdot \boldsymbol{V}^{\pi}$. Both methods, and OLS as a benchmark, are run on a test set `random-mo` that consists of randomly generated 2-agent and 3-agent instances (144 each) of the multi-objective MPP, with 2 to 4 activities and horizons varying between 3 and 10. 4-Agent instances are not considered as a result of the experiments of the previous chapter. Finally, to test the performance of SSII versus AOLS on a prior distribution of weights, a prior $pr$ is defined that focuses on the weight range $w_{profit} \in [0.5, 1]$ (and, correspondingly, $w_{ttl} \in [0, 0.5]$). This prior is distributed uniformly over the range $w_{profit} \in [0.5, 1]$ and satisfies $\sum_{\boldsymbol{w} \in W} pr(\boldsymbol{w}) = 1$. The outcomes of both algorithms produced for the aforementioned instances are compared to the optimal CCS produced by OLS through several error metrics.

$\epsilon_{\max}$ The maximal $\epsilon$ of the approximate CCS over the considered weight range $W$, i.e. the maximal relative error between two CCSes:

$$\epsilon_{\max}(\Psi, \psi, W) = \max_{\boldsymbol{w} \in W} 1 - \frac{V_{\psi}(\boldsymbol{w})}{V_{\Psi}(\boldsymbol{w})} \tag{5.10}$$

$\epsilon_{opt}$ The percentage of instances for which a (near-)optimal solution is found, where a CCS is considered optimal in a range $W$ if $\forall \boldsymbol{w} \in W$: $\epsilon_{\max} < 0.01$.

$\varepsilon_{\max}$ The maximal absolute error between two CCSes in the weight range, not including the prior:

$$\varepsilon_{\max}(\Psi, \psi, W) = \max_{\boldsymbol{w} \in W} V_{\Psi}(\boldsymbol{w}) - V_{\psi}(\boldsymbol{w}) \tag{5.11}$$

$\varepsilon_{\exp}$ The expected coverage of the approximate CCS w.r.t. the CCS, incorporating the prior:

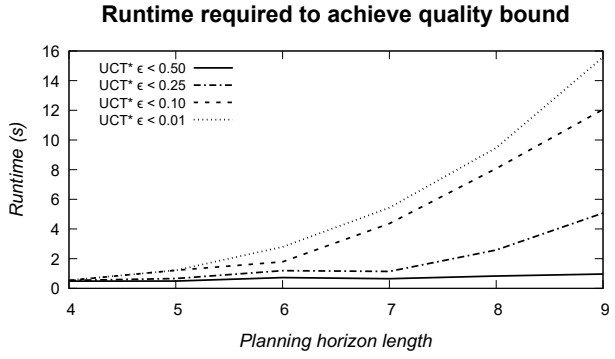
$$\varepsilon_{\exp}(\Psi, \psi, pr) = \int_{[0,1]^m} pr(\boldsymbol{w}) \big(V_{\Psi}(\boldsymbol{w}) - V_{\psi}(\boldsymbol{w})\big) \, \mathrm{d}\boldsymbol{w} \tag{5.12}$$

Finally, OLS, AOLS and SSII all require an (approximate) single-objective solver to produce a policy for the scalarised MDPs. OLS relies on an optimal MDP solver to produce the optimal CCS and hence, with the experiments of the previous chapter in mind, CoRe is used. For the approximate algorithms, the UCT* [134] algorithm is

used (see Section 3.4) and run on an RDDL encoding of MPP (as in Section 3.5. Although this solver is anytime-approximate, it does not provide a guaranteed error bound and therefore AOLS cannot guarantee an $\epsilon$-CCS. The results presented in this section, however, show that the UCT* algorithm performs very well on instances of the MPP. By comparing the approximate CCS value with the optimal one, it can be shown that the $\epsilon$ produced by AOLS in combination with UCT* is typically very low. To establish this, two preliminary experiments are performed that analyse the performance of UCT*, of which the results are shown in Figure 5.5. Note beforehand that UCT* produces only partial policies, i.e. it only includes actions for states and transitions that were visited during its policy search. These partial policies are amended by inserting no-operation actions, with its corresponding transition and new state, whenever the policy does not contain an action.

| Method | Time (s) | $\epsilon_{\max}$ | $\epsilon_{opt}$ | $\varepsilon_{\exp}$ | $\varepsilon_{\max}$ |
|---|---|---|---|---|---|
| OLS | | | | | |
| SPUDD | 614.480 | - | - | - | - |
| CoRe | 391.108 | - | - | - | - |
| AOLS | | | | | |
| UCT* 0.01s | 2.527 | 0.351 | 28 % | 0.186 | 98.659 |
| UCT* 0.1s | 3.933 | 0.234 | 44 % | 0.049 | 33.503 |
| UCT* 1s | 10.170 | 0.099 | 66 % | 0.012 | 13.249 |
| UCT* 10s | 56.138 | 0.022 | 85 % | 0.000 | 0.427 |
| UCT* 20s | 85.976 | 0.015 | 89 % | 0.000 | 0.064 |

(a)




(b)

**Figure 5.5** *Analysis of* AOLS *performance, using* UCT* *as its approximate MDP solver, in terms of solution quality and the time required: (a) summary over 100 solved instances from* random-mo *showing the average runtime and error metrics, and (b) a graph of the average runtime required to achieve various quality bounds* $\epsilon_{\max}$ *for 96 solved instances from* random-mo-N3 *with 3 agents, 3 maintenance activities and increasing horizon.*

First a comparison is done of the CCS produced by each of the algorithms in terms of time and solution quality. The results of this comparison is listed in Table 5.5a for the 100 instances of `random-mo` that are solved by both `AOLS` with `UCT*` `OLS` with `CoRe` and, `OLS` with `SPUDD` (Section 3.5) within at most 20 minutes. Because `UCT*` is stochastic in nature, all `AOLS` runs are done three times and their average is considered as a single result. The tables shows the average runtime and aforementioned error metrics for each of the considered algorithms. Observe that this experiment again confirms the performance increase of `CoRe` with respect to the `SPUDD` approach, as was established previously in Chapter 4. Secondly, when `UCT*` is allowed at most 10 milliseconds to find a policy, already 28% of the instances are solved optimally. Note that the listed runtime of 2.5 seconds is because `AOLS` runs `UCT*` on multiple corner weights, incurring overhead for encoding the MDP as an RDDL instance and starting the `PROST` toolkit. Finally, observe that the quality increases significantly when the allowed runtime increases. By running `UCT*` with at most 20 seconds per policy search, `AOLS` manages to find approximate CCSes with an average $\epsilon$-bound of $0.015$ and solves almost 90% (near-)optimally.

Figure 5.5b shows the average runtime that `UCT*` takes to produce a single policy for various approximation quality levels. `CoRe` and `UCT*` are run on the new set `random-mo-N3` of 100 instances with 3 agents that each have 3 maintenance activities to complete and horizons varying between 4 and 9 (instances with a horizon of 10 often caused `CoRe` to run longer than the maximum allowed time of 15 minutes in this experiment and are therefore omitted), and the expected values of the resulting policies are compared. As `UCT*` does not provide a bounded approximation it is run multiple times, such that each run is allowed more time, until the desired $\epsilon$ is reached. When this is the case, the runtime reported by `UCT*` is used as the time required to reach that quality. Note that this therefore overestimates the required time as a policy with bound $\epsilon$ may have been found earlier but `UCT*` does not terminate until its allowed runtime has been used. Again, `UCT*` is run three times on every instances to reduce the influence of its stochasticity.

The plot of Figure 5.5b shows that `UCT*` quickly produces policies for MPP, even if an error $\epsilon$ of at most $0.01$ is allowed. By comparison, `UCT*` takes on average about 15 seconds to produce a near-optimal policy for the instances with a horizon of 9 whereas `CoRe` averages just above 14 minutes. For this reason, i.e. the `CoRe` runtimes are orders of magnitude larger, they are not shown in this figure, but they are similar to the runtimes reported in the previous chapter. With regard to the scalability of the `UCT*` runtimes, although no larger problems have been considered in this experiment, the shapes of each of the plots expose an (intuitive) underlying structure. When the problem size and complexity increases, it is to be expected that it becomes exponentially harder for `UCT*` to find near-optimal policies, while low-quality solutions can still be produced quickly.

Having established the potential of `UCT*` as anytime approximate single-objective MDP solver, the next experiment compares the runtimes and produced CCSes of all algorithms, where both `AOLS` and `SSII` use `UCT*` as their approximate MDP solver. `SSII` is run both with and without knowledge of a prior weight distribution, thus on ranges $w_{profit} \in [0.5, 1]$ and $w_{profit} \in [0, 1]$ respectively, samples 10 weight vectors

uniformly from its assigned weight range and uses an OCCS based weight-selection heuristic. As before, OLS in combination with the CoRe algorithm is used to produce the optimal CCS to which all other algorithms are compared. The SPUDD solver is no longer considered, on account of its performance in previous experiments. For this experiment, the test set random-mo is again considered, this time with a maximum allowed runtime of 30 minutes. Any run that takes longer is discarded from the results. Furthermore, an optimal CCS is required to evaluate the approximate CCSes against, hence only those instances are compared for which CoRe successfully found the CCS. The results of this comparison are shown in Table 5.1 and Table 5.2 for respectively the 2-agent and 3-agent instances of random-mo.

| | | | $w_{profit} \in [0,1]$ | | | | | $w_{profit} \in [0.5,1]$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Time** | $\lvert\Psi\rvert$ | $V_\Psi$ | $\epsilon_{\max}$ | $\epsilon_{opt}$ | $\varepsilon_{\exp}$ | $\varepsilon_{\max}$ | $V_\Psi$ | $\epsilon_{\max}$ | $\epsilon_{opt}$ | $\varepsilon_{\exp}$ | $\varepsilon_{\max}$ |
| OLS | | | | | | | | | | | | |
| CoRe | 391.2 | 6.6 | 1205.27 | - | - | - | - | 658.35 | - | - | - | - |
| AOLS | | | | | | | | | | | | |
| UCT* 0.01s | 2.7 | 3.4 | 1165.20 | .262 | 14.5 % | .032 | 91.88 | 628.51 | .221 | 14.5 % | .042 | 90.97 |
| UCT* 0.1s | 3.7 | 4.5 | 1197.06 | .100 | 52.6 % | .006 | 24.30 | 651.95 | .058 | 52.6 % | .009 | 23.93 |
| UCT* 1s | 8.6 | 5.6 | 1203.25 | .022 | 82.9 % | .002 | 9.46 | 656.35 | .016 | 82.9 % | .003 | 9.45 |
| UCT* 10s | 43.1 | 6.6 | 1205.27 | .000 | 100.0 % | .000 | 0.03 | 658.35 | .000 | 100.0 % | .000 | 0.03 |
| UCT* 20s | 64.4 | 6.6 | 1205.27 | .000 | 100.0 % | .000 | 0.01 | 658.35 | .000 | 100.0 % | .000 | 0.01 |
| SSII (np) | | | | | | | | | | | | |
| UCT* 1s | 15.4 | 4.7 | 1203.20 | .019 | 68.4 % | .002 | 10.03 | 656.45 | .019 | 68.4 % | .003 | 10.02 |
| UCT* 10s | 40.8 | 5.1 | 1205.23 | .004 | 86.8 % | .000 | 1.13 | 658.31 | .004 | 88.2 % | .000 | 1.12 |
| UCT* 60s | 82.3 | 5.1 | 1205.24 | .004 | 86.8 % | .000 | 1.09 | 658.31 | .004 | 88.2 % | .000 | 1.08 |
| SSII (p) | | | | | | | | | | | | |
| UCT* 1s | 15.4 | 4.8 | 1200.56 | .312 | 56.6 % | .004 | 18.88 | 655.56 | .021 | 80.3 % | .004 | 12.99 |
| UCT* 10s | 40.8 | 5.4 | 1203.62 | .240 | 73.7 % | .001 | 7.47 | 658.34 | .001 | 98.7 % | .000 | 2.67 |
| UCT* 60s | 82.0 | 5.4 | 1203.63 | .240 | 73.7 % | .001 | 7.43 | 658.34 | .001 | 98.7 % | .000 | 2.63 |

**Table 5.1** *Comparison of runtime taken (seconds) and resulting CCS quality of* OLS, AOLS *and* SSII, *in the full weight range* $w_{profit} \in [0,1]$ *and in the prior range* $w_{profit} \in [0.5,1]$ *on 76 solved 2-agent instances of* random-mo. *The time listed next to the algorithm names is the maximum allowed runtime for* UCT* *to produce a single policy.* SSII *is both run with and without a prior, denoted by* p *and* np *respectively.*

Both tables show averages of the runtime it took the algorithms to produce a CCS, the size $|\Psi|$ of the CCS, the value $V_\Psi = \sum_{\boldsymbol{w} \in W} V_\Psi(\boldsymbol{w})$ of the CCS, and the values for the various error metrics defined at the beginning of this section. The CCS value and error metrics are computed for both the entire weight range as well as only over the prior range, to analyse the performance of the SSII algorithm in the prior.

An immediate observation from the 2-agent results (Table 5.1) is that both the AOLS and SSII algorithms perform relatively well compared to OLS in terms of speed versus quality. Whereas OLS on average takes about 6.5 minutes to produce a CCS, AOLS produces an almost equivalent CCS in an average of slightly more than 43 seconds when UCT* is given at most 10 seconds. SSII np that samples the entire weight region also produces a high-quality CCS quickly but cannot approximate the shape of the value function as well as AOLS. This latter claim is intuitively supported by the

resulting CCS size as more value vectors in the CCS lead to more detail in the CCS value function shape. As a consequence, its maximal epsilon $\epsilon_{\max}$ is typically not far from that of AOLS but not as often below $0.01$. When given a prior however, SSII does not perform as well over the full weight range (the columns below $w_{profit} \in [0, 1]$). Its average $\epsilon_{\max}$ and $\varepsilon_{\max}$ are much higher, and the percentage of near-optimal solutions is much lower than its non-prior counterpart and AOLS. Still, the figures show that its average CCS value is close to optimal and almost 74% of the instances are solved near-optimally. This suggests that there are a few instances on which SSII performs terrible, while on average its performance is acceptable. On the other hand, when concentrating on the prior weight range, i.e. $w_{profit} \in [0.5, 1]$, SSII with prior information clearly demonstrates better performance. Nonetheless, the average quality of the CCSes produced by SSII on this range is bested by AOLS in the 2-agent case.

| | | | $w_{profit} \in [0, 1]$ | | | | | $w_{profit} \in [0.5, 1]$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Time** | $\lvert\Psi\rvert$ | $V_\Psi$ | $\epsilon_{\max}$ | $\epsilon_{opt}$ | $\varepsilon_{\exp}$ | $\varepsilon_{\max}$ | $V_\Psi$ | $\epsilon_{\max}$ | $\epsilon_{opt}$ | $\varepsilon_{\exp}$ | $\varepsilon_{\max}$ |
| OLS | | | | | | | | | | | | |
| CoRe | 980.9 | 8.4 | 1244.05 | - | - | - | - | 687.60 | - | - | - | - |
| AOLS | | | | | | | | | | | | |
| UCT* 0.01s | 3.1 | 3.3 | 1170.28 | .381 | 0.0 % | .058 | 160.52 | 633.06 | .358 | 0.0 % | .078 | 160.51 |
| UCT* 0.1s | 4.0 | 3.4 | 1213.12 | .222 | 8.1 % | .024 | 95.77 | 663.65 | .189 | 8.1 % | .034 | 95.37 |
| UCT* 1s | 15.8 | 5.1 | 1235.46 | .088 | 43.2 % | .007 | 37.84 | 680.02 | .064 | 43.2 % | .011 | 37.84 |
| UCT* 10s | 102.4 | 7.4 | 1242.94 | .020 | 89.2 % | .001 | 8.32 | 686.46 | .014 | 89.2 % | .002 | 8.32 |
| UCT* 20s | 168.5 | 7.7 | 1243.52 | .006 | 94.6 % | .000 | 3.71 | 687.02 | .006 | 94.6 % | .001 | 3.71 |
| SSII (np) | | | | | | | | | | | | |
| UCT* 1s | 20.1 | 4.0 | 1234.00 | .085 | 32.4 % | .008 | 42.24 | 679.35 | .078 | 32.4 % | .012 | 42.21 |
| UCT* 10s | 83.6 | 4.7 | 1242.87 | .021 | 64.9 % | .001 | 11.37 | 686.38 | .021 | 64.9 % | .002 | 11.33 |
| UCT* 60s | 239.8 | 5.1 | 1243.94 | .006 | 83.8 % | .000 | 2.25 | 687.49 | .006 | 83.8 % | .000 | 2.22 |
| SSII (p) | | | | | | | | | | | | |
| UCT* 1s | 20.2 | 4.1 | 1220.90 | .333 | 32.4 % | .018 | 58.75 | 676.35 | .095 | 40.5 % | .015 | 46.41 |
| UCT* 10s | 83.6 | 5.6 | 1241.33 | .137 | 70.3 % | .002 | 13.75 | 686.18 | .016 | 81.1 % | .002 | 10.12 |
| UCT* 60s | 239.9 | 5.9 | 1243.01 | .110 | 86.5 % | .001 | 5.10 | 687.59 | .002 | 97.3 % | .000 | 1.45 |

**Table 5.2** *Comparison of runtime taken (seconds) and resulting CCS quality of* OLS, AOLS *and* SSII *for 37 solved 3-agent instances of* random-mo. *The columns are equal to Table 5.1,*

In the three-agent case, presented in Table 5.2, a slightly different picture arises for SSII. The algorithm exhibits better performance within the region of the prior, demonstrated foremost by the lower maximum epsilon, as well as the expected and maximum CCS errors. The optimality percentage is also slightly higher but this corresponds to solving one more instance optimally than AOLS. It must be noted that overall SSII with 60 seconds for UCT* does take more runtime than AOLS with 20 seconds to produce its better CCS in the prior range. Hence the observed better performance may be a result of simply using more time. Nevertheless, when compared to its non-prior counterpart, the algorithm does exploit information regarding a prior weight distribution to produce a better CCS in that range. Again, both approximate methods produce a CCS that is very close to optimal already when allowing UCT* at most 10 seconds to run. One final interesting observation is that while focusing on the prior region SSII solves more instances optimally in the full range weight than its non-prior counterpart, whereas in

the two-agent problems the $\epsilon_{opt}$ score is lower of the prior variant. That is, for the three-agent problems $\epsilon_{opt}$ is higher in the full weight range $w_{profit} \in [0, 1]$ for SSII that samples only prior $w_{profit} \in [0.5, 1]$ than its counterpart that samples the full weight range. While the two-agent case seems intuitive, focusing on the prior range will lead to less optimal approximation over the full weight range, this is not observed for the three-agent problems. Although this has not been studied further, the hypothesis ventured here is that (almost) all of the corner weights lie within the prior range of $w_{profit} \in [0.5, 1]$, therefore performing more sampling in this range will lead to better CCS approximations than distributing the 10 available samples over the full weight range. Further experiments with a different prior (e.g. $w_{profit} \in [0, 0.5]$) or allowing SSII without prior information an equal number of samples in the range $w_{profit}$ can be undertaken to validate this hypothesis.

## 5.5 Further Discussion

This chapter proposes two new approximation techniques for multi-objective planning problems with unknown weights that can be modelled as MDPs, both with their own strengths and weaknesses. The Approximate Optimistic Linear Support (AOLS) algorithm, an approximate adaptation of the OLS algorithm [217], produces an approximate CCS with a bounded error $\epsilon$ that can be specified by the planner and is typically much more time-efficient than its exact predecessor OLS. When exact weights are not known but a prior distribution over the weight simplex, i.e. the planner has imprecise information regarding the weight vectors of interest, the Scalarised Sample-based Iterative Improvement (SSII) algorithm may be more appropriate. While AOLS does not consider such information, SSII focuses specifically on the most interesting areas of the weight simplex and, as a consequence, oftentimes produces a higher-quality CCS in the prior region.

While the experimental evaluation of Section 5.4 provides supporting evidence for the aforementioned strengths and weaknesses, further experiments are required to investigate the scalability of these results. On account of the complexity of 4-agent problems, demonstrated in Chapter 4, no such experiments have been considered. Additional research is required to study whether the results presented in this chapter also apply to larger problems. Given the pattern that emerges from comparing 2-agent and 3-agent instances it seems likely that the strengths of both algorithms will be more pronounced on 4-agent problems, but this has not been (empirically) confirmed. Furthermore, in all experiments regarding SSII the same prior was considered and the algorithm always used 10 uniform samples over that prior. More experiments are required to determine how SSII behaves on different (types of) prior weight distributions and with other sampling methods. Additionally, further research into the weight-selection heuristics may also lead to better performance of the algorithm.

Another interesting avenue for future research lies in harnessing the strengths of both algorithms. A hybrid algorithm could combine the appealing solid theoretical foundation of the inner-loop approach of AOLS with a prior-based heuristic weight selection similar to that of SSII. This could yield an algorithm that will both produce a guaranteed $\epsilon$-approximate CCS but with an (expected) better anytime performance

in the prior. Furthermore, where AOLS now stops if the maximum relative error drops below $\epsilon$, the SSII heuristic may help in still increasing the CCS value until the total available runtime has been exhausted. Finally, when the approximate MDP solver does not guarantee an error bound of $\epsilon$ (such as UCT*), the heuristic approach of SSII may still be able to guide the CCS weight vector selection to quickly identify corner points at which the value can be improved most.

It must be remarked that both OLS and AOLS relied on a single-objective MDP solver that returns both the (optimal) policy and its expected value per objective. Although many existing single-objective MDP solvers return a (scalar) expected value as well as the policy and can be adapted to instead return the value vector over all objectives, this is not true in general. In the latter case a separate policy evaluation step must be included in both algorithms that performs an exact evaluation of the policy. A naive implementation thereof is polynomial in the number of actions and states squared, i.e. $O(|A|\,|S|^2)$, and would compromise the efficiency of both AOLS and SSII. Although for MPP and many other problems a more efficient expected value computation is possible, this is not true for all combinations of solvers and domains. For these (uncommon) situations it must be studied whether approximate policy evaluation can be employed in combination with either algorithm.

# Chapter 6

# Maintenance Planning with Self-interested Agents

All of the approaches to solve the maintenance planning problem that were presented in previous chapters required that the agents participating in the maintenance planning problem are fully cooperative: all participating agents willingly provide all their information regarding revenues, preferences and costs to a central point of computation. Then, this central coordinator – a public or private third-party institution, the road authority, etc. – can develop a contingent maintenance plan that is optimal *for the group as a whole*. This assumption, however, is in many cases not realistic. Service providers in the maintenance planning problem are typically commercial parties that focus mostly on their own interests, i.e. they are *selfish*, leading to a misalignment between the individual goals and the global or group goal. Moreover, information regarding revenues, preferences and costs is almost always considered vital business information that agents are not willing to disclose to others, unless they can profit from it. Note that selfishness does not imply maliciousness: selfish agents strive to maximise their gain, regardless of the impact on other agents, but do not intentionally seek to harm others [212].

In a setting with selfish agents, monetary incentives can be used to elicit private information from the agents, rewarding them for providing the planner with the necessary information to develop a globally optimal plan. A challenges that arises from this approach, however, is that agents might 'cheat' the planner in order to increase their personal gain. If an agent knows how the planning is performed, it can provide false information so that the end result is better for the agent itself, most likely at the cost of other participants. Therefore an incentive scheme to obtain private information should not only reward agents when they disclose it, but also only when they do so *truthfully*. Finding such an incentive scheme is the focus of *mechanism design* theory (see Appendix E for a refresher) and the basis of the solution presented in Section 6.1.2.

Another approach to deal with selfishness in planning problems that is discussed in this chapter concentrates on decentralising the decision making such that each agent develops its own planning. More specifically, if an agent knows the plans of all others at the beginning of his decision making, it can react optimally *with respect to the*

*plans of others*. When this this planning performed iteratively, i.e. each agent gets a turn to react to all others, over multiple rounds, this approach is known as *best-response planning*. This method offers several benefits: agents do not have to disclose (private) information other than their intended plan, agent autonomy is preserved and the computational effort of the planning problem is distributed, albeit almost always at the loss of optimality.

**Contributions** In this chapter, the *dynamic maintenance mechanism* is presented as a novel optimal dynamic mechanism for maintenance planning. This mechanism was first introduced by Scharpff et al. [228]; this thesis extends that work by a formal proof showing that the maintenance mechanism is indeed a *dynamic Vickrey-Clarke-Groves mechanism*, and thus welfare-maximising, strategyproof and budget-balanced in a within-period, ex-post Nash equilibrium (informally, the equilibrium that results when agents are utility maximising, accounting future uncertainty).

Furthermore, in Section 6.2 of this chapter a novel class of congestion games is presented, termed *stochastic planning congestion games*, accompanied by a best-response planning approach that was briefly considered by Scharpff et al. [228], based upon the method of Jonsson and Rovatsos [127]. While this approach no longer guarantees optimal planning, it better preserves the privacy of the agents and is more time-efficient than the dynamic VCG mechanism. Most of the work in Section 6.2 has not been published before and is a contribution made by this thesis.

## 6.1 A Dynamic Mechanism Approach to Maintenance Planning

Standard 'static' mechanism design theory (Appendix E) is suitable for dealing with single-shot games where agents perform one single joint action, resulting in one outcome for which the mechanism computes payments that are incurred once. For almost all single-shot games, the VCG mechanism provides a straightforward solution that is satisfactory; it is both strategyproof, (ex-post) individually rational and (weakly) budget balanced. When applying static VCG mechanisms in a sequential setting where types change over time, as is the case in the maintenance planning problem, it fails to guarantee the properties that held for the single-shot setting. The intuition behind this is that in sequential decision making, decisions influence each other not only for the current decision but might also affect future decisions, and payments in a static mechanism cannot adequately account for this (an example of this is given in Example 6.11). In reaction to this research focused on *dynamic mechanism design*, extending existing mechanism design theory for one-shot games to the setting of sequential games. Moreover, as static mechanisms can be seen as a special case of dynamic mechanisms, all of the impossibility results transfer directly to the dynamic setting. This section introduces first the most important notions of dynamic mechanism design, mainly derived from the presentation by Cavallo [54]. Thereafter, in Section 6.1.2, a dynamic mechanism for maintenance planning is described.

Before going into dynamic mechanism design theory, however, it is first necessary to clarify what is meant by dynamic mechanisms. In the literature there are two main strands of research that each work on dynamic mechanisms, but each address a different type of dynamicity. Arguably the most studied version is that of online mechanism design in which typically agents and/or actions become available or unavailable over the course of time. A few examples of online mechanism design problems can be found in WiFi-pricing [89], where customers come and go, scheduling of electrical vehicle charging [91] and conditional auctions with bidders arriving at varying times [149]. In this thesis online mechanism design is not considered, as they address issues not encountered in the problems presented in this thesis.

Here the focus is on 'offline' dynamic mechanisms in the sense that, although the complete model is known in advance and does not change, the environment or preferences of agents in the sequential game might change over time. Agents in such a game have a dynamic type that includes all possible preferences an agent might have, for current and future decisions, but it is not known in advance what type will be realised and how it will evolve during sequential decision making (otherwise a static mechanism would suffice). Hence, the valuations that an agent has for outcomes of the game can change over time due to for example actions performed by other agents or a random change in the environment. As such, a dynamic type typically defines 'static types' – that can be arbitrarily complex as before – for every anticipated possible state of the world, which may change as a result of new information that becomes available.

Dynamic types can be represented through various models, e.g. black-box simulation or a numerical model, but Markov decision processes have received the most attention in this aspect. In this model, the dynamic type of an agent is represented by an MDP in which the states are possible type realisations, i.e. each state $s^t \in S_i$ corresponds to a type $\theta_i^t \in \Theta_i$ at time $t$ for agent $i$. The transition probability function and reward function model the decisions an agent can make, the uncertainty regarding future types and the expected (future) rewards. At every time step during the execution of the mechanism – typically referred to as rounds – agents submit their dynamic type MDP, with its current type as its initial state, such that the mechanism knows the current and future actions of an agent and its consequential (reported) utility.

For now, it is assumed that the agents have some model to represent their dynamic type space $\Theta_i$ and instances $\theta_i^t$ from this space; the details of a dynamic type do not influence the general concepts and definitions of dynamic mechanism design. Later, in Example 6.8, an example of a dynamic type modelled as a Markov decision process is shown. First, however, dynamic mechanisms are defined after which dynamic extensions of the previously discussed static solution concepts and desired mechanism properties are presented. The definition of a dynamic mechanism is as follows:

**Definition 6.1 Dynamic Mechanism**

A (discrete-time, finite-horizon) dynamic mechanism for a sequential game $\mathcal{G}$ is defined by the tuple $\langle \Gamma, \boldsymbol{\rho}, T \rangle$ in which:

- $\Gamma : \boldsymbol{\Lambda} \mapsto O$ is the sequential social choice function,

- $\boldsymbol{\rho} = \{\rho_i\}_{i \in \boldsymbol{N}}$ such that the dynamic payment for every agent $i$ is defined as a function $\rho_i : \boldsymbol{\Lambda} \times T \mapsto \mathbb{R}$, and
- $T = \{1, 2, \ldots, h\}$ is a discrete set of $h$ time steps.

In every round $t \in T$, the players submit a joint action $\vec{a} \in \boldsymbol{\Lambda}$ based upon the joint dynamic type $\boldsymbol{\theta}^t \in \boldsymbol{\Theta}$ at time $t$ and joint strategy $\vec{\sigma} \in \boldsymbol{\Sigma}$, i.e. $\vec{a} = \vec{\sigma}(\boldsymbol{\theta}^t)$, from which outcome $\Gamma(\vec{a}) \in O$ is computed, and obtain utility

$$u_i^t(\theta_i^t, \vec{a}) = v_i(\theta_i^t, \Gamma(\vec{a})) + p_i^t(\vec{a}) \tag{6.1}$$

such that $p_i^t(\vec{a})$ conveniently denotes $\rho_i(\vec{a}, t)$ and the valuation $v_i$ is derived from the utility function of agent $i$ in the sequential game at time $t$ (similar to Definition E.6).

For dynamic mechanisms an equivalent to the revelation principle exists: any indirect dynamic mechanism that implements a sequential SCF $\Gamma$ in equilibrium can be transformed into an equivalent direct dynamic mechanism [234]. Therefore only direct dynamic mechanisms are considered in the remainder of this chapter. As in the static case, the revelation principle allows the social choice function rule and payments to be defined over the reported types, e.g. $\Lambda_i = \Theta_i$, $\Gamma(\hat{\boldsymbol{\theta}}^t) \in O$ and $p_i^t(\hat{\boldsymbol{\theta}}^t) \in \mathbb{R}$.

Execution of a dynamic mechanism resembles executing $|T|$ static mechanisms consecutively in that at every time step $t$ (also referred to as a round) the game is played, agents report dynamic types $\hat{\boldsymbol{\theta}}^t$ to the mechanism, the mechanism determines the outcome $\Gamma(\hat{\boldsymbol{\theta}}^t)$ of the game and transfers payments $p_i^t(\hat{\boldsymbol{\theta}}^t)$ based upon the reported dynamic types. When one such a round has been completed, the next round is executed identically but now starting from the next state of the game that resulted after execution of the actions for time $t$, with new dynamic types $\hat{\boldsymbol{\theta}}^{t+1}$. The timeline of sequential execution of a direct dynamic mechanism is shown in Figure 6.1.

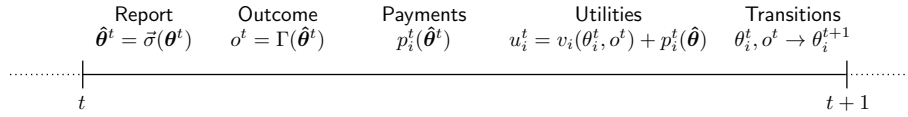



**Figure 6.1** *Illustration of sequential execution of a dynamic mechanism: the agents report a joint type from which an outcome and corresponding payments are computed. When the outcome and payments become known, agents can compute the utility they obtain in this round of the execution. Finally, the agents transition to their new type for the next round. Notice that when computing the utility an agent really obtains, its valuation is determined using its true type $\theta_i^t$.*

Analogous to static mechanisms, the goal of a dynamic mechanisms is to implement the sequential social choice function in a game equilibrium where truthful reporting is the optimal strategy, but the solution concepts for sequential games are different to that of one-shot games. In particular, with dynamic types it is no longer feasible to have a dominant strategy implementation because of the uncertainty regarding the future types that will be realised. Instead a Nash-like equilibrium solution concept is

defined in which agents employ a strategy that *in expectation* maximises their utility over all (remaining) rounds of the mechanism, assuming that all other agents do the same. This is known as the within-period, ex-post Nash equilibrium and is defined as:

**Definition 6.2 Within-period, Ex-post Nash Equilibrium**

A joint strategy $\vec{\sigma}^{\,*} = \langle \sigma_1^*, \sigma_2^*, \ldots, \sigma_n^* \rangle \in \boldsymbol{\Sigma}$ constitutes a within-period, ex-post Nash equilibrium from time $t \in T$ until the execution horizon $h$ if for every time step $x \in [t, h]$ and associated true dynamic type $\boldsymbol{\theta}^x \in \boldsymbol{\Theta}$ every agent $i$ plays strategy $\sigma_i^* \in \Sigma_i$ when sequential SCF $\Gamma$ is used, such that

$$\sigma_i^* = \arg\max_{\sigma_i \in \Sigma_i} \mathbb{E}\Big[ \sum_{x=t}^{h} u_i\big(\boldsymbol{\theta}^x, \big\langle \sigma_i(\theta_i^x), \vec{\sigma}^{\,*}_{-i}(\boldsymbol{\theta}^x_{-i}) \big\rangle\big) \mid \boldsymbol{\theta}^x, \vec{\sigma}^{\,*}_{-i}, \Gamma \Big] \tag{6.2}$$

The within-period, ex-post Nash equilibrium, abbreviated here as dynamic Nash, is the strongest solution concept available for dynamic mechanisms and is also the only one considered here. For mechanisms that implement $\Gamma$ in dynamic Nash, the same properties that are desired in static mechanisms apply, although adapted for the dynamic setting. Before providing the formal definitions of these properties, first shorthand notations for the expected valuation, payment and utility are defined. The valuation that an agent $i$ expects to receive from time step $t \in T$ until the mechanism's execution horizon $h$ when joint strategy $\vec{\sigma} \in \boldsymbol{\Sigma}$ is played and the mechanism uses sequential SCF $\Gamma$ is given by

$$\tilde{V}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \mathbb{E}\Big[ \sum_{x=t}^{h} v_i\big(\hat{\theta}_i^x, \Gamma(\hat{\boldsymbol{\theta}}^x)\big) \;\Big|\; \hat{\boldsymbol{\theta}}^x = \vec{\sigma}(\boldsymbol{\theta}^x) \Big] \tag{6.3}$$

and the sum of all expected valuations is written as $\tilde{V}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \sum_{i \in \boldsymbol{N}} \tilde{V}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma})$. Similarly, the expected payoff for agent $i$ from time $t$ until horizon $h$ when agents play joint strategy $\vec{\sigma}$ is given by

$$\tilde{p}_i(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \mathbb{E}\Big[ \sum_{x=t}^{h} p_i^x\big(\hat{\boldsymbol{\theta}}^x\big) \;\Big|\; \hat{\boldsymbol{\theta}}^x = \vec{\sigma}(\boldsymbol{\theta}^x) \Big] \tag{6.4}$$

for every reported dynamic type $\hat{\boldsymbol{\theta}}^t$. The expected payment sums the total payoff that the agent expects to receive from or transfer to the mechanism. Combining both shorthands, the utility that an agent anticipates under a dynamic mechanism from time $t$ onward when the agents play a joint strategy $\vec{\sigma}$ and sequential SCF $\Gamma$ is used is

$$\begin{aligned} \tilde{U}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) &= \tilde{V}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) + \tilde{p}_i(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) \\ &= \mathbb{E}\Big[ \sum_{x=t}^{h} v_i\big(\hat{\theta}_i^x, \Gamma(\hat{\boldsymbol{\theta}}^x)\big) + p_i^x\big(\hat{\boldsymbol{\theta}}^x\big) \;\Big|\; \hat{\boldsymbol{\theta}}^x = \vec{\sigma}(\boldsymbol{\theta}^x) \Big] \end{aligned} \tag{6.5}$$

Observe that when a model is used in which the dynamic type realisations are uncertain but the probability distribution is known at time $0$, e.g. a Markov decision process,

the expectation becomes a sum over all possible dynamic types times their probability (assuming agents are truthful or a distribution over the misreports is known).[45] For example, the expected utility given a model with known probabilities can be written as

$$\tilde{U}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \sum_{x=t}^{h} \sum_{\hat{\boldsymbol{\theta}}^x \in \boldsymbol{\Theta}} Pr(\hat{\boldsymbol{\theta}}^x | \hat{\boldsymbol{\theta}}^t, \vec{\sigma}) \cdot v_i\big(\hat{\theta}_i^x, \Gamma(\hat{\boldsymbol{\theta}}^x)\big)$$

such that $Pr(\hat{\boldsymbol{\theta}}^x | \hat{\boldsymbol{\theta}}^t, \vec{\sigma})$ denotes the probability of dynamic type $\hat{\boldsymbol{\theta}}^x$ being reported at time $x$ when dynamic type $\hat{\boldsymbol{\theta}}^t$ was reported at time $t$ and joint strategy $\vec{\sigma}$ is played. Similar substitutions can be made for the expected valuations and payments.

A mechanism is said to be incentive compatible in dynamic Nash if following a truthful strategy will always yield the maximum utility in expectation for every agent, given that every other agent will also follow a utility-maximising strategy. The latter condition that is now included in the definition of incentive compatibility for the dynamic setting is due to the strategy assumption required in dynamic Nash. This is the best available solution concept for dynamic mechanisms, an equivalent of a dominant strategy equilibrium cannot exist in the dynamic setting because it is impossible to optimise utility no matter what other agents do when there is uncertainty regarding dynamic types that will be realised in the future (unless agents are completely independent, but then there is no more need for a mechanism).

**Definition 6.3 Within-period, Ex-post Incentive Compatibility**

A dynamic mechanism $\langle \Gamma, \boldsymbol{\rho}, T \rangle$ is said to be (within-period, ex-post) incentive compatible if the sequential social choice function $\Gamma$ is implemented in a within-period, ex-post Nash equilibrium where truthful reporting is utility-maximising in expectation for every agent from the current time onward. Formally, for every agent $i \in \boldsymbol{N}$, all time steps from a time $t \in T$ until horizon $h$ and all corresponding joint dynamic types $\boldsymbol{\theta}^t \in \boldsymbol{\Theta}$, there exists a strategy $\sigma_i^* \in \Sigma_i$ such that

$$\tilde{U}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \langle \sigma_i^*, \vec{\sigma}\,^*_{-i} \rangle) \geq \tilde{U}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \langle \sigma_i, \vec{\sigma}\,^*_{-i} \rangle)$$

and every $\sigma_j^* \in \vec{\sigma}\,^*_{-i}$ is an expected utility-maximising strategy for agent $j$.

Similar adaptations can be derived for individual rationality:

**Definition 6.4 Within-period, Ex-post Individual Rationality**

A dynamic mechanism $\langle \Gamma, \boldsymbol{\rho}, T \rangle$ is (within-period, ex-post) individually rational if the expected utility of every agent is non-negative from the current time onward. Formally, for every agent $i \in \boldsymbol{N}$, all time steps from $t \in T$ until horizon $h$ and all corresponding joint dynamic types $\boldsymbol{\theta}^t \in \boldsymbol{\Theta}$, there exists a strategy $\sigma_i^* \in \Sigma_i$ that is utility-maximising in expectation and

$$\tilde{U}_i^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \langle \sigma_i^*, \vec{\sigma}\,^*_{-i} \rangle) \geq 0$$

[45] *This assumption was already made by Bergemann and Välimäki [30] in their initial work on dynamic mechanisms, and is implicit in the dynamic mechanisms theory of Cavallo [54] and this thesis.*

and every $\sigma_j^* \in \vec{\sigma}^*_{-i}$ is an expected utility-maximising strategy for agent $j$.

and budget balance:

**Definition 6.5 Within-period, Ex-post Budget Balance**

A dynamic mechanism $\langle \Gamma, \boldsymbol{\rho}, T \rangle$ is (within-period, ex-post) budget balanced if the sum of expected payments over all agents from the current time onward is equal to zero. Formally, for all time steps from $t \in T$ until horizon $h$ and all corresponding joint dynamic types $\boldsymbol{\theta}^t \in \boldsymbol{\Theta}$, and joint strategy $\vec{\sigma}^* \in \boldsymbol{\Sigma}$:

$$\sum_{i \in \boldsymbol{N}} \tilde{p}_i(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}^*) = 0$$

where every strategy $\sigma_i^* \in \vec{\sigma}^*$ maximises the expected utility of agent $i$ from the current time onward. When the sum of expected payments is non-positive, the mechanism is known as weakly (within-period, ex-post) budget balanced or no-deficit.

Computational efficiency of a mechanism is naturally also desirable but more often than not difficult to achieve in dynamic mechanisms. This is not necessarily a consequence of using dynamic mechanisms but more of the underlying dynamic setting. The social choice function of a dynamic mechanisms has to deal with a number of dynamic types that is potentially exponential in the execution length $T$, of which the future realisations are typically uncertain, and thus a static outcome is no longer sufficient. For example in the maintenance planning problem, this is the difference between finding an optimal plan in the static setting and finding an optimal contingent plan that is optimal in expectation over all possible future delays in the dynamic setting. The latter planning problem is much harder to solve and this increase in problem complexity due to incorporating dynamicity is typical for sequential social choice functions. Computational efficiency of dynamic mechanisms is simultaneously more desirable but harder to achieve.



### 6.1.1 Dynamic Vickrey-Clarke-Groves Mechanisms

For the within-period ex-post Nash equilibrium a dynamic version of the Vickrey-Clarke-Groves mechanism exists that is concurrently (within-period, ex-post) incentive compatible, individually rational and weakly budget balanced. The dynamic VCG mechanism was initially presented by Bergemann and Välimäki [30] and Cavallo [53] showed that, alike its static counterpart, it is the only dynamic mechanism that achieves this in dynamic Nash.

In the static VCG mechanism, the payments for agent $i$ were defined according to the impact of that agent on the reported valuations $v_{-i}$ of the other agents. In dynamic VCG a similar payment is defined, although based on the impact on the *expected* reported valuations over the remaining time. The expected valuation of all agents other than agent $i$ when all agents participate for every jointly reported dynamic

type $\hat{\boldsymbol{\theta}}^t$ during the remaining period $[t, h]$ when joint strategy $\vec{\sigma}$ is played is given by:

$$\tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t)) + \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t+1}, \vec{\sigma})$$

and when agent $i$ does not participate, the expected valuation of all other agents is:

$$\tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) = v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma_{-i}(\hat{\boldsymbol{\theta}}_{-i}^t)) + \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t+1}, \vec{\sigma})$$

There is a significant difference between the two (similar to the static case): the former computes outcomes using the reported dynamic types of all agents, but does not include the valuation of agent $i$, whereas in the latter only outcomes are considered in which reported dynamic type of $i$ is disregarded, i.e. the outcome is computed over agents $\boldsymbol{N} \setminus \{i\}$ (denoted by $\Gamma_{-i}$).

As with the static VCG mechanism, the dynamic VCG mechanism is only one of the possible mechanisms in the dynamic-Groves mechanism class, the dynamic counterpart of the class of static Groves mechanisms. Both definitions are combined in the following:

**Definition 6.6 Dynamic Vickrey-Clarke-Groves (VCG) Mechanism**

A dynamic mechanism $\langle \Gamma, \boldsymbol{\rho}, T \rangle$ is a dynamic Vickrey-Clarke-Groves mechanism if it is a member of the dynamic-Groves class:

(i) The sequential social choice function $\Gamma$ optimises the expected overall welfare for every reported joint dynamic type $\hat{\boldsymbol{\theta}}^t$ at every time $t \in T$:

$$\Gamma(\hat{\boldsymbol{\theta}}^t) \in \mathbb{E}\Big[ \sum_{x=t}^{h} \arg\max_{o^x \in O} \sum_{i \in \boldsymbol{N}} v_i(\hat{\boldsymbol{\theta}}^x, o^x) \mid \hat{\boldsymbol{\theta}}^x \Big]$$

(ii) The expected payoff $\tilde{p}_i$ at time $t \in T$ for every agent $i$ is relative to its impact on the reported expected valuation of other agents, now and in the future, such that for every joint strategy $\vec{\sigma}$ and all reported dynamic types $\hat{\boldsymbol{\theta}}^t$:

$$\tilde{p}_i(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) \tag{6.6}$$

in which $\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) = \mathbb{E}\big[\sum_{x=t}^{h} H_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^x) \mid \hat{\boldsymbol{\theta}}_{-i}^x = \vec{\sigma}_{-i}(\theta_{-i}^x)\big]$ where $H_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t) \in \mathbb{R}$ is a function that excludes the dynamic type of agent $i$, and the expected payoff $\tilde{p}_i$ is defined as in Equation 6.4.

and said mechanism is a dynamic VCG mechanism if it additionally uses a dynamic Clarke tax:

(iii) Each agent $i$ pays exactly the difference in expected valuation that is caused by its participation by letting $\tilde{H}_{-i}^{\Gamma_{-i}}$ be the expected valuation that the other agents could reportedly have obtained, now and in the future, without agent $i$:

$$\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) = \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i})$$

for every reported dynamic type $\hat{\boldsymbol{\theta}}^t$ and joint strategy $\vec{\sigma}_{-i}$.

The Clarke tax in the dynamic VCG mechanism is enforced by $\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t}, \vec{\sigma}_{-i}) = \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t}, \vec{\sigma}_{-i})$ but, as a result of Lemma 6.7 (see below), it is possible to write the payment of agent $i$ at time $t$ as:

$$p_i^t(\hat{\boldsymbol{\theta}}^t) = v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t)) + \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}^{t+1}, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t}, \vec{\sigma}_{-i}) \tag{6.7}$$

where $\sigma_{-i}^{*}$ is defined as before and $\hat{\boldsymbol{\theta}}^{t+1}$ are the reported dynamic types after observing outcome $\Gamma(\hat{\boldsymbol{\theta}}^t)$ at time $t$ that included all agents. This is shown in the following lemma:

**Lemma 6.7 Dynamic VCG payment**

Setting $\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t}, \vec{\sigma}_{-i}) = \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^{t}, \vec{\sigma}_{-i})$ leads to the payment of Equation 6.7.

*Proof.* Similar to the proof by Bergemann and Välimäki [30], see Appendix A.4. □

Summarising the notions presented in this section, dynamic VCG mechanism provides a standard tool to implement incentive compatibility, individual rationality and (weak) budget balance in a within-period, ex-post Nash equilibrium. Moreover, it is the only possible dynamic mechanism that is able to obtain all these properties concurrently in such an equilibrium. It is for this reason that most of the work on dynamic mechanisms is based upon the dynamic VCG mechanism and that it is also the only one considered in this thesis. In the next section a dynamic VCG mechanism is presented for the maintenance planning problem with selfish agents.

### 6.1.2 The Dynamic Maintenance Mechanism

This section presents a coordination method for maintenance planning with selfish agents that uses the dynamic VCG mechanism of the previous section to achieve truthful maintenance cost reports and thus allows for globally optimal planning. It was observed by Cavallo [53] that the local dynamic types and type reports in (at least) stochastic planning problems such as MPP can be modelled using the Markov decision process framework. Essentially, each agent $i \in \boldsymbol{N}$ has a local MDP $M_i$ that represents its local dynamic type space $\Theta_i$, corresponding to its local planning problem. Then, in every round of the mechanism, each agent reports the state it is currently in, the decisions it can make now and in the future (the transitions in its MDP), and the valuation for each of these decisions (transition rewards). From these reports, the mechanism can construct a joint MDP $\boldsymbol{M} = \times_{i \in n} M_i$, solve it to obtain jointly optimal policy $\pi^*$, and finally compute payments based on this joint policy. At the end of the round, the agents are notified of the outcome, i.e. $\pi^*$, and the resulting payments, and they update the current state of their local type space MDP to their new type.

For the maintenance planning problem, the local dynamic type space MDP is defined by the tuple $M_i = \langle S_i, A_i, P_i, R_i \rangle$ and it is constructed identically to the single-agent MDP model of Section 3.3. To briefly recapitulate: states $S_i$ capture start and end times of maintenance activities $\mathcal{A}_i$, the set of actions $A_i$ contains a start action $\texttt{start}_i$ for each of the agent's activities $\mathrm{a}_i \in \mathcal{A}_i$, $P_i$ specifies a transition for every start action

to a state where the corresponding activity is or is not delayed, and $R_i$ specifies the (local) profit obtained (revenue minus maintenance costs) for performing the activity. Note that the network costs $\ell$ cannot be modelled locally (see Section 3.3.1), they are assumed to be publicly known and included in the sequential SCF of the dynamic mechanism that coordinates the agents. The dynamic type space is illustrated below.

**Example 6.8** Dynamic type space

Again there are two agents, $A$ and $B$, that want to coordinate their planning while maximising their utility. However, this time both agents have to plan two activities, such that their activity sets become $\mathcal{A}_A = \{\mathtt{a}_1, \mathtt{a}_2\}$ and $\mathcal{A}_B = \{\mathtt{b}_1, \mathtt{b}_2\}$, over three time steps. These activities and their associated maintenance costs are defined in Table 6.1a and Table 6.1b.

| $\mathcal{A}_A$ | t1 | t2 | t3 |
|---|---|---|---|
| $\mathtt{a}_1 = \langle 0, 1, 0.2, 1\rangle$ | 6 | 4 | 5 |
| $\mathtt{a}_2 = \langle 0, 1, 0, 0\rangle$ | 12 | 3 | 5 |

(a)

| $\mathcal{A}_B$ | t1 | t2 | t3 |
|---|---|---|---|
| $\mathtt{b}_1 = \langle 0, 1, 0, 0\rangle$ | 7 | 9 | 11 |
| $\mathtt{b}_2 = \langle 0, 2, 0, 0\rangle$ | 5 | 8 | 2 |

(b)

**Table 6.1** *Activity sets and maintenance costs of both agents, the costs are shown per time step in the last three columns of each table.*

From the activity definitions and maintenance cost functions, a local dynamic type space MDP can be constructed for both agents that models their local planning problems. The dynamic type space MDPs that correspond to the activity sets and maintenance costs of Table 6.1a and Table 6.1b, are shown in Figure 6.2a and Figure 6.2b respectively. Note that due to activity $\mathtt{a}_1$ of agent $A$ that can delay with probability $0.2$, the type of that agent is uncertain – i.e. dynamic – and therefore a dynamic mechanism is required. If none of the activities can delay, it would have been possible to coordinate the problem using the one-shot VCG mechanism of the previous section (but only then, see Example 6.11)

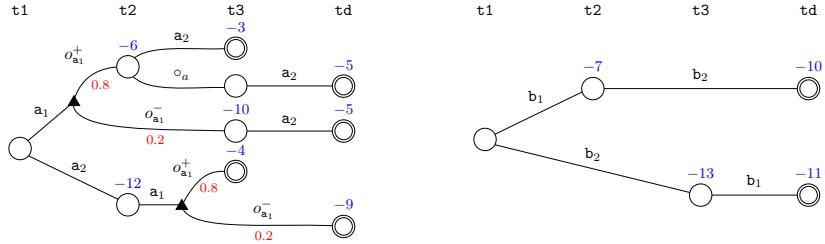



**Figure 6.2** *The dynamic type space MDPs of both agents: (a) type space of agent $A$ that contains one activity $\mathtt{a}_1$ with an uncertain duration and (b) that of agent $B$ with two activities that cannot delay. The reward for every* transition *is shown (in blue) over the result state and only the probabilities of both outcomes for uncertain activity $\mathtt{a}_1$ are shown labelled below the transitions (in red), all other transition probabilities are equal to one.*

The dynamic type that each agent reports to the mechanism is $\hat{\boldsymbol{\theta}}_i^t = \left\langle s_{\hat{\boldsymbol{\theta}}_i^t}, P_{\hat{\boldsymbol{\theta}}_i^t}, R_{\hat{\boldsymbol{\theta}}_i^t} \right\rangle$ such that $s_{\hat{\boldsymbol{\theta}}_i^t} \in S_i$ is the current state of the agent, $P_{\hat{\boldsymbol{\theta}}_i^t}$ the transition probabilities and $R_{\hat{\boldsymbol{\theta}}_i^t}$ the associated transition rewards. The transition probabilities that an agent reports include all possible transitions from the current state until the end of the mechanism period $T$ and implicitly defines the set of possible future static type states of the agent and its (remaining) action set. For every transition, the reward function $R_{\hat{\boldsymbol{\theta}}_i^t}$ contains an associated reward. The reported dynamic type $\hat{\boldsymbol{\theta}}_i^t$ hence defines an MDP itself, with $s_{\hat{\boldsymbol{\theta}}_i^t}$ as its initial state, describing all decisions the agent can make now and in the future. Moreover, if all the agents are truthful, they can submit their dynamic type space MDP to the mechanism at the beginning; thereafter only their current state has to be submitted every round, i.e. the realised dynamic type.

The sequential social choice function $\Gamma$ in dynamic VCG is a function that always returns an outcome that is optimal, given the reported types of all agents. For stochastic planning problems, such as MPP, this means that any optimal stochastic planning algorithm can be used as sequential SCF, e.g. the algorithm presented in Chapter 4. In particular, if agents are guaranteed to report truthfully and submit their full dynamic type space MDP to the mechanism, it suffices to solve the stochastic joint planning problem only once to obtain an optimal joint policy. This policy can then be used as the SCF of the mechanism because it exactly prescribes the optimal joint action for every combination of true types.[46]

In order to coordinate the maintenance planning problem with selfish agents such that the sum of agent valuations is maximised, the dynamic VCG mechanism is used. The valuation function of each agent captures the local value it has for doing maintenance, i.e. the sum of its revenues minus its maintenance and network costs. The revenues and costs for every agent $i \in \boldsymbol{N}$ are represented by the reward function $R_i$ of its (true) dynamic type space MDP (see Example 6.8). The network rewards however are defined over combinations of activities and can therefore not be expressed in terms of local states and actions. These need to be accounted for in the *joint* MDP that results from combining all dynamic type space MDPs, as discussed in Section 3.3. However, where previously the problem was to maximise the expected value of the group as a whole, here each agent strives to optimise its expected individual utility. Therefore, it is no longer sufficient to simply minimise the sum of all network costs, in addition these costs must be attributed to the right agent: only the agent(s) causing hindrance should be penalised for it. In the remainder of this section, a dynamic VCG mechanism will be constructed for the maintenance planning problem. Moreover, it is possible to design its payments so that they correspond to the network costs of MPP, thus making the maintenance mechanism an effective coordination method when dealing with selfish agents.

It is reasonable to assume that in the context of selfish agents, consensus has been established regarding the distribution of the network costs, for instance in the contract. A straightforward method is to evenly divide the network costs over the

[46] *Although $\pi^*_{-i}$, the optimal policy without agent $i$, must still be computed every round because this policy is influenced by* past *decisions of agent $i$, see Example 6.11 at the end of this section.*

agents responsible, as was the case in the game of Example E.4.[47] When dealing only with binary network costs, the valuation of an agent $i$ for a joint activity $\vec{\mathsf{a}}$ when every agent reports truthfully is then (informally) of the form $v_i(\hat{\boldsymbol{\theta}}^t, \vec{\mathsf{a}}) = w_i - c_i(\mathsf{a}_i, t) - 0.5 \times \sum_{\mathsf{a}_j \in \vec{\mathsf{a}}} \ell(\mathsf{a}_i, \mathsf{a}_j, t)$ such that $w_i$ and $c_i$ are respectively the one-time revenue and time-dependent maintenance cost function of activity $\mathsf{a}_i \in \vec{\mathsf{a}}$ of agent $i$ and $\ell$ the function that models inter-agent network costs. In general any asymmetric distribution of network costs can be used as long as its sum equals the total network cost, i.e. $\ell(\vec{\mathsf{a}}, t) = \sum_{i \in \boldsymbol{N}} \ell_i(\vec{\mathsf{a}}, t)$. An example of such a distribution is a weighted sum in which every agent is fined based on the proportion of network costs it generates.

Given such a network cost distribution method, the expected valuation that an agent $i \in \boldsymbol{N}$ has when the mechanism uses joint policy $\pi$ as its sequential SCF and the agents report according to joint strategy $\sigma$ can be modelled alike Equation 3.7 as:

$$\tilde{V}_i^{\boldsymbol{\pi}}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \mathbb{E}\Big[ \sum_{\mathsf{a}_k \in H^h \cap \mathcal{A}_i} w_k(\hat{\boldsymbol{\theta}}_i^t) - \sum_{t \in T} \Big( c_i(\hat{\boldsymbol{\theta}}_i^t, H^t) + \ell_i(H^t) \Big) \;\Big|\; \hat{\boldsymbol{\theta}}^t = \vec{\sigma}(\boldsymbol{\theta}^t), H^t = H^{t-1} \oplus \pi(\hat{\boldsymbol{\theta}}^t) \Big] \tag{6.8}$$

where $w_k(\hat{\boldsymbol{\theta}}_i^t)$ is the reported revenue for completing activity $\mathsf{a}_k$ and $c_i(\hat{\boldsymbol{\theta}}_i^t, \vec{\mathsf{a}}^{\,t}) = c_i(\hat{\boldsymbol{\theta}}_i^t, \mathsf{a}_i, t)$ is the reported maintenance cost for activity $\mathsf{a}_i \in \vec{\mathsf{a}}^{\,t} \cap \mathcal{A}_i$ of agent $i$ at time $t \in T$. Recall that the shorthands $c(H^t)$ and $\ell(H^t)$ denote respectively $c(H^t_{\mathcal{A}}(t), t)$ and $\ell(H^t_{\mathcal{A}}(t), t)$, a notation that was introduced in Chapter 3.

Using these new valuations in which the network costs are assigned to the responsible agents, it is possible to define a dynamic VCG mechanism for the problem. Let the true dynamic type space of every agent $i \in \boldsymbol{N}$ be modelled as an MDP $M_i = \langle S_i, A_i, P_i, R_i \rangle$ then the joint dynamic type MDP $\boldsymbol{M}$ can be constructed as $\langle \times_{i \in n} S_i, \{A\}_{i \in \boldsymbol{N}}, \times_{i \in n} P_i, \times_{i \in n} R_i \cup \ell \rangle$ (the details regarding this construction are explained in Section 3.3). Moreover, the joint MDP without agent $i$'s presence induced by the joint report $\hat{\boldsymbol{\theta}}^t_{-i}$ at time $t$ is defined as $\boldsymbol{M}^t_{-i}$ and is constructed likewise although without agent $i$. With these notations, the dynamic maintenance mechanism can be defined:

**Definition 6.9 Dynamic Maintenance Mechanism**

The Dynamic Maintenance Mechanism is defined as the tuple $\langle \pi^*, \boldsymbol{\rho}, T \rangle$ in which:

- $\pi^*$ is the optimal policy for joint MDP $\boldsymbol{M}$,
- the expected payment for every agent $i \in \boldsymbol{N}$ from time $t \in T$ until the planning horizon $h$ is given by

$$\tilde{p}_i(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) = \sum_{j \in \boldsymbol{N} \setminus \{i\}} \Big( \tilde{V}_j^{\pi^*}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_j^{\pi^*_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i}) \Big) \tag{6.9}$$

[47] *An alternative but intuitive way to model the network costs in a mechanism is by introducing an additional 'society' agent that represents the network users, with $\ell$ as its valuation. The payments that this agent makes correspond to the tax payers collectively compensating the service providers for their 'willingness' to minimise hindrance or, when it receives a payment, a financial compensation for the hindrance they collectively experienced.*

where $\tilde{V}_j^{\pi^*}$ and $\tilde{V}_j^{\pi^*_{-i}}$ are computed using Equation 6.8 and $\pi^*_{-i}$ is the optimal policy for $\boldsymbol{M}_i^t$, i.e. the joint type space MDP induced by the reported type $\hat{\boldsymbol{\theta}}^t_{-i}$ at time $t$, and

- $T$ is the planning period.

The dynamic maintenance mechanism, is efficient, i.e. it optimises the sum of expected utility over all agent. Furthermore, due to its construction, the payments under this mechanism align exactly with the expected harm in terms of network costs caused by concurrent maintenance. As a result, the dynamic maintenance mechanism can be shown to be a dynamic VCG mechanism:

**Theorem 6.10 The Dynamic Maintenance Mechanism is dynamic-VCG**

The dynamic maintenance mechanism is a dynamic VCG mechanism.

*Proof.* For the Dynamic Maintenance Mechanism to be a dynamic VCG mechanism, the three conditions for a dynamic VCG mechanism must be satisfied:

(i) The expected valuation of an agent $i \in \boldsymbol{N}$ is given by $\tilde{V}_i^{\pi^*}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma})$ from Equation 6.8. Summing the expected valuations over all agents results in (using shorthand $\hat{\boldsymbol{\theta}}^t, H^t$ to denote $\hat{\boldsymbol{\theta}}^t = \vec{\sigma}(\boldsymbol{\theta}^t), H^t = H^{t-1} \oplus \pi^*(\hat{\boldsymbol{\theta}}^t)$:

$$\begin{aligned}
&\sum_{i \in \boldsymbol{N}} \tilde{V}_i^{\pi^*}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) \\
&= \sum_{i \in \boldsymbol{N}} \mathbb{E}\Big[ \sum_{\mathsf{a}_k \in H^h \cap \mathcal{A}_i} w_k(\hat{\boldsymbol{\theta}}^t_i) - \sum_{t \in T} \Big( c_i(\hat{\boldsymbol{\theta}}^t_i, H^t) + \ell_i(H^t) \Big) \;\Big|\; \hat{\boldsymbol{\theta}}^t, H^t \Big] \\
&= \mathbb{E}\Big[ \sum_{\mathsf{a}_k \in H^h} w_k(\hat{\boldsymbol{\theta}}^t) - \sum_{t \in T} \Big( c(\hat{\boldsymbol{\theta}}^t, H^t) + \ell(H^t) \Big) \;\Big|\; \hat{\boldsymbol{\theta}}^t, H^t \Big]
\end{aligned}$$

in which $c(\hat{\boldsymbol{\theta}}^t, \vec{\mathsf{a}}, t) = \sum_{i \in \boldsymbol{N}} c_i(\hat{\boldsymbol{\theta}}^t_i, \mathsf{a}_i, t)$ due to Equation 3.5 and $\ell(\vec{\mathsf{a}}, t) = \sum_{i \in \boldsymbol{N}} \ell_i(\vec{\mathsf{a}}, t)$ by construction. This is exactly equal to the expected value of Equation 3.7 that the optimal policy $\pi^*$ optimises for the (reported) joint dynamic type space MDP $\boldsymbol{M}$. As a consequence, policy $\pi^*$ optimises the expected overall welfare.

(ii) and (iii): By definition of $\tilde{V}_j^{\pi}$ (Equation 6.8) it holds that $\sum_{j \in \boldsymbol{N} \setminus \{i\}} \tilde{V}_j^{\pi^*}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma})$ and $\sum_{j \in \boldsymbol{N} \setminus \{i\}} \tilde{V}_j^{\pi^*_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i})$ are equivalent to $\tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma})$ and $\tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i})$ respectively when optimal policy $\pi^*$ is used as sequential SCF and $\pi^*_{-i}$ is the optimal policy for the joint dynamic type space MDP $\boldsymbol{M}^t_{-i}$ that disregard reports $\hat{\boldsymbol{\theta}}^t_i$ by agent $i$ (its states, transitions and rewards are not included in the MDP). □

Because of Theorem 6.10, the dynamic maintenance mechanism is a dynamic VCG mechanism and, as a corollary, it is incentive compatible, individually rational and weakly budget balanced in within-period, ex-post Nash equilibrium.

**Example 6.11** Dynamic Maintenance Mechanism

Consider once more the two agents, $A$ and $B$, from Example 6.8 that need to perform activities $\{\mathtt{a}_1, \mathtt{a}_2\}$ and $\{\mathtt{b}_1, \mathtt{b}_2\}$ respectively. Their activities are defined as shown in Table 6.2a and Table 6.2b. Additionally, the network costs are defined pair-wise and time independent according to the matrix of Table 6.2c. For example, for every time step that activity $\mathtt{a}_2$ and $\mathtt{b}_1$ are concurrently executed, the network cost $\ell$ is $\ell(\mathtt{a}_2, \mathtt{b}_1) = 5$ (omitting $t$ because of the time independence). Observe that activity $\mathtt{a}_1$ of agent $A$ may delay with a probability of $0.1$ and it requires one additional time step when it is delayed.

| $\mathcal{A}_A$ | t1 | t2 | t3 | t4 |
|---|---|---|---|---|
| $\mathtt{a}_1 = \langle 0, 1, 0.1, 1\rangle$ | 6 | 11 | 9 | 18 |
| $\mathtt{a}_2 = \langle 0, 1, 0, 0\rangle$ | 12 | 5 | 3 | 7 |

(a)

| $\mathcal{A}_B$ | t1 | t2 | t3 | t4 |
|---|---|---|---|---|
| $\mathtt{b}_1 = \langle 0, 1, 0, 0\rangle$ | 7 | 2 | 3 | 25 |
| $\mathtt{b}_2 = \langle 0, 1, 0, 0\rangle$ | 5 | 8 | 15 | 17 |

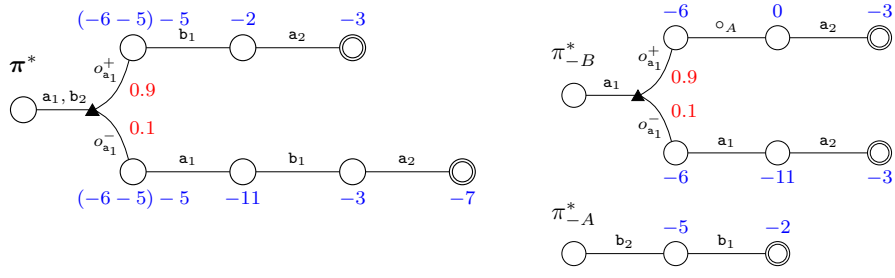
(b)

| $\ell$ | $\mathtt{b}_1$ | $\mathtt{b}_2$ |
|---|---|---|
| $\mathtt{a}_1$ | 12 | 5 |
| $\mathtt{a}_2$ | 9 | 15 |

(c)

**Table 6.2** *Activity sets and maintenance costs of (a) agent $A$ and (b) agent $B$. The network costs are defined over activity pairs as (c).*

The agents agree to using a static VCG mechanism (its construction is illustrated in Example E.13), with again the network costs split evenly over both agents. However, as now the agents are planning in a setting that includes uncertainty, they can no longer simply report their costs and let the mechanism compute an optimal time-slot allocation (plan). Instead, they each report their dynamic type, i.e. their cost model of the current and future decisions as an MDP (as in Example 6.8), that models their costs for every anticipated outcome. The mechanism then employs a stochastic planning algorithm as its outcome rule that computes the optimal *policy* given the dynamic type reports. Finally, the payments of this new mechanism charge each agent the harm in *expected* utility it brings upon others, determined by the difference in expected policy values with and without the agent.




(a) (b)

**Figure 6.3** *The optimal policies of the static mechanism (a) with both agents participating and (b) for each agent without the presence of the other. Probabilities are annotated in red and transition rewards are shown in blue as a label of the result state.*

First, consider the scenario in which both agents report their complete, dynamic type MDPs $M_A$ and $M_B$ truthfully in joint dynamic type report $\hat{\boldsymbol{\theta}}$. The mechanism computes

three policies: the optimal policy $\pi^*$ for the joint type MDP $M_A \times M_B$ in which both agents participate, the optimal policy $\pi^*_{-B}$ for $M_A$ when $B$ would not have participated and the optimal policy $\pi^*_{-A}$ for $M_B$ without $A$'s presence. These are all shown as a state-action diagrams in Figure 6.3.

The payments in the static mechanism are computed as the difference between expected policy values that an agent has when the other agent is and is not participating. For agent $A$, this results in a payment:

$$\begin{aligned}
p_A(\hat{\boldsymbol{\theta}}) = \; & v_{-A}(\hat{\boldsymbol{\theta}}, \pi^*) - v_{-A}(\hat{\boldsymbol{\theta}}, \pi^*_{-A}) \\
= \; & \Big((c_B(b_2, \mathtt{t1}) + \frac{1}{2}\ell(\mathtt{a}_1, \mathtt{b}_2)) + Pr(o^+_{\mathtt{a}_1}) c_B(\mathtt{b}_1, \mathtt{t2}) + Pr(o^-_{\mathtt{a}_1}) c_B(\mathtt{b}_1, \mathtt{t3})\Big) \\
& - \Big(c_B(\mathtt{b}_2, \mathtt{t1}) + c_B(\mathtt{b}_1, \mathtt{t2})\Big) \\
= \; & \Big((-5 + -\frac{1}{2}\ell(\mathtt{a}_1, \mathtt{b}_2)) + 0.9(-2) + 0.1(-3)\Big) - \Big(-5 + -2\Big) \\
= \; & \Big((-5 + -\frac{1}{2} \times 5) + 0.9(-2) + 0.1(-3)\Big) - \Big(-5 + -2\Big) \\
= \; & (-7.5 - 1.8 - 0.3) - (-7) = -2.6
\end{aligned}$$

and, analogously, the payment for agent $B$ can be determined as

$$\begin{aligned}
p_B(\hat{\boldsymbol{\theta}}) = \; & v_{-B}(\hat{\boldsymbol{\theta}}, \pi^*) - v_{-B}(\hat{\boldsymbol{\theta}}, \pi^*_{-B}) \\
= \; & \Big((-6 + -\frac{1}{2}\ell(\mathtt{a}_1, \mathtt{b}_2) + 0.9(-3) + 0.1(-11 + -7)\Big) \\
& - \Big((-6) + 0.9(-3) + 0.1(-11 + -3)\Big) \\
= \; & (-8.5 - 2.7 - 1.8) - (-6 - 2.7 - 1.4) = -2.9
\end{aligned}$$

Thus, both agents pay half of the network costs ($-2.5$) that is caused in the first time step and each agent additionally pays for future decisions they influence negatively. In the case of agent $A$, when his activity $\mathtt{a}_1$ delays, $B$ will start $b_1$ not at time $\mathtt{t2}$ but $\mathtt{t3}$ and its utility is decreased by $1$, with a probability of $0.1$, leading to total payment $-2.5 + 0.1 \times -1 = -2.6$. Vice versa, the presence of $B$ causes $A$ plan $\mathtt{a}_2$ at $\mathtt{t4}$ instead of $\mathtt{t3}$ (if $\mathtt{a}_1$ was previously delayed), decreasing $A$'s utility from $-3$ to $-7$ and thus $B$ pays $-2.5 + 0.1 \times -4 = -2.9$. So far, the VCG mechanism seems to work out as well as in the static mechanism example; nevertheless, it is easy to demonstrate that this mechanism is not strategyproof.

Assuming once again that agent $A$ has received information that $B$ will always report truthfully, it can manipulate its utility in its own favour. Observe that the compensation for the expected harm is discounted by the probability of such harm occurring. In other words, when the probability of harming another agent in the future is relatively small, the payment is also relatively low. This offers an avenue for strategic reporting. Given that $B$ will report the same, truthful dynamic type MDP $M_B$, agent $A$ now reports dynamic type MDP $\hat{M}_A$ that is the same as $M_A$ but for the cost of activity $\mathtt{a}_2$ at time $\mathtt{t4}$. Instead of reporting 7, it now reports $c_A(a_2, \mathtt{t4}) = 40$ and because of the new joint report $\bar{\boldsymbol{\theta}} = \langle M_A, M_B \rangle$, the optimal policy changes such that when $\mathtt{a}_1$ is delayed, now first $\mathtt{a}_2$ and then $\mathtt{b}_1$ is executed. This results in a *reported* joint cost of $-28$, as opposed to the $-43$ if $\mathtt{a}_2$ would have been second. In particular, this increases the *real* utility of agent $A$ by $4$, as $a_2$ is planned for $-3$ instead of $-7$ whereas agent $B$ loses $25$. Both single-agent policies, i.e. without the other agent, remain the same.

Under this joint report, the payment of agent $A$ becomes $p_A(\bar{\boldsymbol{\theta}}) = (-7.5 + 0.9(-2) + 0.1(-28)) - (-7) = -5.1$ (where most of the terms equal the previous payment computation, except for the $-28$). Now although its payment has increased by $2.5$, its costs have decreased by $4$ and therefore, by lying, agent $A$ can increase its overall utility (although this only happens when $\mathtt{a}_1$ is delayed). As a corollary, the static mechanism is not strategyproof in the dynamic setting.

Using the dynamic maintenance mechanism of Definition 6.9, the dynamic setting can be coordinated in a strategyproof way. To illustrate how this mechanism does achieve strategyproofness, the same reports are considered and the payments of both agents are computed. First the scenario where $A$ reports truthfully in joint report $\hat{\boldsymbol{\theta}}$ (and thus the optimal policy is again $\pi^*$). In the first time step, the optimal policy for both agents is then again $\pi^*$ and the individually optimal policies are $\pi^*_{-B}$ and $\pi^*_{-A}$ for agent $A$ and $B$ respectively. This leads to payments

$$\begin{aligned} p_A^{\mathtt{t1}}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*) &= v_{-A}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*(\hat{\boldsymbol{\theta}}^{\mathtt{t1}})) + \tilde{V}_{-A}^{\pi^*_{-A}}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}}, \vec{\sigma}^{\,*}) - \tilde{V}_{-A}^{\pi^*_{-A}}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}_{-A}, \vec{\sigma}^{\,*}_{-A}) \\ &= \Big(-5 + \frac{1}{2} \times -5\Big) + \Big(0.9(-2) + 0.1(-3)\Big) - \Big(-5 + -2\Big) = -2.6 \end{aligned}$$

and

$$\begin{aligned} p_B^{\mathtt{t1}}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*) &= v_{-B}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*(\hat{\boldsymbol{\theta}}^{\mathtt{t1}})) + \tilde{V}_{-B}^{\pi^*_{-B}}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}}, \vec{\sigma}^{\,*}) - \tilde{V}_{-B}^{\pi^*_{-B}}(\hat{\boldsymbol{\theta}}^{\mathtt{t1}}_{-B}, \vec{\sigma}^{\,*}_{-B}) \\ &= \Big(-6 + \frac{1}{2} \times -5\Big) + \Big(0.9(-3) + 0.1(-11 + -7)\Big) \\ &\quad - \Big(-6 + (0.9(-3) + 0.1(-11 + -7))\Big) = -2.5 \end{aligned}$$

in which $\hat{\boldsymbol{\theta}}^2$ is the dynamic type that results after executing optimal decision $\pi^*(\hat{\boldsymbol{\theta}}^1)$ at time $1$ (in this case executing $\mathtt{a}_1$ and $\mathtt{b}_2$). Notice that in the static mechanism the payments were $p_A = -2.6$ and $p_B = -2.9$; in the dynamic setting, agent $A$'s payment remains the same but $B$ pays only the network cost caused by its *current decision* to execute $\mathtt{b}_2$. Any potential impact by future decisions, such as postponing $\mathtt{a}_2$ in favour of $\mathtt{b}_1$ when $\mathtt{a}_1$ delays, will be accounted for in later payments. To see this, all payments are computed for both possible outcomes of $\mathtt{a}_1$. When $\mathtt{a}_1$ does not delay the payments at time $\mathtt{t2}+$ ($+$ to indicate $\mathtt{a}_1$ did not delay) are given by

$$\begin{aligned} p_A^{\mathtt{t2}+}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*) &= v_{-A}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+})) + \tilde{V}_{-A}^{\pi^*_{-A,\mathtt{t2}+}}(\hat{\boldsymbol{\theta}}^{\mathtt{t3}+}, \vec{\sigma}^{\,*}) - \tilde{V}_{-A}^{\pi^*_{-A,\mathtt{t2}+}}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}_{-A}, \vec{\sigma}^{\,*}_{-A}) \\ &= (-2) + (0) - (-2) = 0 \\ p_B^{\mathtt{t2}+}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*) &= v_{-B}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+})) + \tilde{V}_{-B}^{\pi^*_{-B,\mathtt{t2}+}}(\hat{\boldsymbol{\theta}}^{\mathtt{t3}+}, \vec{\sigma}^{\,*}) - \tilde{V}_{-B}^{\pi^*_{-B,\mathtt{t2}+}}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}_{-B}, \vec{\sigma}^{\,*}_{-B}) \\ &= (0) + (-3) - (-3) = 0 \end{aligned}$$

such that $\pi^*_{-A,\mathtt{t2}+}$ and $\pi^*_{-B,\mathtt{t2}+}$ are the optimal policies without the other agent from time $2$ onwards when $a_1$ did not delay (i.e. its outcome was $o^+_{\mathtt{a}_1}$) and like before $\hat{\boldsymbol{\theta}}^{\mathtt{t3}+}$ is the dynamic type that results from executing the decisions of $\pi^*$ for $\hat{\boldsymbol{\theta}}^{\mathtt{t2}+}$. Note that the payments are both equal to zero because there is no network cost interaction and neither of the agents forces the other into planning its activity at a less favourable time. Put otherwise, the decisions in $\pi^*$ match those of $\pi^*_{-A,\mathtt{t2}+}$ and $\pi^*_{-B,\mathtt{t2}+}$. This is also the case for $\mathtt{t3}+$ and $\mathtt{t4}+$, hence now $\mathtt{a}_1$ is considered delayed (omitting formulas):

$$\begin{aligned} p_A^{\mathtt{t2}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}-}, \pi^*) &= (0) + (-3) - (-3) = 0 \\ p_B^{\mathtt{t2}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t2}-}, \pi^*) &= (-11) + (-3) - (-11 + -3) = 0 \end{aligned}$$

The last term in the payment of agent $A$ may seem incorrect, because without agent $A$'s presence from time t2, intuitively agent $B$ could plan $\mathtt{b}_1$ at time t2 instead of t3 and increase its utility by 1. However, although policy $\pi^*_{-A,\mathtt{t2}-}$ does not consider current and future decisions of $A$, past decisions have to be accounted for (and can be derived from joint dynamic type $\hat{\boldsymbol{\theta}}^{\mathtt{t2}-}$). Because of $A$'s decision at time t1 and the delayed outcome, $\mathtt{a}_1$ is still being performed at time t2 and this will lead to a network cost of $\frac{1}{2} \times 5$ if agent $B$ plans $\mathtt{b}_1$ concurrently. For the next time steps the payments are:

$$p_A^{\mathtt{t3}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t3}-}, \pi^*) = (-3) + (0) - (-3) = 0 \qquad p_A^{\mathtt{t4}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t4}-}, \pi^*) = (-7) + (0) - (-7) = 0$$
$$p_B^{\mathtt{t3}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t3}-}, \pi^*) = (0) + (-7) - (-3) = -4 \qquad p_B^{\mathtt{t4}-}(\hat{\boldsymbol{\theta}}^{\mathtt{t4}-}, \pi^*) = (0) + (0) - (0) = 0$$

When both agents are truthful and $a_1$ is delayed, activity $\mathtt{a}_2$ is executed at time t4 instead of t3 and agent $B$ is charged the utility loss $(-4)$ of agent $A$ through payment $p_B^{\mathtt{t3}-}$ at time t3. Recall that the utility loss that $B$ suffers as a result of activity $\mathtt{a}_1$ delaying was already accounted for at time t1. Now the scenario in which agent $A$ lies about its costs – as before reporting $c_A(\mathtt{a}_2, \mathtt{t4}) = 40$ instead of 7 in joint dynamic type report $\bar{\boldsymbol{\theta}}$ – is analysed. The payments are computed for both outcomes of $\mathtt{a}_1$, first at time t1:

$$p_A^{\mathtt{t1}}(\bar{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*) = (-5 + 2.5) + (0.9(-2) + 0.1(-3)) - (-5 + -2) = -2.6$$
$$p_B^{\mathtt{t1}}(\bar{\boldsymbol{\theta}}^{\mathtt{t1}}, \pi^*) = (-6 + 2.5) + (-3) - (-3) = -2.5$$

and then for both outcomes of $\mathtt{a}_1$:

| $\mathtt{a}_1$ **is not delayed** | $\mathtt{a}_1$ **is delayed** |
|---|---|
| $p_A^{\mathtt{t2}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*) = (0) + (-3) - (-3) = 0$ | $p_A^{\mathtt{t2}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t2}-}, \pi^*) = (0) + (-3) - (-3) = 0$ |
| $p_B^{\mathtt{t2}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t2}+}, \pi^*) = (0) + (-3) - (-3) = 0$ | $p_B^{\mathtt{t2}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t2}-}, \pi^*) = (-11) + (-3) - (-11 + -3) = 0$ |
| $p_A^{\mathtt{t3}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t3}+}, \pi^*) = (0) + (0) - (0) = 0$ | $p_A^{\mathtt{t3}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t3}-}, \pi^*) = (0) + (-25) - (-3) = -22$ |
| $p_B^{\mathtt{t3}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t3}+}, \pi^*) = (-3) + (0) - (-3) = 0$ | $p_B^{\mathtt{t3}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t3}-}, \pi^*) = (-3) + (0) - (-3) = 0$ |
| $p_A^{\mathtt{t4}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t4}+}, \pi^*) = (0) + (0) - (0) = 0$ | $p_A^{\mathtt{t4}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t4}-}, \pi^*) = (-25) + (0) - (-25) = 0$ |
| $p_B^{\mathtt{t4}+}(\bar{\boldsymbol{\theta}}^{\mathtt{t4}+}, \pi^*) = (0) + (0) - (0) = 0$ | $p_B^{\mathtt{t4}-}(\bar{\boldsymbol{\theta}}^{\mathtt{t4}-}, \pi^*) = (0) + (0) - (0) = 0$ |

As a result of lying, agent $A$'s total payment increases by $-22$ when activity $\mathtt{a}_1$ is delayed, whereas it only decreases its maintenance costs by 4. Therefore, agent $A$ decreases its utility by misreporting under the dynamic maintenance mechanism.

## 6.2 Selfish Best-response Maintenance Planning

The previous section defined an optimal method to coordinate maintenance planning with self-interested agents. The downside of this method, beside the need to disclose private information and submit autonomy to a centre, is that in order to guarantee incentive compatibility, the sequential social choice function must find optimal contingent maintenance plans. As discussed in Chapter 3, finding these poses a complex computational problem. Moreover, to determine the payment for every agent $i \in \boldsymbol{N}$, the mechanism must compute the policy $\pi^*_{-i}$ that would have been optimal without agent $i$'s presence in *every round of the mechanism*. As a consequence, the computa-

tional effort required to implement such a mechanism may be prohibitively large. This section introduces a best-response coordination method that does not provide the welfare optimality of dynamic VCG but is still robust against strategic behaviour, without agents having to share their knowledge, and may be more feasible when dealing with larger problem sizes.

The main bottleneck of a dynamic VCG mechanism, such as the dynamic maintenance mechanism of the previous section, is that there is one central point of computation that must always determine efficient allocations, i.e. exact solutions. One could try weakening the first condition of Definition 6.6 by considering approximate sequential social choice functions, however none of the guarantees that dynamic VCG provides – incentive compatibility, individual rationality and weak budget balance – transfer (directly) when using sub-optimal allocations. A mechanism that uses dynamic VCG with an inefficient allocation scheme is hence vulnerable to strategic play and potentially requires substantial amounts of external funding, rendering it useless for most practical applications.[48] Instead of pursuing the direction of approximate dynamic mechanisms, here the focus is on 'decentralising' the computational effort while remaining invulnerable to strategic behaviour. Instead of computing an optimal coordination strategy centrally, the agents develop and iteratively improve a policy for their local planning problem given the policies submitted by the other agents in a best-response fashion. Thus in every round each agent gets a turn to respond to the plans of all others.

**Algorithm 6.12** `Best-response Maintenance Planning`

**Require:** MPP instance $M$, utility functions $\{u_i\}_{i \in \boldsymbol{N}}$, number of iterations $X$

$\boldsymbol{\pi}^0 \leftarrow$ initial (random) joint policy for $M$
$k \leftarrow 0$
**repeat**
  $k \leftarrow k + 1$
  $\boldsymbol{\pi}^k \leftarrow \boldsymbol{\pi}^{k-1}$ ▷ Start with previous joint policy
  $\tilde{N} \leftarrow \texttt{OrderAgents}(\boldsymbol{N})$ ▷ Determine order of agents
  **for** $i \in \tilde{N}$ **do**
    $\pi_i' \leftarrow \arg\max_{\pi_i \in \Pi_i} u_i(\left\langle \pi_i, \boldsymbol{\pi}^k_{-i} \right\rangle)$ ▷ Find best-response to $\boldsymbol{\pi}^k_{-i}$
    $\boldsymbol{\pi}^k \leftarrow \left\langle \pi_i', \boldsymbol{\pi}^k_{-i} \right\rangle$
  **end for**
**until** $\boldsymbol{\pi}^k = \boldsymbol{\pi}^{k-1}$ or $k \geq X$
**return** $\boldsymbol{\pi}^k$

The basic outline of the best-response algorithm is presented in Algorithm 6.12. Initially, every agent submits a policy that is optimal for its local planning, resulting in a joint policy $\boldsymbol{\pi}^0$ that neglects all reward interactions between agents. But, as the policies of all other agents become known, every agent can react to these in a best-response fashion: an agent will change its policy when this yields a higher valuation for the agent given the joint policy $\boldsymbol{\pi}_{-i}$ of all other agents. As a result, the overall

[48] *Although examples of successful approximation can be found in the works by, amongst others, Briest et al. [46], Hartline [111] and Mu'Alem and Nisan [180].*

expected utility of the joint policy (and for each agent individually) is likely to improve in every iteration, until a local optimum is attained or $X$ iterations have passed. In general, this procedure is not guaranteed to end up in a local optimum, but for a specific class of games known as *potential games* an iterative best-response method will always result in a pure-strategy Nash equilibrium in a finite number of steps (Definition E.2) [177, 219, 256].[49] A potential game is defined as:

**Definition 6.13 Potential Game**

A game $\mathcal{G}$ (Definition E.1) is a potential game if there exists a potential function $\Phi$: $\boldsymbol{\Lambda} \mapsto \mathbb{R}$ such that when an agent $i \in \boldsymbol{N}$ changes its action from $a_i$ to $a'_i$, its change in utility must equal the change in potential for both outcomes $g(\langle a_i, \vec{a}_{-i}\rangle)$ and $g(\langle a'_i, \vec{a}_{-i}\rangle)$:

$$\Phi(g(\langle a_i, \vec{a}_{-i}\rangle)) - \Phi(g(\langle a'_i, \vec{a}_{-i}\rangle)) = u_i(g(\langle a_i, \vec{a}_{-i}\rangle)) - u_i(g(\langle a_i, \vec{a}_{-i}\rangle)) \tag{6.10}$$

For the maintenance planning problem it is possible to construct a *congestion game* [219] that belongs to a sub-class of potential games.[50] More specifically, the model that is most suitable for maintenance planning is that of the *congestion planning game*, proposed by Jonsson and Rovatsos [127]. In congestion games, agents have to select resources to use at a cost relative to the number of agents using it:

**Definition 6.14 Congestion Game**

A game $\mathcal{G}$ is a congestion game if there exists a set resources $E = \{e_1, e_2, \ldots, e_m\}$ and a monotonic, increasing resource cost function $c_k : \mathbb{Z} \mapsto \mathbb{R}$ for every resource $e_k \in E$ that depends on the number of agents assigned to resource $e_k$ in outcome $o$ given by $\#(e_k, o)$, such that every outcome $o \in O$ is mapping of agents to resources and the utility of every agent $i \in \boldsymbol{N}$ is given by $u_i(o) = -\sum_{e_k \in o_i} c_k(\#(e_k, o))$. Here, $o_i$ represents the resource assignment of agent $i$ in outcome $o$.

Rosenthal [219] showed that every congestion game is a potential game with a potential function that accumulates (negated) resource costs, or

$$\Phi(o) = -\sum_{e_k \in E} \sum_{j=1}^{\#(e_k,o)} c_k(j) \tag{6.11}$$

and a Nash-equilibrium is guaranteed for this type of games if $c$ is non-decreasing. Jonsson and Rovatsos [127] extend the congestion game to a congestion planning game by the addition of a local value function $v_i : O \mapsto \mathbb{R}$ for every agent $i \in \boldsymbol{N}$ that

[49] *Note that the ordering in which agents find their best-response policies does matter as to which of the local optima is obtained, hence the* `OrderAgents` *function in line 6 of Algorithm 6.12.*

[50] *See e.g. the work by Roughgarden [221] for a detailed study of congestion games.*

depends only on the assignment $o_i$ of that agent such that its utility becomes $u_i(o) = v_i(o_i) - \sum_{e_k \in o_i} c_k(\#(e_k, o))$. In this type of games the potential function is given by

$$\Phi(o) = \sum_{i \in \boldsymbol{N}} v_i(o) - \sum_{e_k \in E} \sum_{j=1}^{\#(e_k,o)} c_k(j) \tag{6.12}$$

and it can be shown that this potential function also guarantees the existence of at least one Nash-equilibrium [127].

Following the construction of Jonsson and Rovatsos [127], a potential game can be constructed for MPP when an anonymous, non-decreasing model is used for the network cost, such as the factor-based model of Example 3.5. Here, anonymous means that the network cost only depends on the number of agents concurrently active on a resource and not on the specific set of agents. The non-decreasing property is satisfied when the network cost of performing concurrent maintenance can never decrease when the number of agents working concurrently increases. This is trivially true for maintenance planning: more concurrent maintenance on the network can only lead to the same or more network costs.

The basic setup of the maintenance planning congestion game is a set of agents $\boldsymbol{N}$ of size $n$, that each have to play a policy $\pi_i \in \Pi_i$ (i.e. $\Lambda_i = \Pi_i$) that optimises their utility $u_i$ when their type is $\theta_i \in \Theta_i$. In this game, agents always report their true type with their actual maintenance revenues and costs. The outcome rule in this game is a straightforward function that combines the policies submitted by every player into a joint policy $\boldsymbol{\pi} = \times_{i \in n} \pi_i$ and therefore the set of outcomes in this game corresponds to the set of available joint policies, i.e. $O = \boldsymbol{\Pi} = \times_{i \in n} \Pi_i$ (later it is shown how a joint policy induces a resource assignment). As a consequence, the local valuation $v_i$ and utility $u_i$ of agent $i$ can alternatively be expressed as functions of the joint policy $\boldsymbol{\pi}$ and this notation will be convenient later on.

In Chapter 4 it is shown that, without loss of generality, the maintenance rewards can be divided into a collection of local reward functions $\left\{\bar{R}^i\right\}_{i \in \boldsymbol{N}}$ (local revenues minus maintenance costs) and interaction rewards $\bar{\boldsymbol{R}} = \left\{\bar{R}^k \,|\, k \subseteq \boldsymbol{N}, |k| > 1\right\}$ (the network costs). The local value function for each agent $i \in \boldsymbol{N}$ is defined as $v_i(\boldsymbol{\pi}) = \mathbb{E}[\bar{R}^i \,|\, \pi_i] = V^{\pi_i}$, i.e. the expected value of policy $\pi_i$ (Definition 2.1) *without* the network costs.[51] For each interaction reward $\bar{R}^k \in \bar{\boldsymbol{R}}$ an interaction resource $e_k$ is defined with a function

$$c_k(x,t) = \frac{1}{x}\bar{R}^k(x,t) = \frac{1}{x}\ell_k(x,t) \tag{6.13}$$

that captures the cost per agent when $x$ agents are active at resource $e$ at time $t$.[52] This function is assumed to be non-decreasing at all $t$, i.e. $x' \geq x$ implies $c_k(x',t) \geq c_k(x,t)$

[51] *Technically, the valuation function should be expressed over a resource assignment. It is trivial to introduce $|T| \times |\mathcal{A}_i|$ additional 'activity time slot' resources for all agents such that the resource cost for each such a resource $e_k^t$ models the revenue minus maintenance costs when policy $\pi_i$ states that an activity $\mathrm{a}_k \in \mathcal{A}_i$ starts at time $t$.*

[52] *Again, in congestion game model this should be modelled as $|T|$ resources for every interaction reward, each modelling an interaction at time $t$. It is easy to show both models are equivalent.*

or, put in words, the addition of another agent to the resource will always increase the cost for every agent assigned to it. One could imagine an interaction resource as an area in which concurrent maintenance has a super-additive effect on traffic. The network costs for such an area are computed through the (anonymous) function $\ell_k$, indexed by $k$ to indicate its correspondence to interaction reward $\bar{R}^k$, that is non-decreasing in the number of agents. For example, when activities $a$ and $b$ of two distinct agents result in a (super-additive) network cost, there exists an interaction reward $\bar{R}^k(\{a,b\},t)$ in the MPP instance and for this reward function an interaction resource $e_k$ is created with a cost function $c_k(x,t) = \frac{1}{2}\bar{R}^k(x,t)$ based on the number of agents active on this resource at time $t$. This number can be zero ($a$ and $b$ are not performed), one ($a$ or $b$ is performed) or two (both $a$ and $b$).

**Remark 6.15 Non-anonymous Network Costs**

The resource costs must be anonymous with regard to agents in the sense that the increase in network cost does not depend on which agent is assigned. There can however be non-anonymity as to *what activity* each agent performs. Two activities $\mathtt{a}_i$ and $\mathtt{b}_i$ that result in different network costs within the same area when the activities $\mathtt{a}_{-i}$ of the others do not change, can be modelled using two interaction reward functions $\bar{R}^k_1(\langle \mathtt{a}_i, \mathtt{a}_{-i}\rangle)$ and $\bar{R}^k_2(\langle \mathtt{b}_i, \mathtt{a}_{-i}\rangle)$ (with according network costs $\ell_1$ and $\ell_2$) such that $\mathtt{a}_i \in \mathrm{Dom}(\bar{R}^k_1)$, $\mathtt{a}_i \notin \mathrm{Dom}(\bar{R}^k_2)$, $\mathtt{b}_i \in \mathrm{Dom}(\bar{R}^k_2)$ and $\mathtt{b}_i \notin \mathrm{Dom}(\bar{R}^k_1)$. From the agent's policy $\pi_i$ it is derived to which interaction resource an agent is mapped, i.e. $\#(e_1,t,\pi_i) = 1$ if $\pi_i(t) = \mathtt{a}_i$ or $\#(e_2,t,\pi_i) = 1$ if $\pi_i(t) = \mathtt{b}_i$.

In a standard congestion game, the number of agents that concurrently use a resource is computed using the function $\#$. In the stochastic setting, it is only possible to determine the *expected* number of agents on a resource, given a joint policy $\boldsymbol{\pi}$. For a single agent, the probability of it being active at resource $e_k$ at time $t$ can be derived from its policy $\pi_i$ by considering all possible histories $\mathcal{H}^h$ of length $h$:

$$Pr(i \in e_k | \pi_i, t) = \sum_{H^h \in \mathcal{H}^h | \pi_i} Pr(H^h) \cdot \begin{cases} 1, & \text{if } H^h_{\mathcal{A}}(t) = \mathtt{a}_i,\ \mathtt{a}_i \in \mathcal{A}_i \\ & \text{and } \mathtt{a}_i \in \mathrm{Dom}(\bar{R}^k) \\ 0, & \text{otherwise} \end{cases} \tag{6.14}$$

The probability of $j$ agents being concurrently assigned to resource $e_k$ is then given by the sum of probabilities of all $\binom{n}{j}$ (unique) subsets $\boldsymbol{N}^j$ of agents with size $j$ being assigned to it at time $t$, or

$$Pr(|e_k| = j \,|\, \boldsymbol{\pi}, t) = \sum_{\boldsymbol{N}^j \subseteq \boldsymbol{N}} \Big( \prod_{i \in \boldsymbol{N}^j} Pr(i \in e_k | \pi_i, t) \times \prod_{i \in \boldsymbol{N} \setminus \boldsymbol{N}^j} Pr(i \notin e_k | \pi_i, t) \Big) \tag{6.15}$$

Alternatively, this can be written recursively which will be useful later on:

$$Pr(|e_k| = j \,|\, \boldsymbol{\pi}, t) = \begin{cases} 0, & j < 0 \\ 0, & j > |\boldsymbol{\pi}| \\ 1, & j = 0 \\ Pr(i \in e_k | \pi_i, t)\, Pr(|e_k| = j-1 \,|\, \boldsymbol{\pi}_{-i}, t) & \\ \quad + Pr(i \notin e_k | \pi_i, t)\, Pr(|e_k| = j \,|\, \boldsymbol{\pi}_{-i}, t), & \text{otherwise} \end{cases} \tag{6.16}$$

Given the expected occupation of a resource, the corresponding expected cost is computed by summing over all potential occupation levels and the probability thereof:

$$\tilde{c}_k(\boldsymbol{\pi}, t) = \sum_{j=1}^{n} Pr(|e_k| = j \mid \boldsymbol{\pi}, t)\, c_k(j, t) \tag{6.17}$$

As the execution of a joint policy is inherently uncertain, agents in the maintenance congestion planning game optimise their expected utility. For each agent $i \in \boldsymbol{N}$, the utility it obtains when it is potentially assigned to resources $e_k \in E$ (i.e. every $e_k \in E$ for which $Pr(i \in e_k | \pi_i, t) > 0$ for at least one $t \in T$) when the joint policy is $\boldsymbol{\pi}$ is:

$$\begin{aligned} u_i(\boldsymbol{\pi}) &= v_i(\boldsymbol{\pi}) - \sum_{e_k \in E} \sum_{t \in T} \sum_{j=1}^{n} Pr(|e_k| = j, i \in e_k | \boldsymbol{\pi}_i, t)\, c_k(j, t) \\ &= v_i(\pi_i) - \sum_{e_k \in E} \sum_{t \in T} \sum_{j=1}^{n} \big( Pr(|e_k| = j - 1 | \boldsymbol{\pi}_{-i}, t) Pr(i \in e_k | \pi_i, t) \\ &\qquad - Pr(|e_k| = j | \boldsymbol{\pi}_{-i}, t) Pr(i \in e_k | \pi_i, t) \big)\, c_k(j, t) \end{aligned} \tag{6.18}$$

The first term accounts for its expected local valuation, according to Equation 2.1, while the second term counts the expected resource costs – the network costs – for only those resources the agent is assigned to. The terms in the summation of the last equality respectively express the probability that $i$ creates a set of size $j$ minus the probability that a set of size $j$ was already formed by the other agents while $i$ is also assigned to $e_k$. The latter is accounted for in the next summation, when $j = j + 1$, and thus prevents overcounting of costs at set size $j$. Note that there is no term for $Pr(i \notin e_k | \pi_i, t)$ as there is no cost to agent $i$ if it does not make use of the resources.

Finally, the potential function corresponding to this stochastic planning game is defined as

$$\Phi(\boldsymbol{\pi}) = \sum_{i \in \boldsymbol{N}} v_i(\boldsymbol{\pi}) + \sum_{e_k \in E} \sum_{t \in T} \tilde{c}_k(\boldsymbol{\pi}, t) \tag{6.19}$$

such that the potential sums the expected resource costs over all interaction resources.

**Theorem 6.16 Maintenance Congestion Planning Game**

The maintenance congestion planning game is a congestion planning game.

*Proof.* To prove the theorem, it must be shown that the conditions for a congestion planning game still hold:

(i) When two joint policies $\boldsymbol{\pi} = \langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$ and $\boldsymbol{\pi}' = \langle \pi_i', \boldsymbol{\pi}_{-i} \rangle$ differ only in policies $\pi_i$ and $\pi_i'$ for agent $i$, it must be that $\Phi(\boldsymbol{\pi}) - \Phi(\boldsymbol{\pi}') = u_i(\boldsymbol{\pi}) - u_i(\boldsymbol{\pi}')$.

(ii) The cost function $\tilde{c}_k(j, t, \boldsymbol{\pi})$ for every resource $e_k \in E$ is monotonic and increasing in the number of agents.

(iii) The local value function $v_i$ for every agent $i \in \boldsymbol{N}$ is independent from the resource assignment of other agents.

Condition (i) is proven in Appendix A.5 and thus the maintenance congestion planning game is a potential game. In fact, by proving this condition, the existence of Nash equilibria in the maintenance planning game is guaranteed. Additionally, it can be shown that the maintenance planning game is a congestion planning game by proving conditions (ii) and (iii).

The expected resource cost must be monotonic in the *expected* number of agents concurrently on the resource according to the joint policy. The interaction resource costs $c_k$ are defined as Equation 6.13 and by construction monotonic increasing in the number of agents at the resource. Now consider the other term of $\tilde{c}_k$, the probability $Pr(|e_k| = j|\boldsymbol{\pi}, t)$ that $j$ agents are assigned to the resource. Observe that the expected number of agents at resource $e_k$ is given by $\tilde{n}_k(\boldsymbol{\pi}) = \sum_{j=0}^{n} Pr(|e_k| = j|\boldsymbol{\pi}, t) \times j$. To prove condition (ii) it must be shown that when this expected number increases, $\tilde{c}_k$ does also. Given two joint policies $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ such that $\tilde{n}_k(\boldsymbol{\pi}) > \tilde{n}_k(\boldsymbol{\pi}')$ then there must exist at least one $j$ for which $Pr(|e_k| = j|\boldsymbol{\pi}, t) > Pr(|e_k| = j|\boldsymbol{\pi}', t)$ such that

$$\sum_{j'=0}^{j-1} Pr(|e_k| = j'|\boldsymbol{\pi}, t) < \sum_{j'=0}^{j-1} Pr(|e_k| = j'|\boldsymbol{\pi}', t) \quad \text{and}$$
$$\sum_{j''=j+1}^{n} Pr(|e_k| = j''|\boldsymbol{\pi}, t) \geq \sum_{j''=j+1}^{n} Pr(|e_k| = j''|\boldsymbol{\pi}', t)$$

Assume now that the sum of probabilities from $j+1$ to $n$ remains equal and that the increase in probability at $j$, i.e. $Pr(|e_k| = j|\boldsymbol{\pi}, t) - Pr(|e_k| = j|\boldsymbol{\pi}', t)$, is equal to $p$. Then, because $c_k$ is monotonic in $j$, $p \cdot c_k(j, k) > p \cdot c_k(j-1, t)$, which is the maximal resource cost increase due to the probability increasing for any agent set size smaller than $j$. Therefore, the increase in probability $p$ at $j$ results in an overall increase of expected resource cost. When the sum of probabilities from $j+1$ to $n$ increases, a similar argument can be made to show that again the expected resource cost can only increase. The expected resource cost is hence monotonic, increasing in the expected number of agents and, consequentially, condition (ii) is also satisfied. Condition (iii) holds because the local value function is defined as $v_i(\boldsymbol{\pi}) = V^{\pi_i}$, i.e. the expected value of the policy for only agent $i$, that by definition cannot include any inter-agent rewards. □

As a direct result of Theorem 6.16, the existence of a Nash equilibrium is guaranteed. Therefore, any algorithm based on Nash-like improvements, such as the best-response procedure of Algorithm 6.12, is guaranteed to find at least a local optimum of the potential function. Note that the existence of such an equilibrium does not provide any guarantees as to the value thereof. Furthermore, in a Nash equilibrium no agent is better off by *unilaterally* changing its policy. Better solutions might be achieved if agents collectively switch or coordinate policies.

## 6.3 Further Discussion

In this chapter two methods are presented to solve the maintenance planning problem when dealing with self-interested agents, an optimal dynamic mechanism design solu-

tion and an approximate best-response congestion game. The dynamic maintenance mechanism is a dynamic Vickrey-Clarke-Groves mechanism, which is the only type of dynamic mechanism that concurrently achieves within-period, ex-post incentive compatible, individually rational and weakly budget balanced. Nonetheless, there are several caveats when using dynamic VCG mechanisms: it is vulnerable to opportunistic behaviour by groups of agents or when two or more (dummy) agents are controlled by the same actor, the payments/revenues can be arbitrarily high, profits for the participants are minimised and the determination of payments may be – or quickly become – computationally infeasible for some problems. As a consequence, other types of and sub-optimal mechanisms have been proposed that counter some of these weaknesses, dropping or trading off some of dynamic VCG's desiderata.

The most famous counterpart of VCG is the class of d'Aspremont-Gérard-Varet (AGV) mechanisms [15].[53] Alike VCG, mechanisms from the AGV class are efficient, but additionally they are *strictly* budget balanced, made possible by defining the payments based on expected utilities. The trade-off is that AGV mechanisms can only guarantee ex-ante individual rationality, i.e. players *expect* to benefit from participating, and incentive compatibility is implemented in a Bayesian-Nash equilibrium as opposed to the dominant strategy equilibrium. An extension of this mechanism to the dynamic setting is shown by Athey and Segal [16], where in fact a more general class of efficient dynamic mechanisms is proposed which they dub the (balanced) team mechanism. Another interesting avenue for efficiency is that of approximate mechanism design, especially when dealing with complex (sequential) social choice functions and payments as for instance MPP. More on this can found in the works by Briest et al. [46], Hartline [111], Mu'Alem and Nisan [180], Nisan et al. [188], amongst many others.

With regards to the maintenance planning congestion game of Section 6.2, approximation of the policy search does not interfere with the Nash equilibria of the game as long as it is deterministic, for instance using the RDDL encoding of Chapter 5. When policies can be found quickly many iterations of the best-response game can be played, leading to local optima fast. In particular, it would then be possible to include random restarts or planning heuristics to increase the chances of finding a good or even optimal local minimum. Finally, note that although agents may cheat the best-response approach by strategically reporting plans that cause other agents to divert from their best choices, this behaviour can be countered by instilling fines for not following the established joint plan. Because the agent is never sure what the last round will be, he is not guaranteed to be able to resubmit a plan that maximises his own reward at a later time. Consequentially, opportunistic agents either have to adhere to the joint plan, thus performing activities at a sub-optimal time that was strategic to report, or divert from the plan and incur steep penalties. In both cases the agent suffers from its misreporting. Another approach can be to incentivise the sharing of planning information through explicit rewards [192, 244] so that the joint plan converges to a truly optimal outcome for all. Lastly it remains to note that the approach described here resembles the "better-response strategy" of Jordán et al. [128] that was published recently.

[53] *Independent from this work, Arrow [14] proposed a similar mechanism class.*

# Chapter 7

# The Game of Maintenance Planning

The algorithms and techniques of the previous chapters combined provide a solid theoretical basis for the design of incentives and the coordination of agents in self-regulating planning problems such as the MAINTENANCE PLANNING PROBLEM. Here, a first step is taken into the direction of bringing these techniques into real-world contracts. This chapter investigates the effectiveness of incentives in the group setting and the potential of self-regulation as a viable alternative to complete regulation when confronted with human decision-makers. Recall from Section 1.1 that performance-based approaches such as self-regulation offer many promising benefits, e.g. increased flexibility, preservation of autonomy and authority, stimulation of performance of each individual participant as well as the group, less demand on governmental resources and, consequentially, makes better use of public funding [7, 47, 75, 124, 141, 233]. Before these benefits can be realised, however, a number of challenges must be overcome. Inherently, self-regulation implies that the coordination amongst individual parties becomes more complex and uncertain. This is particularly noticeable in the form of increased potential for opportunistic, self-interested behaviour and possible misalignment of (societal) objectives that may lead to sub-optimal performance or failures.

For bilateral performance-based contracts, i.e. a partnership between the asset manager and a single service provider, a lot of research has been performed on (the success of) incentives and practical examples thereof [43, 62, 268]. The management of a *network* (see Remark 7.1 below) through performance-based incentives, however, has not received as much attention and is still paired with great uncertainties. Of particular interest is the use of incentives to instill self-regulation. This approach is actively being pursued by, amongst others, Bresnen and Marshall [45], Rose and Manley [218], Love et al. [163], Volker et al. [254] and Hosseinian and Carmichael [118]. The design of monetary incentives is considered key to the success of these approaches. Therefore, of at least equal importance is the effectiveness of said incentives to influence the decision-making process of contractors. A perfectly designed payment mechanism is rendered useless if contractors are not stimulated by these incentives in their planning,

coordination and execution. Furthermore, many authors state the paramount role of social relationships in group performance-based contracts and argue that the social dimension may be more effective in maximising performance and inciting self-regulation of a network [41, 47, 140, 218]. Although the use of incentives and requirements for their design have received much attention in literature, the effectiveness of such incentives to actually influence decision making has not been widely studied in the context of networks, let alone seen any empirical evaluation. This is surprising since on the one hand many authors argue the importance and benefits of using performance-based innovative contract forms in network tenders while others report limitations on the effectiveness of incentives in said methods.

**Remark 7.1 (Public) Network Management, Networks and Network Members**

At first the use of the terms network and network management may seem confusing in this chapter but this terminology is widely recognised in literature. The field of network management studies the composition, governance and interactions of inter-firm teams, commonly in the context of multi-agent service delivery or group contracts. An inter-firm group is referred to as a network and its members are hence the network members. Public network management (PNM) is the more specific branch that applies if the network is "led or managed by government representatives" Agranoff and McGuire [7]. It does not mean the management of an actual infrastructural or IT network, although these can be the reason for the network to exist.

This chapter addresses this uncertainty with respect to the effectiveness of monetary incentives in network tenders and the role of social cohesion of the network. In particular, it investigates the emergence of coordination in a setting that is designed to reward contractors for coordinating their interactions or, in other words, the manifestation of self-regulation as a result of monetary incentives. This setting is created through *serious gaming*, a method to perform research in a controlled simulated environment. This approach enables research into changes in behaviour or decision making without the risks and potential costly failures that may occur in real-world testing [49, 169]. Using such a serious game, it is investigated whether contractors change their decision-making behaviour in the presence of network incentives and that this leads to intrinsic coordination of the network. Furthermore, as several articles report on the importance of relationships [45, 218] and their impact on contract outcomes [140], the social cohesion of the network is investigated as a potential moderator for self-regulation. To this end, the following research questions are considered in this chapter:

$RQ_1$ Are monetary incentives effective in influencing the decision making of contractors in the network setting?

$RQ_2$ Does the use of monetary incentives that reward coordination of the network lead to self-regulation?

$RQ_3$ Does the relationship between network members influence the effectiveness of the incentives and, indirectly, the manifestation of self-regulation?

**Contributions** This chapter contributes a serious game to study the application of monetary incentives in a controlled setting with human decision makers. In particular, the serious game is used to evaluate the potential of monetary incentives to incite self-regulation. The results of the gaming experiment provide initial evidence that the use of incentives to steer decision making also applies in the team setting with human-decision makers, but also show that monetary incentives may lead to unintended consequences. The findings revealed that monetary incentives based on traffic penalties introduces competition in the network which hinders coordination. Only in networks with strong social cohesion this adversarial setting can be overcome and emergent coordination arises as intended. This suggests that theoretically-engineered incentives may lead to unexpected and undesired outcomes in practice and that the social dimension is key to the successful implementation of self-regulation. Moreover, the chapter additionally presents the design and implementation of the serious game itself that can be used as a 'sandbox environment' in future and related research. The contributions of this chapter are presented by Scharpff et al. [232] and a complete description of the serious game and all data that was collected is given by Scharpff et al. [231]. The source code and data from gaming sessions can be retrieved from the repository at `https://github.com/AlgTUDelft/road-maintenance-game`.

The contributions of this chapter are currently under review for publication and this chapter is at large a duplication of that article with some alterations to better fit the style and terminology of the thesis. Most noticeably, the introduction, discussion and conclusion sections of the paper have been condensed as they are covered in other parts of this thesis, and sections have been retitled and refactored in accordance with the format of the thesis. Finally, references to other parts of this work have been integrated into the body text while explanations of game theory, MPP and other already established concepts have been removed.

## 7.1 The Road Maintenance Game

This chapter addresses the uncertainty in network management literature with respect to the effectiveness of monetary incentives to influence human decision making and incite self-regulation, and the impact of relationships on this effectiveness, by addressing the three aforementioned research questions. While ideally these questions are answered through real-world experiments, this introduces additional and complex variables into the decision making and the cost of failure in practical tenders prohibits such an approach without prior evidence. Therefore the concept of self-regulation though incentives is first validated in a controlled, agent-based simulation in the form of a serious game. While serious games are primarily known as a tool for instruction, they can also be employed as a research method comparable to simulation or experimentation [49]. Serious games have the potential to integrate a multiplicity of elements including motivation, expertise and social structures, and can be considered a means to explore, explain, and assess the complex interactions between ecosystems and human actions [18]. This research methodology is also known as (agent-based) participatory simulation [107] and can serve to enhance our scientific understanding or to recommend corrective policy action, as in the studies of e.g. Le Bars and Le Grusse [23], Altami-

rano et al. [9] and Meijer et al. [169]. Although the closed-system of a game does not allow direct generalisation to real-world tenders, it does enable a controlled evaluation of changes in the decision-making process and allows reflection on the concept of implementing self-regulation through incentives.

This section describes three parts of the serious gaming setup. First, it is detailed what measurements are performed in the experiment, both from the game as well as through a preliminary questionnaire. Thereafter hypotheses are formed with respect to the measurements that serve to investigate the research questions from experimental data. Finally, the section concludes with an outline of the gameplay to illustrate the game process. The game design and its measurements are summarised here, for a more complete explanation on the game and its model see the on line appendix [231] or Appendix B for a detailed description on the measurements.

### 7.1.1 Operationalising Decision-making, Coordination and Cohesion

The decision preferences of players are expressed in terms of the three distinct decision parameters or objectives of the problem domain, namely *profit*, *traffic time lost (ttl)* and *risk aversion*. A preference for profit reveals itself in predominantly making choices that lead to a higher profit to the player. A player with a ttl preference is more likely to strive for minimisation of expected impact on the traffic hindrance. Players that are risk-averse tend to favour approaches with low delay probability, preferring more robust planning with low variance in both the profits and ttl.

The decision preferences of players before playing the game are captured in their *profile score*. This score is measured through a questionnaire (Figure 7.1) that participants submit 2 weeks prior to the game. This questionnaire consists of 8 increasingly complex game situations and participants are asked to rank the options according to their preferences. From this the profile scores are computed using the methodology from Triantaphyllou [248]. First the questionnaire is modelled as a multi-criteria decision making in which participants rank their preference over alternatives for every question to determine the relative weights of alternatives. Then, from the responses that specify rankings for the alternatives, a preference score per objective is computed using the weighted-sum method from Roszkowska [220]. Finally, the preference scores are normalised to obtain the relative importance of each objective. That is, if $\hat{Q}_p(\mathbf{X})$ expresses the preference score for the objective profit (hence the subscript '$p$') computed from questionnaire responses $\mathbf{X}$, the profile score for the profit parameter is given by

$$Q_p(\mathbf{X}) = \frac{\hat{Q}_p(\mathbf{X})}{\hat{Q}_p(\mathbf{X}) + \hat{Q}_t(\mathbf{X}) + \hat{Q}_r(\mathbf{X})} \tag{7.1}$$

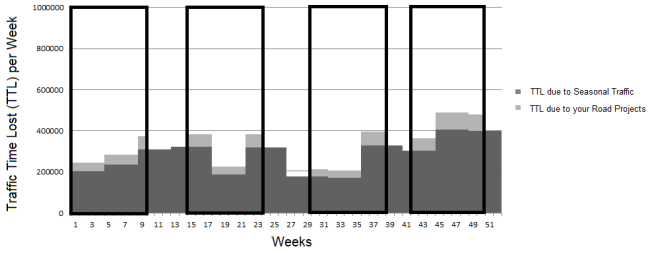
Similarly, the profile scores $Q_t(\mathbf{X})$ and $Q_r(\mathbf{X})$ express the relative preference for respectively ttl (subscript '$t$') and risk aversion (subscript '$r$'). Finally, the rationality of these responses is determined by comparing them to the Pareto optimal trade-off [223], which serves as a measured for how well participants are able to make trade-offs.

**Question 5**

You have chosen four potential periods in which you can perform your project. Using quarterly figures, you determine the following prospects regarding four possible maintenance periods:

| Situation | Risk of delay | | Period 1 | Period 2 | Period 3 | Period 4 |
|---|---|---|---|---|---|---|
| **Project as planned** | 67% | *TTL* | 370.000 | 416.000 | 333.000 | 615.000 |
| | | *Profit* | € 1.369.000 | € 1.323.000 | € 1.406.000 | € 1.124.000 |
| **Project is delayed** | 33% | *TTL* | 503.000 | 571.000 | 493.000 | 809.000 |
| | | *Profit* | € 1.236.000 | € 1.168.000 | € 1.246.000 | € 930.000 |

In addition, you also possess information regarding the TTL figures of the previous year.




a) Can you specify the order of periods in which you prefer to perform the maintenance? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Period 1 | |
| Period 2 | |
| Period 3 | |
| Period 4 | |

b) Please motivate your ranking.

**Figure 7.1** *Example problem from the questionnaire.*

The strategy scores are determined from the actions that are played in the game, roughly similar to the multi-criteria method used on the questionnaire. Each of the actions of the game have been designed for a specific goal and are assigned weights accordingly. Then the alternative weights of the actions performed are summed to form a preference score which in turn is normalised to express relative importance of the objective as in Equation 7.1. For example, the 'low hindrance' action will have only a minor impact on ttl and have a high ttl strategy score $G_t$. On the other hand, its costs are relatively high compared to the other available actions and thus the profit strategy score $G_p$ for this action will be low. Given a maintenance plan plan $\mathbf{Y}$ that describes the assignment of chosen maintenance methods to time slots, the profit strategy score for that agent is denoted by $G_p(\mathbf{Y})$, using the same subscripts as before.

The outcomes of the game are measured in terms of the profit and ttl that are to be expected when executing the (current) plan $\mathbf{Y}$, accounting for the risk of potential

delays using the standard approach of probability times utility. The functions $P(\mathbf{Y})$ and $T(\mathbf{Y})$ express respectively the total expected profit and total expected traffic time lost, given joint plan $\mathbf{Y}$. Additionally, the functions $P_{wc}(\mathbf{Y})$ and $T_{wc}(\mathbf{Y})$ represent respectively the profits and ttl in the worst case, e.g. when all maintenance operations delay. Note that the ttl depends on the joint plan of all players, i.e. all concurrent maintenance within the infrastructure. The utility to a player is defined as the sum of its profits from maintenance work minus its costs due to traffic penalties (1 euro for every hour of ttl in this game model). Thus the sum of utilities $u(\mathbf{Y})$ for a plan $\mathbf{Y}$ is given by $P(\mathbf{Y}) - T(\mathbf{Y})$, and similar for the worst case. Finally, an outcome of the game can be thought of as 'better' if either the total profit increases, the total ttl decreases, or both. Therefore the *performance ratio* $\phi$ of a joint plan $\mathbf{Y}$ is computed by the formula $\phi(\mathbf{Y}) = P(\mathbf{Y})/\ell(\mathbf{Y})$. For clarity of presentation, the questionnaire response $\mathbf{X}$ or joint plan $\mathbf{Y}$ are assumed implicit. For instance $Q_t(\mathbf{X})$ and $P(\mathbf{Y})$ are written as $Q_t$ and $P$.

| | **Description** |
|---|---|
| **Coordination level** | |
| Low | Conflict-driven coordination of interactions via bilateral or trilateral negotiations |
| Medium | Coordination of network via democratic, plenary negotiations |
| High | Centralised planning that governs network decisions |
| **Cohesion level** | |
| Unfamiliar | Players have (had) limited to no interaction previously |
| Familiar | Players see and/or work with each other on a regular basis |

**Table 7.1** *Definition of the qualitative categories for the coordination and cohesion level variables and their descriptions.*

In addition to the quantitative information that is measured from the questionnaire and the game on decision preferences and the outcomes of the game, two qualitative parameters are considered in this study. These are the *coordination level* that is showcased by the network and the *social cohesion* of network members. Both parameter are established by classification rules, shown in Table 7.1. The coordination level of a network is established trough observing the communications and interactions that take place between the players outside of the game. Depending on the types of interaction displayed by the players, the game session is classified into one of the categories. For example, a session where players only resolve planning conflicts in pairs or triples would be categorised as 'Low'. If in addition plenary negotiations are used (at least once) it would be considered 'Medium'. The designation 'High' is given to game sessions in which central regulation takes place. The social cohesion of the group is a qualification of the relation between participants outside of the game, established through inquiry at the start. Because a classification of social cohesion is hard to capture exactly, two clearly disjoint categories are used. If players in a gaming session have never or rarely interacted with each other, either during work or during social events, the group is classified as 'Unfamiliar'. Other groups, in which participants do frequently interact, are classified as 'Familiar'.

### 7.1.2 Hypothesising Self-regulation in Road Networks

Using the methodology of the preceding sections, the research questions that were proposed in the introduction are examined. Therefore, these questions are translated into hypotheses that can be tested in the closed system of the serious game. To ascertain that the game is designed correctly and the human decision makers act according to the model adopted here, first a set of validation hypotheses are formulated. Thereafter it is discussed how the original questions (albeit in a compacter form) can be answered through hypothesis testing.

**Is the model valid?** As a first step, the model of the game itself is analysed to confirm the validity of the assumptions underlying the design and further experiments. To this end, three hypotheses are proffered. The first hypothesis is that the decision-making of human decision makers can be considered at least boundedly rational. While an exact degree of rationality cannot be determined, decision optimality can be used as an approximation of the ability to rationally maximise utility. Hence, the rationality of a questionnaire response, and thus an indicator for the rationality of the participant, is defined as its relative position on a scale from the lowest possible score to the Pareto-optimal trade-off that is closest to the response score. The resulting metric is an indicator for the rationality on the $[0, 1]$ scale and agents are considered boundedly rational if this their rationality score is at least $0.8$. Therefore it is hypothesised that the mean of rationality scores of the agents is at least $0.8$ with a confidence level of $95\%$.

The second and third hypotheses establish the correctness of the game design. Validation hypothesis two states that the actions affect their designated objective. That is, the hypothesis is that there is a strong correlation between the played strategy scores $G_p$, $G_t$ and $G_r$ and their corresponding impact on the outcome $\mathbf{Y}$, respectively the expected profit $P(\mathbf{Y})$ and ttl $T(\mathbf{Y})$ for the first two and the worst case profit $P_{wc}(\mathbf{Y})$ and ttl $T_{wc}(\mathbf{Y})$ for the latter. Finally, to confirm that coordination of decisions is beneficial to the players and hence the premise that network-based incentives stimulate coordination, a third hypothesis states that coordination is strongly correlated to utility.

**Are monetary incentives effective to influence decision making?** It is hypothesised that applying monetary incentives in a network-based tender is an effective way to influence the decision-making process contractors. Hence there should be a notable difference in the decisions-making preferences of players when comparing the settings with and without monetary incentives. Although both the questionnaire and the game confront players with networks, only the latter actually includes monetary incentives. This hypothesis is tested by comparing the means of the decision scores of both in all decision parameters and showing that they are significantly different.

**Do network incentives lead to self-regulation?** Before the relation between the incentives and self-regulation can be studied, first the notion of self-regulation must be translated into the context of the game. When is a network considered to be self-regulating? And how can self-regulation be quantified or at least partially ordered?

Answering these questions with an exact metric is infeasible, hence two hypotheses are formulated that together approximate the original research question. The first is that due to the presence of incentives, the observed coordination level of every game is at least 'Medium', which corresponds to a network-wide coordination of interactions. In games with a 'Low' rating, coordination arises from conflict that need to be resolved between two or three players and is not considered coordination of the network. Secondly, in a self-regulated network, members are expected to act more in favour of the network, which is expressed in the game through a traffic or risk-averse play-style. Thus, the hypothesis is that a strong positive correlation exists between the coordination level and the strategy scores for ttl $G_t$ and risk-aversion $G_r$.

**Do relationships influence the effectiveness of incentives and, indirectly, self-regulation?** To confirm that decision making is influenced by social cohesion and not a predisposed preference of individual participants, first a comparison is performed between the a priori profile scores and the in-game strategy scores. The hypothesis is that there exists no correlation between all pairs $Q_x$ and $G_x$, where $x$ represents the profit, ttl and risk-aversion objectives. Assuming that this first hypothesis is confirmed, the inverse is studied for social cohesion. That is, it is hypothesised that social cohesion is strongly correlated to changes in strategy scores only and the change in preferences is statistically significant. Finally, it is to be expected that players that are more familiar with each other are more likely to coordinate their operations, expressed in a strong correlation between cohesion and coordination.

### 7.1.3 Playing the Game

The hypotheses of the previous section are validated in a controlled environment that simulates MAINTENANCE PLANNING PROBLEM in a game called "Road Maintenance Game". In this game, players take on the role of one of the five service providers (SPs) that need to plan and execute maintenance work for the client, the asset manager (AM), on the road network visualised in Figure 7.2. As monetary incentives to incite self-regulation, the original network payments of Equation 3.6 are used such that SPs are charged 1 euro per additional hour of ttl caused. The design of the game entails a complex model of actions, rewards and rules, based upon figures of actual road maintenance projects from Brandt [42], of which an elaborate description can be found in the online appendix [231]. Here the core parts of the game are explained to understand the methodology and value the obtained results correctly.

Each players is responsible for an equivalent portfolio of 4 maintenance projects, for which they must select one of the four alternative methods to perform the work – 'low cost', 'low hindrance', 'low risk' and 'fast completion' – and plan them in time. The methods differ in their duration and direct impact on the profits and traffic hindrance, but also in their indirect impact in the form of risk of maintenance delays. All alternatives except for the 'low risk' have a delay probability of 33% and varying delay durations. If delayed, the project may incur additional costs and ttl. The alternatives also vary in network impact: whereas the 'low hindrance' alternative results in low ttl even when it is planned concurrently with other works, the 'low cost' alternative will

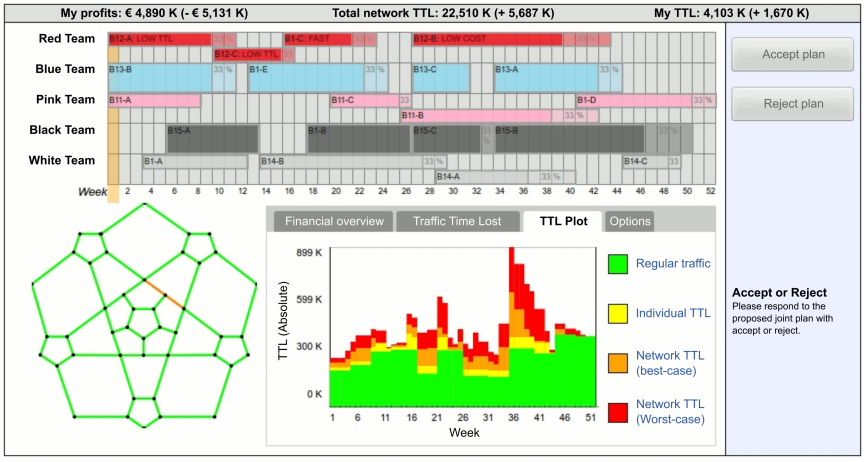



**Figure 7.2** *The user interface of a game in progress with the infrastructure (bottom left), the current joint plan as Gantt chart (top) and a graph showing its impact on ttl (bottom right). In the action bar on the right are the actions currently available to the player (Red Team).*

lead to a substantial amount of ttl if planned concurrently with others. Finally, players may decide to pass up on a project altogether and miss out on the contracted revenue for that task but perhaps avoid steep ttl penalties. During the game, the interface of Figure 7.2 functions as a decision-support system that computes and displays the impact of their decisions, incorporating the last-known decisions made by other players.

The game is run in a client-server fashion, where the game master (the AM) hosts a game server and all players (the SPs) connect to the game with Tablet PCs. Every tablet corresponds to one of the players and shows the planning interface of which an example is shown in Figure 7.2. This interface visualises the planning decisions and their impact on profit and ttl and presents the available actions to the players.

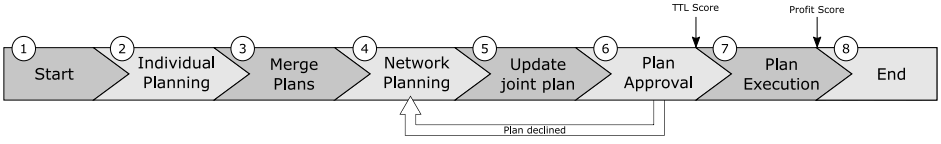



**Figure 7.3** *The various stages of the game and their succession.*

The game play follows the process flow of Figure 7.3:

1. At the start of the game players are introduced to the network, briefed and finally assigned one of the five maintenance portfolios for which they will be responsible during the rest of the game.

2. In the 'individual planning round' they are given the task to develop a maintenance schedule according to their preferences. Their scheduling decisions can be summarised into two actions for every maintenance project: a) choose the

preferred maintenance method and plan it in time or b) choose to not plan the method. Each player individually submits their schedule to the AM when satisfied.

3. Once all players submitted their plans, the AM merges them into a single joint plan and sends that plan to all players thus informing them about the decisions made by others.

4. Now a 'network planning round' starts. With the newly received information about the plans of other players, every player is again requested to submit a maintenance schedule for their operations. This schedule can be the same as before, a slightly modified one to account for the other players or a completely new one. Once they are again satisfied with their schedule, they submit it to the AM and wait for the other players to do the same. Note that there are no real-time updates of the other players' plans during this round, only when all plans are again submitted will this information be updated.

5. Once the (new) plans have again been received by the AM, the joint maintenance plan is updated and sent back to the players. This time an approval round is requested from all players.

6. In the approval round, every player either accepts or declines the joint plan. If any of the players declines, a new network planning round is started and the process is reset to the network planning round of step 4.

7. If all players accept the joint plan, the planning phase ends. From this point onward no more changes can be made to the joint maintenance schedule and the (expected) group ttl score is recorded. Now the execution phase starts, and the only action left in the game is the 'realisation' of outcomes of the maintenance projects. In the execution phase, the plan execution is (gradually) simulated one week at a time until a maintenance operation starts that may delay. The player to which the task belongs then rolls a dice. If the dice lands on a green square, there will be no delay in the execution of the task, whereas a red square means that the task is delayed (effectuated by the game master). This process is continued until all tasks have been fulfilled and a year has passed in the game. Then the game ends.

8. At the end of the game, the session winner is the player that has the highest profit after the execution of the joint maintenance plan.

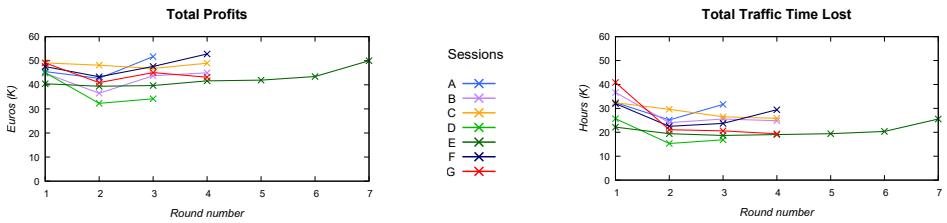
Players can win the game in two ways. In every session, the player that has the highest profit after execution of the joint maintenance plan is declared the winner of the session. On the other hand, the group of players that achieves the lowest expected traffic time lost compared to all other sessions will win as a group (before execution). These two ways to win mimic the typical misalignment between the goals of the service providers (maximum profit) and that of the contractor (minimal nuisance) that is seen in practice. To emphasise this misalignment and provoke competition, there is only a small price for the winning player in a single session (€ 2.50 scratch ticket) but a more valuable one for the players in the winning session (€ 10 vouchers for all participants).

## 7.2 Gaming Results

This section presents visual and tabular summaries of the data gathered through the combination of the questionnaire and the Road Maintenance Game. The full data set can be found in Appendix C and the repository available at `https://github.com/AlgTUDelft/road-maintenance-game`. For the sessions both public institutions and companies were contacted that focus on asset-management related activities. Getting together a large enough group of people able to participate in a gaming session of approximately 3 hours proved to be challenging, especially when dealing with practitioners from the industry. Nonetheless, 7 sessions with a total of 95 participants were hosted in groups of varying composition, skill and social coherence. Furthermore, 60 questionnaire responses were received from the participants. Table C.1 gives a descriptive overview of all sessions. The overall results of all sessions in terms of the game goals, i.e. maximising profit and minimising ttl, are summarised in the graphs of Figure 7.4. The figures show the expected profit and ttl of the group as a whole, as recorded at the end of each round (based on the data of Section C.2 of Appendix C).

| ID | Company/institute Profile | #P | #Q | Category | Coordination | Cohesion |
|---|---|---|---|---|---|---|
| A | University, Computer Science | 9 | 9 | Students | Low | Unfamiliar |
| B | ICT-focused R&D Company | 10 | 9 | Engineers | Low | Familiar |
| C | Utility provider, mainly power | 15 | 3 | Professionals | Low | Unfamiliar |
| D | Dutch national road authority | 17 | 16 | Trainees | High | Familiar |
| E | Dutch national road authority | 8 | 5 | Trainees | Medium | Familiar |
| F | AM Professionals | 20 | 9 | Professionals | Medium | Unfamiliar |
| G | AM and Health-care Consultants | 16 | 9 | Professionals | High | Familiar |

**Table 7.2** *Outline of game session characteristics, from left to right the columns are: session identifier, company/institute, number of participants, number of questionnaires reviewed, participants skill category, and the observed coordination and social cohesion of participants.*




(a) (b)

**Figure 7.4** *Quality of the joint plan at the end of each round in terms of the goals profit and ttl: (a) shows the expected profit in thousands of euros, (b) the expected ttl in thousands of hours. Note that the ranges on the vertical axes do not align, but the sizes of the ranges do. The line styles associated with each session are shown in the legend in the middle.*

Outcomes and decision-preference scores are computed over $N-1$ rounds, such that $N$ is the number of rounds played in a session. This is to correct for the 'last round' effect due to the design of the game that the player with the highest expected profit *at the end of the game* wins the session. This caused players to radically change their strategy in the last round to a profit-driven one in an effort to win the prize. With the exception of session $G$, all games suffered from this effect. This can also be observed from the graphs in Figure 7.4: all lines show a substantial growth in profits in the last round, often paired with an increase in ttl.

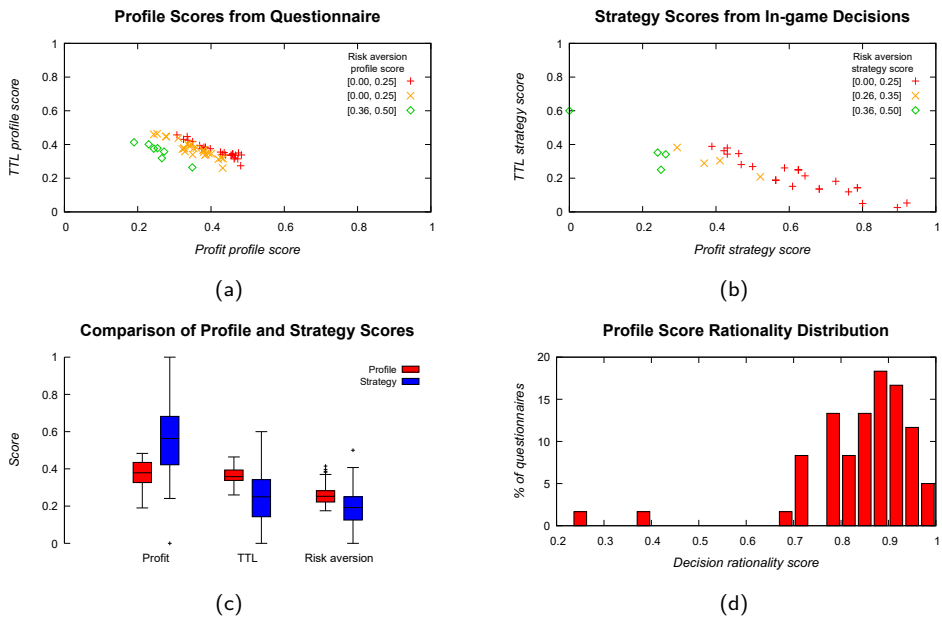



**Figure 7.5** *Visualisation of (a) the profile scores determined from the questionnaire responses, (b) the strategy scores computed from in-game action decisions, (c) a comparison of their distributions and (d) the computed rationality of questionnaire responses.*

The decision preferences of participants are visualised through the graphs in Figure 7.5, where Figure 7.5a plots the profile scores and Figure 7.5b the strategy scores. Both figures show the profit and ttl preferences on the x and y axis respectively, and use different point styles to categorise the risk aversion scores associated with each point. For example, points visualised by a diamond shape correspond to a preference score with a preference score for risk aversion between $0.36$ and $0.5$. Risk aversion scores greater than $0.5$ are not found in either the profile score or the strategy score. Figure 7.5c visualises the distribution of profile scores and strategy scores using a box plot. This picture illustrates the range in which preferences are typically expressed. The means are illustrated by the thick lines in the vertical middle of the boxes, that represent the first and third quartile of the data set. The whiskers visualise the most distant point on both ends that are at most $1.5$ away from the inter-quartile distance.

Finally, the histogram of Figure 7.5d shows the distribution of rationality as computed from the questionnaires. All visualisations use the scores of Tables C.2 and C.6.

| | $Q_p$ | | $Q_t$ | | $Q_r$ | |
|---|---|---|---|---|---|---|
| $G_p$ | -0.10 | 57% | -0.10 | 54% | 0.22 | 91% |
| $G_t$ | 0.14 | 73% | 0.12 | 63% | -0.29 | 98% |
| $G_r$ | 0.00 | 2% | 0.15 | 75% | -0.13 | 68% |

**Table 7.3** *The correlation strengths and associated confidence levels (grey) for every pair of profile and strategy scores.*

Based upon the same data sets, a correlation analysis is performed on the relationship between profile scores and strategy scores. Correlation coefficients are computed using the Pearson method [247, 269] and correlation confidence is computed through two-tailed sample t-tests with confidence levels $95\%$ and $99\%$. Interpretation follows the model for social sciences [66] and, conform the labelling by Taylor [247], defined as weak for coefficient values in the range $[0, 0.35]$, moderate for $[0.36, 0.67]$ and strong for $[0.68, 1.0]$. The summary of this analysis is shown in Table 7.3.

| | *Expected outcomes* | | | | *Worst-case outcomes* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Variable** | $P$ | $T$ | $u$ | $\phi$ | $P_{wc}$ | $T_{wc}$ | $u_{wc}$ | $\phi_{wc}$ | CD | CH |
| $Q_p$ | -0.15 | *0.31* | **-0.47** | **-0.34** | -0.13 | -0.19 | 0.13 | 0.07 | **-0.51** | -0.18 |
| $Q_t$ | -0.01 | -0.13 | 0.13 | 0.14 | -0.11 | -0.15 | 0.11 | 0.07 | 0.10 | 0.02 |
| $Q_r$ | 0.21 | *-0.30* | **0.52** | **0.34** | *0.27* | **0.38** | *-0.27* | -0.15 | **0.60** | 0.22 |
| $G_p$ | **0.87** | **0.64** | 0.09 | *-0.41* | **0.80** | **0.68** | -0.31 | *-0.43* | -0.37 | **-0.69** |
| $G_t$ | **-0.82** | **-0.53** | -0.18 | 0.28 | **-0.81** | **-0.55** | 0.13 | 0.24 | 0.26 | **0.68** |
| $G_r$ | **-0.78** | **-0.67** | 0.05 | **0.50** | **-0.63** | **-0.72** | **0.48** | **0.59** | **0.58** | **0.58** |
| Coordination | -0.23 | *-0.45* | *0.37* | **0.59** | -0.12 | **-0.51** | **0.58** | **0.68** | - | **0.57** |
| Cohesion | **-0.57** | *-0.42* | -0.05 | 0.33 | **-0.50** | **-0.47** | 0.25 | *0.38* | **0.57** | - |

**Table 7.4** *Summary of the correlation analysis between the variables and outcome. All correlation coefficients with a statistical confidence level greater than $99\%$ are shown in bold, greater than 95% in italic and all correlations with lower likelihoods of being correct predictors are depicted in grey.*

To gain insight into the interactions between the parameters in the environment and their relation to the outcomes of the game a comprehensive correlation analysis is performed, of which the summary is presented in Table 7.4. Correlations shown in bold correspond to a confidence level of at least $99\%$, italic to $95\%$ and the others are shown in grey. For the game outcomes, the expected profit $P$ and ttl $T$ is listed along with the associated performance ratio $\phi$ and the worst-case profit $P_{wc}$ and ttl $T_{wc}$ with their performance ratio $\phi_{wc}$. The abbreviations CD and CH refer to respectively the coordination level and the cohesion level of games. The figures underlying this table

can be found in Table C.1 (coordination and cohesion level), Table C.2 (profile scores), Table C.6 (strategy scores) and Table C.4 (outcomes) of the appendix.

Table 7.5 takes a closer look into the variables coordination level and cohesion level. This table lists the averages per category for the expected outcomes and associated performance ratio, the worst-case outcomes, profile scores and strategy scores of either variable. The same data set as for Table 7.4 was used in a similar fashion to construct this table. Note that the correlation coefficient included in this table are the same as in Table 7.4 but restated to provide a more complete overview.

| | *Expected outcomes* | | | | *Worst-case outcomes* | | | | *Profile scores* | | | *Strategy scores* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $P$ | $T$ | $u$ | $\phi$ | $P_{wc}$ | $T_{wc}$ | $u_{wc}$ | $\phi_{wc}$ | $Q_p$ | $Q_t$ | $Q_r$ | $G_p$ | $G_t$ | $G_r$ |
| **Coord.** | | | | | | | | | | | | | | |
| Low | 45.7 | 28.6 | 17.1 | 1.62 | 33.0 | 35.2 | -2.1 | 0.95 | 0.43 | 0.36 | 0.21 | 0.60 | 0.23 | 0.18 |
| Medium | 43.3 | 23.6 | 19.6 | 1.87 | 31.9 | 28.1 | 3.8 | 1.16 | 0.35 | 0.37 | 0.28 | 0.53 | 0.23 | 0.24 |
| High | 43.2 | 22.4 | 20.8 | 2.00 | 31.8 | 26.1 | 5.7 | 1.27 | 0.34 | 0.37 | 0.29 | 0.50 | 0.27 | 0.24 |
| *Corr.* | -0.23 | *-0.45* | *0.37* | **0.59** | -0.12 | **-0.51** | **0.58** | **0.68** | **-0.51** | 0.10 | **0.60** | *-0.37* | 0.26 | **0.58** |
| **Cohesion** | | | | | | | | | | | | | | |
| Unfam. | 47.7 | 28.3 | 19.4 | 1.70 | 35.0 | 34.7 | 0.3 | 1.02 | 0.39 | 0.37 | 0.24 | 0.65 | 0.18 | 0.17 |
| Familiar | 42.0 | 23.1 | 18.9 | 1.90 | 30.7 | 27.2 | 3.4 | 1.18 | 0.36 | 0.37 | 0.27 | 0.48 | 0.28 | 0.24 |
| *Corr.* | **-0.57** | *-0.42* | -0.05 | 0.33 | **-0.50** | **-0.47** | 0.25 | *0.38* | -0.18 | 0.02 | 0.22 | **-0.69** | **0.68** | **0.58** |

**Table 7.5** *Average values for the outcomes and scores per coordination and cohesion category. Again correlation coefficients with a confidence level of at least 99% are shown bold, 95% in italic and the rest in grey.*

### 7.2.1 Findings

Using the measurements and observations of the previous section, the hypotheses formulated for the research questions are again considered. The same structure as before is used in presenting the findings relevant to each question.

**Validity of the Model** To assert that human decision makers are at least boundedly rational, the hypothesis is that the mean of rationality scores is at least $0.8$. In other words, human decision makers are capable in finding at least $80\%$ optimal solutions on average. This assumption is validated by performing a one-tailed, one-sample t-test where the mean of rationality scores $\mu_\theta$ is compared against a mean of $0.8$ using null-hypothesis $H_0$: $\mu_\theta > 0.8$ and a confidence level of $95\%$. The resulting probability value is $0.010$ and since this is lower than the desired confidence value of $0.05$, it is concluded that the null-hypothesis is valid. Furthermore, there are two outliers at $0.255$ and $0.385$ that likely correspond to misunderstanding the questionnaire, as the rationality of the next lowest is close to $0.7$. When these outliers are removed, the mean of rationalities becomes greater than $0.84$ with the same confidence level.

The correctness of the game design is illustrated by the correlations between the strategy scores and the outcomes listed in Table 7.4. Starting with the profit strategy score $G_p$, the table shows a strong positive correlation between this score and the total expected profits $P$ and worst-case profits $P_{wc}$ of the game outcome as the absolute value of the coefficient is greater than $0.67$. Profit strategy scores show a moderate to strong positive correlation to the ttl in the resulting outcomes. Note that this positive correlation is conform the intended design as higher ttl values mean more network hindrance. For ttl, a similar analysis shows that it is strongly negatively correlated to profits, and moderately negatively correlated to ttl. Risk aversion is also strongly negative correlated with both and in addition shows a moderately positive correlation with the performance ration, e.g. the ratio of profit versus ttl, thus the risk-averse actions are likely to lead to performance increases. In conclusion, although the original hypothesis of strong correlations is not proven, the correlation coefficients are all in the range of moderate to strong correlation and hence the hypothesis is highly plausible.

Note that although it might seem that playing a risk-averse strategy is more effective to reduce the overall ttl, this cannot be concluded from the table. A quick linear regression analysis reveals a minor flaw in the design of the actions however. The slopes of both functions show that risk-averse actions are approximately $1.6$ times more effective in reducing ttl ($-43.8$K versus $-71.6$K). This effect was not intended but is a consequence of the factor-based ttl model used that amplifies ttl for every concurrent maintenance operation. Observe also that there is no evidence for a relationship between strategy and the utility of the players.

It remains to show that coordination is beneficial to the outcomes. Table 7.5 illustrates the correlations of interests. These figures do not provide sufficient evidence for the hypothesis of a strong correlation, nonetheless they do reveal a moderate correlation between the coordination level and the utilities. This suggest that again it is at least plausible that coordination is beneficial to the players, also supported by the observed average utility per category. Moreover, the performance ratio shows a moderate to strong correlation to the coordination level, indicating that more coordination is likely to improve the trade-off between profits and ttl. Finally, coordination seems to substantially improve the worst case scenarios in the game to the benefit of all players.

**Influence of Monetary Incentives on Decision Making** The effectiveness of the monetary incentives is measured through the hypothesis that a significant difference can be observed between the decision making with and without monetary incentives. The decision-making preferences in the former situation are given by the strategy scores from the game, the latter through the profile scores computed based on the questionnaire. Indeed, Figure 7.5a to Figure 7.5c appear to visualise a difference between the two categories. In Figure 7.5a, the profile scores seem relatively balanced with a slight preference towards profit and ttl over risk aversion. The strategy scores of Figure 7.5b seemingly indicate that when monetary incentives are used, a profit-focused play style is preferred. This assessment is supported by the box plot in Figure 7.5c that visualises the distributions of both score types. A two-tailed, paired t-test between the means of all $Q_x$ and $G_x$ is performed with null-hypotheses $H_0 : \mu_{Q_x} = \mu_{G_x}$ and a confidence level of $95\%$ to confirm the statistical significance of the difference. The resulting

probability values are $0.009$, $0.001$ and $0.313$ for respectively the profit, ttl and risk-aversion scores. Hence, with high statistical likelihood, the changes in preference for profit and ttl can be ascribed to the presence of monetary incentives, but risk-aversion seems unaffected.

**Self-regulation of the Network** The results indicate that a change of behaviour can be observed in the presence of incentives. Hence it is interesting to know whether that change corresponds to the manifestation self-regulation. The first hypothesis in this regard is that the presence monetary incentives always leads to a coordination level of 'Medium' or 'High'. A quick glance at Table C.1 is sufficient to invalidate this hypothesis. In three games coordination was limited to only conflict-driven negotiations. The second hypothesis states that monetary incentives lead to a change in decision-making strategy towards a ttl-driven or risk-averse play style in the game. Table 7.5 however shows only a moderate correlation between coordination and risk aversion, and no such a relationship is established between coordination and ttl strategy scores. Hence the incentives are unlikely to incite self-regulation.

An interesting additional find from the same table is that whereas Table 7.3 illustrates that a correlation between profile scores and game scores is highly unlikely, the coordination level is moderately correlated to both the profile and strategy scores for risk. This does not reveal a relationship between risk aversion and coordination, but it does suggest that network members with a predisposition against risks are more likely to coordinate their operations, with or without the presence of incentives.

**Role of Social Cohesion** As incentives do not consistently incite self-regulation, the influence of relationships on coordination of even greater interest. First the hypothesis is tested that preference changes are due to the presence of familiar network members and not from player profiles. The absence of any relationship between the profile scores and social cohesion can be concluded from Table 7.5 with a high likelihood. All profile parameters are at most weakly correlated with low likelihood (respectively $83\%$, $10\%$ and $91\%$ for profit, ttl and risk aversion). The strategy scores, on the other hand, appear to be related to cohesion. With risk aversion being at least moderately correlated, a strong correlation is revealed for profit and ttl strategy scores. A one-tailed paired t-test is performed to analyse the statistical significance of this change in strategy means. The use of a one-tailed test follows from the strong directions of the coefficients. As null-hypotheses $H_0 : \mu_{G_{x,U}} > \mu_{G_{x,F}}$ are used, such that $U$ and $F$ correspond to the categories 'Unfamiliar' and 'Familiar' and $x$ to the objectives. The tests yield probability values $0.020$, $0.028$ and $0.035$ for respectively the profit, ttl and risk-aversion strategy scores with a confidence level $95\%$ and hence the behavioural change to a collaborative play style is statistically significant. With respect to the correlation between cohesion and coordination, a similar conclusion can be drawn as twice before. Table 7.4 provides evidence of a moderate correlation but the original hypothesis of a strong correlation is not satisfied, although a relationship is very plausible. Summarising these findings, social cohesion is a probable moderator for the effectiveness of monetary incentives.

## 7.3 Evaluation of Gaming Results

Here the findings of the experiments are discussed and their relevance to the broader context is addressed. First, the effectiveness of monetary incentives to influence decision making is considered and in particular how behaviour is changed and how this relates to current literature. Thereafter the role of social cohesion is investigated.

The effectiveness of incentives to influence decision making was already highlighted by other scholars such as Bresnen and Marshall [45], Bower et al. [41] and Rose and Manley [218]. This study contributes an empirical confirmation that indeed incentives changes behaviour of participants within the controlled experimental setting and provides an insight into *how behaviour is influenced* (Figures (a) to (c)). The findings show that the payment mechanism used in this game led to a competitive play style focused on profits, whereas the questionnaire responses of the same group of participants were more balanced in their decision preferences. Even more so, all but a few participants motivated their responses in the questionnaire among the lines of "this seems to optimally balance profits and ttl" when asked why they chose a specific alternative. In the game, however, profit was observed to be the key driver of all but a few players. Nonetheless, the profit-driven behaviour did not result in the emergence of coordination, even though players could improve their utility by coordinating their actions. Further investigation into the existence of any correlation between coordination and agent decision making reveals moreover that no substantial differences are found in the strategic preferences for different levels of coordination, except for a stronger risk aversion (Table 7.5). The latter effect is explained by an increased focus on robust planning so that coordination of activities is guaranteed. The experiments here therefore contribute to an improved understanding of the influence of incentives on decision making, but also lead to wondering why coordination does not emerge, even if players can benefit from doing so.

One explanation that is offered by literature is the inability of players to fully comprehend the complexity of the domain and thus fail to maximise their gains, as observed by for instance Eriksson [83] and Shadid [235]. Although the participants performed near optimal in their questionnaire responses, the complex dynamics of a coordinating a 5-player network under pressure of time via a new interface may justify this explanation. Along the same lines is the relative novelty of the role the players assume which requires an additional set of skills and capabilities [112, 141, 246] that the participants may not possess. In either case, a mitigating measure is to employ computer-aided decision support techniques that help to maximise the value of planning and suggest coordination to network members when this is beneficial, leading even the most isolated or selfish network members to coordinate decisions. Positive examples in similar decision-support scenarios can be found in the works by e.g. Cheung et al. [61], Le Bars and Le Grusse [23] and Douma et al. [82]. Of additional value is the use of collaboration tools and knowledge sharing, as proposed by Eriksson [83]. Whether these extra measures result in more coordination could be established through additional gaming experiments and would be a valuable empirical contribution by future work.

A more plausible explanation offered by e.g. Bresnen and Marshall [45] and Fehr and Falk [85] is that the monetary incentives lead to an effect that is opposite to their

intended design. The payment mechanism used in the game penalises the players per additional hour of traffic time lost. Although the mechanism was intended to stimulate reduction of hindrance due to maintenance and encourage efficient and innovative operations, it can be experienced as a painshare mechanism that only penalises bad behaviour. This type of mechanism has been found much less effective to motivate contractors. For instance, Choi et al. [62] conclude from a large survey of completed projects that agreements that incorporate both positive and negative incentives successfully improved performance while agreements with only penalties led to performance worse than conventional contracts. Through a serious gaming study not unlike the one performed here, Altamirano and de Jong [11] demonstrated that high penalties seem to create incentives for collusive behaviour, while a combination of moderate penalties with significant bonuses creates a positive atmosphere of trust. Similarly, Hosseinian and Carmichael [118] remark that "sharing gain/pain provides a strong motivational factor for all parties to work together, rather than in a confrontational or adversarial fashion, with the desired result of producing a successful project". Particularly the 'adversarial fashion' seems to fit the observations made here. Instead of motivating players to collaborate, the incentives appeared to stimulate a competitive attitude with a decrease in ttl and risk-averse preferences in favour of a profit-driven play style. Not only is competition within the network regarded detrimental to its performance [69, 235], it is counterproductive to an open and cooperative network environment in which decision coordination and co-creation is stimulated [118, 113].

In contrast, the experimental results regarding *social cohesion* present a new empirical confirmation of the positive impact of the social dimension on network performance. A contribution of this work is that while networks with familiar members demonstrate a more collaborative attitude in their decision making, as per expectation, it is newly shown that no such inclination was observed in questionnaire responses of the same participants. Another new finding is that while the 'socially-cohesive' networks were confronted with the same monetary incentives as the unfamiliar networks, they did not show a similar competitiveness. As a corollary, the presence of familiar network members is seemingly influencing their decision making towards a more collaborative style and supports the incentive mechanism in achieving its intended results. This once more stresses the importance of building a socially cohesive network, as many authors have before [45, 7, 140, 255], but contributes also the learning that the relationships within the network may be a necessary condition for (inadequately designed) monetary incentive schemes, or are at least beneficial to realising its goal of maximising network performance.

### 7.3.1 Conclusion

Concluding the experimental evaluation, the results show that incentives are effective in changing decision preferences but may lead to counterproductive effects. Instead of inciting better performance, the findings demonstrate that players are driven to a competitive attitude, detrimental to collaboration. Furthermore, it was observed that networks with strong social cohesion can overcome this competition and will self-regulate their performance as intended by the incentive design. These results confirm

previous findings on the importance of relationships within the network and additionally signify the role of social cohesion on the effectiveness of incentive mechanisms, suggesting that strong social cohesion is prerequisite to performance-based monetary incentive mechanisms.

The recommendation ventured from this study is therefore that if self-regulation is to be successfully implemented in contracts, it should be based on positive incentives structures that incite better performance in a cooperative atmosphere of co-creation and trust. Consequentially, self-regulation is deemed most effective in collaborative team settings such as alliances and early contractor involvements or integrated project delivery. Future work is needed to clarify the impact of the incentive design on self-regulation, compare the findings here to the non-incentivised setting and perform repeated studies to strengthen the conclusion drawn in this study on road maintenance. The serious game setup of this study can be reused in future settings and is available online to be used in related studies.

## 7.4 Further Discussion

Ultimately, the failure of monetary incentives to incite coordination in every gaming session may be interpreted as evidence against the use of incentives to achieve self-regulation, but a closer look at the results suggests otherwise. The experiments in a controlled setting reveal a definite potential for monetary incentives but their potential is determined by the type of incentives that are used, the relationships within the network or a combination of both. From the gaming sessions it is yet impossible to conclude the exact role either element. As a consequence, further study is strongly recommended to isolate either the design of the mechanism or social cohesion as the main contributor to self-regulation, or conclude that both are prerequisite to network self-regulation. Regardless, the experiments performed provide new insights and recommendations for both the research on and the application of incentives in networks. With the next step of bringing self-regulation into practice through monetary incentives, and evaluating their effect in real-world scenarios, three main directions for current and future work are identified.

First of all, adequate design of the incentive scheme is vital to the success of performance-based tenders and care should be taken in the type of incentives that are implemented, especially when good relationships cannot be guaranteed. The use of painshare without the gainshare led to competition and selfish optimisation within the network and this should be considered in game-theoretically engineered incentive schemes such as that by [106, 228, 254] or [117]. Although on paper such mechanisms result in optimal coordination of the network, they may be counterproductive to self-regulation in practice if based upon an incompatible payment mechanism. The effect of incentive design, and in particular the type of mechanism employed, needs further investigation to establish the exact mediators that realise the desired changes in decision making towards self-regulation.

Secondly, the findings here again underline the paramount importance of the social dimension in network-based approaches and emphasise that relationships between network members are to be fostered. Even though the precise effect of social cohesion

is not fully determined, monetary incentives to stimulate self-regulation seem most effective in cooperative networks and its promise is hence the greatest in managing the interactions in collaborative networks such as alliances, early contractor involvements or integrated project delivery. In such networks cooperative behaviour between the parties is established and measures are taken to get all parties to work to the same goal [50, 118]. Moreover, alliances are becoming the preferred delivery for large and complex projects [114, 163] and hence the contribution of self-regulation as a method to optimise network performance is very relevant for future work.

Thirdly, to bring self-regulation into realistic tenders, concerns will have to be addressed with respect to accountability and the possibility of failure to stimulate desired performance. While the application of performance-based contracting on the network level is certainly promising, the lack of a principal in the network may lead to gaps in the responsibility of the network [7]. Self-regulation is inherently in tension with the accountability of the network and requires a solid legal basis before practical use can be realised. Furthermore, as also prevalent in the experiments, decentralised tools and lack of coordination by a central institution or agency can certainly worsen the performance and output of a network, as noted by [249], and may in the worst case lead to a fall-back to regulatory approaches [239]. More research is needed into mechanisms and their enforcement to prevent such fall-backs in practice. One typical approach to reduce the probability of failure and increase the effectiveness of incentives is to perform regular monitoring [124, 90], or at least increase the perception thereof [185], but this is typically paired with substantial costs to the client.

Finally, care must be taken in the interpretation of the results from the experiments and their translation to the real world. Although players often show behaviour similar as they would in comparable real-world situations, the game will always remain a simplified model of the world and hence behaviour cannot be directly extrapolated to realistic scenarios. The absence of real-world consequence may also cause players to take more risks than they normally would. The gaming setting however helps to explore potential changes in decision-making preferences and patterns in behaviour under controlled conditions that may guide further experimentation and research. On that note, the serious game itself is an additional contribution of this work and available online for future research. It can be used for instance to play the same game with different incentive mechanisms, or the absence of incentives, to further investigate their impact, but also to explore the potential related approaches such as the Dynamic Contracting framework of [254] or the gainshare/painshare mechanisms of [118]. Alternatively, it can be employed as a tool to get practitioners acquainted to the concept of self-regulation, alike [9], [110] and [82], or to perform many experiments using automated agents to play the game [262]

# Chapter 8

# Discussion and Conclusions

Self-regulation is the next step in performance-based contracting with the potential to transfer the desirable properties of performance-based approaches to group contracts. Examples of such benefits are increased flexibility, better preservation of autonomy and authority of participants, maximal utilisation of knowledge and expertise, emergent coordination and, consequentially, better use of public funding. The main focus of this research is to combine algorithmic techniques of decision-theoretic planning and game theory to enable this transfer. The key concept is that through careful design of monetary incentives and the availability of automated planning support the contracted group of agents will become *self-regulating* and successful outcomes can be guaranteed. In other words, agents will actively seek to coordinate their joint decisions amongst themselves and optimise their contribution to the contracted goal. The previous chapters have proposed various approaches with the purpose of bringing self-regulation closer to application in group contracts, addressing specific parts of the main challenge. Here they are discussed within the broader context of the main research question of this thesis:

**Main Research Question**

> Can algorithmic techniques be employed to efficiently coordinate planning in self-regulating contracts and ensure successful outcomes while preserving the autonomy and interests of the agents?

All of the algorithms and techniques presented in Chapters 3 to 7 contribute to this goal of achieving successful self-regulation in team contracts but concentrate on different elements of this challenge. Whereas Chapters 3 to 5 focus on the decision-coordination aspect of self-regulation, Chapters 6 and 7 address the design and application of these incentives. In Section 8.1 the research questions are reviewed against the contributions each of the aforementioned chapters. From the evaluation of the questions an answer to the main question is formulated in Section 8.2. Thereafter, in Section 8.3 the position of this work is discussed in a broader context and the direct and indirect implications of the contributions of this thesis are reviewed in the various areas of research that are touched upon.

## 8.1 The Challenges of Self-regulation

In Section 1.3, the main research question of this thesis was broken down into several smaller questions that address various aspects of self-regulation in the context of contracts. Here these challenges are re-evaluated and, now equipped with the algorithms, mechanisms and empirical evaluations presented throughout the chapters of this thesis, it is discussed if and how they can be overcome.

**RQ1 Coordinating self-regulating planning with existing techniques**

Can the vast body of existing planning literature, with its tools and techniques, be employed to develop joint policies for self-regulating planning problems?

Yes. Chapter 3 demonstrates two ways in which the MAINTENANCE PLANNING PROBLEM (MPP), the main example problem and an instance of the more generic SELF-REGULATING PLANNING PROBLEM, can be represented by the Markov decision process model that is widely supported in stochastic planning literature and tools. The first is a direct translation of the problem into a multi-agent MDP (MMDP), thus making it possible for any MMDP solver to produce solutions for MPP. Additionally, Chapter 3 shows that for MPP it is also possible to 'flatten' the MMDP into a joint but single-agent MDP hence allowing *any* generic MDP solver to develop policies for the MAINTENANCE PLANNING PROBLEM, of which there are plenty in the literature of stochastic planning. The experiment at the end of the chapter illustrate that for instance the SPUDD solver [116] is able to produce optimal solutions.

**RQ2 Leverage the structure of self-regulating planning**

Can the structural properties of self-regulating planning problems be leveraged to produce optimal joint policies significantly more efficient than currently available methods?

Yes. Instances of the SELF-REGULATING PLANNING PROBLEM, such as MPP, exhibit a particular structure in their transition and reward functions as a consequence of their definition. In essence, the decisions of agents in self-regulating planning are completely independent except for the reward interaction introduced by the performance-based payment functions. Chapter 4 presents the CoRe algorithm that exploits this particular absence of transition dependence. In the empirical evaluation at the end of the same chapter it is demonstrated that this algorithm manages to solve 'bigger' instances of MPP and performs at least one order of magnitude faster than its available alternatives, `Dynamic Programming` and SPUDD. For many instances runtime reductions of a factor 100 are observed, while some are even solved 1000 times faster.

**RQ3 Self-regulating planning with multi-dimensional objectives**

Can multi-objective planning methods be applied to self-regulating planning problems with multi-dimensional objectives to efficiently find an optimal joint policy for every linear trade-off between objectives?

No. In the special case where objective weights are known beforehand, the multi-objective MDP (MOMDP) that represents the multi-objective MPP can be transformed into a 'regular' multi-agent MDP and solving this special case is therefore at least as hard as finding an optimal policy for MMDP. In general, these weights are not known a priori and hence multi-objective solvers have to produce a set of optimal policies for *every* combination of objective weights. The `OLS` method discussed in Chapter 5 reduces the number of optimal policy computations to a minimum by focusing on the *Convex Coverage Set* (CCS), i.e. the set of optimal policies that cover all combinations of linear weights, but still many time-consuming exact solving runs have to be performed to generate this set.

Given that exact solving is typically too computationally-demanding for all but the smallest instances of MPP, the attention shifts towards finding approximate solutions for multi-objective MPP. In Chapter 5 it is shown that without the requirement of optimality it becomes possible to solve a much broader range of instances while the loss of quality is shown to be relatively small or can even be bounded. The `Approximate Optimistic Linear Support` (`AOLS`) algorithm provides a bounded guarantee on the relative error with respect to the optimal convex coverage set, allowing a fully configurable trade-off between solution quality and the maximum available or allotted runtime. The empirical evaluation shows that `AOLS` is able to produce a near-optimal CCS with errors lower than $2\%$ for most instances, and in some cases it even produces the optimal CCS in just a fraction of the time that the exact `OLS` algorithm needs (even compared against the `CoRe` solver that specifically leverages the characteristics of MPP). In situations where more is known about the distribution of the objective weights, the `Scalarised Sample-based Iterative Improvement` (`SSII`) algorithm may be more effective than `AOLS` in producing a high-quality coverage *for a specific weight region*, albeit without guarantees on the solution quality. Empirical evaluation shows that `SSII` on average can produce a higher-quality CCS than `AOLS` for the specific area of focus.

Despite the positive results obtained through approximation, the answer to the original question still remains negative. While approximation techniques seem promising and are able to produce high-quality solutions for MOMDPs, they do not offer the guarantee of optimality that was sought after. Furthermore, as current exact algorithms only manage to solve small instances of MPP, it is unknown how the results obtained here generalise. Further research is required to evaluate how well the approximation algorithms presented here scale in terms of quality and runtime when faced with larger or harder problems.

**RQ4 Self-regulating planning with self-interested agents**

Can game-theoretical techniques be employed to guarantee optimal joint decision policies for self-regulating planning problems if agents are autonomous and self-interested?

Yes. However not all conditions of the question can be satisfied at the same time. In Chapter 6, two algorithmic techniques have been proposed to counter the self-interested behaviour of agents, required to ensure that self-regulation does not suffer

from opportunistic behaviour. The first is the Dynamic Maintenance Mechanism, a dynamic-Vickrey-Clarke-Groves mechanism that is tailored to MPP in this thesis but can trivially be extended to other self-regulating planning problems. As it is a dynamic-VCG mechanism, using this mechanism guarantees that the sum of agent rewards is maximised, no agent can unilaterally benefit from misreporting its information (e.g. cheating) and that there is no need for external funding to operate this mechanism. In essence, through a carefully designed payment mechanism, it makes it in the best interest of the agent to report its planning problem honestly. Reporting anything else will result in a loss of reward for the agent. From all these truthful reports the mechanism operator establishes an optimal joint decision policy and imposes this on the participating agents.

Exactly this prerequisite for the dynamic mechanism approach, that is, the exchange of private information and central coordination, makes it impossible to satisfy the condition of agent autonomy. Furthermore, the dynamic maintenance mechanism has to compute several optimal joint decision policies in each round of its execution, resulting in a prohibitively large computational effort for most realistically-sized MPP instances. If either autonomy is principal or computational efficiency is required, a decentralised approach based on best-response planning is more suitable. Chapter 6 introduces the stochastic planning congestion game and proves that an iterative, myopic improvement of individual plans eventually reaches an equilibrium from which no agent wants to divert. Put differently, after a finite number of rounds in which agents optimise their plans with respect to the known plans of other agents, a joint plan is guaranteed to exist in which no single agent could be better off by switching plans. Note however that the existence of such an equilibrium merely implies that a best-response planning algorithm terminates after a finite number of rounds. This work has not investigated the quality of the obtained outcomes of best-response planning; existing work on for instance the "price of anarchy" [221] could be used to quantify outcomes. Still, its preservation of autonomy and computational efficiency make this an appealing approach to combat self-interested behaviour.

**RQ5 Confronting self-regulating planning with the real world**

Can the theoretical guarantees of self-regulating incentives be transferred to real-world group tenders to ensure successful outcomes if planning decision are made by human decision makers?

Unclear. The empirical validation of Chapter 7 shows that monetary incentives can indeed be successful in inciting self-regulation between agents and stimulating them towards favourable outcomes. In the experiments, however, it was only observed in sessions with groups in which the social relationships between agents are strong. For groups in which members are not familiar with one another, the monetary incentives led to competitive behaviour with detrimental impact on the coordination of the group. Currently it has not been shown whether the design of the incentives or the social cohesion is the main contributor to the emergence of coordination, or that self-regulation only manifests as a combination of both. Still, the restricted setting of the serious game provides first evidence for its potential in settings with human decision makers, espe-

cially in the context of collaborative partnerships such as alliances. Further research is needed before a successful transfer to real-world group tenders can be ensured.

## 8.2 Conclusion

From the answers to the research questions above a conclusion is summarised in answer to the main research question of this thesis:

**Main Research Question**

> Can algorithmic techniques be employed to efficiently coordinate planning in self-regulating contracts and ensure successful outcomes while preserving the autonomy and interests of the agents?

Yes, although not all requirements of the main research question can be met concurrently by any of the approaches presented in this thesis and their implementation in real-world contracts is still far away. No one-size-fits-all method exists to implement self-regulation in contracts that simultaneously achieves efficient coordination, preservation of agent autonomy and interests, absence of opportunistic behaviour and guaranteed solution quality. Instead several techniques to implement and support self-regulation in contracts can be discerned, each with its own strengths and weaknesses.

The dynamic mechanism of Chapter 6 is the most promising approach from the perspective of contract design. Through carefully designed performance payments this mechanism provably discourages opportunistic behaviour, makes coordination of decisions in the best interest of agents and optimises the expected value. To compute these payments, however, the mechanism requires full information about the (private) decision processes of all the participants and needs to solve the underlying self-regulating planning problem, for example the MAINTENANCE PLANNING PROBLEM, optimally in every round of its execution. Even with optimised solving algorithms such as CoRe of Chapter 4 this is computationally very demanding for even single-objective models; multi-objective models with their inherent increase in complexity are infeasible to implement in such a mechanism. Successful outcomes can only be guaranteed if the agents 'stick to the plan', i.e. they partially yield their autonomy to the centrally developed joint policy. Furthermore, the assumption of rationality is shown to be rather strong in practice and more research is required to obtain similar guarantees in the case of bounded rationality. This technique is most suitable for settings in which the cost of opportunistic behaviour is significantly high, problem sizes allow for optimal solution computation within reasonable time or optimal solutions are demanded.

Best-response planning offers a more efficient method of coordination that preserves autonomy, interest and privacy of agents, but it can only partly ensure successful outcomes. In particular, opportunistic behaviour can be discouraged but no guarantees on the quality of the resulting coordination can be given. Nevertheless, rationality or optimality are no requirements and hence agents can employ any type of algorithm to produce a plan in the best-response approach. This means that approximations such as the UCT* algorithm of Chapter 3 could be used, but it also opens the way for multi-objective planning as in Chapter 5. On the other hand, the increased efficiency is paired

with a lack of guarantees regarding the quality of the outcomes, due to the limited information available and lack of global coordination. This method is most applicable to adversarial or competitive scenarios in which mistrust or commercial reasons makes agents reluctant to share private information or accept centralised coordination.

Central coordination is suitable if efficiency and solution quality are considered the most important desiderata in the implementation of self-regulation. This method implements the aforementioned mechanism-based payments to motivate self-regulation but coordinates decisions centrally using one of the algorithms for self-regulating planning problems presented in this thesis. Alternatively, this approach can also be implemented in the form of a collaborative decision support system that allows human decision-makers to express preferences and quickly analyse many high-quality solutions. It does require a complete submission of autonomy and needs an external mechanism to mitigate the impact of opportunistic behaviour, e.g. regulation, past-performance or simply trust. This approach is most appropriate in scenarios with fully-cooperative agents, collaborative platforms such as decision support systems or settings in which opportunistic behaviour is already mitigated, not expected or has limited impact on the outcomes.

## 8.3 Implications and Next Steps

The previous section summarised the challenges of self-regulation and the contributions made by this thesis to overcome them from an algorithmic point of view. Although significant progress is made in various areas, still much work is required to bridge the gap between the theoretical concept of self-regulation and its implementation in real-world contracts. This section discusses the impact of the contributions made by this thesis in terms of immediate next steps that can be taken and open questions that are to be addressed in subsequent research. These next steps are discussed for the three main topics around self-regulation addressed by this thesis: decision coordination, incentive design and its implementation within contracts.

### 8.3.1 Decision Coordination

With respect to the topic of decision coordination, this thesis contributes mostly to the field of decision theory, in particular to the area of stochastic decision-theoretic planning. Although the chapters throughout this work mainly focus on the MAINTENANCE PLANNING PROBLEM, a special case of SELF-REGULATING PLANNING PROBLEM set in the domain of infrastructural maintenance, the insights and algorithms presented in this work can be generalised to other domains as well. The results obtained here apply broadly to any group decision-coordination setting in which the individual goals of agents do not align with the global goal, as long as objectives can be operationalised into rewards and payments. Examples of such settings can be found in abundance in planning literature, e.g. scheduling the loading and unloading of vessels in harbours [82], supply-chain optimisation with autonomous links [136] and taxi scheduling [260]. Many more examples can be found throughout the literature [28, 35, 36, 48, 56, 103, 148, 208, 215, 243, 261].

The techniques used in Chapter 3 to encode self-regulating problems as a multi-agent MDP or even a 'flat' MDP immediately provides access to a broad portfolio of already existing solving techniques, both exact as well as approximate. For problems that fit the self-regulation model this alleviates the need to design and develop problem-specific solvers. Not only that, it allows researchers and practitioners to rely upon on the state-of-the-art of decision-theoretic planning to optimise their problems; a field that is actively being pushed further every day. The ideas used in this chapter to encode and flatten MMDP may also serve as an inspiration for other problems – for example the compact encoding of histories in the states of a "memoryless" MDP (e.g. Markovian), the decomposition of activities into unit-time actions, or the structure of complex rewards – and provide new leads for research on MDP algorithms.

One of these leads, inspired by the way rewards are encoded in the MDP, is that of conditional reward independence which lead to the conception of the `Conditional Return Policy Search` solver of Chapter 4. The key insight is that in many decision-coordination problems, including but not limited to self-regulating planning problems, reward dependencies between agents are limited to a small number of interactions [26, 79, 170, 179, 196, 251, 264]. When these interactions can no longer occur, for instance because a particular action cannot be performed anymore, the reward functions can be decoupled and optimised independently. For MPP this particular idea enabled finding optimal solutions to instances that were previously deemed too hard to solve. Whether the same applies to other domains is an open question.

The extension to generic MMDPs, as outlined in the discussion of Chapter 4, introduces a multitude of new challenges to the solver that lead to new insights and, vice versa, may prompt many more positive results for problems that exhibit a similar conditional-reward structure [26, 40, 80, 103, 171]. An additional benefit of this extension is that, besides the `SPUDD` solver, `CoRe` can be compared to an array of state-of-the-art (M)MDP solvers, e.g. those proposed by Boutilier et al. [40], Dai [71], Oliehoek et al. [196], Plutowski [207] and Ruiz and Hernández [224]. Challenges for future research are to extend the algorithm to generic MMDP without substantially hampering performance and to determine the existence of characteristics that help classify problems as 'suitable' for `CoRe`.

The potential of `CoRe` itself can be increased by interleaving the action selection with the agent decoupling to better exploit independence, by implementing heuristics for action-selection to quickly obtain tight bounds, and by adding pruning techniques to rapidly remove sub-optimal paths from the reward structures. Furthermore, the concept of conditional reward graphs can be leveraged to develop an approximate algorithm for domains where efficiency is required. The upper and lower bounds on the rewards stored in these graphs can directly be employed to produce bounded-approximation algorithm. Additionally, grouping of sequences with $\epsilon$-equivalent rewards using arbitrary constraint validations as done by de Nijs et al. [190] or approaches such as those taken by Guestrin et al. [104], Koller and Parr [145] or Oliehoek et al. [194] can be used to approximate the reward function itself, leading to sparser interaction graphs. Finally, techniques like the factored upper bounds of Oliehoek et al. [195] may be used to approximate the return bounds and lead to faster pruning at the cost of solution quality.

A substantial contribution of this thesis is made to the domain of multi-objective planning. Although the optimality requirement of RQ 3 instigates a negative answer, significant positive results are obtained in the area of approximate MOMDP solving. Chapter 5 proposes two new approaches to produce approximate convex coverage sets that in many cases closely resemble optimal solutions while taking only a fraction of the time required by optimal solvers. The `Approximate Optimistic Linear Support` algorithm solves general MOMDPs with a bounded error on the quality of the solution while `SSII` is more tailored to exploit preliminary knowledge regarding the distribution of weights, e.g. when it is roughly known what trade-offs will be made or only parts of the decision problem are considered viable solutions. Both approaches target the generic multi-objective MDP model and can make use of any available ($\epsilon$-approximate) single-objective MDP solver to compute solutions. As a consequence, not only does this imply that any available state-of-the-art MDP solver can be capitalised on, both methods can directly be applied to a wide range of existing multi-objective planning problems [132, 139, 160, 175, 187, 215] or employed as part of decision-support systems [23, 61, 65, 82]. An interesting next step would be to harness the strengths of both algorithms, i.e. have the theoretical guarantees of `AOLS` combined with the focused improvements of `SSII`, although the how remains uncharted. Another next step is to employ recent techniques that re-use parts of the solution space when computing policies [133, 213], which is particularly interesting in this domain as many of the scalarised multi-objective MDPs are similar for a broad range of scalarisation weights.

Still, computational complexity remains a challenge for the implementation of self-regulation in real-world contract [29, 173]. Solving the underlying decision-coordination problems of the service providers in an optimal fashion can be argued unsuitable for realistic problem instances. In Chapter 3 it is demonstrated that approximate approaches produce coordination solutions of near-optimal quality within a fraction of the time. Additionally, the optimal approach assumes a model in which all uncertainties are known beforehand and do not change over time, an assumption that is unrealistic in many decision-making problems. As the planning horizon grows, the probability of the environment changing in unforeseen ways increases. Therefore an approximate or 'online' approach may be more effective and robust in coordinating service providers in long-term, performance-based contracts.

### 8.3.2 Incentive Design

On the topic of incentive design, and the enforcement thereof, the major contribution of this thesis is the toolkit of methods to deal with selfish behaviour in decision-coordination problems, presented in Section 8.2. The dynamic maintenance mechanism of Chapter 6 is a novel optimal mechanism that shows the potential of game theory and mechanism design in sequential decision-making problems such as MPP. In particular, although the dynamic maintenance mechanism is designed for MPP, the steps of its construction can be applied directly to any SELF-REGULATING PLANNING PROBLEM or in general any non-cooperative problem that can be modelled using the MDP framework, e.g. [26, 35, 48, 103, 143, 148, 173, 243]. In effect, this work guides the

design of mechanisms for stochastic planning problems that maximise the value of the planning process while simultaneously preventing strategic behaviour.

While the theoretical properties of the dynamic maintenance mechanism are appealing, its immediate practical application is still hampered by the computational effort it requires. The payments that ensure truthful reporting of agents demand optimal solutions to the underlying coordination problem, hence limiting its application to only those problems where the coordination can be determined efficiently or time is not of the essence. Expanding the scope of dynamic mechanisms such as the dynamic maintenance mechanism is an important next step; a step that seems very viable in light of related work on approximate mechanisms [46, 111, 147, 180].

Along the line of broadening the scope of mechanisms to deal with selfish behaviour, this thesis contributes a best-response coordination method that is significantly more efficient than dynamic mechanisms and preserves the autonomy of agents. Moreover, Chapter 6 demonstrates that for SELF-REGULATING PLANNING PROBLEM this approach always converges to a joint coordination that is a Nash equilibrium. In other words, a joint coordination is guaranteed to be produced by the method within a finite number of rounds for such problems. As a result, this approach can be applied directly to competitive settings in which agents are adversarial such as considered by Buzing et al. [48], Jonsson and Rovatsos [127], Van der Krogt et al. [148]. Best-response planning does not rely on optimal coordination. This enables the use of any approximation algorithm to compute local plans and means that it is not affected by the bounded-rationality typical to human decision makers [94, 130], which was also observed in Chapter 7.

A point of attention of best-response approach is its inability to provide any assurance on quality of the produced outcome. The procedure is certain to terminate in an equilibrium after a finite number of steps, however there are no guarantees on the rewards obtained in such a local optimum. A first idea could be to compute the optimal coordination, or at least an $\epsilon$-approximation for notoriously complex problems, to determine the "price of anarchy" [221], i.e. a bound on the optimality loss due to decentralisation of the decision problem. In addition, an empirical evaluation of the approach can establish an intuition with respect to typical outcomes. Furthermore, algorithmic techniques and heuristics can be employed to guide the coordination search and improve the optima found by the method [95, 120, 172, 178, 198]. Other ideas such as learning [68] and hybrid algorithms [240] can further strengthen this approach.

An aspect that is not covered in this thesis is the impact that malicious agents may have on the outcomes obtained under either one of the methods. The work here assumed that agents are selfish but not necessarily harmful to others; they simply do not care about the rewards obtained by others. Malicious agents, however, may actively seek to harm other agents and for this they may have justifiable reasons in a competitive setting [212]. For instance contractors may try to force competition out of the market. But also a less malevolent intention such as continuation of business can drive agents to make strategic decisions that are not rational or not included in models of their rationality function. Dealing with this type of behaviour is typically very hard in mechanisms because it either violates the underlying assumption of agent rationality or implies models of rationality that are too complex to compute [72, 86].

Furthermore, both the dynamic maintenance mechanism and the self-response counter strategic behaviour by single agents, how both approaches hold up against strategic coalitions is unclear. In parallel to the theoretical research into maliciousness [63, 182] and collusion [58, 59, 96, 151], the game of Chapter 7 could be used as a testing environment to get a first insight into the impacts of these types of strategic behaviour an investigate potential countermeasures, especially if automated agents are used to mimic the behaviour of any type of player.

### 8.3.3 Self-regulation in Contracts

Whereas the main focus of this thesis has been on algorithmic techniques for the design and enforcement of self-regulation incentives, and the consequential decision-coordination problem, the original motivation of this research is the introduction of self-regulation into innovative contract forms. The dynamic contracting procedure of Volker et al. [254] – but also other novel contract forms such performance-driven contracts [246] and Best Value contracts [239], and research similar to that of Bower et al. [41], Bresnen and Marshall [45], Rose and Manley [218], Turrini et al. [249] – directly benefit from the research in this thesis. A significant contribution to contracting is again the 'toolkit' proposed in the conclusion of Section 8.2 as it provides the machinery to implement monitoring, performance-based steering and self-regulation, as well as the equipment to effectively coordinate decisions between contract participants. Ideas from mechanism design and the solutions presented in Chapters 3 to 6 may be of inspiration to the construction of efficient payment mechanisms within performance-based agreements and these mechanisms can be empirically evaluated using the serious game of Chapter 7. Moreover, the gaming setup offers a sandbox environment for the empirical evaluation of the execution of contractual frameworks to further study agent behaviour, incentive design, coordination approaches, etc. without the risks and costs of real-world testing [9, 20, 27, 155, 225]. Finally, this thesis takes the first steps towards a practical implementation of self-regulation in real-world tenders.

The serious gaming experiment of Chapter 7 forms a first proof of the concept of self-regulation in a setting with human decision makers but at the same time shows that there is still a long way to go before its implementation in actual contracts. The experiments showed that monetary incentives are certainly capable of changing decision-making preferences but their effect may be contradictory to their design. Still, emergent coordination of interactions to improve team performance was observed in groups with a strong social cohesion and hence the serious gaming experiment provides initial evidence for the potential of monetary incentives to implement self-regulation in performance-based contracts. Moreover, the apparent influence of the relationships between contractors on the effectiveness of incentives confirms once more the importance of the social factor in partnerships – a conclusion drawn earlier by others such as Agranoff and McGuire [7], Bresnen and Marshall [45], Klijn et al. [140] and Volker et al. [255] – and the experiments newly show that social cohesion is at least positively correlated to the emergence of coordination as a consequence of monetary incentives. Hence, this suggests that incentives to incite self-regulation are most effective in settings where contractors are familiar and collaborative, such as strategic partnerships or

alliances [50, 114, 118, 163], and emphasises the importance of nurturing social relationships in group tenders [44, 76, 154, 218]. Whether social cohesion is a necessary condition for monetary incentive structures or merely beneficial to coordination for a particular set of mechanisms can not be concluded from the experiments. This open problem is one of the most important ones to be addressed in future research and determines the applicability of the incentive-based approach in domains and settings where contractors are not inherently collaborative. In a broader sense, it is currently unclear how the results of the gaming experiment generalise to real-world tenders and hence additional empirical studies are necessary to establish the potential of self-regulation in a (more) realistic setting. In this endeavour, the serious game of 7 can be refined to gradually better approximate realistic contract execution, but ultimately the proof of the pudding is in the eating.

Notwithstanding the need for a real-world proof, the failure of incentives to consistently achieve self-regulation in the controlled experiments demonstrates the precariousness of relying solely on incentives to optimise team performance. Carefully designed incentives that in theory optimise performance and counter opportunism may cause unexpected or undesirable behaviour of human decision makers and can lead to difficult questions with respect to the responsibility and accountability of agents. This is a strong concern to recognise when depending on incentive schemes to incite performance. If the contracted objectives are not specific enough, malicious service providers will seek to exploit the performance-based payments while still complying to the contracted goals. Such outcomes are difficult to prevent from a legal perspective as the service provider is not actually at fault. But even cooperative service providers may fail to produce the desired results due to incomplete, incorrect or imprecise goal and payment formulations. In its essence, the potential of self-regulation is only as strong as the design of its incentives. Naturally, the effectiveness of incentives is paired the willingness of service providers to adhere to these incentives and their ability to optimise their decisions in the light of these incentives. The lack of control and outsourcing of expertise makes it difficult to detect and prevent scenarios in which service providers ignore or misinterpret incentives. These cases may lead to costly failures, interventions or 'bail-outs'. In situations where trust is insufficient, uncertainties are great or objectives are complex, the traditional governed contract is often preferable as it offers tight control over the process and its outcomes. At any rate, the current understanding on team incentive structures, e.g. [45, 118, 185, 218], and the outcomes of the experiments of Chapter 7 together advice against relying completely on monetary incentives as the single mechanism to concurrently incite optimal contractor performance, discourage opportunistic behaviour and stimulate cooperation in group tenders. Instead, further research should consider pairing monetary incentives with additional measures to mitigate its potential pitfalls, such as past-performance [81, 84], trust [7, 140, 150, 255], social control [85], training and knowledge sharing [83, 249], governmental supervision [90, 185] or automated decision support [23, 61]

Important to consider for contract designers is that while self-regulation stimulates efficient and flexible outsourcing of expertise and work, there may be additional drawbacks to this approach besides the aforementioned lack of control. The cost of outsourcing is typically much higher than having in-house resources to perform work

while quality levels are not necessarily better [12]. Outsourcing may also lead to less knowledge and expertise, making it harder to effectively assess the quality of delivered services and design the right objectives in tenders. Furthermore, outsourcing degrades the learning ability of an organisation [146]. For these reasons insourcing is again preferred in several domains [13, 137]. Also, this thesis has only considered optimal performance in terms of maximising the total value obtained as a result of contracted work, regardless of the value to individual contractors. Other aspects such as fairness [267] may play a role in the success of contracts and alternative value-maximisation strategies may be more suitable in different setting, e.g. the distribution mechanism of Cavallo [53]. Certainly, before self-regulation can be implemented in contracts, further research is required into the legal aspects and implications of self-regulation [201, 211, 222]. Employing monetary incentives based on group performance may lead to difficult questions about shared responsibility, intentionally harming other contractors or who is at fault in case of performance penalties.

In conclusion, this thesis has taken the first step in bringing self-regulation into contracting frameworks by the introduction of a mathematical framework, contribution of several techniques to incentivise and coordinate service providers, and through a first empirical evaluation of the concept in a simulated environment. Nonetheless, many more steps need to be taken before self-regulation can be implemented in real-world contracts and they should be taken pairwise with other fields such as contracting theory, network (interaction) management, legal studies, behavioural psychology, and many others. Ultimately, guided by the tools, techniques and learnings contributed by this thesis, and paired with counselling from related disciplines, self-regulation has the potential to revolutionise the traditionally hierarchical service-delivery model into partnerships based on co-creation, respect and trust.

# Bibliography


[1] Rijksbegroting - 3. de agentschappen (Dutch only). Technical report, Rijkswaterstaat, 2013. Available at http://www.rijksbegroting.nl/2013/voorbereiding/begroting,kst173855_37.html [16 December 2019].

[2] Oxford dictionary: self-regulation, 2015. Available at https://www.oxfordlearnersdictionaries.com/definition/english/self-regulation?q=self-regulation [03 April 2019].

[3] Verkeersintensiteit; rijkswegen (Dutch only). Technical report, Centraal Bureau voor de Statistiek, 2018. Available at https://opendata.cbs.nl/statline/#/CBS/nl/dataset/82855NED/table?ts=1519907724922 [16 December 2019].

[4] Motor vehicles; type, age class, 1 january. Technical report, Centraal Bureau voor de Statistiek, 2019. Available at https://opendata.cbs.nl/statline/#/CBS/en/dataset/82044ENG/table?ts=1576522293313 [16 December 2019].

[5] Traffic performance motor vehicles; kilometres, type of vehicle, territory. Technical report, Centraal Bureau voor de Statistiek, 2019. Available at https://opendata.cbs.nl/statline/#/CBS/en/dataset/80302ENG/table?ts=1576522182197 [16 December 2019].

[6] Rijksbegroting - agentschap rijkswaterstaat (Dutch only). Technical report, Rijkswaterstaat, 2019. Available at http://www.rijksbegroting.nl/2020/voorbereiding/begroting,kst264853_25.html [16 December 2019].

[7] R. Agranoff and M. McGuire. Big questions in public network management research. *Journal of Public Administration Research and Theory*, 11(3):295–326, 2001.

[8] M. Allen, M. Petrik, and S. Zilberstein. Interaction structure and dimensionality in decentralized problem solving. *Proc. of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1440–1441, 2008.

[9] M. Altamirano, P. Herder, and M. De Jong. Road roles using gaming simulation as decision technique for future asset management practices. In *Proc. of the International Conference on Systems, Man and Cybernetics*, pages 2297–2302, 2008.

[10] M. A. Altamirano. *Innovative contracting practices in the road sector: cross-national lessons in dealing with opportunistic behaviour*. PhD thesis, Delft University of Technology, 2010.

[11] M. A. Altamirano and W. M. de Jong. Opportunistic behavior in road maintenance markets. *Transportation Research Record: Journal of the Transportation Research Board*, 2108(1):13–22, 2009.

[12] J. S. Arlbjørn and T. Lüthje. Global operations and their interaction with supply chain performance. *Industrial Management & Data Systems*, 2012.

[13] J. S. Arlbjørn and O. S. Mikkelsen. Backshoring manufacturing: notes on an important but under-researched theme. *Journal of Purchasing and Supply Management*, 20(1):60–62, 2014.

[14] K. J. Arrow. *The property rights doctrine and demand revelation under incomplete information*. Elsevier, 1979.

[15] C. d'Aspremont and L.-A. Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11(1):25–45, 1979.

[16] S. Athey and I. Segal. An efficient dynamic mechanism. *Econometrica*, 81(6): 2463–2485, 2013.

[17] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[18] R. Axelrod. Advancing the art of simulation in the social sciences. In *Japanese Journal for Management Information System, Special Issue on Agent-Based Modeling*, volume 12, pages 1–19, 12 2003.

[19] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebric decision diagrams and their applications. *Formal Methods in System Design*, 10(2-3):171–206, 1997.

[20] O. Barreteau, F. Bousquet, and J.-M. Attonaty. Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems. *Journal of Artificial Societies and Social Simulation*, 4(2):5, 2001.

[21] L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proc. of the 25th International Conference on Machine Learning*, pages 41–47. ACM, 2008.

[22] F. H. Barron and B. E. Barrett. Decision quality using ranked attribute weights. *Management science*, 42(11):1515–1523, 1996.

[23] M. Le Bars and P. Le Grusse. Use of a decision support system and a simulation game to help collective decision-making in water management. *Computers and Electronics in Agriculture*, 62(2):182–189, 2008.

[24] J. Bates. Values of time and reliability in passenger and freight transport in The Netherlands. Technical Report Project 08064, Significance,Vrije Universiteit Amsterdam, November 2012.

[25] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 302–309. IEEE Computer Society, 2004.

[26] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[27] G. Bekebrede. *Experiencing complexity: a gaming approach for understanding infrastructure systems*. Delft University of Technology, 2010.

[28] R. Bellman. A Markovian decision process. Technical report, DTIC Document, 1957.

[29] D. Bergemann and M. Said. Dynamic auctions. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[30] D. Bergemann and J. Välimäki. *Efficient dynamic auctions*. Yale University, Cowles Foundation for Research in Economics, 2006.

[31] D. Bernstein and R. Givan. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 2002.

[32] A. Beynier and A. Mouaddib. A polynomial algorithm for decentralized Markov decision processes with temporal constraints. In *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 963–969. ACM, 2005.

[33] A. Beynier and A. Mouaddib. An iterative algorithm for solving constrained decentralized Markov decision processes. *Proc. of the National Conference on Artificial Intelligence*, 2006.

[34] M. Blencowe. Review of Scottish public sector procurement in construction. Technical report, Scottish Futures Trust, January 2016. Available at `https://www.scottishfuturestrust.org.uk/storage/uploads/Target_Cost_Guidance.pdf` [28 April 2019].

[35] P. Bogetoft. *Non-cooperative planning theory*, volume 418. Springer Science & Business Media, 2012.

[36] C. Boutilier. Planning, learning and coordination in multiagent decision processes. *Proc. of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

[37] C. Boutilier and N. Friedman. Context-specific independence in Bayesian networks. *Proc. of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, 1996.

[38] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proc. of the International Joint Conference on Artificial Intelligence*, volume 14, pages 1104–1113, 1995.

[39] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1—-94, 1999.

[40] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 2000.

[41] D. Bower, G. Ashby, K. Gerald, and W. Smyk. Incentive mechanisms for project success. *Journal of Management in Engineering*, 18(1):37–43, 2002.

[42] M. H. Brandt. Kwartaalmonitor bereikbaarheidsontwikkeling hoofdwegennet 1e kwartaal 2011 (Dutch only). Technical report, Ministerie van Infrastructuur en Milieu Rijkswaterstaat, Dienst Verkeer en Scheepvaart, April 2011. Available at `http://publicaties.minienm.nl/documenten/kwartaalmonitor-bereikbaarheidsontwikkeling-hoofdwegennet-1e-k-2` [28 April 2019].

[43] M. H. Brandt, G. Loos, and S. van Houten. Fileminuten als gunningscriterium (Dutch only). Technical report, Cobouw, July 2011. Available at `https://www.cobouw.nl/infra/nieuws/2011/07/fileminuten-als-gunningscriterium-10122517` [28 April 2019].

[44] M. Bresnen and N. Marshall. Partnering strategies and organizational cultures in the construction industry. In *Proceedings, ARCOM 14th Annual Conference, University of Reading*, volume 2, pages 465–76, 1998.

[45] M. Bresnen and N. Marshall. Motivation, commitment and the use of incentives in partnerships and alliances. *Construction Management and Economics*, 18(5): 587–598, 2000.

[46] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *Proc. of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pages 39–48. ACM, 2005.

[47] E. M. Van Bueren, E.-H. Klijn, and J. F. Koppenjan. Dealing with wicked problems in networks: analyzing an environmental debate from a network perspective. *Journal of Public Administration Research and Theory*, 13(2):193–212, 2003.

[48] P. Buzing, A. Ter Mors, J. Valk, and C. Witteveen. Coordinating self-interested planning agents. *Autonomous Agents and Multi-Agent Systems*, 12(2):199–218, 2006.

[49] A. Calderón and M. Ruiz. A systematic literature review on serious games evaluation: an application to software project management. *Computers & Education*, 87:396–422, 2015.

[50] D. G. Carmichael. *Contracts and international project management*. CRC Press, 2000.

[51] A. R. Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes*. PhD thesis, Brown University, 1998.

[52] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proc. of the National Conference on Artificial Intelligence*, volume 94, pages 1023–1028, 1994.

[53] R. Cavallo. Efficiency and redistribution in dynamic mechanism design. In *Proc. of the 9th ACM Conference on Electronic Commerce*, pages 220–229. ACM, 2008.

[54] R. Cavallo. *Social welfare maximization in dynamic strategic decision problems*. PhD thesis, Harvard University, 2008.

[55] R. Cavallo, D. C. Parkes, and S. Singh. Optimal coordinated planning amongst self-interested agents with private state. In *Proc. of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 55–62. AUAI Press, 2006.

[56] R. Cavallo, D. C. Parkes, and S. Singh. Efficient mechanisms with dynamic populations and dynamic types. Technical report, Harvard University, Division of Engineering and Applied Physics, 2009.

[57] A. P. Chan, D. W. Chan, and J. F. Yeung. *Relational contracting for construction excellence: principles, practices and case studies*. Routledge, 2009.

[58] J. Chan and W. Zhang. Collusion enforcement with private information and private monitoring. *Journal of Economic Theory*, 157:188–211, 2015.

[59] Y.-K. Che and J. Kim. Optimal collusion-proof auctions. *Journal of Economic Theory*, 144(2):565–603, 2009.

[60] H.-T. Cheng. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, 1988.

[61] W. Cheung, L. C. Leung, and P. C. Tam. An intelligent decision support system for service network planning. *Decision Support Systems*, 39(3):415–428, 2005.

[62] K. Choi, Y. H. Kwak, J.-H. Pyeon, and K. Son. Schedule effectiveness of alternative contracting strategies for transportation infrastructure improvement projects. *Journal of Construction Engineering and Management*, 138(3):323–330, 2011.

[63] A. K. Chorppath and T. Alpcan. Adversarial behavior in network mechanism design. In *Proc. of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, pages 506–514, 2011.

[64] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[65] R. T. Clemen and T. Reilly. *Making hard decisions with DecisionTools*. Cengage Learning, 2013.

[66] J. Cohen. Statistical power analysis. *Current directions in psychological science*, 1(3):98–101, 1992.

[67] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proc. of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 103–110. Morgan Kaufmann Publishers Inc., 2002.

[68] V. Conitzer and T. Sandholm. AWESOME: a general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

[69] D. A. Conrad, S. H. Cave, M. Lucas, J. Harville, S. M. Shortell, G. J. Bazzoli, R. Hasnain-Wynia, S. Sofaer, J. A. Alexander, E. Casey, et al. Community care networks: linking vision to outcomes for community health improvement. *Medical Care Research and Review*, 60(4_suppl):95S–129S, 2003.

[70] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo Tree Search. In *Proc. of the 5th international conference on Computers and games*, pages 72–83. Springer, 2006.

[71] P. Dai. *Decision making under uncertainty: scalability and applications*. University of Washington, 2011.

[72] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-mechanism design: a call to arms. *Intelligent Systems, IEEE*, 18(6):40–47, 2003.

[73] A. Davies, S. MacAulay, and T. Brady. Delivery model innovation: insights from infrastructure projects. *Project Management Journal*, 50(2):119–127, 2019.

[74] H. A. Davies and E. K. Chan. Experience of energy performance contracting in Hong Kong. *Facilities*, 19(7/8):261–268, 2001.

[75] H. Ç. Demirel, W. Leendertse, L. Volker, and M. Hertogh. Flexibility in PPP contracts–dealing with potential change in the pre-contract phase of a construction project. *Construction Management and Economics*, 35(4):196–206, 2017.

[76] G. Dewulf and A. Kadefors. Collaboration in public construction-contractual incentives, partnering schemes and trust. *Engineering Project Organization Journal*, 2(4):240–250, 2012.

[77] J. S. Dibangoye, C. Amato, and A. Doniec. Scaling up decentralized MDPs through heuristic search. In *Proc. of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012.

[78] J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. In *Proc. of the Twenty-Third international Joint Conference on Artificial Intelligence*. AAAI Press, 2013.

[79] J. S. Dibangoyea, C. Amatob, O. Buffeta, and F. Charpilleta. Exploiting separability in multiagent planning with continuous-state MDPs. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1281–1288. IFAAMAS, 2014.

[80] D. Dolgov and E. Durfee. Graphical models in local, asymmetric multi-agent Markov decision processes. In *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 956–963. IEEE Computer Society, 2004.

[81] H. Doloi, K. Iyer, and A. Sawhney. Structural equation model for assessing impacts of contractor's performance on project success. *International Journal of Project Management*, 29(6):687–695, 2011.

[82] A. M. Douma, J. van Hillegersberg, and P. C. Schuur. Design and evaluation of a simulation game to introduce a multi-agent system for barge handling in a seaport. *Decision Support Systems*, 53(3):465–472, 2012.

[83] P. E. Eriksson. Partnering: what is it, when should it be used, and how should it be implemented? *Construction management and economics*, 28(9):905–917, 2010.

[84] P. E. Eriksson and M. Westerberg. Effects of cooperative procurement procedures on construction project performance: a conceptual framework. *International Journal of Project Management*, 29(2):197–208, 2011.

[85] E. Fehr and A. Falk. Psychological foundations of incentives. *European Economic Review*, 46(4-5):687–724, 2002.

[86] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: recent results and future directions. In *Current Trends in Theoretical Computer Science: The Challenge of the New Century*, pages 403–434. World Scientific, 2004.

[87] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1972.

[88] M. S. Fox, M. Barbuceanu, and R. Teigen. Agent-oriented supply-chain management. In *Information-Based Manufacturing*, pages 81–104. Springer, 2001.

[89] E. J. Friedman and D. C. Parkes. Pricing wifi at starbucks: issues in online mechanism design. In *Proc. of the 4th ACM Conference on Electronic Commerce*, pages 240–241. ACM, 2003.

[90] R. Gao and J. Liu. Selection of government supervision mode of PPP projects during the operation stage. *Construction Management and Economics*, pages 1–20, 2019.

[91] E. H. Gerding, V. Robu, S. Stein, D. C. Parkes, A. Rogers, and N. R. Jennings. Online mechanism design for electric vehicle charging. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 811–818. IFAAMAS, 2011.

[92] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41 (4):587–601, 1973.

[93] R. Gibbons. *A primer in game theory*. Harvester Wheatsheaf, 1992.

[94] G. Gigerenzer and R. Selten. *Bounded rationality: the adaptive toolbox*. MIT Press, 2002.

[95] F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[96] A. V. Goldberg and J. D. Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proc. of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 620–629. Society for Industrial and Applied Mathematics, 2005.

[97] C. Goldman and S. Zilberstein. Decentralized control of cooperative systems: categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 2004.

[98] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proc. of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144. ACM, 2003.

[99] J. J. Green, Jand Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica: Journal of the Econometric Society*, pages 427–438, 1977.

[100] A. Griffith. Delivering best value in the small works portfolio of public sector organizations when using preferred contractors. *Construction Management and Economics*, 29(9):891–900, 2011.

[101] T. Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, pages 617–631, 1973.

[102] C. Guestrin and G. Gordon. Distributed planning in hierarchical factored MDPs. In *Proc. of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 197–206. Morgan Kaufmann Publishers Inc., 2002.

[103] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. *Advances in Neural Information Processing Systems*, 2001.

[104] C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored MDPs. In *Proc. of the National Conference on Artificial Intelligence*, pages 253–259, 2002.

[105] A. Guo and V. R. Lesser. Planning for weakly-coupled partially observable stochastic games. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1715–1716, 2005.

[106] D. Gupta, A. Vedantam, and J. Azadivar. Optimal contract mechanism design for performance-based contracts. Technical report, Minnesota Department of Transportation Research Services Section, 2011.

[107] P. Guyot and S. Honiden. Agent-based participatory simulations: merging multi-agent systems and role-playing games. *Journal of Artificial Societies and Social Simulation*, 9(4), 10 2006.

[108] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proc. of the National Conference on Artificial Intelligence*, volume 4, pages 709–715, 2004.

[109] M. O'Hare, R. Leone, and M. Zegans. Privatization of imprisonment: a managerial perspective. In *Private Prisons and the Public Interest*, pages 107–129. Rutgers University Press, 1990.

[110] C. Harteveld, R. Guimarães, I. S. Mayer, and R. Bidarra. Balancing play, meaning and reality: the design philosophy of LEVEE PATROLLER. *Simulation & Gaming*, 41(3):316–340, 2010.

[111] J. D. Hartline. Approximation in mechanism design. *The American Economic Review*, pages 330–336, 2012.

[112] A. Hartmann and M. Hietbrink. An exploratory study on the relationship between stakeholder expectations, experiences and satisfaction in road maintenance. *Construction Management and Economics*, 31(4):345–358, 2013.

[113] A. Hartmann, J. Roehrich, L. Frederiksen, and A. Davies. Procuring complex performance: the transition process in public infrastructure. *International journal of operations & production management*, 34(2):174–194, 2014.

[114] A. J. Hauck, D. H. Walker, K. D. Hampson, and R. J. Peters. Project alliancing at national museum of Australia: collaborative process. *Journal of Construction Engineering and Management*, 130(1):143–152, 2004.

[115] S. Hedborg Bengtsson. Coordinated construction logistics: an innovation perspective. *Construction Management and Economics*, 37(5):294–307, 2019.

[116] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: stochastic planning using decision diagrams. *Proc. of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[117] S. Hong, C. Wernz, and J. D. Stillinger. Optimizing maintenance service contracts through mechanism design theory. *Applied Mathematical Modelling*, 40 (21-22):8849–8861, 2016.

[118] S. M. Hosseinian and D. G. Carmichael. Optimal gainshare/painshare in alliance projects. *Journal of the Operational Research Society*, 64(8):1269–1278, 2013.

[119] R. A. Howard. *Dynamic Programming and Markov Processes*. John Wiley, 1970.

[120] X. Hu, R. Shonkwiler, and M. C. Spruill. Random restarts in global optimization. Technical report, Georgia Institute of Technology, 2009.

[121] W. Hughes and S. Kabiri. Performance-based contracting in the construction sector. Technical report, University of Reading, 2013.

[122] P. Hypko, M. Tilebein, and R. Gleich. Clarifying the concept of performance-based contracting in manufacturing industries: a research synthesis. *Journal of Service Management*, 21(5):625–655, 2010.

[123] H. B. Isik, B. Sohngen, et al. Performance-based voluntary group contracts for nonpoint source pollution. In *2003 Annual meeting, July 27-30, Montreal, Canada*, number 22064. American Agricultural Economics Association, 2003.

[124] A. A. Javed, P. T. Lam, and A. P. Chan. Change negotiation in public-private partnership projects through output specifications: an experimental approach based on game theory. *Construction Management and Economics*, 32(4):323–348, 2014.

[125] M. C. Jensen and W. H. Meckling. *Theory of the firm: managerial behavior, agency costs, and ownership structure*. Springer, 1979.

[126] A. Jonsson and A. Barto. A causal approach to hierarchical decomposition of factored MDPs. *Proc. of the 22nd International Conference on Machine Learning*, 2005.

[127] A. Jonsson and M. Rovatsos. Scaling up multiagent planning: a best-response approach. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2011.

[128] J. Jordán, A. Torreño, M. M. de Weerdt, and E. Onaindia. A better-response strategy for self-interested planning agents. *Applied Intelligence: the international journal of artificial intelligence, neural networks, and complex problem-solving technologies*, 48(4):1020–1040, 2018. doi: 10.1007/s10489-017-1046-5.

[129] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[130] D. Kahneman. A perspective on judgment and choice: mapping bounded rationality. *American Psychologist*, 58(9):697, 2003.

[131] S. M. Kakade, I. Lobel, and H. Nazerzadeh. Optimal dynamic mechanism design and the virtual-pivot mechanism. *Operations Research*, 61(4):837–854, 2013.

[132] S. Kalyanasundaram, E. K. Chong, and N. B. Shroff. Optimal resource allocation in multi-class networks with user-specified utility functions. *Computer Networks*, 38(5):613–630, 2002.

[133] T. Keller and P. Eyerich. PROST: probabilistic planning based on UCT. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2012.

[134] T. Keller and M. Helmert. Trial-based heuristic tree search for finite horizon MDPs. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2013.

[135] R. Kenley, K. London, and J. Watson. Strategic procurement in the construction industry mechanisms for public sector clients to improve performance in the australian public sector. *Journal of Construction Procurement*, 6(1):4–19, 2000.

[136] C. Kilger, B. Reuter, and H. Stadtler. Collaborative planning. In *Supply Chain Management and Advanced Planning*, pages 257–277. Springer, 2015.

[137] S. Kinkel. Future and impact of backshoring - Some conclusions from 15 years of research on german practices. *Journal of Purchasing and Supply Management*, 20(1):63–65, 2014.

[138] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 6, pages 739–743. IEEE, 1999.

[139] L. Klein, J.-Y. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham, and M. Tambe. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in Construction*, 22:525–536, 2012.

[140] E.-H. Klijn, J. Edelenbos, and B. Steijn. Trust in governance networks its impacts on outcomes. *Administration & Society*, 42(2):193–221, 2010.

[141] E.-H. Klijn, B. Steijn, and J. Edelenbos. The impact of network management on outcomes in governance networks. *Public Administration*, 88(4):1063–1082, 2010.

[142] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[143] J. R. Kok and N. Vlassis. Sparse cooperative Q-learning. In *Proc. of the International Conference on Machine Learning*, pages 481–488, 2004.

[144] C. J. Koliba, J. W. Meek, and A. Zia. Gordian knot or integrated theory? Critical conceptual considerations for governance network analysis. *The Future of Governance*, 2010.

[145] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, volume 99, pages 1332–1339, 1999.

[146] M. Kotabe, M. J. Mol, and J. Y. Murray. Outsourcing, performance, and the role of e-commerce: a dynamic perspective. *Industrial Marketing Management*, 37(1):37–45, 2008.

[147] V. Krishna and M. Perry. Efficient mechanism design. Technical report, Pennsylvania State University, 1998. Available at SSRN 64934.

[148] R. Van der Krogt, M. M. De Weerdt, and Y. Zhang. Of mechanism design and multiagent planning. In *ECAI*, pages 423–427, 2008.

[149] P. Krysta and B. Vöcking. Online mechanism design (randomized rounding on the fly). In *Automata, Languages, and Programming*, pages 636–647. Springer, 2012.

[150] A. Laan, N. Noorderhaven, H. Voordijk, and G. Dewulf. Building trust in construction partnering projects: an exploratory case-study. *Journal of Purchasing and Supply Management*, 17(2):98–108, 2011.

[151] J.-J. Laffont and D. Martimort. Mechanism design with collusion and correlation. *Econometrica*, 68(2):309–342, 2000.

[152] T. Lam and K. Gale. Highway maintenance: impact of framework agreements upon project financial performance. *Construction Management and Economics*, 32(5):460–472, 2014.

[153] T. C. Lam and K. A. Small. The value of time and reliability: measurement from a value pricing experiment. *Transportation Research Part E: Logistics and Transportation Review*, 37(2):231–251, 2001.

[154] E. Larson. Partnering on construction projects: a study of the relationship between partnering activities and project success. *IEEE Transactions on engineering management*, 44(2):188–195, 1997.

[155] S. Lavy, J. A. Garcia, P. Scinto, and M. K. Dixit. Key performance indicators for facility performance assessment: simulation of core indicators. *Construction Management and Economics*, 32(12):1183–1204, 2014.

[156] R. Leiringer, S. D. Green, and J. Z. Raja. Living up to the value agenda: the empirical realities of through-life value creation in construction. *Construction Management and Economics*, 27(3):271–285, 2009.

[157] J. Levin. Relational incentive contracts. *American Economic Review*, 93(3): 835–857, 2003.

[158] R. Li and R. Roberti. Optimal scheduling of railway track possessions in large-scale projects with multiple construction works. *Journal of Construction Engineering and Management*, 143(6):04017007, 2017.

[159] X. Li, T. Geng, Y. Yang, and X. Xu. Multiagent AGVs dispatching system using multilevel decisions method. In *Proc. of the American Control Conference*, volume 2, pages 1135–1136. IEEE, 2002.

[160] G. J. Lim and S. S. Desai. Markov decision process approach for multiple objective hazardous material transportation route selection problem. *International Journal of Operational Research*, 7(4):506–529, 2010.

[161] D. J. Lizotte, M. H. Bowling, and S. A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In *Proc. of the 27th International Conference on Machine Learning*, pages 695–702, 2010.

[162] D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-q iteration with multiple reward functions. *The Journal of Machine Learning Research*, 13(1): 3253–3295, 2012.

[163] P. E. Love, P. R. Davis, R. Chevis, and D. J. Edwards. Risk/reward compensation model for civil engineering infrastructure alliance projects. *Journal of Construction Engineering and Management*, 137(2):127–136, 2011.

[164] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.

[165] J. Marecki and M. Tambe. On opportunistic techniques for solving decentralized Markov decision processes with temporal constraints. In *Proc. of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 219. ACM, 2007.

[166] J. Marecki and M. Tambe. Towards faster planning with continuous resources in stochastic domains. In *Proc. of the National Conference on Artificial Intelligence*, pages 1049–1055, 2008.

[167] J. Marecki and M. Tambe. Planning with continuous resources for agent teams. In *Proc. of The 8th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 1089–1096. IFAAMAS, 2009.

[168] E. Maskin. Nash equilibrium and welfare optimality. *The Review of Economic Studies*, 66(1):23–38, 1999.

[169] S. A. Meijer, I. S. Mayer, J. van Luipen, and N. Weitenberg. Gaming rail cargo management: exploring and validating alternative modes of organization. *Simulation & Gaming*, 43(1):85–101, 2 2012.

[170] F. Melo and M. Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 2011.

[171] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. L. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proc. of the Fifteenth National Conference on Artificial Intelligence*, pages 165–172, 1998.

[172] V. Mirrokni, N. Thain, and A. Vetta. A theoretical examination of practical game playing: lookahead search. In *Algorithmic Game Theory*, pages 251–262. Springer, 2012.

[173] V. Mirrokni, R. Paes Leme, R. Ren, and S. Zuo. Dynamic mechanism design in the field. In *Proc. of the 2018 World Wide Web Conference*, pages 1359–1368. International World Wide Web Conferences Steering Committee, 2018.

[174] J. P. Mo. System support engineering: the foundation knowledge for performance based contracting. In *ICOMS 2009: asset Management Conference Proceedings: Sydney, 1-5 June 2009*, page 205. Asset Management Council, 2009.

[175] K. Van Moffaert, M. M. Drugan, and A. Nowé. Scalarized multi-objective reinforcement learning: novel design techniques. In *Symposium on Adaptive Dynamic Programming And Reinforcement Learning*, pages 191–199. IEEE, 2013.

[176] G. E. Monahan. State of the art – a survey of partially observable Markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[177] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[178] B. Monien, D. Dumrauf, and T. Tscheuschner. Local search: simple, successful, but sometimes sluggish. In *International Colloquium on Automata, Languages, and Programming*, pages 1–17. Springer, 2010.

[179] H. Mostafa and V. Lesser. Offline planning for communication by exploiting structured interactions in decentralized MDPs. *Web Intelligence and Intelligent Agent Technologies*, 2009.

[180] A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, 64(2):612–631, 2008.

[181] R. B. Myerson. *Game Theory*. Harvard University Press, 2013.

[182] P. Naghizadeh and A. Sinha. Adversarial contract design for private data commercialization. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 681–699, 2019.

[183] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: towards efficient policy computation for multiagent settings. *Proc. of the International Joint Conference on Artificial Intelligence*, 2003.

[184] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: a synthesis of distributed constraint optimization and POMDPs. *Proc. of the National Conference on Artificial Intelligence*, 2005.

[185] H. R. Nalbantian and A. Schotter. Productivity under group incentives: an experimental study. *The American Economic Review*, pages 314–341, 1997.

[186] J. F. Nash et al. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36(1):48–49, 1950.

[187] T. A. Nguyen, M. Do, A. E. Gerevini, I. Serina, B. Srivastava, and S. Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 190:1–31, 2012.

[188] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[189] F. de Nijs, M. T. J. Spaan, and M. M. de Weerdt. Decoupling a resource constraint through fictitious play in multi-agent sequential decision making. In *Proceedings - 22nd European Conference on Artificial Intelligence, ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1724–1725, Netherlands, 2016. IOS Press.

[190] F. de Nijs, E. Walraven, M. M. de Weerdt, and M. T. J. Spaan. Bounding the probability of resource constraint violations in multi-agent MDPs. In *Proceedings of the 31st Conference on Artificial Intelligence, AAAI 2017*, pages 3562–3568, United States, 2017. American Association for Artificial Intelligence (AAAI).

[191] F. de Nijs, M. M. de Weerdt, and M. T. J. Spaan. Multi-agent planning under uncertainty for capacity management. In *Intelligent Integrated Energy Systems*, pages 197–213. Springer, 2019.

[192] A. Noroozian, M. M. de Weerdt, and C. Witteveen. Incentivizing cooperation in P2P file sharing: indirect interaction as an incentive to seed. In L. Cao, Y. Zeng, A. Symeonidis, V. Gorodetsky, P. Yu, and M. Singh, editors, *Proceedings - 8th International Workshop on Agents and Data Mining Interaction*, pages 36–50. Springer, 2013. doi: 10.1007/978-3-642-36288-0_5. Harvest Book Part II.

[193] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proc. of the 7th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 517–524. IFAAMAS, 2008.

[194] F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Approximate solutions for factored Dec-POMDPs with many agents. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 563–570, 2013.

[195] F. A. Oliehoek, M. T. J. Spaan, and S. J. Witwicki. Factored upper bounds for multiagent planning problems under uncertainty with non-factored value functions. In *IJCAI 2015 - Proceedings of the 24th International Joint Conference on Artificial Intelligence*, volume 2015-January, pages 1645–1651. International Joint Conferences on Artificial Intelligence, 1 2015.

[196] F. A. Oliehoek, M. T. J. Spaan, B. Terwijn, P. Robbel, and J. V. Messias. The MADP toolbox: an open source library for planning and learning in (multi-) agent systems. *The Journal of Machine Learning Research*, 18(1):3112–3116, 2017.

[197] A. Opdyke, F. Lepropre, A. Javernick-Will, and M. Koschmann. Inter-organizational resource coordination in post-disaster infrastructure recovery. *Construction Management and Economics*, 35(8-9):514–530, 2017.

[198] A. V. Orlov, A. S. Strekalovsky, and S. Batbileg. On computational search for Nash equilibrium in hexamatrix games. *Optimization Letters*, 10(2):369–381, 2016.

[199] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT press, 1994.

[200] S. Osborne. *Public-private partnerships: theory and practice in international perspective*. Routledge, 2002.

[201] M. Painter and J. Pierre. Why legality cannot be contracted out: exploring the limits of new public management. In *Reasserting the Public in Public Services*, pages 61–74. Routledge, 2010.

[202] C. Papadimitriou, G. Pierrakos, C.-A. Psomas, and A. Rubinstein. On the complexity of dynamic mechanism design. In *Proc. of the Twentyseventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1458–1475. SIAM, 2016.

[203] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[204] D. C. Parkes, R. Cavallo, F. Constantin, and S. Singh. Dynamic incentive mechanisms. *AI Magazine*, 31(4):79–94, 2010.

[205] A. Parkhe. Strategic alliance structuring: a game theoretic and transaction cost examination of interfirm cooperation. *Academy of Management Journal*, 36(4): 794–829, 1993.

[206] P. A. Pathak. The mechanism design approach to student assignment. *Annual Review of Economics*, 3(1):513–536, 2011.

[207] M. Plutowski. MDP solver for a class of location-based decisioning tasks. In *Proc. of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2003.

[208] M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley, 2009.

[209] M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

[210] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein. Non-approximability of decentralized control. Technical report, Leibniz Center for Computer Science, 2002.

[211] M. Raskin. The law and legality of smart contracts. Technical report, Georgetown Law Technology, 2016.

[212] M. Rehák, M. Pěchouček, and J. Tožička. Adversarial behavior in multi-agent systems. In *International Central and Eastern European Conference on Multi-Agent Systems*, pages 470–479. Springer, 2005.

[213] D. Roijers, E. Walraven, and M. T. J. Spaan. Bootstrapping LPs in value iteration for multi-objective and partially observable MDPs. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling*, pages 218–226. Association for the Advancement of Artificial Intelligence (AAAI), 2018.

[214] D. M. Roijers. *Multi-Objective Decision-Theoretic Planning*. PhD thesis, University of Amsterdam, 2016.

[215] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[216] D. M. Roijers, J. Scharpff, M. T. J. Spaan, F. A. Oliehoek, M. M. de Weerdt, and S. Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In *Proc. of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[217] D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Linear support for multi-objective coordination graphs. In *AAMAS 2014: Proc. of the Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, pages 1297–1304. IFAAMAS, 2014.

[218] T. Rose and K. Manley. Motivation toward financial incentive goals on construction projects. *Journal of Business Research*, 64(7):765–773, 2011.

[219] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[220] E. Roszkowska. Rank ordering criteria weighting methods–a comparative overview. Technical report, Wydawnictwo Uniwersytetu w Białymstoku, 2013.

[221] T. Roughgarden. *Selfish routing and the price of anarchy*, volume 174. MIT Press Cambridge, 2005.

[222] J. Rouse. Performance management, quality management and contracts. In *Public Management in Britain*, pages 76–93. Springer, 1999.

[223] D. Rousis. *A Pareto frontier intersection-based approach for efficient multi-objective optimization of competing concept alternatives*. PhD thesis, Georgia Institute of Technology, 2011.

[224] S. Ruiz and B. Hernández. A parallel solver for Markov decision process in crowd simulations. In *Proc. of the Fourteenth Mexican International Conference on Artificial Intelligence*, pages 107–116. IEEE, 2015.

[225] T. Ryan. The role of simulation gaming in policy-making. *Systems Research and Behavioral Science*, 17(4):359, 2000.

[226] S. Sanner. Relational dynamic influence diagram language (RDDL): language description. Technical report, NICTA and the Australian National University, 2010. Available at `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf` [3 May 2019].

[227] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[228] J. Scharpff, M. T. J. Spaan, M. M. de Weerdt, and L. Volker. Planning under uncertainty for coordinating infrastructural maintenance. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2013.

[229] J. Scharpff, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. M. de Weerdt. Solving transition-independent multi-agent MDPs with sparse interactions. In *Proc. of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[230] J. Scharpff, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. M. de Weerdt. Solving transition-independent multi-agent MDPs with sparse interactions. *arXiv preprint arXiv:1511.09047*, 2016. (Extended version).

[231] J. Scharpff, D. Schraven, L. Volker, M. T. J. Spaan, and M. M. De Weerdt. The road maintenance planning game – game design and first results. Technical report, Delft University of Technology, 2019.

[232] J. Scharpff, D. Schraven, L. Volker, M. Spaan, and M. de Weerdt. Can multiple contractors self-regulate their joint service delivery? a serious gaming experiment on road maintenance planning. *Construction Management and Economics*, 2020. doi: 10.1080/01446193.2020.1806336. In publication.

[233] D. Schraven, A. Hartmann, and G. Dewulf. Effectiveness of infrastructure asset management: challenges for public agencies. *Built Environment Project and Asset Management*, 1(1):61–74, 2011.

[234] I. Segal and J. Toikka. Revenue equivalence, profit maximization, and transparency in dynamic mechanisms. Technical report, Stanford University, 2007.

[235] W. K. Shadid. A framework for managing organizations in complex environments. *Construction Management and Economics*, 36(4):182–202, 2018.

[236] H. Sharma, C. McIntyre, Z. Gao, and T.-H. Nguyen. Developing a traffic closure integrated linear schedule for highway rehabilitation projects. *Journal of Construction Engineering and Management*, 135(3):146–155, 2009.

[237] W. Shen, Z. Wang, and S. Zuo. Ex-post IR dynamic auctions with cost-per-action payments. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2076–2078. IFAAMAS, 2018.

[238] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2006.

[239] T. Snippert, W. Witteveen, H. Boes, and H. Voordijk. Barriers to realizing a stewardship relation between client and vendor: the best value approach. *Construction Management and Economics*, 33(7):569–586, 2015.

[240] Y. S. Son and R. Baldick. Hybrid coevolutionary programming for nash equilibrium search in games with local optima. *IEEE Transactions on Evolutionary Computation*, 8(4):305–315, 2004.

[241] E. J. Sondik. *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University, 1971.

[242] M. T. J. Spaan and F. Melo. Local interactions in decentralized multiagent planning under uncertainty. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 525–532. Citeseer, 2008.

[243] M. T. J. Spaan, C. Amato, and S. Zilberstein. Decision making in multiagent settings: team decision making. Online tutorial. Available at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.598&rep=rep1&type=pdf` [3 May 2019], 2011.

[244] M. T. J. Spaan, T. S. Veiga, and P. U. Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157–1185, 2015. doi: 10.1007/s10458-014-9279-8.

[245] S. Stein, E. Gerding, V. Robu, and N. R. Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 669–676. IFAAMAS, 2012.

[246] A. Straub. Competences of maintenance service suppliers servicing end-customers. *Construction Management and Economics*, 28(11):1187–1195, 2010.

[247] R. Taylor. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography*, 6(1):35–39, 1990.

[248] E. Triantaphyllou. *Multi-criteria decision making methods: a comparative study*, volume 44. Springer Science & Business Media, 2013.

[249] A. Turrini, D. Cristofoli, F. Frosini, and G. Nasi. Networking literature about determinants of network effectiveness. *Public Administration*, 88(2):528–550, 2010.

[250] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting loose a SPIDER on a network of POMDPs: generating quality guaranteed policies. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, page 218, New York, New York, USA, 2007. ACM, ACM Press.

[251] P. Varakantham, J. Kwak, M. Taylor, and J. Marecki. Exploiting coordination locales in distributed POMDPs via social model shaping. *Proc. of the International Conference on Automated Planning and Scheduling*, 2009.

[252] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[253] B. Viswanathan, V. V. Aggarwal, and K. P. K. Nair. Multiple criteria Markov decision processes. *TIMS Studies Management Science*, 6:263–272, 1977.

[254] L. Volker, J. Scharpff, and M. M. de Weerdt. Designing a dynamic network based approach for asset management activities. In *Proc. of the 28th Annual ARCOM Conference*, pages 655–664, 2012.

[255] L. Volker, M. Altamirano, P. Herder, and T. van der Lei. The impact of innovative contracting on asset management of public infrastructure networks. In *Engineering Asset Management*, pages 665–676. Springer, 2014.

[256] M. Voorneveld, P. Borm, F. Van Megen, S. Tijs, and G. Facchini. Congestion games and potentials reconsidered. *International Game Theory Review*, 1 (03n04):283–299, 1999.

[257] K. Wakuta and K. Togawa. Solution procedures for Markov decision processes. *Optimization: a Journal of Mathematical Programming and Operations Research*, 43(1):29–46, 1998.

[258] E. Walraven and M. T. J. Spaan. Planning under uncertainty with weighted state scenarios. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 912–921, 7 2015.

[259] E. Walraven and M. T. J. Spaan. Point-based value iteration for finite-horizon POMDPs. *The Journal of Artificial Intelligence Research*, 65:307–341, 2019.

[260] M. M. de Weerdt. *Plan Merging in Multi-Agent Systems*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2003.

[261] M. M. de Weerdt and B. Clement. Introduction to planning in multiagent systems. *Multiagent and Grid Systems*, 5(4):345–355, 2009.

[262] I. Wenzler and D. Chartier. Why do we bother with games and simulations: an organizational learning perspective. *Simulation & Gaming*, 30(3):375–384, 1999.

[263] C. C. White and K. W. Kim. Solution procedures for vector criterion Markov decision processes. *Large Scale Systems*, 1(4):129–140, 1980.

[264] S. Witwicki and E. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. *Proc. of the International Conference on Automated Planning and Scheduling*, 2010.

[265] J. Wu and E. H. Durfee. Mixed-integer linear programming for transition-independent decentralized MDPs. In *Proc. of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, page 1058. IFAAMAS, ACM Press, 2006.

[266] J. Yu, R. Buyya, and C. K. Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *First International Conference on e-Science and Grid Computing*. IEEE, 2005.

[267] T. R. Zenger and C. Marshall. Determinants of incentive intensity in group-based rewards. *Academy of Management Journal*, 43(2):149–163, 2000.

[268] G. Zietlow. Cutting costs and improving quality through performance-based road management and maintenance contracts-the Latin American and OECD experiences. *Senior Road Executives Programme, Restructuring Road Management, German Development Cooperation, Birmingham*, 2005.

[269] K. H. Zou, K. Tuncali, and S. G. Silverman. Correlation and simple linear regression. *Radiology*, 227(3):617–628, 2003.

# Publications and Supplementary Material

## Publications

J. Scharpff, M. T. J. Spaan, M. M. de Weerdt, and L. Volker. Planning under uncertainty for coordinating infrastructural maintenance. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2013.

J. Scharpff, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. M. de Weerdt. Solving transition-independent multi-agent MDPs with sparse interactions. In *Proc. of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

J. Scharpff, D. Schraven, L. Volker, M. Spaan, and M. de Weerdt. Can multiple contractors self-regulate their joint service delivery? a serious gaming experiment on road maintenance planning. *Construction Management and Economics*, 2020. doi: 10.1080/01446193.2020.1806336. In publication.

## Co-authored Publications

D. M. Roijers, J. Scharpff, M. T. J. Spaan, F. A. Oliehoek, M. M. de Weerdt, and S. Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In *Proc. of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

L. Volker, J. Scharpff, and M. M. de Weerdt. Designing a dynamic network based approach for asset management activities. In *Proc. of the 28th Annual ARCOM Conference*, pages 655–664, 2012.

## Non Peer-reviewed Publications

J. Scharpff, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. M. de Weerdt. Solving transition-independent multi-agent MDPs with sparse interactions. *arXiv preprint arXiv:1511.09047*, 2016. (Extended version).

J. Scharpff, D. Schraven, L. Volker, M. T. J. Spaan, and M. M. De Weerdt. The road

maintenance planning game – game design and first results. Technical report, Delft University of Technology, 2019. https://repository.tudelft.nl/islandora/object/uuid:15778996-6b67-42d7-a923-f31cd6ec0952.

## Source and Data disclosure

The source code of the toolkit that contains all the solvers used in the experiments of Chapters 3 to 5 can be found in the Stochastic Planning Toolkit at https://github.com/AlgTUDelft/SPTK.

For a stand-alone implementation of the Conditional Return Policy Search (CoRe) solver, see https://github.com/AlgTUDelft/core-solver.

The maintenance planning game and the results of the gaming sessions can be found at https://github.com/AlgTUDelft/road-maintenance-game.

# TRAIL Thesis Series

This work is part of the TRAIL Thesis Series, a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics. The following list contains the most recent dissertations in this series. For a complete overview of more than 250 titles see the TRAIL website: `www.rstrail.nl`.

Scharpff, J.C.D., *Collective Decision Making trough Self-regulation*, T2020/17, November 2020, TRAIL Thesis Series, the Netherlands

Guo, W., *Optimization of Synchromodal Matching Platforms under Uncertainties*, T2020/16, November 2020, TRAIL Thesis Series, the Netherlands

Narayan, J., *Design and Analysis of On-Demand Mobility Systems*, T2020/15, October 2020, TRAIL Thesis Series, the Netherlands

Gong, X., *Using Social Media to Characterise Crowds in City Events for Crowd Management*, T2020/14, September 2020, TRAIL Thesis Series, the Netherlands

Rijal, A., *Managing External Temporal Constraints in Manual Warehouses*, T2020/13, September 2020, TRAIL Thesis Series, the Netherlands

Alonso González, M.J., *Demand for Urban Pooled On-Demand Services: Attitudes, preferences and usage*, T2020/12, July 2020, TRAIL Thesis Series, the Netherlands

Alwosheel, A.S.A., *Trustworthy and Explainable Artificial Neural Networks for choice Behaviour Analysis*, T2020/11, July 2020, TRAIL Thesis Series, the Netherlands

Zeng, Q., *A New Composite Indicator of Company Performance Measurement from Economic and Environmental Perspectives for Motor Vehicle Manufacturers*, T2020/10, May 2020, TRAIL Thesis Series, the Netherlands

Mirzaei, M., *Advanced Storage and Retrieval Policies in Automated Warehouses*, T2020/9, April 2020, TRAIL Thesis Series, the Netherlands

Nordhoff, S., *User Acceptance of Automated Vehicles in Public Transport*, T2020/8, April 2020, TRAIL Thesis Series, the Netherlands

Winter, M.K.E., *Providing Public Transport by Self-Driving Vehicles: User preferences, fleet operation, and parking management*, T2020/7, April 2020, TRAIL Thesis Series, the Netherlands

Mullakkal-Babu, F.A., *Modelling Safety Impacts of Automated Driving Systems in Multi-Lane Traffic*, T2020/6, March 2020, TRAIL Thesis Series, the Netherlands

Krishnakumari, P.K., *Multiscale Pattern Recognition of Transport Network Dynamics and its Applications: A bird's eye view on transport*, T2020/5, February 2020, TRAIL Thesis Series, the Netherlands Wolbertus, Evaluating Electric Vehicle Charging Infrastructure Policies, T2020/4, February 2020, TRAIL Thesis Series, the Netherlands

Yap, M.D., Measuring, *Predicting and Controlling Disruption Impacts for Urban Public Transport*, T2020/3, February 2020, TRAIL Thesis Series, the Netherlands

Luo, D., *Data-driven Analysis and Modeling of Passenger Flows and Service Networks for Public Transport Systems*, T2020/2, February 2020, TRAIL Thesis Series, the Netherlands

Erp, P.B.C. van, *Relative Flow Data: New opportunities for traffic state estimation*, T2020/1, February 2020, TRAIL Thesis Series, the Netherlands

Zhu, Y., *Passenger-Oriented Timetable Rescheduling in Railway Disruption Management*, T2019/16, December 2019, TRAIL Thesis Series, the Netherlands

Chen, L., *Cooperative Multi-Vessel Systems for Waterborne Transport*, T2019/15, November 2019, TRAIL Thesis Series, the Netherlands

Kerkman, K.E., *Spatial Dependence in Travel Demand Models: Causes, implications, and solutions*, T2019/14, October 2019, TRAIL Thesis Series, the Netherlands

Liang, X., *Planning and Operation of Automated Taxi Systems*, T2019/13, September 2019, TRAIL Thesis Series, the Netherlands

Ton, D., *Unravelling Mode and Route Choice Behaviour of Active Mode Users*, T2019/12, September 2019, TRAIL Thesis Series, the Netherlands

Shu, Y., *Vessel Route Choice Model and Operational Model Based on Optimal Control*, T2019/11, September 2019, TRAIL Thesis Series, the Netherlands

Luan, X., *Traffic Management Optimization of Railway Networks*, T2019/10, July 2019, TRAIL Thesis Series, the Netherlands

Hu, Q., *Container Transport inside the Port Area and to the Hinterland*, T2019/9, July 2019, TRAIL Thesis Series, the Netherlands

Andani, I.G.A., *Toll Roads in Indonesia: transport system, accessibility, spatial and equity impacts*, T2019/8, June 2019, TRAIL Thesis Series, the Netherlands

Ma, W., *Sustainability of Deep Sea Mining Transport Plans*, T2019/7, June 2019, TRAIL Thesis Series, the Netherlands

Alemi, A., *Railway Wheel Defect Identification*, T2019/6, January 2019, TRAIL Thesis Series, the Netherlands

Liao, F., *Consumers, Business Models and Electric Vehicles*, T2019/5, May 2019, TRAIL Thesis Series, the Netherlands

Tamminga, G., *A Novel Design of the Transport Infrastructure for Traffic Simulation Models*, T2019/4, March 2019, TRAIL Thesis Series, the Netherlands

# Appendices

# Appendix A

# Proofs

## A.1 MPP is a Self-regulating Planning Problem

Through a series of theorems and proofs, this section will demonstrate that the MAINTENANCE PLANNING PROBLEM (MPP, Definition 3.3) is in fact an instance of SELF-REGULATING PLANNING PROBLEM (SRP), a MDP-based formulation of self-regulating planning that was introduced in Definition 1.1. This self-regulating planning problem can be reformulated in terms of the MDP model as

**Definition A.1 Self-regulating Planning Problem**

Given agents $\boldsymbol{N} = \{1, 2, \ldots, n\}$, a reward-independent multi-agent MDP $\boldsymbol{M} = \{M_i\}_{i \in \boldsymbol{N}} = \{S, \boldsymbol{A}, P, \boldsymbol{R}\}_{i \in \boldsymbol{N}}$, a set $\boldsymbol{p}$ of $n$ payment functions $p_i \colon S \times \boldsymbol{A} \times S \mapsto \mathbb{R}$ and a finite planning horizon $h$, the SELF-REGULATING PLANNING PROBLEM is to find a joint policy $\boldsymbol{\pi}^* = \langle \pi_1, \pi_2, \ldots, \pi_n \rangle$ that maximises the utility of all agents in expectation within the planning period, that is the policy $\boldsymbol{\pi}^* *$ given by

$$\arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} \sum_{i \in \boldsymbol{N}} \Big( R_i(s^t, \pi_i(s^t), s^{t+1}) + p_i(s^t, \boldsymbol{\pi}(s^t), s^{t+1}) \Big) \mid P, s^0 \in S \Big] \quad \text{(A.1)}$$

such that $s^t, s^{t+1} \in S$ are the current and next state at every time $t \leq h$.

Now, through a series of reductions – first from the MPP to the optimisation problem of its MDP encoding, then from that problem to SRP – it can be shown that all instances of MPP can be transformed into equivalent instances of SRPÀs a corollary, any algorithm that solves the SRP can also solve instances of the MPP. Note that this proof is a new contribution by this thesis and has not been published. First, the optimisation problem for the MMDP model of Definition 3.9 is defined:

**Definition A.2 Maintenance Planning MDP Problem (MPP-MDP)**

Given an MMDP instance $M = \langle \boldsymbol{N}, S, \boldsymbol{A}, P, R \rangle$ that encodes an MPP instance using

Definition 3.9, the MAINTENANCE PLANNING MDP PROBLEM (MPP-MDP) is to find an optimal, joint policy $\boldsymbol{\pi}^*$ for $M$ that maximises the expected multi-agent reward $R$ according to (the finite version of) Equation 2.12.

Given this optimisation problem for the MMDP encoding of the MPP problem, it must first be shown that MPP reduces to MPP-MDP. That is, any optimal policy for such an MMDP corresponds to a solution for the MPP instance it encodes.

**Theorem A.3 MPP is an MPPMDP**

Any instance $M = \langle \boldsymbol{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{c}, \ell, T \rangle$ of the MAINTENANCE PLANNING PROBLEM is reducible to an instance $M' = \langle \boldsymbol{N}', S, \boldsymbol{A}, P, \boldsymbol{R} \rangle$ of MPP-MDP (cf. Definition A.2).

*Proof.* The relation between the two problems can be demonstrated by a reduction from MAINTENANCE PLANNING PROBLEM to MPP-MDP. The construction of instance $M'$ follows Definitions 3.7 and 3.9. Here it remains to show that a solution to MPP-MDP is also a solution for MPP. Hence it must be shown that any optimal joint policy $\boldsymbol{\pi}^*$ for $M'$ constitutes a solution for the MAINTENANCE PLANNING PROBLEM, i.e. it maximises $V^{\mathcal{P}}$ of Equation 3.7. Any optimal joint policy $\boldsymbol{\pi}^*$ for the MMDP $M$ of MPP-MDP optimises the expected value of function $R$ over every state/action-history $\theta$ reachable under that policy. That is, it optimises the function sum of expected return as in Equation 4.5:

$$\begin{aligned} V^*(s^0) &= \sum_{\theta^h | \boldsymbol{\pi}^*, s^0} Pr(\theta^h) Z(\theta^h) = \sum_{\theta^h | \boldsymbol{\pi}^*, s^0} Pr(\theta^h) \Big( \sum_{(s,\vec{a},\hat{s}) \in \theta^h} R(s, \vec{a}, \hat{s}) \Big) \\ &= \sum_{\theta^h | \boldsymbol{\pi}^*, s^0} Pr(\theta^h) \Big( \sum_{(s,\vec{a},\hat{s}) \in \theta^h} \sum_{i \in \boldsymbol{N}} R_i(s_i, a_i, \hat{s}_i) + \ell(s, \vec{a}, \hat{s}) \Big) \end{aligned} \tag{A.2}$$

The main observation is that it is possible to construct a history $H_i^h$ for agent $i$ from every execution sequence $\theta^h$ by adding an entry to the history for every $\texttt{start}_{k,i} \in \vec{a}$ of each transition $(s, \vec{a}, \hat{s}) \in \theta^h$:

$$H_i^h = \big\{ \langle \mathsf{a}_{k,i}, t(s,i), o(\mathsf{a}_{k,i}, \hat{s}) \rangle \mid \exists (s, \vec{a}, \hat{s}) \in \theta^h : \texttt{start}_{k,i} \in \vec{a} \wedge \mathsf{a}_{k,i} \in \mathcal{A}_i \big\}$$

such that $t(s,i) \in T$ is the start time of activity $\mathsf{a}_{k,i}$ derived from the time variable of agent $i$ from state $s$ and $o(\mathsf{a}_{k,i}, t(s,i), t(\hat{s},i)) \in [o^+_{k,i}, o^-_{k,i}]$ the function that determines the outcome of activity $\mathsf{a}_{k,i}$ given the time variable of agent $i$ in the current and new state. Furthermore, the joint history $H^h$ can be constructed by aggregating all agent histories $H_i^h$, e.g. $H^h = \bigcup_{i \in \boldsymbol{N}} H_i^h$.

Recall that the probability of a state/action execution sequence $\theta^h$ occurring is equal to the product of state transition probabilities $P(s, \vec{a}, \hat{s})$ for all transitions $(s, \vec{a}, \hat{s}) \in \theta^h$. By Definition 3.9, the transition probabilities are 1 for all transitions except when the joint action $\vec{a}$ contains one or more $\texttt{start}_{k,i}$ actions. In these cases the transition probability is equal to the product of outcome probabilities for all the start actions,

each given by $p_{k,i}$ if the activity is delayed or $1 - p_{k,i}$ otherwise. In other words, the probability of such a joint action $\vec{a}$ as time $t \in T$ is given by $Pr(H_o^t(H_{\mathcal{A}}^t(t)))$ and the total probability of a sequence $\theta^h$ is therefore given by $\prod_{t \in T} H_o^t(H_{\mathcal{A}}^t(t))$ which is equal to $Pr(H^h)$ if $h = |T|$ (Equation 3.3).

Now because $R_i$ is only non-zero when $a_i = \texttt{start}_k$, summing rewards over all state transitions $(s, \vec{a}, \hat{s}) \in \theta^h$ is equal to summing rewards over all activities in the history as just defined. Thus the agent rewards $R_i$ can be replaced by their definition from Definition 3.9. The network cost $\ell(s, \vec{a}, \hat{s})$ is by its definition (Equation 3.11) only summed when $\hat{s}$ is a terminal state or, in other words, only once per execution history $\theta^h$ that must end in such a state $\hat{s}$. Combining this with the construction of $H^h$ from $\theta^h$, $\ell$ can replaced by (the convenient notation of) its definition (Equation 3.11).

All of the above results in an updated version of Equation A.2:

$$V^*(s^0) = \sum_{H^h \in \mathcal{H}^h | \boldsymbol{\pi}^*} Pr(H^h) \Big( \big( \sum_{i \in \boldsymbol{N}} \sum_{\mathsf{a}_{k,i} \in H_i^h} w_{k,i} - \sum_{t=t_s^{k,i}}^{t_e^{k,i}} c_i(\mathsf{a}_{k,i}, t) \big) + \sum_{t \in T} \ell(H^t) \Big)$$

such that $t_s^{k,i}$ and $t_e^{k,i}$ are respectively the start and end time of activity $\mathsf{a}_{k,i}$ as specified by history $H^h$. Observe that $\sum_{i \in \boldsymbol{N}} \sum_{t=t_s^{k,i}}^{t_e^{k,i}} c_i(\mathsf{a}_{k,i}, t)$ can also be written as the sum of costs over all joint activities $\vec{\mathsf{a}} = H_{\mathcal{A}}^t(t)$ that are being performed at time $t$ according to joint history $H^t$, i.e. $\sum_{t \in T} c(H_{\mathcal{A}}^t(t), t) = \sum_{t \in T} c(H^t)$ (shorthand). Furthermore, the per-agent sum of activity revenues $\sum_{i \in \boldsymbol{N}} \sum_{\mathsf{a}_k \in H^h} w_k, i$ can simply be replaced by the revenues of all activities in the joint history $H^h$, hence:

$$V^*(s^0) = \sum_{H^h \in \mathcal{H}^h | \boldsymbol{\pi}^*} Pr(H^h) \Big( \sum_{\mathsf{a}_k \in H^h} w_k - \sum_{t \in T} c(H^t) + \sum_{t \in T} \ell(H^t) \Big)$$

Which is equal to the contingent plan value of Equation 3.7 after grouping the latter two sums and choosing the policy as the contingent plan, e.g. $\mathcal{P}^* = \boldsymbol{\pi}^*$. □

Thus any instance of MPP can be transformed into an instance of MPP-MDP and solved by an algorithm for the latter problem. Now it remains to show that MPP-MDP is reducible to SRP because of the transitivity of the reduction relation. In fact, it is possible to show that MPP-MDP and SRP are equivalent problems, e.g. any instance of MPP-MDP can be solved as an instance of SRP and vice versa. This is shown in the following lemmas.

**Lemma A.4 MPPMDP is an SRP**

For any instance $M = \langle \boldsymbol{N}, S, \boldsymbol{A}, P, R \rangle$ of MPP-MDP there exists a reduction to an instance $M' = \langle \boldsymbol{N}', S', \boldsymbol{A}', P', \boldsymbol{R}', \boldsymbol{p} \rangle$ of SRP.

*Proof.* Let $M$ be the instance of MPP-MDP, i.e. the multi-agent MDP that is constructed according to Definition 3.9. Then it is possible to construct an equivalent

instance $M' = \langle \boldsymbol{N}', S', \boldsymbol{A}', P', \boldsymbol{R}', \boldsymbol{p} \rangle$ of SRP as follows. The sets $\boldsymbol{N}'$, $S'$, $\boldsymbol{A}'$ and $P'$ are exactly the same as respectively $\boldsymbol{N}$, $S$, $\boldsymbol{A}$ and $P$ of the multi-agent MDP, only the way the reward function is constructed differs in both models. Recall from Definition 3.9 that the joint reward function $R$ of the MDP $M$ is obtained by summing the individual reward functions $R_i$ of all agents $i \in \boldsymbol{N}$ and the inter-agent reward function $\ell$. Without loss of generality one can assume that there exists a distribution of inter-agent costs over all agents such that $\forall t \leq h, \vec{a} \in \boldsymbol{A} \colon \ell(\vec{a}, t) = \sum_{i \in \boldsymbol{N}} \ell_i(\vec{a}, t)$. Here $\ell_i(\vec{a}, t)$ represents the costs to agent $i$ as a consequence of joint action $\vec{a}$ being performed (in a state $s^t$) at a time $t$. A dummy agent $0$ may be introduced if these costs are not incurred to any of the agents $i \in \boldsymbol{N}$ but to a centre. With such a cost distribution function it is possible to define $\boldsymbol{R}'$ as the collection of individual agent rewards, i.e. $\boldsymbol{R}' = \{R_i\}_{i \in \boldsymbol{N}}$, and the mechanism payments $\boldsymbol{p}$ as the distributed inter-agent rewards, i.e. $\boldsymbol{p} = \{\ell_i\}_{i \in \boldsymbol{N}}$.

Now to show that finding an optimal policy for MDP $M$ is equivalent to solving SRP. Based upon the multi-agent formulation of the Bellman equation Equation 2.12, the optimal joint policy $\boldsymbol{\pi}^*$ can be found through the formula

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} R(s^t, \boldsymbol{\pi}(s^t), s^{t+1}) \,|\, P, s^0 \in S \Big]$$

which, following Definition 3.9, can be rewritten as

$$\begin{aligned}
\boldsymbol{\pi}^* &= \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} \sum_{i \in \boldsymbol{N}} R_i(s^t, \pi_i(s^t), s^{t+1}) + \ell(\vec{a}, t(s)) \,|\, P, s^0 \in S \Big] \\
&= \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} \sum_{i \in \boldsymbol{N}} R_i(s^t, \pi_i(s^t), s^{t+1}) + \sum_{i \in \boldsymbol{N}} p_i(s^t, \boldsymbol{\pi}(s^t), s^{t+1}) \,|\, P, s^0 \in S \Big] \\
&= \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} \sum_{i \in \boldsymbol{N}} \Big( R_i(s^t, \pi_i(s^t), s^{t+1}) + p_i(s^t, \boldsymbol{\pi}(s^t), s^{t+1}) \Big) \,|\, P, s^0 \in S \Big]
\end{aligned}$$

by construction of $\boldsymbol{R}' = \{R_i\}_{i \in \boldsymbol{N}}$ and $\boldsymbol{p} = \{\ell_i\}_{i \in \boldsymbol{N}}$, which in turn is exactly Equation A.1. □

and the reduction the other way can also be shown:

**Lemma A.5 SRP is an MPPMDP**

For any instance $M = \langle \boldsymbol{N}, S, \boldsymbol{A}, P, \boldsymbol{R}, \boldsymbol{p} \rangle$ of SRP there exists a reduction to an instance $M' = \langle \boldsymbol{N}', S', \boldsymbol{A}', P', R' \rangle$ of MPP-MDP.

*Proof.* Re-using some of the work in Lemma A.4 it is easy to show that for any instance $M$ an equivalent MMDP $M'$ can be constructed. Again, the sets $\boldsymbol{N}'$, $S'$, $\boldsymbol{A}'$ and $P$ are equal to their SRP counterparts $\boldsymbol{N}$, $S$, $\boldsymbol{A}$ and $P$ respectively. The reward

function $R'$ can trivially be constructed by the sum of agent rewards and payments, or $R'(s, \vec{a}, \hat{s}) = \boldsymbol{R}(s, \vec{a}, \hat{s}) + \boldsymbol{p}(s, \vec{a}, \hat{s}) = \sum_{i \in \boldsymbol{N}} R_i(s, \vec{a}, \hat{s}) + \sum_{i \in \boldsymbol{N}} p_i(s, \vec{a}, \hat{s})$ for all possible transitions $(s, \vec{a}, \hat{s}) \in S \times \boldsymbol{A} \times S$.

Showing that any solution to an instance of $M$ is a solution to the constructed instance of SRP is demonstrated similarly to the proof of Lemma A.4. A solution for SRP is given by the policy $\boldsymbol{\pi}^*$ that satisfies Equation A.1:

$$\arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}\Big[ \sum_{t=0}^{h-1} \sum_{i \in \boldsymbol{N}} \Big( R_i(s^t, \pi_i(s^t), s^{t+1}) + p_i(s^t, \boldsymbol{\pi}(s^t), s^{t+1}) \Big) \mid P, s^0 \in S \Big] \quad \text{(A.3)}$$

which is equivalent to optimising the the multi-objective Bellman equation that is the objective function in MPP-MDP if

$$R'(s, \boldsymbol{\pi}(s^t), s^{t+1}) = \sum_{i \in \boldsymbol{N}} R_i(s^t, \pi_i(s^t), s^{t+1}) + \sum_{i \in \boldsymbol{N}} p_i(s^t, \boldsymbol{\pi}(s^t), s^{t+1})$$

which is the case due to the construction of the reward function just described. □

---

As a corollary of Lemmas A.4 and A.5 it can be concluded that both problems are equivalent:

**Theorem A.6 SRP is equal to MPPMDP**

The SELF-REGULATING PLANNING PROBLEM is equivalent to MPP-MDP.

*Proof.* The proof is a corollary of Lemmas A.4 and A.5 that show that any instance of the former can be modelled and solved as an instance of the latter and vice versa. □

---

From the previous theorems it can be concluded that any instance of MPP is also an instance of SRP, which is the conclusion that was sought after.

**Theorem A.7 The MPP is a SRPfull**

Any instance $M = \langle \boldsymbol{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{c}, \ell, T \rangle$ of the MAINTENANCE PLANNING PROBLEM is reducible to an equivalent instance $M' = \langle \boldsymbol{N}', S, \boldsymbol{A}, P, \boldsymbol{R}, \boldsymbol{p} \rangle$ of the SELF-REGULATING PLANNING PROBLEM (cf. Definition A.1).

*Proof.* The relation between the two problems can be demonstrated by a reduction from MPP to MPP-MDP and subsequently from MPP-MDP to SRP. Because the reduction relation is transitive and Theorems A.3 and A.6 show the existence of the required reductions, MPP is reducible to SRP. □

---

As a corollary of Theorem A.7, any technique or algorithm to solve SELF-REGULATING PLANNING PROBLEM can solve instances of the MAINTENANCE PLANNING PROBLEM as instances of the latter can always be transformed into instances of the former.

## A.2 Lemma 4.9: Admissible heuristics

The bounding heuristics $L(s)$ and $U(s)$ are admissible with respect to the expected value $V(s)$ obtained from a joint state $s \in \boldsymbol{S}$ onward.

*Proof.* The admissibility of the bounding heuristics is shown by induction. Only the proof for the upper bound is shown, the proof for the lower bound is conceived accordingly. Recall that $\boldsymbol{R} = \bigcup_{i \in \boldsymbol{N}} \boldsymbol{R}_i$ is the disjoint partitioning of reward functions over the CRGs and $e_i$ is the set of agent in the scope of $\boldsymbol{R}_i$, i.e. $e_i = \left\{ j \in \boldsymbol{N} \mid \bar{R}_e \in \boldsymbol{R}_i \wedge j \in e \right\}$. First, consider a joint state $s^{h-1}$ at the very last decision stage, $h-1$, for which there is no future reward:

$$
\begin{aligned}
V(s^{h-1}) &= \max_{\vec{a}^{\,h-1} \in \boldsymbol{A}} \sum_{s^h \in \boldsymbol{S}} P(s^{h-1}, \vec{a}^{\,h-1}, s^h) \boldsymbol{R}(s^{h-1}, \vec{a}^{\,h-1}, s^h) \\
&= \max_{\vec{a}^{\,h-1} \in \boldsymbol{A}} \sum_{s^h \in \boldsymbol{S}} P(s^{h-1}, \vec{a}^{\,h-1}, s^h) \sum_{i \in \boldsymbol{N}} \boldsymbol{R}_i(s^{h-1}_{e_i}, \vec{a}^{\,h-1}_{e_i}, s^h_{e_i}) \\
&\leq \max_{\vec{a}^{\,h-1} \in \boldsymbol{A}} \max_{s^h \in \boldsymbol{S}} \sum_{i \in \boldsymbol{N}} \boldsymbol{R}_i(s^{h-1}_{e_i}, \vec{a}^{\,h-1}_{e_i}, s^h_{e_i}) \\
&\leq \sum_{i \in \boldsymbol{N}} \max_{(s^{h-1}_{e_i}, \vec{a}^{\,h-1}_{e_i}, s^h_{e_i}) \in \phi_i(s^{h-1}_i)} \boldsymbol{R}_i(s^{h-1}_{e_i}, \vec{a}^{\,h-1}_{e_i}, s^h_{e_i}) = \sum_{i \in \boldsymbol{N}} U(s^{h-1}_i)
\end{aligned}
$$

Then it can be shown that if for a next stage $t+1$ a valid upper bound exists, the value for a state $s^t$ is upper bounded by $\sum_{i \in \boldsymbol{N}} U(s^t_i)$. And therefore, because $U(s^{h-1}_i)$ is a valid upper bound on $V(s^{h-1})$, the upper bound is admissible for all stages before $h-1$:

$$
\begin{aligned}
V(s^t) &= \max_{\vec{a}^{\,t} \in \boldsymbol{A}} \sum_{s^{t+1} \in \boldsymbol{S}} P(s^t, \vec{a}^{\,t}, s^{t+1}) \left( \boldsymbol{R}(s^t, \vec{a}^{\,t}, s^{t+1}) + V(s^{t+1}) \right) \\
&= \max_{\vec{a}^{\,t} \in \boldsymbol{A}} \sum_{s^{t+1} \in \boldsymbol{S}} P(s^t, \vec{a}^{\,t}, s^{t+1}) \left( \sum_{i \in \boldsymbol{N}} \boldsymbol{R}_i(s^t_{e_i}, \vec{a}^{\,t}_{e_i}, s^{t+1}_{e_i}) + V(s^{t+1}) \right) \\
&\leq \max_{\vec{a}^{\,t} \in \boldsymbol{A}} \sum_{s^{t+1} \in \boldsymbol{S}} P(s^t, \vec{a}^{\,t}, s^{t+1}) \sum_{i \in \boldsymbol{N}} \left( \boldsymbol{R}_i(s^t_{e_i}, \vec{a}^{\,t}_{e_i}, s^{t+1}_{e_i}) + U(s^{t+1}_i) \right) \\
&\leq \sum_{i \in \boldsymbol{N}} \max_{(s^t_{e_i}, \vec{a}^{\,t}_{e_i}, s^{t+1}_{e_i}) \in \phi_i(s^t_i)} \left( \boldsymbol{R}_i(s^t_{e_i}, \vec{a}^{\,t}_{e_i}, s^{t+1}_{e_i}) + U(s^{t+1}_i) \right) = \sum_{i \in \boldsymbol{N}} U(s^t_i)
\end{aligned}
$$

Thus, by induction, the upper bound $U$ is admissible with respect to the expected value for any given state $s^t$. Following a similar reasoning for the lower bound it is possible to derive

$$
L(s^t) = \sum_{i \in \boldsymbol{N}} L(s^t_i) \leq V^{\pi}(s^t) \leq \sum_{i \in \boldsymbol{N}} U(s^t_i) = U(s^t) \tag{A.4}
$$

□

## A.3 Lemma 4.11: CRI decouples returns

Given an execution history $\theta^t = [s^0, \vec{a}^{\,0}, \ldots, s^u, \ldots, s^t]$ up to time $t$ that can be partitioned into two histories, $\theta^u = [s^0, \ldots, s^u]$ and $\theta^{u'} = [s^u, \ldots, s^t]$, and a disjoint partitioning of agent sets $\boldsymbol{N} = N_1 \cup N_2 \cup \ldots \cup N_k$ such that for every pair $N_a, N_b \in \boldsymbol{N}$ it holds that $CRI(N_a, N_b, \theta^u)$ when $a \neq b$, then the return can be decoupled as:

$$Z(\theta^t) = Z(\theta^u) + \sum_{i=1}^{k} Z_{N_i}(\theta^{u'}_{N_i}) \tag{A.5}$$

Here, $\theta^{u'}_{N_i}$ is the execution history of the agents in the set $N_i \subseteq \boldsymbol{N}$, starting from time $u$.

*Proof.* The following notational shorthand is used. For two (sub)sets of agents $A, B \subseteq \boldsymbol{N}$, $\boldsymbol{R}_{AB} \subseteq \boldsymbol{R}$ is the set of all rewards for which $A \cap B \cap e \neq \emptyset$. $\boldsymbol{R}_{A\not B} \subseteq \boldsymbol{R}$ is the set of rewards such that $A \cap e \neq \emptyset$ and $B \cap e = \emptyset$. Observe that the individual rewards for all agents of $A$ are thus contained within $\boldsymbol{R}_{A\not B}$ (and similarly for all agents $b \in B$, $R_b$ is included in $\boldsymbol{R}_{\not AB}$).

Let $A$ and $B$ be disjoint subsets of agents such that $A \cup B = \boldsymbol{N}$ and let the reward functions be partitioned accordingly as disjoint sets $\boldsymbol{R} = \boldsymbol{R}_{A\not B} \cup \boldsymbol{R}_{\not AB} \cup \boldsymbol{R}_{AB}$. Now assume that as a result of a given execution history $\theta^u$, such that $\theta^t = \theta^u \cup \theta^{u'}$, now $CRI(A, B, \theta^u)$ becomes true. From the state $s^u$ that results from execution history $\theta^u$ all future rewards can only be local with respect to subset $A$ or subset $B$ because every reward $\boldsymbol{R}_{AB}$ must be zero by definition of CRI. Therefore the (future) global reward $\boldsymbol{R}(\tau)$ of every possible joint transition $\tau = (s, \vec{a}, \hat{s})$ can be written as:

$$\begin{aligned} \boldsymbol{R}(\tau) &= \boldsymbol{R}_{A\not B}(\tau_A) + \boldsymbol{R}_{\not AB}(\tau_B) + \boldsymbol{R}_{AB}(\tau_{AB}) \\ &= \sum_{\bar{R}_A \in \boldsymbol{R}_{A\not B}} \bar{R}_A(\tau_A) + \sum_{\bar{R}_B \in \boldsymbol{R}_{\not AB}} \bar{R}_B(\tau_B) \end{aligned} \tag{A.6}$$

and note that $\boldsymbol{R}_{A\not B}(\tau_A)$ is equal to $\boldsymbol{R}_{A\not B}(\tau)$ because the reward is zero for every transition not in the scope of $\boldsymbol{R}_{A\not B}$. Remember that the returns for an execution history $\theta^h$ are expressed by $Z(\theta^t) = \sum_{x=0}^{t-1} \boldsymbol{R}(\tau^x) = \sum_{x=0}^{t-1} \sum_{\bar{R}_e \in \boldsymbol{R}} \bar{R}_e(\tau_e^x)$ (Equation 4.4), where $\tau_e^x$ denotes the transition in the execution history $\theta^t$ local to agents $e$ at time $x$. Then, for two disjoint agent subsets $A \cup B = \boldsymbol{N}$ with $CRI(A, B, \theta^u)$ as a result of $\theta^u$:

$$\begin{aligned} Z(\theta^t) &= Z(\theta^u) + Z(\theta^{u'}) \\ &= Z(\theta^u) + \sum_{x=u}^{t-1} (\boldsymbol{R}_{A\not B}(\tau_A^x) + \boldsymbol{R}_{\not AB}(\tau_B^x)) \\ &= Z(\theta^u) + \sum_{x=u}^{t-1} \boldsymbol{R}_{A\not B}(\tau_A^x) + \sum_{x=u}^{t-1} \boldsymbol{R}_{\not AB}(\tau_B^x) \\ &= Z(\theta^u) + Z_A(\theta^{u'}_A) + Z_B(\theta^{u'}_B) \end{aligned}$$

and, consequentially, the returns for agent sets $A$ and $B$ are independent from time $u$.[54]

This result can be extended from two agent sets $A$ and $B$ to the desired arbitrary disjoint partitioning of agents such that $N_1 \cup N_2 \cup \ldots \cup N_k = \boldsymbol{N}$ and $\forall N_a, N_b \in \boldsymbol{N}$: $CRI(N_a, N_b, \theta^u)$. Without loss of generality, let $A = N_1$ and $B = N_2 \cup \ldots \cup N_k$ and decouple the return as $Z(\theta^u) + Z_A(\theta_A^{u'}) + Z_B(\theta_B^{u'})$. Observe that in turn $Z_B$ can be rewritten as $Z_{N_2}(\theta_{N_2}^{u'}) + Z_{B\setminus\{N_2\}}(\theta_{B\setminus\{N_2\}}^{u'})$ by following the same argument, because both sets again satisfy conditional reward independence. By continuing this process Equation A.5 is obtained. □

## A.4 Lemma 6.7: Dynamic VCG payment

Setting $\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) = \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i})$ leads to the payment of Equation 6.7.

*Proof.* It is possible to obtain Equation 6.7 by substituting $\tilde{H}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i})$ into Equation 6.6 by the term $\tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i})$ and reducing it to the desired payoff function $p_i^t$. Substitution results in:

$$\tilde{p}_i(\hat{\boldsymbol{\theta}}_i^t, \vec{\sigma}) = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) \tag{A.7}$$

substituting the expected payment by its definition (Equation 6.4) and extracting the payoff for the current time yields

$$\mathbb{E}\Big[\sum_{t=t'}^{h} p_i^t\big(\Gamma(\hat{\boldsymbol{\theta}}^t)\big) \;\Big|\; \hat{\boldsymbol{\theta}}^t = \vec{\sigma}(\boldsymbol{\theta}^t)\Big] = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) \quad \Rightarrow$$

$$p_i^t(\hat{\boldsymbol{\theta}}^t) + \mathbb{E}\Big[\sum_{t=t'+1}^{h} p_i^t\big(\Gamma(\hat{\boldsymbol{\theta}}^t)\big) \;\Big|\; \hat{\boldsymbol{\theta}}^t = \vec{\sigma}(\boldsymbol{\theta}^t)\Big] = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i})$$

using again Equation 6.4 to replace the sum of expected payments and rearranging terms leads to

$$p_i^t(\hat{\boldsymbol{\theta}}^t) = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) - \tilde{p}_i(\bar{\boldsymbol{\theta}}^{t+1}, \vec{\sigma})$$

such that $\tilde{p}_i(\bar{\boldsymbol{\theta}}^{t+1}, \vec{\sigma})$ is the expected payment for the jointly reported dynamic type $\bar{\boldsymbol{\theta}}^{t+1}$ that is reported based on the outcome that included *all agents* at time $t$, i.e. the outcome $\Gamma(\hat{\boldsymbol{\theta}}^t)$. Now, the expected payment from $t+1$ can be substituted using Equation A.7:

$$p_i^t(\hat{\boldsymbol{\theta}}^t) = \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}_{-i}^t, \vec{\sigma}_{-i}) - \Big(\tilde{V}_{-i}^{\Gamma}(\bar{\boldsymbol{\theta}}^{t+1}, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\bar{\boldsymbol{\theta}}_{-i}^{t+1}, \vec{\sigma}_{-i})\Big)$$

[54] *Although the execution history $\theta^{u'}$ ranges from time $u$ to $t$, it is trivial to transform it to the $[0, t-u]$ range such that all previously defined formulas apply.*

from which eventually the desired Equation 6.7 can be obtained (omitting the conditions $\hat{\boldsymbol{\theta}}^t$ and $\bar{\boldsymbol{\theta}}^t$ of the expectation sums for clarity):

$$
\begin{aligned}
p_i^t(\hat{\boldsymbol{\theta}}^t) &= \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i}) - \Big(\tilde{V}_{-i}^{\Gamma}(\bar{\boldsymbol{\theta}}^{t+1}, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\bar{\boldsymbol{\theta}}^{t+1}_{-i}, \vec{\sigma}_{-i})\Big) \\
&= \tilde{V}_{-i}^{\Gamma}(\hat{\boldsymbol{\theta}}^t, \vec{\sigma}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i}) - \tilde{V}_{-i}^{\Gamma}(\bar{\boldsymbol{\theta}}^{t+1}, \vec{\sigma}) + \tilde{V}_{-i}^{\Gamma_{-i}}(\bar{\boldsymbol{\theta}}^{t+1}_{-i}, \vec{\sigma}_{-i}) \\
&= \mathbb{E}\Big[\sum_{t=t'}^{h} v_{-i}\big(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t)\big)\Big] - \mathbb{E}\Big[\sum_{t=t'}^{h} v_{-i}\big(\hat{\boldsymbol{\theta}}^t_{-i}, \Gamma_{-i}(\hat{\boldsymbol{\theta}}^t_{-i})\big)\Big] \\
&\quad - \mathbb{E}\Big[\sum_{t=t'+1}^{h} v_{-i}\big(\bar{\boldsymbol{\theta}}^{t+1}, \Gamma(\bar{\boldsymbol{\theta}}^{t+1})\big)\Big] + \mathbb{E}\Big[\sum_{t=t'+1}^{h} v_{-i}\big(\bar{\boldsymbol{\theta}}^{t+1}_{-i}, \Gamma_{-i}(\bar{\boldsymbol{\theta}}^{t+1}_{-i})\big)\Big] \quad \text{(c.f. Eq. 6.3)} \\
&= v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t)) + \mathbb{E}\Big[\sum_{t=t'+1}^{h} v_{-i}\big(\bar{\boldsymbol{\theta}}^{t+1}_{-i}, \Gamma_{-i}(\bar{\boldsymbol{\theta}}^{t+1}_{-i})\big)\Big] - \mathbb{E}\Big[\sum_{t=t'}^{h} v_{-i}\big(\hat{\boldsymbol{\theta}}^t_{-i}, \Gamma_{-i}(\hat{\boldsymbol{\theta}}^t_{-i})\big)\Big] \\
&= v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t)) + \tilde{V}_{-i}^{\Gamma_{-i}}(\bar{\boldsymbol{\theta}}^{t+1}_{-i}, \vec{\sigma}_{-i}) - \tilde{V}_{-i}^{\Gamma_{-i}}(\hat{\boldsymbol{\theta}}^t_{-i}, \vec{\sigma}_{-i})
\end{aligned}
$$

which is exactly Equation 6.7. The last two reductions are possible because respectively 1) the difference of sums over expected valuation for the outcomes including agent $i$ (the first and third expectation sums) differ only in the current time $t$, and 2) the expected valuations can be replaced by their compact forms using Equation 6.3. □

Observe that in Lemma 6.7 it is not possible to combine the second and fourth expectation sums in the second-last step similar to the combination of the other two summations. This is because their reported dynamic types $\hat{\boldsymbol{\theta}}^{t+1}$ and $\bar{\boldsymbol{\theta}}^{t+1}$ are not the same at time $t+1$: in the outcomes of the former expected value summation agent $i$ is never included or considered, in the latter agent $i$ *was* participating in the outcome at time $t$ (although its valuation is discarded) but not thereafter. In other words, $\hat{\boldsymbol{\theta}}^{t+1}$ is the new type after outcome $\Gamma_{-i}(\hat{\boldsymbol{\theta}}^t_{-i})$ and $\bar{\boldsymbol{\theta}}^{t+1}$ the new type after outcome $\Gamma(\hat{\boldsymbol{\theta}}^t)$. The dynamic type at $t+1$ can therefore be highly dissimilar between both summations, depending on the impact of agent $i$'s presence at time $t$. The difference between the two expectation sums can be intuitively seen as the impact that agent $i$'s current decision has on future valuations. The immediate impact of agent $i$'s action is accounted for by the first term of the dynamic VCG payment, $v_{-i}(\hat{\boldsymbol{\theta}}^t, \Gamma(\hat{\boldsymbol{\theta}}^t))$.

## A.5 Theorem 6.16: Condition (i)

When two joint policies $\boldsymbol{\pi} = \langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$ and $\boldsymbol{\pi}' = \langle \pi'_i, \boldsymbol{\pi}_{-i} \rangle$ differ only in policies $\pi_i$ and $\pi'_i$ for agent $i$, it must be that $\Phi(\boldsymbol{\pi}) - \Phi(\boldsymbol{\pi}') = u_i(\boldsymbol{\pi}) - u_i(\boldsymbol{\pi}')$.

*Proof.* In order to prove the condition it must be shown that $u_i(\boldsymbol{\pi}) - u_i(\boldsymbol{\pi}')$ equals

$$
\begin{aligned}
\Phi(\boldsymbol{\pi}) - \Phi(\boldsymbol{\pi}') &= \sum_{i \in \boldsymbol{N}} v_i(\boldsymbol{\pi}) + \sum_{e_k \in E} \sum_{t \in T} \tilde{c}_k(\boldsymbol{\pi}, t) - \sum_{i \in \boldsymbol{N}} v_i(\boldsymbol{\pi}') + \sum_{e_k \in E} \sum_{t \in T} \tilde{c}_k(\boldsymbol{\pi}', t) \\
&= v_i(\boldsymbol{\pi}) - v_i(\boldsymbol{\pi}') + \sum_{e_k \in E} \sum_{t \in T} \big(\tilde{c}_k(\boldsymbol{\pi}, t) - \tilde{c}_k(\boldsymbol{\pi}', t)\big) \qquad \text{(A.8)}
\end{aligned}
$$

Let $p$ be shorthand for $Pr(i \in e_k|\pi_i, t)$ and, as a consequence, $1-p = Pr(i \notin e_k|\pi_i, t)$. It is possible to rewrite $\tilde{c}_k(\boldsymbol{\pi}, t)$ such that all terms involving agent $i$ are extracted (using the recursive formulation of Equation 6.16 to obtain the second equation):

$$
\begin{aligned}
\tilde{c}_k(\boldsymbol{\pi}, t) &= \sum_{j=1}^{n} Pr(|e_k| = j|\boldsymbol{\pi}, t)\, c_k(j, t) \\
&= \sum_{j=1}^{n} \Big( p\, Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) + (1-p) Pr(|e_k| = j|\boldsymbol{\pi}_{-i}, t) \Big) c_k(j, t) \\
&= p \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) + (1-p) \sum_{j=1}^{n} Pr(|e_k| = j|\boldsymbol{\pi}_{-i}, t) c_k(j, t) \\
&= p \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) + (1-p) \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \\
&= p \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - p\tilde{c}_k(\boldsymbol{\pi}_{-i}, t) + \tilde{c}_k(\boldsymbol{\pi}_{-i}, t)
\end{aligned}
$$

Then, let $q$ denote $Pr(i \in e_k|\pi'_i, t)$ and consider $\tilde{c}_k(\boldsymbol{\pi}, t) - \tilde{c}_k(\boldsymbol{\pi}', t)$ of Equation A.8:

$$
\begin{aligned}
\tilde{c}_k(\boldsymbol{\pi}, t) - \tilde{c}_k(\boldsymbol{\pi}', t) &= p \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - p\tilde{c}_k(\boldsymbol{\pi}_{-i}, t) + \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \\
&\quad - \Big( q \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - q\tilde{c}_k(\boldsymbol{\pi}_{-i}, t) + \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \Big) \\
&= p \Big( \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \Big) \\
&\quad - q \Big( \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \Big) \\
&= (p-q) \Big( \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - \tilde{c}_k(\boldsymbol{\pi}_{-i}, t) \Big) \\
&= (p-q) \Big( \sum_{j=1}^{n} Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) c_k(j, t) - \sum_{j=1}^{n} Pr(|e_k| = j|\boldsymbol{\pi}_{-i}, t) c_k(j, t) \Big) \\
&= (p-q) \Big( \sum_{j=1}^{n} \big( Pr(|e_k| = j-1|\boldsymbol{\pi}_{-i}, t) - Pr(|e_k| = j|\boldsymbol{\pi}_{-i}, t) \big) c_k(j, t) \Big)
\end{aligned}
$$

Now, substituting this result back into Equation A.8 yields

$$\begin{aligned}
&\Phi(\boldsymbol{\pi}) - \Phi(\boldsymbol{\pi}') = v_i(\boldsymbol{\pi}) - v_i(\boldsymbol{\pi}') + \sum_{e_k \in E} \sum_{t \in T} \big(\tilde{c}_k(\boldsymbol{\pi}, t) - \tilde{c}_k(\boldsymbol{\pi}', t)\big) \\
&= v_i(\boldsymbol{\pi}) - v_i(\boldsymbol{\pi}') \\
&\quad + \sum_{e_k \in E} \sum_{t \in T} (p - q) \Big( \sum_{j=1}^{n} \big( Pr(|e_k| = j - 1 | \boldsymbol{\pi}_{-i}, t) - Pr(|e_k| = j | \boldsymbol{\pi}_{-i}, t) \big) c_k(j, t) \Big) \\
&= v_i(\boldsymbol{\pi}) + \sum_{e_k \in E} \sum_{t \in T} p \sum_{j=1}^{n} \big( Pr(|e_k| = j - 1 | \boldsymbol{\pi}_{-i}, t) - Pr(|e_k| = j | \boldsymbol{\pi}_{-i}, t) \big) c_k(j, t) \\
&\quad - v_i(\boldsymbol{\pi}') - \sum_{e_k \in E} \sum_{t \in T} q \sum_{j=1}^{n} \big( Pr(|e_k| = j - 1 | \boldsymbol{\pi}_{-i}, t) - Pr(|e_k| = j | \boldsymbol{\pi}_{-i}, t) \big) c_k(j, t) \\
&= u_i(\boldsymbol{\pi}_i) - u_i(\boldsymbol{\pi}_i')
\end{aligned}$$

which proves the condition. □

# Appendix B

# Computing Game Scores

This appendix describes the data gathering methodology used to validate the hypotheses of Chapter 7 in full detail. This section is separated into two parts corresponding to the source from which the measurements are obtained, a priori through a questionnaire and from measurements during game play. A complete overview of the game design and the measurements can be found in the on line appendix by Scharpff et al. [231]. Furthermore, the source code for of the Road Maintenance Game and its results can be found at `https://github.com/AlgTUDelft/road-maintenance-game`.

## B.1 Agent decision preference and rationality

The a priori decision rationality and preference of participants is established by a questionnaire. This questionnaire poses 7 increasingly more complex decision-making scenarios from the maintenance planning domain, asking participants to rank alternatives according to their preference. Whereas the first question is relatively easy and has a 'correct' answer, i.e. the alternatives can be clearly ordered according to their ttl impact, the subsequent questions become increasingly more complex. This is due to the introduction of new factors into the decision-making process such as profits, delays and the presence of other service providers. Furthermore, the alternatives are designed in such a way that no one answer is optimal in all objectives. Therefore the ranking of alternatives mostly depends on personal preference, that is, the decision rationality of the participants. The questionnaire is included in Figure B.1.

**Dynamic Network Planning Questionnaire**

Name:
Date:
Occupation and position:

**Question 1**

You are a service provider responsible for the maintenance of a road segment in a regional network. To this end, you have studied the impact on traffic of four possible alternatives. This results in the following congestion figures, expressed in hours of traffic time lost (TTL), caused by each alternative.

| | Alternative A | Alternative B | Alternative C | Alternative D |
|---|---|---|---|---|
| *TTL* | 352.000 | 578.000 | 440.000 | 370.000 |

a) Can you specify the order in which you would choose from the various alternatives? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Alternative A | |
| Alternative B | |
| Alternative C | |
| Alternative D | |

b) Please motivate your ranking.

**Question 2**

In addition you perform cost computation, resulting in the following figures:

| | Alternative A | Alternative B | Alternative C | Alternative D |
|---|---|---|---|---|
| *TTL* | 352.000 | 578.000 | 440.000 | 370.000 |
| *Profit* | € 1.450.000 | € 2.108.000 | € 1.500.000 | € 1.739.000 |

a) Can you specify the order in which you would choose from the various alternatives? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Alternative A | |
| Alternative B | |
| Alternative C | |
| Alternative D | |

b) Please motivate your ranking.

**Question 3**

The road authority decides to implement a traffic penalty payment that charges the service provider 1 euro for each hour of TTL. After some recalculation you find out that this has the following impact on your project:

| | Alternative A | Alternative B | Alternative C | Alternative D |
|---|---|---|---|---|
| *TTL* | 352.000 | 578.000 | 440.000 | 370.000 |
| *Profit* | € 1.098.000 | € 1.529.000 | € 1.060.000 | € 1.369.000 |

a) Can you specify the order in which you would choose from the various alternatives? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Alternative A | |
| Alternative B | |
| Alternative C | |
| Alternative D | |

b) Please motivate your ranking.

**Question 4**

You are aware of the possibility that your project execution might be delayed and you are wondering how much that will affect the figures from before. Therefore you decide to also consider this delay in your computations:

| Situation | Risk of delay | | Alt. A | Alt. B | Alt. C | Alt. D |
|---|---|---|---|---|---|---|
| **Project as planned** | 67% | *TTL* | 352.000 | 578.000 | 440.000 | 370.000 |
| | | *Profit* | € 1.098.000 | € 1.529.000 | € 1.060.000 | € 1.369.000 |
| **Project is delayed** | 33% | *TTL* | 443.000 | 885.000 | 440.000 | 503.000 |
| | | *Profit* | € 1.006.000 | € 1.223.000 | € 1.060.000 | € 1.236.000 |

a) Can you specify the order in which you would choose from the various alternatives? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Alternative A | |
| Alternative B | |
| Alternative C | |
| Alternative D | |

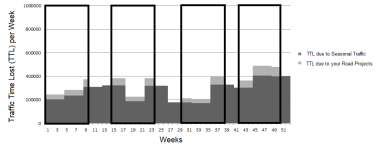
b) Please motivate your ranking.

**Question 5**

You have chosen four potential periods in which you can perform your project. Using quarterly figures, you determine the following prospects regarding four possible maintenance periods:

| Situation | Risk of delay | | Period 1 | Period 2 | Period 3 | Period 4 |
|---|---|---|---|---|---|---|
| **Project as planned** | 67% | *TTL* | 370.000 | 416.000 | 333.000 | 615.000 |
| | | *Profit* | € 1.369.000 | € 1.323.000 | € 1.406.000 | € 1.124.000 |
| **Project is delayed** | 33% | *TTL* | 503.000 | 571.000 | 493.000 | 809.000 |
| | | *Profit* | € 1.236.000 | € 1.168.000 | € 1.246.000 | € 930.000 |

In addition, you also possess information regarding the TTL figures of the previous year.




a) Can you specify the order of periods in which you prefer to perform the maintenance? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Period 1 | |
| Period 2 | |
| Period 3 | |
| Period 4 | |

b) Please motivate your ranking.

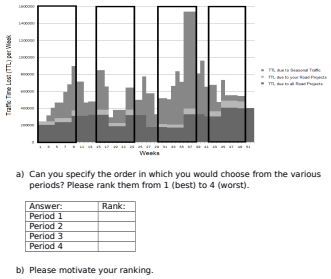
**Question 6**

Other service providers are also working in this region and they, in combination with your project, cause additional traffic hindrance:

| Situation | Risk of delay | | Period 1 | Period 2 | Period 3 | Period 4 |
|---|---|---|---|---|---|---|
| **Project as planned** | 67% | *TTL* | 370.000 | 416.000 | 333.000 | 615.000 |
| | | *Profit* | € 1.369.000 | € 1.323.000 | € 1.406.000 | € 1.124.000 |
| **Project is delayed** | 33% | *TTL* | 503.000 | 571.000 | 493.000 | 809.000 |
| | | *Profit* | € 1.236.000 | € 1.168.000 | € 1.246.000 | € 930.000 |

| Situation | Risk of delay | | Period 1 | Period 2 | Period 3 | Period 4 |
|---|---|---|---|---|---|---|
| **Project as planned** | 67% | *TTL Ind.* | 370.000 | 416.000 | 333.000 | 615.000 |
| | | *TTL Net.* | 990.000 | 2.033.000 | 3.966.000 | 1.302.000 |
| | | *Profit* | € 1.171.000 | € 916.000 | € 613.000 | € 863.000 |
| **Project is delayed** | 33% | *TTL Ind.* | 414.000 | 471.000 | 406.000 | 667.000 |
| | | *TTL Net.* | 2.030.000 | 2.662.000 | 6.545.000 | 1.436.000 |
| | | *Profit* | € 919.000 | € 736.000 | € 24.000 | € 785.000 |

In this table, the individual TTL denotes the TTL caused solely by your project, ignoring others. The network TTL captures the 'combined effect' of multiple service providers working concurrently in the same region. Next to this table, you are also given a plot of the TTL distribution over time.



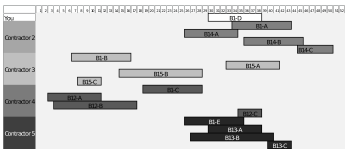

a) Can you specify the order in which you would choose from the various periods? Please rank them from 1 (best) to 4 (worst).

| Answer: | Rank: |
|---|---|
| Period 1 | |
| Period 2 | |
| Period 3 | |
| Period 4 | |

b) Please motivate your ranking.

**Question 7**

You have chosen the third period and you wish to reduce the incurred traffic time losses. Through communication with the other contractors, you have been able to create the following joint schedule:




a) a) If you were given the opportunity to change plans made by others, what project(s) would you modify? Can you rank the projects based on which one you would modify first (1) to last (4)?

| Project: | Rank: |
|---|---|
| Contractor 2: B1-A | |
| Contractor 2: B14-B | |
| Contractor 3: B15-A | |
| Contractor 5: B13-A | |

b) Please motivate your ranking.

**Thank you very much for completing this questionnaire!**

**Figure B.1** *The seven questions of the questionnaire that is used to determine the a priori decision preference of game participants.*

To measure the decision preference and rationality, the submitted responses are evaluated against pre-determined rankings of alternatives per question and objective. In other words, for every question $n$ and objective $m$ there is a ranking of objectives $O_n^m$ that lists the alternatives in their order from best to worst. For example, the profit ordering of question 2 is the ranking $O_2^p = (4, 1, 3, 2)$ as alternative B yields the most profit and A the least. For the ttl, on the other hand, the ranking is given by $O_2^t = (1, 4, 3, 2)$ as the lowest hindrance is caused by alternative A. Note that here numerical indexes are used instead of the alphabetical index in the questionnaire, this is more convenient when computing scores. For each objective the ranking is determined according to the following rules:

- The profit ranking $O^p$ is given by the expected profit of an alternative, ordered from highest to lowest. When the probability of delay is zero, the profit is simply the profit as listed (questions 1 to 4). In the case of potential delay, the expected profit is computed as $(1-p) \times P_{planned} + p \times P_{delayed}$ in which $p$ is the probability of delay ($0$ or $0.33$) and $P_{planned}$ and $P_{delayed}$ the listed profit when respectively the maintenance is performed according to plan or a delay is encountered.

- The ttl ranking $O^t$ is determined by the expected ttl, ordered from lowest (best) to highest (worst). Similar to profit, the ttl score is also computed by the expected ttl in the presence of delay. Moreover, in the presence of other service providers an additional ttl component 'Network' is factored into the computation, i.e. $(1-p) \times (T_{ind,planned} + T_{net,planned}) + p \times (T_{ind,delayed} + T_{net,delayed})$ such that $T_{ind,x}$ and $T_{net,x}$ express the listed individual and network ttl for the planned

and delayed scenarios. Note that it is assumed that work by other contractors does not delay (or always does) to keep the influence of the network on ttl simple.

- The risk(-aversion) ranking $O^r$ is ordered on the expected loss of revenue due to delay from lowest to highest potential loss. In other words, the risk aversion score is the highest when the effect of delay is the least and lowest when a delay causes high revenue decreases. The formula to compute this loss is $(P_{planned} - T_{planned}) - (P_{delayed} - T_{delayed})$ due to the design of payments such that every hour of ttl incurs a cost of 1 euro (with $T_x = T_{ind,x} + T_{net,x}$).

| | *Profit* | | | | *TTL* | | | | *Risk-aversion* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Question #** | A | B | C | D | A | B | C | D | A | B | C | D |
| 1 | - | - | - | - | 1 | 4 | 3 | 2 | - | - | - | - |
| 2 | 4 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | - | - | - | - |
| 3 | 3 | 1 | 4 | 2 | 1 | 4 | 3 | 2 | - | - | - | - |
| 4 | 2 | 4 | 3 | 1 | 1 | 4 | 3 | 2 | 2 | 4 | 1 | 3 |
| 5 | 2 | 3 | 1 | 4 | 2 | 3 | 1 | 4 | 1 | 2 | 3 | 4 |
| 6 | 1 | 2 | 4 | 3 | 1 | 3 | 4 | 2 | 3 | 2 | 4 | 1 |
| 7 | - | - | - | - | 3 | 4 | 2 | 1 | - | - | - | - |

**Table B.1** *Rank of alternatives per question and objective from best (1) to worst (4). Entries marked as '-' indicate a no-score in that objective, due to absence of the objective.*

Using these pre-determined rankings per objective, Table B.1 is obtained that contains the rankings for every alternative per question and objective. Now, given a complete questionnaire response that is composed of rankings $\mathbf{X} = (x_1, x_2, \ldots, x_7)$, such that $x_i$ ranks the alternatives for question $i$ from best to worst, the rankings per objective can be used to compute a relative score that expresses how the participant performs in each of the objectives. This relative score is termed the *player profile score* and is an indication for the decision-making preference of the participants. To compute the profile scores, first the objective rankings of Table B.1 are converted into weights for each alternative. Then, the submitted ranking of alternatives for each question is aggregated into a single score using the multi-criteria decision-making scoring of Roszkowska [220] and Triantaphyllou [248]. These steps are explained in more detail below.

The weighting of alternatives is performed according to the rank-order centroid (ROC) formula proposed initially by Barron and Barrett [22]. ROC is often used in decision-making theory when the relative rank ordering is known but no meaningful quantitative information is available about the alternatives. It has the property of minimising the maximum error of each weight and typically generates weight vectors that are comparable to those produced by panels of subject matter experts. Assuming that the four weights are uniformly distributed, the weight for each alternative is computed as its expected value by $\mathbb{E}(w_j) = 1/n \sum_{k=j}^{n} 1/k$, where $w_j$ is the weight for the alternative at rank position $j$. In the case of four alternatives this yields the weight vector $(.521, .271, .146, .063)$, ordered from best to worst alternative. By combining

the relative rank ordering with the rank weights, the weighted rank score $S_i^m$ per attribute $m$ is determined for every problem $i$ of the questionnaire by simply replacing the ranks $j \in [1, 4]$ of ordering $O_i^m$ by their respective rank weights $w_j$. Thus, the ordering $O_5^p = (2, 3, 1, 4)$ of alternatives of question 5 with respect to expected profits becomes the score vector $S_5^p = (.271, .146, .521, .063)$. Similar substitutions can be performed to generate all alternative weights.

The weighted rank scores enable to measuring and comparing questionnaire responses on a quantified scale. Again a single, complete questionnaire response is denoted by $\mathbf{X} = (x_1, x_2, \ldots, x_7)$, with $x_i$ being the ranking of alternatives for question $i$ as submitted by the participant. As before, a ranking is a vector that for every alternative specifies the preferred order from best (1) to worst (4). Given a ranking $x_i$ for question $i$, the (unscaled) profile score for objective $m$ is then computed by $q_m(x_i) = \sum_{k=1}^{4}(5 - x_{i,k}) \times S_{i,k}^m$, such that $5 - x_{i,k}$ ensures that the first ranked alternative has a weight of 4 and the least preferred option gets a weight of 1. Given a questionnaire response $\mathbf{X}$, the preference score of a participant for objective $m$ is the normalised sum of scores over all questions:

$$\hat{Q}_m(\mathbf{X}) = \frac{\sum_{i=1}^{|\mathbf{X}|} q_m(x_i) - Q_m^{min}}{Q_m^{max} - Q_m^{min}} \tag{B.1}$$

where $Q_m^{min}$ and $Q_m^{max}$ are respectively the minimum and maximum attainable scores for objective $m$ computed over all possible rankings of alternatives. Then from the preference scores over all objectives, the actual profile score can be computed as its relative importance using the formula:

$$Q_m(\mathbf{X}) = \frac{\hat{Q}_m(\mathbf{X})}{\sum_{k \in \{p,t,r\}} \hat{Q}_k(\mathbf{X})} \tag{B.2}$$

Finally, the *rationality* of the questionnaire responses is measured in terms of their distances to the closest Pareto-optimal score and closest minimum score, also known as the normalised Pareto distance [223]. Given a complete questionnaire score $\mathbf{q} = (Q_p(\mathbf{X}), Q_t(\mathbf{X}), Q_r(\mathbf{X}))$ with the symbols $p$ for profit, $t$ for ttl and $r$ for risk-aversion, decision rationality is then expressed as the Euclidean distance of $\mathbf{q}$ to the closest Pareto-optimal score $\mathbf{b}$ inversely related to the sum of Euclidean distances to score $\mathbf{b}$ and closest lowest score $\mathbf{w}$:

$$\theta(\mathbf{q}) = 1 - \frac{||\mathbf{b} - \mathbf{q}||}{||\mathbf{b} - \mathbf{q}|| + ||\mathbf{w} - \mathbf{q}||} \tag{B.3}$$

Here, the closest Pareto scores $\mathbf{b}$ and $\mathbf{w}$ are determined by checking the distance from $\mathbf{q}$ to all other scores. The sets of the worst and best Pareto scores for the questionnaire responses are computed using a simple Java program that can be found in the online code base (`https://github.com/AlgTUDelft/road-maintenance-game`).

## B.2 Player/team Strategy

The in-game actions are scored based on their impact on each of the objectives. Each maintenance alternative available to the players in the game is attributed scores for profit, ttl and risk-aversion, and they are ranked from best to worst, similar to the ranking of the previous section. These rankings are shown in Table B.2.

| Method | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| LOW TTL | 4 | 1 | 2 |
| LOW COST | 1 | 4 | 4 |
| NO RISK | 3 | 3 | 1 |
| FAST | 2 | 2 | 3 |

**Table B.2** *Rank of each maintenance method per objective from best (1) to worst (2).*

From these rankings the *strategy score* for a player is computed from his/her submitted maintenance plan as follows. For a single player $i \in \boldsymbol{N}$, a maintenance plan is given by

$$\mathbf{Y}_k = (\langle m_1^i, t_1^i \rangle, \langle m_2^i, t_2^i \rangle, \langle m_3^i, t_3^i \rangle, \langle m_4^i, t_4^i \rangle)$$

such that each $\langle m_k^i, t_k^i \rangle$ represents the chosen alternative $m_k^i$ and start time $t_k^i$ of maintenance task $k$. The played preference score $G_p$ of player $i$ is then

$$\hat{G_p}(\mathbf{Y}_i) = \frac{\sum_{m_k^i \in \mathbf{Y}_i} g_p(m_k^i) - G_p^{min}}{G_p^{max} - G_p^{min}} \tag{B.4}$$

in which $g_p(m_k^i)$ is the (non-normalised) profit score for method $m_k^i$ of task $k$ and $G_{min}^p$ and $G_{max}^p$ denote respectively the minimum and maximum profit scores attainable in game. As with the profile scores of Section B.1, the profit strategy score $G_p$, ttl strategy score $G_t$ and risk-aversion strategy score $G_r$ are computed relative to the other preferences, thus:

$$G_m(\mathbf{Y}_i) = \frac{\hat{G}_m(\mathbf{Y}_i)}{\sum_{k \in \{p,r,t\}} \hat{G}_k(\mathbf{Y}_i)} \tag{B.5}$$

and strategy scores of a session are aggregated using the average strategy score over all teams.

The performance of players with respect to game outcomes is also scored. To this end, the expected profit, traffic time lost and performance are measured as a function of the former two, similar to the model of Scharpff et al. [228]. The expected profit of a player $i$ given its plan $\mathbf{Y}_i$, denoted by $P_i(\mathbf{Y}_i)$, is defined as the expected reward of completing work minus the expected costs thereof, or

$$P_i(\mathbf{Y}_i) = \sum_{m_k^i \in \mathbf{Y}_i} \left( W(m_k^i) - \sum_{t=t_k^i}^{t_k^i + d(m_k^i)} c(m_k^i, t) - p(m_k^i) \sum_{t=t_k^i}^{t_k^i + \hat{d}(m_k^i)} c(m_k^i, t) \right) \tag{B.6}$$

such that $d(m_k^i)$ and $\hat{d}(m_k^i)$ denote respectively the regular and extended maintenance period, the latter only applies when the task is delayed with probability $p(m_k^i)$. Furthermore, $W(m_k^i)$ is the fixed, contracted reward received upon completion of the task associated with $m_k^i$ (thus independent from the chosen method) and $c(m_k^i, t)$ the maintenance cost of performing method $m_k^i$ at time $t$.[55]

For the traffic time lost $T_i(\mathbf{Y})$ caused by player $i$, given *joint* plan $\mathbf{Y} = \bigcup_{i \in \boldsymbol{N}} \mathbf{Y}_i$ with the set of players $\boldsymbol{N} = \{1, 2, \ldots, 5\}$, a similar expected value computation is made. Notice that for the computation of ttl a joint plan is required as concurrent maintenance can have super-linear impact on traffic. The ttl model of the game is defined through a function $\ell_i(\mathbf{Y}, t)$ that returns the ttl caused by player $i \subseteq \boldsymbol{N}$ at time $t$ when joint plan $\mathbf{Y}$ is executed. Therefore, the total ttl caused by an individual player $i$ is given by[56]

$$T_i(\mathbf{Y}) = \sum_{m_k^i \in \mathbf{Y}_i} \Big( \sum_{t=t_k^i}^{t_k^i + d(m_k^i)} \ell_i(\mathbf{Y}, t) + p(m_k^i) \sum_{t=t_k^i}^{t_k^i + \hat{d}(m_k^i)} \ell_i(\mathbf{Y}, t) \Big) \tag{B.7}$$

With the aforementioned formulas the expected utility of a player $i$ is expressed as the sum of its expected revenue minus the monetary value of the expected ttl. Consequentially, expected profit for a player $i$ given a joint plan $\mathbf{Y}$, such that $\mathbf{Y}_i \in \mathbf{Y}$ is the plan of player $i$, is given by $u_i(\mathbf{Y}) = P_i(\mathbf{Y}_i) - T_i(\mathbf{Y})$. Finally, similar to the decision rationality of profile scores, an indication of the quality of the in-game decisions can be defined over the strategy scores. The *performance ratio* $\phi(\mathbf{Y})$ for a given joint plan $\mathbf{Y}$ expresses the ratio between profit and ttl:

$$\phi(\mathbf{Y}) = \frac{P(\mathbf{Y})}{T(\mathbf{Y})} \tag{B.8}$$

and observe that this value increases either when the joint profit increases, the joint ttl decreases or both. Hence a higher performance ratio indicates a better overall outcome.

[55] *The reward and cost functions have no player index as the underlying model is the same.*

[56] *No time step $t$ is counted more than once due to the one-task-at-a-time restriction.*

# Appendix C

# Game Session Outcomes

This chapter summarises the results of all measurements taken from this session, both a priori as well as during the session. In total, 7 gaming sessions have been performed with 95 players from various ages, institutions and backgrounds. Each of the sessions is given a letter for identification purpose and the characteristics of these sessions are listed in Table C.1.

| | Company/institute Profile | #P | #Q | Category | Coordination | Cohesion |
|---|---|---|---|---|---|---|
| A | University, Computer Science | 9 | 9 | Students | Low | Unfamiliar |
| B | ICT-focused R&D Company | 10 | 9 | Engineers | Low | Familiar |
| C | Utility provider, mainly power | 15 | 3 | Professionals | Low | Unfamiliar |
| D | Dutch national road authority | 17 | 16 | Trainees | High | Familiar |
| E | Dutch national road authority | 8 | 5 | Trainees | Medium | Familiar |
| F | AM Professionals Course | 20 | 9 | Professionals | Medium | Unfamiliar |
| G | AM and Health-care Consultants | 16 | 9 | Professionals | High | Familiar |

**Table C.1** *Outline of game session characteristics, from left to right the columns are: session identifier, company/institute, number of participants and questionnaire responses, participants skill category, and the observed coordination and social cohesion levels.*

## C.1 Questionnaire Responses

From the 95 participants, 59 valid questionnaires were collected. All of these responses have been scored according to Equation B.2 and are listed in Table C.2 on the next page. The columns capture respectively the session name, the computed profit, ttl and risk-aversion profile scores ($Q_p$, $Q_t$ and $Q_r$), and the decision rationality $\theta$. The average profile score of the session is included in the bottom row, included only for reference and is not used. The rationality scores are found using Equation B.3 of Appendix B, where the optimal Pareto trade-offs have been computed using a Java program that can be found in the repository at `https://github.com/AlgTUDelft/road-maintenance-game`.

| | $Q_p$ | $Q_t$ | $Q_r$ | $\theta$ |
|---|---|---|---|---|
| **Session A** | | | | |
| | 0.455 | 0.336 | 0.209 | 0.795 |
| | 0.436 | 0.352 | 0.212 | 0.802 |
| | 0.481 | 0.274 | 0.244 | 0.855 |
| | 0.440 | 0.338 | 0.221 | 0.815 |
| | 0.322 | 0.373 | 0.305 | 0.932 |
| | 0.475 | 0.350 | 0.175 | 0.730 |
| | 0.465 | 0.321 | 0.213 | 0.773 |
| | 0.385 | 0.384 | 0.231 | 0.845 |
| | 0.253 | 0.464 | 0.284 | 0.385 |
| *avg* | *0.412* | *0.355* | *0.233* | *0.770* |
| **Session B** | | | | |
| | 0.244 | 0.375 | 0.382 | 0.955 |
| | 0.459 | 0.343 | 0.198 | 0.789 |
| | 0.230 | 0.401 | 0.369 | 0.883 |
| | 0.244 | 0.460 | 0.296 | 0.691 |
| | 0.277 | 0.449 | 0.274 | 0.878 |
| | 0.375 | 0.375 | 0.250 | 0.897 |
| | 0.369 | 0.395 | 0.236 | 0.870 |
| | 0.360 | 0.380 | 0.259 | 0.904 |
| | 0.459 | 0.343 | 0.198 | 0.789 |
| *avg* | *0.335* | *0.391* | *0.274* | *0.851* |
| **Session C** | | | | |
| | 0.379 | 0.367 | 0.254 | 0.913 |
| | 0.428 | 0.321 | 0.251 | 0.849 |
| | 0.329 | 0.358 | 0.314 | 0.962 |
| *avg* | *0.379* | *0.348* | *0.273* | *0.908* |
| **Session D** | | | | |
| | 0.272 | 0.358 | 0.370 | 0.958 |
| | 0.401 | 0.345 | 0.254 | 0.255 |
| | 0.455 | 0.336 | 0.209 | 0.795 |
| | 0.459 | 0.343 | 0.198 | 0.789 |
| | 0.426 | 0.355 | 0.219 | 0.884 |
| | 0.398 | 0.375 | 0.227 | 0.855 |
| | 0.350 | 0.417 | 0.233 | 0.877 |
| | 0.326 | 0.378 | 0.297 | 0.916 |
| | 0.327 | 0.377 | 0.296 | 0.982 |
| | 0.381 | 0.384 | 0.235 | 0.853 |
| | 0.431 | 0.342 | 0.227 | 0.880 |
| | 0.335 | 0.447 | 0.218 | 0.710 |
| | 0.266 | 0.319 | 0.415 | 0.838 |
| | 0.347 | 0.400 | 0.253 | 0.907 |
| | 0.336 | 0.404 | 0.260 | 0.938 |
| | 0.349 | 0.264 | 0.387 | 0.974 |
| *avg* | *0.366* | *0.365* | *0.269* | *0.838* |

| | $Q_p$ | $Q_t$ | $Q_r$ | $\theta$ |
|---|---|---|---|---|
| **Session E** | | | | |
| | 0.386 | 0.341 | 0.273 | 0.948 |
| | 0.379 | 0.367 | 0.254 | 0.913 |
| | 0.420 | 0.313 | 0.267 | 0.815 |
| | 0.473 | 0.315 | 0.212 | 0.845 |
| | 0.463 | 0.314 | 0.223 | 0.891 |
| *avg* | *0.424* | *0.330* | *0.246* | *0.882* |
| **Session F** | | | | |
| | 0.307 | 0.457 | 0.236 | 0.713 |
| | 0.432 | 0.260 | 0.308 | 0.862 |
| | 0.483 | 0.338 | 0.178 | 0.715 |
| | 0.459 | 0.343 | 0.198 | 0.789 |
| | 0.190 | 0.413 | 0.397 | 0.802 |
| | 0.325 | 0.431 | 0.243 | 0.887 |
| | 0.383 | 0.336 | 0.281 | 0.987 |
| | 0.337 | 0.425 | 0.237 | 0.731 |
| | 0.433 | 0.316 | 0.251 | 0.827 |
| *avg* | *0.372* | *0.369* | *0.259* | *0.813* |
| **Session G** | | | | |
| | 0.380 | 0.355 | 0.266 | 0.933 |
| | 0.340 | 0.393 | 0.266 | 0.933 |
| | 0.467 | 0.336 | 0.197 | 0.790 |
| | 0.353 | 0.377 | 0.270 | 0.932 |
| | 0.311 | 0.438 | 0.252 | 0.879 |
| | 0.350 | 0.340 | 0.310 | 0.961 |
| | 0.254 | 0.378 | 0.368 | 0.963 |
| | 0.278 | 0.446 | 0.276 | 0.886 |
| | 0.394 | 0.347 | 0.260 | 0.922 |
| *avg* | *0.347* | *0.379* | *0.274* | *0.911* |

**Table C.2** *Complete overview of questionnaire profile scores, grouped per session. For each response the profit, ttl and risk-aversion profile scores are computed (resp. $Q_p$, $Q_t$ and $Q_r$) and the decision rationality $\theta$ according to Equation B.3.*

## C.2 Session Outcomes

The in-game results are listed as a single table per game session. Each table contains multiple sub-tables, one for every round played in the game, and the listed figures are the values measured exactly when all players submitted their plan. Per round, the tables list for both the profit and ttl objectives the maximum value that can be obtained, the maximum impact of delay on that value and the expected value. For example, at the end of round $1$ of session $A$, the Red player can potentially achieve a maximum profit of € $5,425$. If the player is really unlucky and all of its activities are delayed, its profit decreases by € $6,464$, resulting in a total loss of € $1,039$. In expectation, however, its profit is € $3,270$ which is of course much better than the worst-case scenario. The ttl columns are similar but for the fact that the figure in the delay column is *added* to the ttl figure in the case of delay. Note that the score listed in the last round is the score before the execution starts.

| Session A | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 4.878 | -6.748 | 2.629 | 6.179 | 2.721 | 7.086 |
| Blue | 3.273 | -3.446 | 2.124 | 6.390 | 1.107 | 6.759 |
| Pink | 2.907 | -4.028 | 1.564 | 6.216 | 893 | 6.514 |
| Red | 5.425 | -6.464 | 3.270 | 5.632 | 2.438 | 6.445 |
| White | 5.347 | -5.647 | 3.465 | 5.158 | 1.441 | 5.638 |
| *Total* | 21.830 | -26.333 | 13.052 | 29.575 | 8.600 | 32.442 |
| **Round 2** | | | | | | |
| Black | 4.878 | -6.748 | 2.629 | 5.512 | 2.735 | 6.424 |
| Blue | 5.646 | -6.094 | 3.615 | 5.095 | 2.578 | 5.954 |
| Pink | 5.745 | -5.796 | 3.813 | 5.799 | 2.139 | 6.512 |
| Red | 5.456 | -6.279 | 3.363 | 550 | 2.177 | 1.276 |
| White | 6.183 | -6.208 | 4.114 | 4.322 | 2.001 | 4.989 |
| *Total* | 27.908 | -31.125 | 17.533 | 21.278 | 11.630 | 25.155 |
| **Round 3** | | | | | | |
| Black | 4.989 | -6.226 | 2.914 | 5.912 | 2.387 | 6.708 |
| Blue | 5.646 | -6.094 | 3.615 | 6.264 | 1.711 | 6.834 |
| Pink | 6.449 | -6.019 | 4.443 | 5.029 | 2.955 | 6.014 |
| Red | 7.242 | -7.292 | 4.811 | 5.590 | 3.246 | 6.672 |
| White | 5.927 | -5.010 | 4.257 | 4.816 | 1.854 | 5.434 |
| *Total* | 30.253 | -30.641 | 20.039 | 27.611 | 12.153 | 31.662 |

| Session B | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 3.918 | -5.986 | 1.923 | 5.766 | 1.946 | 6.415 |
| Blue | 3.696 | -5.289 | 1.933 | 7.257 | 2.079 | 7.950 |
| Pink | 4.269 | -6.521 | 2.095 | 6.824 | 2.506 | 7.659 |
| Red | 5.842 | -7.297 | 3.410 | 6.287 | 3.234 | 7.365 |
| White | 88 | -3.873 | -1.203 | 6.934 | 984 | 7.262 |
| *Total* | 17.813 | -28.966 | 8.158 | 33.068 | 10.749 | 36.651 |
| **Round 2** | | | | | | |
| Black | 4.688 | -5.636 | 2.809 | 3.240 | 1.158 | 3.626 |
| Blue | 4.089 | -4.935 | 2.444 | 4.639 | 1.726 | 5.214 |
| Pink | 4.269 | -6.521 | 2.095 | 3.753 | 2.389 | 4.549 |
| Red | 6.971 | -6.352 | 4.854 | 4.968 | 2.832 | 5.912 |
| White | 1.519 | -3.771 | 262 | 4.450 | 841 | 4.730 |
| *Total* | 21.536 | -27.215 | 12.464 | 21.050 | 8.946 | 24.032 |
| **Round 3** | | | | | | |
| Black | 5.095 | -5.472 | 3.271 | 3.144 | 803 | 3.412 |
| Blue | 6.693 | -6.808 | 4.424 | 5.673 | 2.148 | 6.389 |
| Pink | 6.491 | -6.176 | 4.432 | 4.569 | 1.975 | 5.227 |
| Red | 7.762 | -6.724 | 5.521 | 4.166 | 2.588 | 5.029 |
| White | 1.899 | -4.043 | 551 | 5.123 | 1.144 | 5.504 |
| *Total* | 27.940 | -29.223 | 18.199 | 22.675 | 8.658 | 25.561 |
| **Round 4** | | | | | | |
| Black | 5.095 | -5.472 | 3.271 | 3.086 | 1.009 | 3.422 |
| Blue | 7.565 | -6.492 | 5.401 | 4.825 | 2.379 | 5.618 |
| Pink | 6.927 | -5.927 | 4.951 | 4.591 | 1.820 | 5.198 |
| Red | 8.005 | -6.469 | 5.849 | 4.465 | 2.277 | 5.224 |
| White | 1.960 | -3.943 | 646 | 5.045 | 998 | 5.378 |
| *Total* | 29.552 | -28.303 | 20.118 | 22.012 | 8.483 | 24.840 |

| Session C | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 6.797 | -6.692 | 4.566 | 5.837 | 2.579 | 6.697 |
| Blue | 5.602 | -6.532 | 3.425 | 5.659 | 2.404 | 6.460 |
| Pink | 5.040 | -6.427 | 2.898 | 4.793 | 2.382 | 5.587 |
| Red | 3.377 | -4.433 | 1.899 | 5.747 | 1.650 | 6.297 |
| White | 6.204 | -6.810 | 3.934 | 6.429 | 2.697 | 7.328 |
| *Total* | 27.020 | -30.894 | 16.722 | 28.465 | 11.712 | 32.369 |
| **Round 2** | | | | | | |
| Black | 6.797 | -6.692 | 4.566 | 5.744 | 2.271 | 6.501 |
| Blue | 5.859 | -6.329 | 3.749 | 5.402 | 2.277 | 6.161 |
| Pink | 5.741 | -6.449 | 3.591 | 4.058 | 2.000 | 4.725 |
| Red | 3.377 | -4.433 | 1.899 | 5.271 | 1.650 | 5.821 |
| White | 6.978 | -6.742 | 4.731 | 5.660 | 2.207 | 6.396 |
| *Total* | 28.752 | -30.645 | 18.537 | 26.135 | 10.405 | 29.603 |
| **Round 3** | | | | | | |
| Black | 5.479 | -4.901 | 3.845 | 5.097 | 1.482 | 5.591 |
| Blue | 7.163 | -6.599 | 4.963 | 5.212 | 2.086 | 5.907 |
| Pink | 6.524 | -6.636 | 4.312 | 3.791 | 1.858 | 4.410 |
| Red | 3.566 | -3.292 | 2.469 | 4.565 | 1.092 | 4.929 |
| White | 6.978 | -6.742 | 4.731 | 4.882 | 2.245 | 5.630 |
| *Total* | 29.710 | -28.170 | 20.320 | 23.547 | 8.763 | 26.468 |
| **Round 4** | | | | | | |
| Black | 5.662 | -5.224 | 3.921 | 4.711 | 1.820 | 5.318 |
| Blue | 6.855 | -4.857 | 5.236 | 4.579 | 1.369 | 5.035 |
| Pink | 7.691 | -6.654 | 5.473 | 4.055 | 2.084 | 4.750 |
| Red | 3.566 | -3.292 | 2.469 | 4.697 | 1.044 | 5.045 |
| White | 8.100 | -6.179 | 6.040 | 4.951 | 2.195 | 5.683 |
| *Total* | 31.874 | -26.206 | 23.139 | 22.993 | 8.512 | 25.830 |

| Session D | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 2.582 | -1.326 | 2.140 | 4.729 | 361 | 4.849 |
| Blue | 6.044 | -5.722 | 4.137 | 4.461 | 1.515 | 4.966 |
| Pink | 3.825 | -2.941 | 2.845 | 4.158 | 846 | 4.440 |
| Red | 7.898 | -6.326 | 5.789 | 4.735 | 2.212 | 5.472 |
| White | 5.986 | -4.675 | 4.428 | 5.489 | 1.462 | 5.976 |
| *Total* | 26.335 | -20.990 | 19.338 | 23.572 | 6.396 | 25.704 |
| **Round 2** | | | | | | |
| Black | 3.583 | -1.326 | 3.141 | 2.657 | 204 | 2.725 |
| Blue | 6.241 | -5.853 | 4.290 | 2.972 | 920 | 3.279 |
| Pink | 3.249 | -2.248 | 2.500 | 3.078 | 486 | 3.240 |
| Red | 5.884 | -5.033 | 4.206 | 2.541 | 964 | 2.862 |
| White | 4.010 | -3.498 | 2.844 | 3.012 | 609 | 3.215 |
| *Total* | 22.967 | -17.958 | 16.981 | 14.260 | 3.183 | 15.321 |
| **Round 3** | | | | | | |
| Black | 3.702 | -1.326 | 3.260 | 3.171 | 204 | 3.239 |
| Blue | 6.241 | -5.853 | 4.290 | 2.774 | 1.110 | 3.144 |
| Pink | 3.011 | -2.248 | 2.262 | 3.312 | 446 | 3.461 |
| Red | 6.152 | -5.329 | 4.376 | 3.141 | 1.324 | 3.582 |
| White | 4.418 | -3.760 | 3.165 | 3.193 | 678 | 3.419 |
| *Total* | 23.524 | -18.516 | 17.352 | 15.591 | 3.762 | 16.845 |

| Session E | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 5.993 | -4.215 | 4.588 | 4.223 | 973 | 4.547 |
| Blue | 3.678 | -1.835 | 3.066 | 3.932 | 482 | 4.093 |
| Pink | 5.712 | -4.779 | 4.119 | 4.084 | 1.173 | 4.475 |
| Red | 5.001 | -3.865 | 3.713 | 4.291 | 970 | 4.614 |
| White | 4.512 | -5.109 | 2.809 | 3.996 | 1.263 | 4.417 |
| *Total* | 24.896 | -19.803 | 18.295 | 20.526 | 4.861 | 22.146 |
| **Round 2** | | | | | | |
| Black | 5.993 | -4.215 | 4.588 | 3.751 | 923 | 4.059 |
| Blue | 4.417 | -1.726 | 3.842 | 3.218 | 403 | 3.352 |
| Pink | 5.568 | -4.493 | 4.070 | 3.925 | 850 | 4.208 |
| Red | 5.802 | -5.005 | 4.134 | 3.727 | 1.205 | 4.129 |
| White | 5.006 | -5.026 | 3.331 | 3.378 | 892 | 3.675 |
| *Total* | 26.786 | -20.465 | 19.964 | 17.999 | 4.273 | 19.423 |
| **Round 3** | | | | | | |
| Black | 5.924 | -4.187 | 4.528 | 3.420 | 1.025 | 3.762 |
| Blue | 4.561 | -1.700 | 3.994 | 3.267 | 347 | 3.383 |
| Pink | 6.377 | -4.678 | 4.818 | 3.307 | 1.054 | 3.658 |
| Red | 6.032 | -5.034 | 4.354 | 3.722 | 983 | 4.050 |
| White | 5.006 | -5.023 | 3.332 | 3.580 | 720 | 3.820 |
| *Total* | 27.900 | -20.622 | 21.026 | 17.296 | 4.129 | 18.672 |
| **Round 4** | | | | | | |
| Black | 5.790 | -4.243 | 4.376 | 3.569 | 1.015 | 3.907 |
| Blue | 6.688 | -3.910 | 5.385 | 3.355 | 698 | 3.588 |
| Pink | 6.377 | -4.678 | 4.818 | 3.179 | 1.114 | 3.550 |
| Red | 6.413 | -5.156 | 4.694 | 3.660 | 1.156 | 4.045 |
| White | 5.006 | -5.026 | 3.331 | 3.675 | 826 | 3.950 |
| *Total* | 30.274 | -23.013 | 22.603 | 17.438 | 4.809 | 19.041 |

| Session E (2) | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 5** | | | | | | |
| Black | 5.790 | -4.243 | 4.376 | 3.651 | 1.015 | 3.989 |
| Blue | 6.655 | -4.065 | 5.300 | 3.501 | 853 | 3.785 |
| Pink | 6.377 | -4.678 | 4.818 | 3.230 | 1.176 | 3.622 |
| Red | 6.413 | -5.156 | 4.694 | 3.722 | 1.156 | 4.107 |
| White | 5.006 | -5.026 | 3.331 | 3.643 | 826 | 3.918 |
| *Total* | 30.241 | -23.168 | 22.518 | 17.747 | 5.025 | 19.422 |
| **Round 6** | | | | | | |
| Black | 5.790 | -4.243 | 4.376 | 3.805 | 1.180 | 4.198 |
| Blue | 6.547 | -3.955 | 5.229 | 3.706 | 764 | 3.965 |
| Pink | 7.406 | -5.464 | 5.585 | 3.521 | 1.585 | 4.049 |
| Red | 6.413 | -5.156 | 4.694 | 3.794 | 1.171 | 4.184 |
| White | 4.899 | -5.077 | 3.207 | 3.681 | 851 | 3.965 |
| *Total* | 31.055 | -23.895 | 23.090 | 18.507 | 5.551 | 20.357 |
| **Round 7** | | | | | | |
| Black | 5.779 | -4.348 | 4.330 | 4.715 | 1.297 | 5.147 |
| Blue | 6.796 | -4.164 | 5.408 | 4.686 | 1.189 | 5.082 |
| Pink | 7.406 | -5.464 | 5.585 | 4.143 | 1.849 | 4.759 |
| Red | 7.893 | -5.949 | 5.910 | 5.261 | 1.859 | 5.881 |
| White | 4.899 | -5.077 | 3.207 | 4.296 | 1.188 | 4.692 |
| *Total* | 32.773 | -25.002 | 24.439 | 23.101 | 7.382 | 25.562 |

| Session F | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 4.008 | -4.699 | 2.442 | 6.208 | 1.457 | 6.694 |
| Blue | 5.119 | -6.227 | 3.043 | 5.386 | 2.020 | 6.059 |
| Pink | 4.752 | -4.061 | 3.398 | 5.044 | 1.115 | 5.416 |
| Red | 5.480 | -6.861 | 3.193 | 6.481 | 2.910 | 7.451 |
| White | 5.242 | -5.183 | 3.514 | 5.712 | 1.974 | 6.370 |
| *Total* | 24.601 | -27.031 | 15.591 | 28.831 | 9.476 | 31.990 |
| **Round 2** | | | | | | |
| Black | 5.625 | -4.619 | 4.085 | 3.323 | 1.378 | 3.782 |
| Blue | 5.420 | -6.081 | 3.393 | 4.317 | 1.492 | 4.814 |
| Pink | 6.556 | -4.204 | 5.155 | 3.766 | 963 | 4.087 |
| Red | 7.267 | -6.324 | 5.159 | 4.912 | 1.488 | 5.408 |
| White | 4.752 | -5.097 | 3.053 | 3.914 | 1.477 | 4.406 |
| *Total* | 29.620 | -26.325 | 20.845 | 20.232 | 6.798 | 22.498 |
| **Round 3** | | | | | | |
| Black | 5.625 | -4.619 | 4.085 | 4.054 | 1.143 | 4.435 |
| Blue | 7.308 | -6.260 | 5.221 | 3.814 | 1.395 | 4.279 |
| Pink | 6.556 | -4.204 | 5.155 | 4.524 | 1.068 | 4.880 |
| Red | 7.097 | -6.453 | 4.946 | 4.763 | 1.743 | 5.344 |
| White | 6.463 | -5.936 | 4.484 | 4.170 | 1.836 | 4.782 |
| *Total* | 33.049 | -27.472 | 23.892 | 21.325 | 7.185 | 23.720 |
| **Round 4** | | | | | | |
| Black | 5.588 | -5.216 | 3.849 | 4.911 | 2.200 | 5.644 |
| Blue | 7.308 | -6.260 | 5.221 | 4.745 | 2.013 | 5.416 |
| Pink | 5.628 | -4.385 | 4.166 | 5.230 | 1.446 | 5.712 |
| Red | 7.864 | -6.807 | 5.595 | 5.550 | 2.207 | 6.286 |
| White | 6.720 | -6.560 | 4.533 | 5.512 | 2.607 | 6.381 |
| *Total* | 33.108 | -29.228 | 23.365 | 25.948 | 10.473 | 29.439 |

| Session G | $P$ | $P_{del}$ | $\mathbb{E}[P]$ | $T$ | $T_{del}$ | $\mathbb{E}[T]$ |
|---|---|---|---|---|---|---|
| **Round 1** | | | | | | |
| Black | 4.415 | -6.525 | 2.240 | 8.219 | 2.412 | 9.023 |
| Blue | 3.683 | -6.793 | 1.419 | 6.822 | 2.586 | 7.684 |
| Pink | 4.716 | -6.798 | 2.450 | 7.917 | 2.684 | 8.812 |
| Red | 3.972 | -5.413 | 2.168 | 7.504 | 2.200 | 8.237 |
| White | 1.851 | -5.718 | -55 | 6.405 | 2.069 | 7.095 |
| *Total* | 18.637 | -31.247 | 8.221 | 36.867 | 11.951 | 40.851 |
| **Round 2** | | | | | | |
| Black | 6.342 | -5.908 | 4.373 | 4.414 | 1.413 | 4.885 |
| Blue | 4.323 | -3.642 | 3.109 | 3.870 | 851 | 4.154 |
| Pink | 5.947 | -3.488 | 4.784 | 3.716 | 1.149 | 4.099 |
| Red | 6.455 | -4.536 | 4.943 | 3.630 | 1.262 | 4.051 |
| White | 4.383 | -5.228 | 2.640 | 3.522 | 1.112 | 3.893 |
| *Total* | 27.450 | -22.802 | 19.849 | 19.152 | 5.787 | 21.081 |
| **Round 3** | | | | | | |
| Black | 6.194 | -3.813 | 4.923 | 3.909 | 581 | 4.103 |
| Blue | 5.456 | -2.990 | 4.459 | 3.553 | 800 | 3.820 |
| Pink | 6.022 | -3.793 | 4.758 | 4.102 | 570 | 4.291 |
| Red | 7.081 | -4.437 | 5.602 | 3.804 | 863 | 4.092 |
| White | 6.216 | -4.414 | 4.745 | 3.918 | 1.120 | 4.291 |
| *Total* | 30.969 | -19.447 | 24.487 | 19.286 | 3.934 | 20.597 |
| **Round 4** | | | | | | |
| Black | 6.309 | -3.813 | 5.038 | 3.793 | 581 | 3.987 |
| Blue | 5.456 | -2.990 | 4.459 | 3.649 | 724 | 3.890 |
| Pink | 6.022 | -3.793 | 4.758 | 3.847 | 540 | 4.027 |
| Red | 7.081 | -4.437 | 5.602 | 3.453 | 1.068 | 3.809 |
| White | 5.299 | -4.012 | 3.962 | 3.313 | 939 | 3.626 |
| *Total* | 30.167 | -19.045 | 23.819 | 18.055 | 3.852 | 19.339 |

**Table C.3** *Outcomes at the end of every round per team and total for the session. The first three data columns represent the maximum profit $P$, the potential profit loss due to uncertainties $P_{del}$ and the expected profit $\mathbb{E}[P]$. Similarly, the columns $T$, $T_{del}$ and $\mathbb{E}[T]$ show the minimum ttl, the potential increase in ttl due to delays and the expected ttl $T_{wc}$. Note that the utility is not listed but can easily be computed as $u = P - T$ as every 1 hour of ttl corresponds to 1 € of penalty to the players (and similarly for expected utility).*

Table C.4 contains a summarised overview of the previous detailed session outcome listing per player. The columns R1 to R7 represent the rounds of the game. The profit and utility are the total session values in thousands of euros, the ttl are the session totals in hours.

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | min | max | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Profit** | | | | | | | | | | |
| A | 45.494 | 42.688 | 51.701 | | | | | 42.688 | 51.701 | 46.628 |
| B | 44.809 | 36.496 | 43.760 | 44.957 | | | | 36.496 | 44.957 | 42.506 |
| C | 49.091 | 48.140 | 46.788 | 48.969 | | | | 46.788 | 49.091 | 48.247 |
| D | 45.042 | 32.302 | 34.197 | | | | | 32.302 | 45.042 | 37.180 |
| E | 40.441 | 39.388 | 39.698 | 41.644 | 41.941 | 43.447 | 50.001 | 39.388 | 50.001 | 42.366 |
| F | 47.580 | 43.343 | 47.612 | 52.804 | | | | 43.343 | 52.804 | 47.835 |
| G | 49.072 | 40.930 | 45.084 | 43.158 | | | | 40.930 | 49.072 | 44.561 |
| **TTL** | | | | | | | | | | |
| A | 32.442 | 25.155 | 31.662 | | | | | 25.155 | 32.442 | 29.753 |
| B | 36.651 | 24.032 | 25.561 | 24.840 | | | | 24.032 | 36.651 | 27.771 |
| C | 32.369 | 29.603 | 26.468 | 25.830 | | | | 25.830 | 32.369 | 28.568 |
| D | 25.704 | 15.321 | 16.845 | | | | | 15.321 | 25.704 | 19.290 |
| E | 22.146 | 19.423 | 18.672 | 19.041 | 19.422 | 20.357 | 25.562 | 18.672 | 25.562 | 20.661 |
| F | 31.990 | 22.498 | 23.720 | 29.439 | | | | 22.498 | 31.990 | 26.912 |
| G | 40.851 | 21.081 | 20.597 | 19.339 | | | | 19.339 | 40.851 | 25.467 |
| **Utility** | | | | | | | | | | |
| A | 13.052 | 17.533 | 20.039 | | | | | 13.052 | 20.039 | 16.875 |
| B | 8.158 | 12.464 | 18.199 | 20.118 | | | | 8.158 | 20.118 | 14.735 |
| C | 16.722 | 18.537 | 20.320 | 23.139 | | | | 16.722 | 23.139 | 19.679 |
| D | 19.338 | 16.981 | 17.352 | | | | | 16.981 | 19.338 | 17.890 |
| E | 18.295 | 19.964 | 21.026 | 22.603 | 22.518 | 23.090 | 24.439 | 18.295 | 24.439 | 21.705 |
| F | 15.591 | 20.845 | 23.892 | 23.365 | | | | 15.591 | 23.892 | 20.923 |
| G | 8.221 | 19.849 | 24.487 | 23.819 | | | | 8.221 | 24.487 | 19.094 |

**Table C.4** *Outcomes per objective summed over all players in the session per round and summarised over all rounds. Note that every gaming session is presented in its own table.*

## C.3 Strategy Scores

| Session A | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 0.786 | 0.143 | 0.071 |
| Blue | 0.563 | 0.188 | 0.250 |
| Pink | 0.222 | 0.389 | 0.389 |
| Red | 0.786 | 0.143 | 0.071 |
| White | 0.625 | 0.250 | 0.125 |
| *avg.* | 0.596 | 0.222 | 0.181 |
| **Round 2** | | | |
| Black | 0.786 | 0.143 | 0.071 |
| Blue | 0.563 | 0.188 | 0.250 |
| Pink | 0.786 | 0.143 | 0.071 |
| Red | 0.786 | 0.143 | 0.071 |
| White | 0.625 | 0.250 | 0.125 |
| *avg.* | 0.709 | 0.173 | 0.118 |
| **Round 3** | | | |
| Black | 0.786 | 0.143 | 0.071 |
| Blue | 0.563 | 0.188 | 0.250 |
| Pink | 0.786 | 0.143 | 0.071 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.563 | 0.188 | 0.250 |
| *avg.* | 0.739 | 0.132 | 0.129 |

| Session B | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 0.500 | 0.333 | 0.167 |
| Blue | 0.714 | 0.071 | 0.214 |
| Pink | 0.500 | 0.313 | 0.188 |
| Red | 0.786 | 0.143 | 0.071 |
| White | 0.000 | 0.600 | 0.400 |
| *avg.* | 0.500 | 0.292 | 0.208 |
| **Round 2** | | | |
| Black | 0.400 | 0.400 | 0.200 |
| Blue | 0.714 | 0.071 | 0.214 |
| Pink | 0.500 | 0.313 | 0.188 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.000 | 0.600 | 0.400 |
| *avg.* | 0.523 | 0.277 | 0.200 |
| **Round 3** | | | |
| Black | 0.400 | 0.400 | 0.200 |
| Blue | 1.000 | 0.000 | 0.000 |
| Pink | 0.786 | 0.143 | 0.071 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.000 | 0.600 | 0.400 |
| *avg.* | 0.637 | 0.229 | 0.134 |
| **Round 4** | | | |
| Black | 0.400 | 0.400 | 0.200 |
| Blue | 1.000 | 0.000 | 0.000 |
| Pink | 0.786 | 0.143 | 0.071 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.000 | 0.600 | 0.400 |
| *avg.* | 0.637 | 0.229 | 0.134 |

| Session C | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 1.000 | 0.000 | 0.000 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.222 | 0.389 | 0.389 |
| White | 1.000 | 0.000 | 0.000 |
| *avg.* | 0.647 | 0.206 | 0.147 |
| **Round 2** | | | |
| Black | 1.000 | 0.000 | 0.000 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.222 | 0.389 | 0.389 |
| White | 1.000 | 0.000 | 0.000 |
| *avg.* | 0.647 | 0.206 | 0.147 |
| **Round 3** | | | |
| Black | 0.714 | 0.071 | 0.214 |
| Blue | 1.000 | 0.000 | 0.000 |
| Pink | 0.625 | 0.250 | 0.125 |
| Red | 0.278 | 0.278 | 0.444 |
| White | 1.000 | 0.000 | 0.000 |
| *avg.* | 0.723 | 0.120 | 0.157 |
| **Round 4** | | | |
| Black | 0.714 | 0.071 | 0.214 |
| Blue | 0.714 | 0.071 | 0.214 |
| Pink | 0.786 | 0.143 | 0.071 |
| Red | 0.278 | 0.278 | 0.444 |
| White | 1.000 | 0.000 | 0.000 |
| *avg.* | 0.698 | 0.113 | 0.189 |

| Session D | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 0.250 | 0.250 | 0.500 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.333 | 0.333 | 0.333 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.714 | 0.071 | 0.214 |
| *avg.* | 0.585 | 0.181 | 0.235 |
| **Round 2** | | | |
| Black | 0.250 | 0.250 | 0.500 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.200 | 0.350 | 0.450 |
| Red | 0.400 | 0.400 | 0.200 |
| White | 0.000 | 0.600 | 0.400 |
| *avg.* | 0.295 | 0.370 | 0.335 |
| **Round 3** | | | |
| Black | 0.250 | 0.250 | 0.500 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.200 | 0.350 | 0.450 |
| Red | 0.500 | 0.333 | 0.167 |
| White | 0.100 | 0.550 | 0.350 |
| *avg.* | 0.335 | 0.347 | 0.318 |

| Session E | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.300 | 0.300 | 0.400 |
| Pink | 0.375 | 0.375 | 0.250 |
| Red | 0.333 | 0.333 | 0.333 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.392 | 0.317 | 0.291 |
| **Round 2** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.300 | 0.300 | 0.400 |
| Pink | 0.375 | 0.375 | 0.250 |
| Red | 0.389 | 0.389 | 0.222 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.403 | 0.328 | 0.269 |
| **Round 3** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.300 | 0.300 | 0.400 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.389 | 0.389 | 0.222 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.406 | 0.331 | 0.263 |
| **Round 4** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.500 | 0.313 | 0.188 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.457 | 0.311 | 0.232 |
| **Round 5** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.500 | 0.313 | 0.188 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.457 | 0.311 | 0.232 |
| **Round 6** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 0.500 | 0.313 | 0.188 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.504 | 0.283 | 0.213 |
| **Round 7** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.563 | 0.188 | 0.250 |
| Pink | 0.389 | 0.389 | 0.222 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 0.389 | 0.389 | 0.222 |
| *avg.* | 0.628 | 0.203 | 0.169 |

| Session F | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.438 | 0.250 | 0.313 |
| Red | 0.643 | 0.214 | 0.143 |
| White | 0.714 | 0.071 | 0.214 |
| *avg.* | 0.596 | 0.195 | 0.209 |
| **Round 2** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 0.643 | 0.214 | 0.143 |
| White | 0.563 | 0.188 | 0.250 |
| *avg.* | 0.591 | 0.205 | 0.204 |
| **Round 3** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 0.643 | 0.214 | 0.143 |
| White | 0.786 | 0.143 | 0.071 |
| *avg.* | 0.636 | 0.196 | 0.168 |
| **Round 4** | | | |
| Black | 0.714 | 0.071 | 0.214 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 1.000 | 0.000 | 0.000 |
| White | 1.000 | 0.000 | 0.000 |
| *avg.* | 0.780 | 0.102 | 0.118 |

| Session G | $G_p$ | $G_t$ | $G_r$ |
|---|---|---|---|
| **Round 1** | | | |
| Black | 1.000 | 0.000 | 0.000 |
| Blue | 0.625 | 0.250 | 0.125 |
| Pink | 1.000 | 0.000 | 0.000 |
| Red | 0.714 | 0.071 | 0.214 |
| White | 0.350 | 0.350 | 0.300 |
| *avg.* | 0.738 | 0.134 | 0.128 |
| **Round 2** | | | |
| Black | 0.786 | 0.143 | 0.071 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 0.563 | 0.188 | 0.250 |
| White | 0.350 | 0.350 | 0.300 |
| *avg.* | 0.541 | 0.229 | 0.230 |
| **Round 3** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 0.563 | 0.188 | 0.250 |
| White | 0.563 | 0.188 | 0.250 |
| *avg.* | 0.539 | 0.206 | 0.256 |
| **Round 4** | | | |
| Black | 0.563 | 0.188 | 0.250 |
| Blue | 0.444 | 0.278 | 0.278 |
| Pink | 0.563 | 0.188 | 0.250 |
| Red | 0.563 | 0.188 | 0.250 |
| White | 0.444 | 0.278 | 0.278 |
| *avg.* | 0.515 | 0.224 | 0.261 |

**Table C.5** *Strategy scores per team and average for the entire session per round of the game. The data columns present the profit strategy score $G_p$, ttl strategy score $G_r$ and the risk-aversion strategy score $G_r$, as computed following Section C.3.*

And, as with the outcomes, Table C.6 summarises the strategy scores per session over all rounds of the game.

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | min | max | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Profit** | | | | | | | | | | |
| A | 0.596 | 0.709 | 0.739 | | | | | 0.596 | 0.739 | 0.681 |
| B | 0.500 | 0.523 | 0.637 | 0.637 | | | | 0.500 | 0.637 | 0.574 |
| C | 0.647 | 0.647 | 0.723 | 0.698 | | | | 0.647 | 0.723 | 0.679 |
| D | 0.585 | 0.295 | 0.335 | | | | | 0.295 | 0.585 | 0.405 |
| E | 0.392 | 0.403 | 0.406 | 0.457 | 0.457 | 0.504 | 0.628 | 0.392 | 0.628 | 0.464 |
| F | 0.596 | 0.591 | 0.636 | 0.780 | | | | 0.591 | 0.780 | 0.651 |
| G | 0.738 | 0.541 | 0.539 | 0.515 | | | | 0.515 | 0.738 | 0.583 |
| **TTL** | | | | | | | | | | |
| A | 0.222 | 0.173 | 0.132 | | | | | 0.132 | 0.222 | 0.176 |
| B | 0.292 | 0.277 | 0.229 | 0.229 | | | | 0.229 | 0.292 | 0.256 |
| C | 0.206 | 0.206 | 0.120 | 0.113 | | | | 0.113 | 0.206 | 0.161 |
| D | 0.181 | 0.370 | 0.347 | | | | | 0.181 | 0.370 | 0.299 |
| E | 0.317 | 0.328 | 0.331 | 0.311 | 0.311 | 0.283 | 0.203 | 0.203 | 0.331 | 0.298 |
| F | 0.195 | 0.205 | 0.196 | 0.102 | | | | 0.102 | 0.205 | 0.175 |
| G | 0.134 | 0.229 | 0.206 | 0.224 | | | | 0.134 | 0.229 | 0.198 |
| **Risk aversion** | | | | | | | | | | |
| A | 0.181 | 0.118 | 0.129 | | | | | 0.118 | 0.181 | 0.143 |
| B | 0.208 | 0.200 | 0.134 | 0.134 | | | | 0.134 | 0.208 | 0.169 |
| C | 0.147 | 0.147 | 0.157 | 0.189 | | | | 0.147 | 0.189 | 0.160 |
| D | 0.235 | 0.335 | 0.318 | | | | | 0.235 | 0.335 | 0.296 |
| E | 0.291 | 0.269 | 0.263 | 0.232 | 0.232 | 0.213 | 0.169 | 0.169 | 0.291 | 0.238 |
| F | 0.209 | 0.204 | 0.168 | 0.118 | | | | 0.118 | 0.209 | 0.175 |
| G | 0.128 | 0.230 | 0.256 | 0.261 | | | | 0.128 | 0.261 | 0.219 |

**Table C.6** *Strategy profile scores objective averaged over all players in the session per round and summarised over all rounds.*

# Appendix D

# Computational Complexity Theory

This appendix briefly recapitulates on the most essential notions of computational complexity theory that are required in order to comprehend the work presented in this thesis. The theory presented here originates mainly from the book by Sipser [238] and the interested reader can find a more thorough treatment of complexity theory and additional material there.

Computational complexity theory provides a framework for the analysis and classification of the difficulty of solving computational problems. Through formal analysis of algorithms it is possible to relate the size of the problem's input to the time and/or space that is required to solve the problem using the algorithm, and in particular it allows reasoning about the algorithm's scalability. For one problem, multiple algorithms might exists that each scale differently in terms of runtime or memory when the size of problem instances increases. Computational complexity theory helps to classify algorithms according to their (asymptotic) *worst-case* runtime or memory, expressed as a function over the input size.

The asymptotic bound is commonly denoted using the 'big-O' notation $O(g(n))$, where $g$ is the asymptotic worst-case complexity of an algorithm that is a function of the input size $n$, and it can be interpreted as 'this algorithm takes at most $g(n)$ time/space for an input of size $n$'. Observe thus that it is possible to analyse both time and space this way, however in most occasions computational complexity is implicitly assumed to consider runtime; space complexity analysis is often mentioned explicitly (also in the names complexity classes, discussed later). Conversely, $\Omega(g(n))$ denotes that an algorithm requires *at least* $g(n)$ time or space.

In addition to analysis of algorithms, computational complexity theory can also be used to classify problems. Essentially, solving a computational problem requires as least as much runtime as the best known algorithm for it. If for a problem one can prove that no faster or more memory-efficient algorithm can exist, then the problem can be classified according to the complexity of that algorithm. This is fundamental to the theory of complexity classes, that defines a hierarchy of problem difficulty.

The class of problems that is often considered easy, tractable or efficiently solvable, is that of polynomial time, denoted by P. All problems in this class allow a polynomial algorithm[57] in the size of the input, i.e. their asymptotic worst-case complexity is given by $O(n^c)$ where $c$ is a constant. Algorithms for this class are said to be tractable or efficient as they scale well with the input size compared to other classes and hence they are widely considered applicable in practice.

The polynomial-time problems class is contained[58] within the larger class of non-deterministic polynomial time problems, abbreviated with NP. For every problem in this class it is possible to quickly verify that the outcome of an algorithm is indeed correct, but to produce this outcome it requires a non-deterministic algorithm. Informally one can say that a non-deterministic algorithm arrives at a correct solution by guessing the intermediate steps. Such a guessing algorithm does not exist and instead all possible search directions and their resulting solutions have to be explored by an algorithm to produce a correct solution in the worst case. Inherently this leads to algorithms that have asymptotic worst-case bounds of the form $O(c^n)$, again $c$ being a constant, and scale much worse than problems in P.

As P is contained in the set of NP, there are problems in NP that can be solved efficiently. However there exist also many problems outside P that require at least exponential time, known as NP-hard problems. Formally NP-hard problems need not to be in the set NP themselves but are 'at least as hard' as any other problem in NP, i.e. there cannot be any problem in NP that is harder to solve than this problem, see Figure D.1b. Such a relation between problem complexities can be shown through reduction. If for two problems $A$ and $B$ there exists a method to transform instances of problem $A$ into instance of problem $B$ such that a solution for $B$ can only be correct if and only if a correct solution is produced for $A$, it is said that $A$ is reducible to $B$ denoted $A \leq B$. Consequential from such a reduction is that any algorithm for $B$ is also able to solve problem $A$ and hence if a polynomial-time algorithm exists for $B$, $A$ can also be solved in polynomial time. Observe that this reduction is one way: it does not have to be that $B$ is reducible to $A$.

For NP-hard problems there must exist a reduction from every other problem in NP because they are at least as hard as all other problems in NP. For this reason it is also true that if there exists a polynomial algorithm for any of the NP-hard problems, all problems in NP can be solved in polynomial time. If a problem is both hard for a complexity class and it also contained within that class, it is said to be complete. The set of complete problems for a complexity class in significant because it characterises the hardest problems within that class. The set of NP-complete problems has been studied extensively by computer scientists; it contains famous examples such as BOOLEAN SATISFIABILITY (SAT), KNAPSACK, TRAVELING SALESMAN (TSP) and many more.

The class NP is its turn a subset of PSPACE, i.e. the class of all problems that can be solved using polynomial space, which itself is a subset of the exponential time solvable problems class EXPTIME (EXP). The latter contains problems that can be solved in

[57] *Exponential algorithms may exists for the problem, for example as a result of poor programming, but if at least one polynomial algorithm exists for a problem, it is a member of* P.

[58] *In this thesis it is assumed that* $\text{P} \neq \text{NP}$*, according to common belief. See below for a more in-depth discussion.*

exponential time by deterministic algorithms, bounded in the worst case by $O(c^{g(n)})$ where $g(n)$ is a polynomial function. EXP contains problems that are considered highly intractable and may also not be contained in PSPACE. Similar to P the non-deterministic counterpart of EXP abbreviated NEXP(TIME), is defined as the set of problems that are solvable with a non-deterministic algorithm in $O(c^{g(n)})$ and therefore algorithms for a problem are bounded by $O(c^{h(n))})$, where $h(n)$ is now an exponential function of $n$. Finally both EXP and NEXP are sub classes of the class of problems that are solvable using at most exponential space EXPSPACE, and between the EXP classes there exists a hierarchy alike that of the P classes.

The complexity classes and relations between them that have discussed in this appendix are all summarised in Figure D.1a. In Figure D.1b an illustration of hardness and completeness is shown for NP problems; a similar can be made for EXP but is not included here.


**Figure D.1** *Diagram illustrating the complexity discussed in this appendix: (a) shows the complexity class hierarchy and (b) illustrates the sets of hard and complete problems for the class* NP.

**P versus NP** Currently it is still an open question whether P is merely contained in NP, i.e. it is a sub-class of NP, or that both classes are equivalent. This conundrum, commonly referred to as $\text{P} = \text{NP}$, is one of the most fundamental open problems in theoretical computer science with enormous consequences if true. If it turns out that indeed $\text{P} = \text{NP}$ then many of the currently known problems can be solved within polynomial time and most of the hierarchy between complexity classes (Figure D.1a) would collapse. Some classes however, such as EXP discussed in the previous section, have been shown to be a strict super set of P and will therefore remain considered intractable.

Ever since the question $\text{P} = \text{NP}$ was poised there has been a colossal amount of research to either prove or disprove it, but so far no conclusive answer has been found. Nonetheless, as still no polynomial algorithm has been found for *any* of the difficult problems in NP, it is widely believed that the classes are not equivalent. This thesis will adhere to this common belief and present all complexity relates issues as if they are shown disjoint to avoid having to repeat this discussion every time.

# Appendix E

# Game Theory and Mechanism Design

This appendix serves to provide the reader unfamiliar with game theory and/or mechanism design a brief introduction into the main notions of both strands of literature. This appendix is by no means a complete overview but concentrates on providing the required background to the work presented in Chapter 6. Most of the concepts described here are based upon the presentation by Nisan et al. [188], (pointers for) further reading on game theory may be found in the works by Gibbons [93], Myerson [181] and Osborne and Rubinstein [199], and for mechanism design in those by Conitzer and Sandholm [67], Dash et al. [72], Krishna and Perry [147].

## E.1 Game Theory

Game Theory is the strand of literature that models (multi-)agent decision making as a game and studies the outcomes of such a game. Here, every agent is represented by a player that must play an action – i.e. make one of its decisions – and it obtains utility for the corresponding joint outcome. For example, in a maintenance planning game the players must submit a plan, each possible plan being an available action, and the utility of each player is the sum of the its revenues minus the maintenance and network costs of the resulting outcome. Game theory mainly focuses is on the (type of) outcomes that can be obtained, given a definition of a game; mechanism design aims to influence the outcomes of such a game (discussed in the next section).

A very important assumption required to reason about outcomes of games is that of agent rationality.[59] An agent is considered to be rational if at any time it has to make a decision, it chooses the decision that optimises the utility it expects to receive as a result from that decision. The (expected) utility of an agent $i$ is captured through

[59] *This assumption is vital for all game theoretical literature but does not necessarily hold true in practice (see Section 8.2).*

its utility function $u_i$ that expresses the – usually monetary – gain or loss an agent has for every possible outcome $o \in O$ of the game, which typically depends on the private information of the agent known as its *type* $\theta_i$. In general, this type can be any structure that represents the private information, from a single constant to a complex, high-dimensional system, although using more elaborate types can make a mechanism impractical in terms of tractability (discussed more later) and therefore most mechanism design approaches focus on types of only one or a few dimensions. Important for now is that the type of an agent is assumed fixed in the sense that the agent is initially given a type $\theta_i$ from $\Theta_i$, the set of types available to that agent, and it will not change type during the mechanism's execution. Later the dynamic setting in which an agent's type can change is discussed. Summarising, for every outcome $o \in O$ the utility function specifies a value $u_i(\theta_i, o) \in \mathbb{R}$ that this outcome has to the agent and a rational agent of type $\theta_i$ will always try to achieve the outcome in which its utility is maximised.

The decisions that agent $i$ can make in the game are known as its actions $\Lambda_i$ and the agents chooses the action it will play using its strategy $\sigma_i : \theta_i \mapsto \Lambda_i$ from the set of strategies $\Sigma_i$ available to the agent. The joint strategy $\vec{\sigma} = \times_{i \in n} \sigma_i$ dictates the joint action $\vec{a} \in \boldsymbol{\Lambda}$ that is played in a game and the corresponding outcome of such a joint action is implemented through the outcome rule $g : \boldsymbol{\Lambda} \mapsto O$. An example of actions and an outcome rule can be a planning algorithm that takes as its input the activities each agent wants to perform and returns the corresponding optimal plan. Together, all of the previous can be summarised into a formal definition of a game:



**Definition E.1 (One-shot) Game**

A (one-shot) game $\mathcal{G}$ is defined as the tuple $\langle \boldsymbol{N}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}, \Sigma_i, O, g, \{u_i\}_{i \in \boldsymbol{N}} \rangle$ in which:

- $\boldsymbol{N} = \{1, 2, \ldots, n\}$ is the set of agents (players),
- $\boldsymbol{\Theta} = \times_{i \in n} \Theta_i$ is the joint type space where each $\Theta_i$ is type space available to agent $i$,
- $\boldsymbol{\Lambda} = \times_{i \in n} \Lambda_i$ is the set of joint actions where each $\Lambda_i$ is the set of actions available to agent $i$,
- $\boldsymbol{\Sigma} = \times_{i \in n} \Sigma_i$ is the set of joint strategies where each $\Sigma_i$ is the set of strategies available to agent $i$ and a single (joint) strategy $\vec{\sigma} \in \boldsymbol{\Sigma}$ is a mapping of types to actions $\vec{\sigma} : \boldsymbol{\Theta} \mapsto \boldsymbol{\Lambda}$,
- $O$ is the outcome space of the game that contains all possible outcomes the game can have,
- $g : \boldsymbol{\Lambda} \mapsto O$ is the outcome rule of the game such that for every jointly played action $\vec{a} \in \boldsymbol{\Lambda}$ the outcome rule $g(\vec{a})$ specifies the outcome $o \in O$ that results, and
- $\{u_i\}_{i \in \boldsymbol{N}}$ is the collection of utility functions where $u_i : \Theta_i \times O \mapsto \mathbb{R}$ specifies the utility agent $i \in \boldsymbol{N}$ has for every outcome $o \in O$ for every possible type $\theta_i \in \Theta_i$.

One play of the game requires every agent $i$ to choose an action $a_i \in \Lambda_i$, according to their (predetermined) strategy $\sigma_i \in \Sigma_i$. The resulting joint action $\vec{a} = \langle a_1, a_2, \ldots, a_n \rangle$ is 'executed', leading to outcome $o = g(\vec{a})$ and utilities $u_i(\theta_i, o)$, such that $o \in O$ and $\theta_i \in \Theta_i$ is the (predetermined) type of agent $i$. The game is one-shot, also called single-shot or static, because agents can only perform one a single action, leading to one outcome.

---

Although the formal definition of a game includes all possible types and strategies available, at the start of the game both are 'realised' for every agent. In other words, when the game starts, the agent taken on a type $\theta_i \in \Theta_i$ and strategy $\sigma_i \in \Sigma_i$. The main use of a game is to reason about the outcomes and associated utilities that can result from every combination of types and strategies. For any given combination of joint type $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and joint strategy $\vec{\sigma} \in \boldsymbol{\Sigma}$, the utility of agent $i$ is given by $u_i(\theta_i, g(\vec{\sigma}(\boldsymbol{\theta})))$ and therefore directly dependent on the types and strategies of other agents. Different joint strategies lead to different outcomes and game theory analyses the different outcomes that can occur in equilibrium, known as solution concepts.

An important solution concept is that of a Nash equilibrium, originating from the work by Nash et al. [186], that results when all agents maximise their own utility given the knowledge that all other agents will do the same for theirs. A game is in Nash equilibrium if no single agent can benefit from adopting a different strategy. Let $\boldsymbol{\theta}_{-i}$ and $\vec{\sigma}_{-i}$ denote respectively the joint type and joint strategy without that of agent $i$, i.e. $\boldsymbol{\theta}_{-i} = \boldsymbol{\theta} \setminus \theta_i = \langle \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n \rangle$ and similarly for $\vec{\sigma}_{-i}$, such that together $\theta_i$ and $\boldsymbol{\theta}_{-i}$ again form a complete type $\boldsymbol{\theta} = \langle \theta_i, \boldsymbol{\theta}_{-i} \rangle$ for all $n$ agents (likewise $\langle \sigma_i, \vec{\sigma}_{-i} \rangle$ forms a complete joint strategy) then the Nash equilibrium can be defined as:

**Definition E.2 Nash Equilibrium**

A joint strategy $\vec{\sigma} \in \boldsymbol{\Sigma}$ constitutes a Nash equilibrium if for every agent $i$:

$$\sigma_i^* \in \arg\max_{\sigma_i \in \Sigma_i} u_i\big(\theta_i, g\big(\langle \sigma_i(\theta_i), \vec{\sigma}_{-i}(\boldsymbol{\theta}_{-i}) \rangle\big)\big) \tag{E.1}$$

where $\theta_i \in \Theta_i$ is the true type of agent $i$.

---

When agents are not certain about strategies taken by others, but they have a common belief over the distribution of strategies, a solution concept known as a Bayes-Nash equilibrium can be achieved. In such an equilibrium, agents can do no better by deviating from their current strategy given the belief they have over the strategies played by others.

A more powerful solution concept than Nash is that of a dominant strategy equilibrium, in which every agent has at least one strategy that yields the highest utility regardless of the strategies played by the other agents. Formally:

**Definition E.3 Dominant Strategy Equilibrium**

A game has a dominant strategy equilibrium if for every agent $i$ there exists a

dominant strategy $\sigma_i^*$ such that

$$\forall \boldsymbol{\theta}_{-i} \in \boldsymbol{\Theta}_{-i}, \vec{\sigma}_{-i} \in \boldsymbol{\Sigma}_{-i} \colon \sigma_i^* \in \arg\max_{\sigma_i \in \Sigma_i} u_i\big(\theta_i, g\big(\langle \sigma_i(\theta_i), \vec{\sigma}_{-i}(\boldsymbol{\theta}_{-i})\rangle\big)\big) \tag{E.2}$$

where $\theta_i \in \Theta_i$ is the true type of agent $i$ and $\boldsymbol{\Theta}_{-i}$ and $\boldsymbol{\Sigma}_{-i}$ denote respectively the joint types and strategies available to all other agents.

Note that Equation E.2 is almost equal to the Nash equilibrium of Equation E.1 but it differs in its quantifiers such that the equation must hold for *every* joint type $\boldsymbol{\theta}_{-i}$ and strategy $\vec{\sigma}_{-i}$ of the other agents. This solution concept is therefore much stronger and, because the optimal strategy no longer depends on other agents, allows individual reasoning about optimal strategies.

**Example E.4** Maintenance Planning Game

As an example game, consider maintenance planning problem in which players $A$ and $B$ have to coordinate a single activity of unit duration with no probability of delay. Each agent has to decide in which of the two available time steps $\{\mathtt{t1}, \mathtt{t2}\}$ it wants to perform its activity. Both agents are contractually obliged to complete the activity and will therefore not receive a reward for completing it, instead they want to minimise the cost of it. Agent $1$ has costs $2$ and $6$ for the two time steps respectively, whereas agent $2$ has costs $3$ and $10$. When both agents plan their activities in the same time step, they are both penalised $5$. This can be formulated as the following maintenance planning problem:

- $\boldsymbol{N} = \{A, B\}$,
- $\mathcal{A} = \{\{\mathtt{a}\}, \{\mathtt{b}\}\}$ such that $\mathtt{a} = \langle 0, 1, 0, 0\rangle$ and $\mathtt{b} = \langle 0, 1, 0, 0\rangle$,
- $c_A(\mathtt{a}) = \langle 2, 6\rangle$ and $c_B(\mathtt{b}) = \langle 3, 10\rangle$,
- $\ell(\mathtt{a}, \mathtt{b}, \cdot) = 10$ ($5$ penalty for both agents), and
- $T = \{\mathtt{t1}, \mathtt{t2}\}$.

This maintenance planning problem can be converted into a game in which the agents play by selecting a time slot for their activity. The type of an agent is defined as a function that describes the revenues and costs for all of its activities in each time step, and its utility is defined as the profit it obtains for an outcome. This leads to the game (purposely omitting strategies for now):

- $\boldsymbol{N} = \{A, B\}$,
- $\boldsymbol{\Theta} = \{\{c_A\} \times \{c_B\}\}$,
- $\boldsymbol{\Lambda} = \big\{\big\{a^{\mathtt{t1}}, a^{\mathtt{t2}}\big\} \times \big\{b^{\mathtt{t1}}, b^{\mathtt{t2}}\big\}\big\}$ in which $a^t$ represents agent $A$ 'playing' activity $\mathtt{a}$ in time step $t$ (and correspondingly for $b^t$),
- $O = \{o_{1,1}, o_{1,2}, o_{2,1}, o_{2,2}\}$ such that $o_{x,y}$ represents agent $A$ performing its activity in time step $x$, agent $B$ in $y$,
- $g(a^x, b^y) = o_{x,y}$, and
- $u_1(\theta_1, o_{x,y}) = -c_1(\mathtt{a}, x) - 0.5\ell(\mathtt{a}, \mathtt{b}, x) \times X$ such that $X = 1$ if $x = y$ and zero otherwise, $u_2$ is defined analogously as $u_2(\theta_2, o_{x,y}) = -c_2(\mathtt{b}, y) - 0.5\ell(\mathtt{a}, \mathtt{b}, y) \times X$.

Now the agents in this game both want to minimise their maintenance costs and therefore $A$ plays a strategy $\sigma_A(\theta_A) = \arg\min_{a^t \in \Lambda} c_A(a^t, t)$ (and $B$ plays a similar strategy). Following this strategy, both agents will play the action that corresponds to the first time step, resulting in joint action $\vec{a} = \langle a^{\texttt{t1}}, b^{\texttt{t1}} \rangle$ and outcome $o_{1,1}$, illustrated in Figure E.1a.

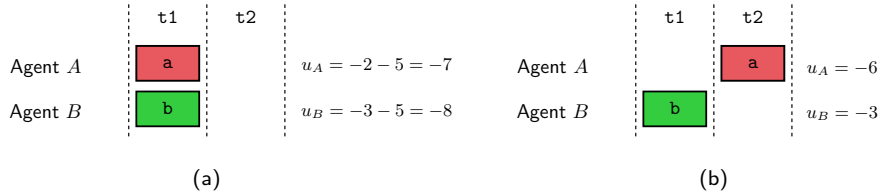



**Figure E.1** *Outcomes of the game corresponding to various joint strategies: (a) both players minimise their maintenance costs and (b) player $B$ still minimises its maintenance costs but player $A$ now always selects time step* `t2`.

Although both players minimise their maintenance cost, their utility is not maximal in the outcome of Figure E.1a because of the network cost penalty that both players receive in this outcome. Nevertheless, as both players can only play one single strategy, their joint strategy trivially constitutes a Nash equilibrium and both their strategies are dominant. However, this will not hold true when more strategies are available.

Each agent now additionally has two more simple strategies at its disposal: 'always play time step `t1`' and 'always play time step `t2`'. In the presence of these extra strategies, the previous outcome no longer forms a Nash equilibrium. Consider for instance agent $A$, if it switches strategy to 'always `t2`' its utility will increase from $-7$ to $-6$ and therefore it can benefit from changing its strategy, as shown in Figure E.1b. Notice that this new combination of strategies – $A$ plays 'always `t2`' and $B$ minimises its maintenance costs – does form a Nash equilibrium.

In this specific example there exists a dominant strategy for agent $B$: irregardless of what strategy agent $A$ plays, it always maximises its utility by playing either the 'cost minimisation' or 'always `t1`' strategy (although the actual value of the utility depends on what strategy $A$ plays). For agent $A$ however such a dominant strategy is not available. When agent $B$ plays either of the previously mentioned strategies that will result in $b$ being planned at time step `t1`, agent $A$ is best off by playing 'always `t2`'. If $B$ plays the 'always `t2`' strategy, agent $A$ maximises its utility by playing either one of the other strategies that are available to it. Therefore, this game has no dominant strategy equilibrium.

## E.2 Mechanism Design

Mechanism design bases on game theory to model multi-agent decision making as $n$-player games with actions and outcomes, but it additionally steers agents towards globally favourable outcomes (and therefore sometimes referred to as 'inverse game theory'). By far, most of the work on mechanism design focuses on monetary incentives where agents get a payoff for their participation in the mechanism and these

are therefore also the only type of mechanism considered in this thesis.[60] In monetary mechanisms, agents are rewarded for their contribution toward a global goal, thereby making it in their personal best interest to consider the global goal even if they are selfish.

The main purpose of a mechanism is to impose a function upon the game that specifies (globally) desirable outcomes that are achieved in the equilibria of the game, even when agents act in their own best interest. This function is known as the *social choice function (SCF)* of the mechanism and it maps the actions of all agents to an outcome, i.e. $f : \mathbf{\Lambda} \mapsto O$, similar to the outcome rule of the game. Examples of SCFs are: a winner determination algorithm that allocates an auctioned item to the bidder with the highest bid, or an optimal planner that uses agent's preferences to develop an optimal maintenance plan.

Nevertheless, whereas in a game the utility functions of the agents were assumed public knowledge, mechanism design assumes a *private-values setting* in which the agent's realised type – its preferences or bid function – is only known by the agent itself. The set of types available to each agent, i.e. $\Theta_i$, is public knowledge, but the mechanism does not know which of these types the agent has. Intuitively, one can imagine that a mechanism tries to optimise some global goal but it will not know the (precise) motivation of each of its participants although it can reason about the various motivations an agent can have. This private value setting is modelled through a valuation function $v_i : \Theta_i \times O \mapsto \mathbb{R}$ such that the value of an outcome $o \in O$ depends on the type $\theta_i$ that is known only agent $i$ and the SCF is a function expressed over the valuation functions. The goal of a mechanism is then typically to maximise or minimise the SCF exactly in the equilibria of the underlying game, *when all players act truthfully with respect to their intentions or preferences* If a mechanism always succeeds to achieve outcomes desired by the SCF $f$ in an equilibrium where all agents act truthfully with respect to their intention or valuation, then the mechanism *implements* $f$ and is *incentive compatible* (discussed in more detail later).

Before going into the details of a mechanism, however, one very important negative result immediately limits the scope of mechanisms that can implement an SCF in a truthful equilibrium. This result is known as the Gibbard-Satterthwaite theorem:

**Theorem E.5 Gibbard-Satterthwaite [92, 227]**

For a game with unrestricted types, $|\boldsymbol{N}| \geq 2$, $|O| \geq 3$, and a social choice function $f$ such that $\forall o \in O, \exists \vec{a} \in \mathbf{\Lambda} : f(\vec{a}) = o$, a mechanism can implement $f$ in a dominant strategy equilibrium if and only if $f$ is dictatorial.

Here, dictatorial means that there is one agent that can determine the outcome of the game, thereby disregarding the preferences of the other agents. For this reason, a mechanism that implements a dictatorial SCF is considered unacceptable for most applications. Although this theorem is rather strong, it can be circumvented by weak-

[60] *There exist mechanisms that do not require monetary transfers, see e.g. Chapter 10 of the book by Nisan et al. [188].*

ening any of its conditions: using restricted types, considering only problems with 2 outcomes, or implementing $f$ in a weaker solution concept.

It turns out there exists a rather natural restriction on the types of agents under which this result no longer holds. Using monetary transfers such that the utilities of agents become quasilinear, e.g. of the form $u_i + y$ for some function $y$, it is possible to design mechanisms that implement an SCF and are non-dictatorial (e.g. the Vickrey-Clarke-Groves mechanism of Section E.4). The intuition behind this is that using transfers, a mechanism has means to reward or penalise agents based on their contribution to the SCF and, by using well-designed payments, the SCF is implemented by the mechanism exactly when agents maximise their personal utility (accounting for the payment it will receive or make). In summary, a (one-shot) mechanism with monetary transfers can be defined as:

**Definition E.6 (One-shot) Mechanism**

A (one-shot) mechanism for a (one-shot) game $\mathcal{G}$ is defined as the tuple $\langle f, \boldsymbol{p} \rangle$ where $f : \boldsymbol{\Lambda} \mapsto O$ is the social choice function and $\boldsymbol{p} = \{p_i\}_{i \in \boldsymbol{N}}$ is a collection of payoff functions such that each $p_i : \boldsymbol{\Lambda} \mapsto \mathbb{R}$ defines the payment to agent $i$ depending on the actions played. For each agent $i \in \boldsymbol{N}$ participating in the mechanism, the (quasilinear) utility of an agent $i$ for outcome $f(\vec{a})$ when joint action $\vec{a} \in \boldsymbol{\Lambda}$ is taken and its (private) type is $\theta_i \in \Theta_i$ is given by:

$$u_i(\theta_i, \vec{a}) = v_i(\theta_i, f(\vec{a})) + p_i(\vec{a}) \tag{E.3}$$

where $\forall \theta_i \in \Theta_i, o \in O : v_i(\theta_i, o) = u'_i(\theta_i, o)$ such that $u'_i$ is the utility function of agent $i$ in game $\mathcal{G}$.

A mechanism is said to be executed if the game is played and payments have been made, based upon the outcome of the SCF. Recall from the previous section that players use their strategy to determine the action they are going to play in the game. The joint action that is played is $\vec{a} = \vec{\sigma}(\boldsymbol{\theta})$, which is reported to the mechanism, and the outcome thereof is $f(\vec{\sigma}(\boldsymbol{\theta})) \in O$. Thereafter, the mechanism computes and incurs payments $p_i(\vec{a})$ for every agent $i$ based upon the joint action that is taken.

In mechanism design two types of mechanisms are distinguished based upon the type of information that is reported by the agents to the mechanism. In *direct-revelation* mechanisms, agents are required to report their complete private information (its type) to the mechanism, whereas in indirect-revelation mechanisms other types of actions are allowed.[61] Alternatively one could say that in a direct mechanism the set of actions for each agent is restricted to reporting its types, i.e. $\Lambda_i = \Theta_i$ and therefore $\sigma_i : \Theta_i \mapsto \Theta_i$ and $O = \boldsymbol{\Theta} = \times_{i \in n} \Theta_i$. Due to an important theorem in mechanism design, known as the revelation principle, it is again possible to limit the scope of mechanisms:

**Theorem E.7 Revelation principle [92]**

An indirect mechanism that successfully implements social choice function $f(\vec{\sigma}(\boldsymbol{\theta}))$

[61] *The term 'revelation' is often omitted in mechanism design literature and will be omitted from now.*

in at least a Bayes-Nash equilibrium can be transformed into an equivalent direct-revelation mechanism that implements SCF $f(\boldsymbol{\theta})$ directly over the joint type $\boldsymbol{\theta}$.

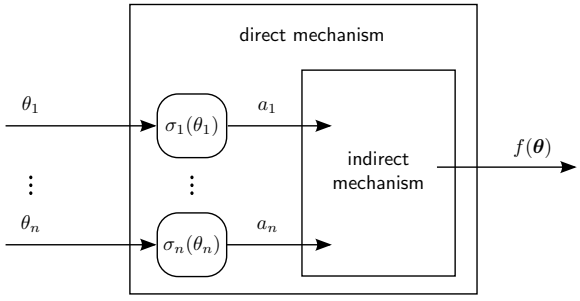



**Figure E.2** *The relation between direct and indirect mechanisms: an indirect can always be obtained by simulating the use of strategies as a part of the direct mechanism.*

As a consequence of the revelation principle only direct mechanisms need to be considered because of two immediate corollaries of Theorem E.7: either a direct mechanism exists that implements $f$ truthfully and it can be transformed to an equivalent indirect mechanism, or no such direct mechanism exists and therefore also no indirect mechanism can exist that implements $f$ truthfully. This relation is visualised in Figure E.2: if an indirect mechanism exists for a game, one can always construct a direct mechanism, that takes type reports as its input, apply strategies to determine the input to the indirect mechanism and use the indirect mechanism to compute the SCF outcome. In the remainder of this chapter the focus will henceforth be on direct mechanisms.

In a direct mechanism, agents report a joint type $\hat{\boldsymbol{\theta}}$ instead of a joint action. The joint type that is reported does not necessarily contain the true type of every agent: it is determined by the strategies of the agents such that $\hat{\boldsymbol{\theta}} = \vec{\sigma}(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the true joint type of the agents. To avoid confusion between the two, the true type of an agent $i$ will be always denoted by $\theta_i$ whereas its reported type is written as $\hat{\theta}_i$. As in a direct mechanism the agents of the underlying game play a type based on their strategy, i.e. $\vec{\sigma}(\boldsymbol{\theta}) = \hat{\boldsymbol{\theta}}$, it is possible to write $f(\vec{\sigma}(\boldsymbol{\theta})) = f(\hat{\boldsymbol{\theta}})$. Moreover, the payments in a direct mechanism can be defined directly over the reported type: $p_i(\hat{\boldsymbol{\theta}}) = p_i(\vec{\sigma}(\boldsymbol{\theta}))$. Finally, the utility of an agent $i$ under a direct mechanism is given by

$$u_i(\theta_i, \hat{\boldsymbol{\theta}}) = v_i(\theta_i, f(\hat{\boldsymbol{\theta}})) + p_i(\hat{\boldsymbol{\theta}}) \tag{E.4}$$

when joint type $\hat{\boldsymbol{\theta}}$ is reported to the mechanism (and $\theta_i$ is the true type of agent $i$). Observe that a negative payoff means that the agent should pay the mechanism and vice versa for a positive payoff. The valuation of an agent is defined over the joint outcome and can possibly also capture inter-agent valuations. For instance, an agent might have negative valuation if a certain other agent gets assigned an item by an auctioning mechanism. An example of a mechanism for the maintenance planning game of Example E.4 is presented later in Example E.13 in Section E.4.

## E.3 Mechanism Design Desiderata

Typically, there are several desirable properties of which at least a few need to be satisfied in order for the mechanism to be applicable in practice. Of course, the main purpose of a mechanism is to implement a chosen SCF, but there are more characteristics that are favourable (or often required) in a mechanism. Here the most important *mechanism desiderata* are discussed: incentive compatibility, individual rationality, budget balance and computational efficiency. The first one, incentive compatibility, is defined as:

**Definition E.8 Incentive Compatible (IC)**

A mechanism is incentive compatible if it implements SCF $f$ in an equilibrium where a truthful strategy is utility-maximising for every agent. Formally, for every agent $i \in \boldsymbol{N}$ and every joint type $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ there exists a strategy $\sigma_i^*$ such that

$$u_i(\theta, \langle \sigma_i^*(\theta_i), \vec{\sigma}_{-i}(\boldsymbol{\theta}_{-i}) \rangle) \geq u_i(\theta, \langle \sigma_i(\theta_i), \vec{\sigma}_{-i}(\boldsymbol{\theta}_{-i}) \rangle)$$

In a dominant strategy equilibrium this equation holds true for every joint strategy $\vec{\sigma}_{-i} \in \boldsymbol{\Sigma}_{-i}$ of the other players, and the mechanism is said to be strategyproof.

Incentive compatibility is more often than not a compulsory property in mechanism design literature. Under a direct incentive compatible mechanism, the best strategy for every agent is to report their type truthfully to the mechanism, e.g. without lying about their preferences. This is very desirable because under such mechanisms agents are honest about for instance their valuation and therefore the mechanism can find a 'fair' outcome. For example in an auctioning setting, under an IC mechanism each bidder always truthfully reports the valuation it has for the auctioned item and the mechanism can therefore assign it to the bidder that values it most. In the case of a mechanism being incentive compatible in dominant strategies it is said to be *strategyproof* and for a direct strategyproof mechanism it must be that $\sigma_i(\theta_i) = \theta_i$, i.e. its strategy is to report its type truthfully.

Another property that is typically essential in a mechanism is individual rationality. Individual rationality guarantees that it is in the best interest of agents to participate in the mechanism or, in other words, it will not run at a loss because of the mechanism:

**Definition E.9 Individual Rationality (IR)**

A mechanism is individually rational if for all joint actions $\vec{a} \in \boldsymbol{\Lambda}$, all types $\theta_i \in \Theta_i$ and all agents $i \in \boldsymbol{N}$:

$$u_i(\theta_i, \vec{a}) > 0$$

When the total utility is always non-negative, i.e. $\forall i \in \boldsymbol{N} : u_i(\theta_i, \vec{a}) \geq 0$, the mechanism is weakly individually rational.

There are different sorts of individual rationality that can be demanded from the mechanism, depending on the time at which IR should hold. The strongest type of individual

rationality is ex-post IR, in which agents only participate if they always have positive utility, regardless of the types and strategies of other agents. When agents expect to obtain positive utility given knowledge about their private type and beliefs about types of other agents, the mechanism is said to be ex-interim IR. The weakest rationality notion is that of ex-ante IR, in which agents expect to have positive utility at least prior to realisation of all agent types, including his own. In other words, in ex-ante IR agents do not expect to run at a loss given their beliefs about all reported agent types; in practice, however, negative utilities are possible.

As the mechanism relies on monetary transfers, a property similar to IR is preferable for the agent that employs the mechanism. Having a mechanism that allocates items fairly or optimises overall utility is beneficial, but if its payments require (substantial) external funding it is likely unacceptable. This is the budget balance property:

**Definition E.10 Budget Balance (BB)**

A mechanism is budget balanced if for all joint actions $\vec{a} \in \mathbf{\Lambda}$:

$$\sum_{i \in \boldsymbol{N}} p_i(\vec{a}) = 0$$

If it never runs at a loss, i.e. $\sum_{i \in \boldsymbol{N}} p_i(\vec{a}) \leq 0$, the mechanism is weakly budget balanced (or no-deficit).

Finally, computational efficiency[62] of a mechanism is used to refer to the computational effort that is required to compute social choice function $f$ of the mechanism. Even though a mechanism might satisfy IC and IR, if computing the objective function is intractable, the mechanism can still be of little use in practice. A famous example is that of combinatorial auctions, for which a simple payment scheme exists that leads to a strategyproof mechanism however determining the winner of the auction poses an NP-hard problem. Moreover, although often good approximations are available for such problems, the mechanism design desiderata do not transfer (immediately) to such approaches. Computational efficiency in mechanisms is the basis for an entirely own branch of research known as computational or algorithmic game theory. For more reading on this, the reader is referred to the book by Nisan et al. [188].

## E.4 The Vickrey-Clarke-Groves Mechanism

Given the mechanism desiderata presented in the previous section and the impossibility result by Gibbard and Sattertwaithe (Theorem E.5), the range of mechanisms that can be successfully applied in practice is limited. Moreover, Green [99] showed that there exists only one mechanism that is concurrently strategyproof, individually rational and

[62] *The term computational is included here to avoid confusion with the terminology of an efficient mechanism that is used in the literature to denote a mechanism that optimises the sum of agent valuations (e.g. the Vickrey-Clarke-Groves mechanism of the next section).*

(weakly) budget balanced for arbitrary types and quasi-linear utilities. This mechanism is known as the Vickrey-Clarke-Groves (VCG) mechanism, due to the Nobel-prize winning work by Vickrey [252], Clarke [64] and Groves [101]. Here the VCG mechanism will be discussed, starting from the broader class of Groves mechanisms. Let $v_{-i}(\hat{\boldsymbol{\theta}}, o) = \sum_{j \in \boldsymbol{N} \setminus i} v_j(\hat{\theta}_j, o)$ (which is equal to $v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, o)$ as type $\hat{\theta}_i$ is not used) denote the *reported* valuation[63] of all agents except agent $i$ for outcome $o$ when joint type $\hat{\boldsymbol{\theta}}$ is reported, then the class of Groves mechanisms is defined as:

**Definition E.11 Mechanisms in the Groves Class**

A (direct) mechanism $\langle f, \boldsymbol{p} \rangle$ belongs to the *Groves class* if it satisfies the following two conditions:

(i) The social choice function $f$ is one that results in outcomes that always optimise group welfare:

$$\forall \hat{\boldsymbol{\theta}} \in \boldsymbol{\Theta} \colon f(\hat{\boldsymbol{\theta}}) \in \arg\max_{o \in O} \sum_{i \in \boldsymbol{N}} v_i(\hat{\theta}_i, o)$$

(ii) The payoff for an agent $i$ is relative to its impact on the valuation of other agents:

$$p_i(\hat{\boldsymbol{\theta}}) = v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}})) - h_{-i}(\hat{\boldsymbol{\theta}}_{-i})$$

for every reported type $\hat{\boldsymbol{\theta}}$ and function $h_{-i}(\hat{\boldsymbol{\theta}}_{-i}) \in \mathbb{R}$ that excludes the type of agent $i$.

Mechanisms in the Groves class aim to optimise the social welfare of the group, i.e. the sum of individual agent valuations. A mechanism that implements such a social choice function is termed an *efficient* mechanism. As a result of Green [99], this is the only social choice function that allows for an IC mechanism in dominant strategies. The Groves class provides a straightforward method to design a mechanism that is both incentive compatible as well as individually rational, although it requires a condition known as the no negative externalities property to guarantee the latter. This condition is defined as $\forall i \in \boldsymbol{N}, \hat{\boldsymbol{\theta}} : v_i(\hat{\theta}_i, f(\hat{\boldsymbol{\theta}}_{-i})) \geq 0$ and it prevents agents from having a negative valuation for outcomes in which they do not participate. This property is naturally more often than not present in many realistic mechanism design problems such as (combinatorial) auctions, voting games and selfish planning. Notice that it is of vital importance for incentive compatibility that the payment for agent $i$ is independent from its own reported type: it is only subject to the reported joint type of all other agents. If this were not the case, an agent is able to influence its own payoff, which makes it impossible to devise an incentive compatible mechanism (see Example E.13).

One of the shortcomings of the Groves class, is that it does not guarantee budget balance in general. Not having the budget balance property can be an important drawback in many applications as such mechanisms may require external funding. Clarke [64]

[63] *As the mechanism does not know the true type of an agent, the mechanism can only rely on the valuation that agents report having for an outcome or, in direct mechanisms, the reported type.*

proposed a specific payment function, known as the Clarke tax or Clarke's pivot rule, as a choice for the function $h$ of a Groves-class mechanism in order to obtain the no-deficit property. The Clarke tax charges each agent the value others could have obtained if it would not have participated, or formally $h_{-i}(\hat{\boldsymbol{\theta}}_{-i}) = v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}}_{-i}))$. Groves0class mechanisms that implement Clarke's pivot rule are known as Vickrey-Clarke-Groves mechanisms:

**Definition E.12 Vickrey-Clarke-Groves (VCG) Mechanism**

A direct mechanism $\langle f, \boldsymbol{p} \rangle$ is a *Vickrey-Clarke-Groves mechanism* if it satisfies two conditions:

(i) The mechanism is a member of the Groves class (Definition E.11).

(ii) Each agent $i$ pays exactly the difference in valuation that is caused by its participation by letting $h_i$ be the valuation that the other agents could reportedly have obtained without agent $i$:

$$p_i(\hat{\boldsymbol{\theta}}) = v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}})) - v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}}_{-i}))$$

for every reported type $\hat{\boldsymbol{\theta}}$.

As the VCG mechanism is a member of the Groves class, it is automatically strategyproof, i.e. incentive compatible in a dominant strategy equilibrium. Furthermore, the VCG mechanism is also individually rational and weakly budget balanced if the no single-agent effect is satisfied. This latter condition states that the sum of valuations of all $\boldsymbol{N} \setminus \{i\}$ agents in an outcome where an agent $i$ is not present cannot exceed the sum of valuations over the same set of agents in an outcome where all agents participate, i.e. it must be that $v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}})) \geq v_{-i}(\hat{\boldsymbol{\theta}}_{-i}, f(\hat{\boldsymbol{\theta}}_{-i}))$. In other words, disregarding an agent can never lead to a better outcome for all others collectively (although agents can potentially benefit individually from disregarding an agent, this is always at the cost of others as a corollary of the no single-agent effect).

**Example E.13** VCG for Maintenance Planning

The two agents of Example E.4 agree to coordinate their activity planning using a mechanism as long as their joint valuation is optimised. Their valuation $v_i$ for an outcome of is given by the utility function $u_i'$ of the maintenance game, e.g. $v_A(\theta_A, o_{x,y}) = u_A'(\theta_A, o_{x,y}) = c_A(\mathsf{a}, x) + 0.5\ell(\mathsf{a}, \mathsf{b}, x) \times X$ (see Example E.4). As a first attempt, they consider a direct mechanism that does not use monetary transfers (or $p_i(\vec{a}) = 0$ for every agent $i \in \boldsymbol{N}$ and all joint actions $\vec{a} \in \boldsymbol{\Lambda}$) with an optimal planning algorithm as its social choice function $f$. Nevertheless, the agents quickly realise that this mechanism will only work when agents are truthful; if not, they can cheat the system by misreporting their valuation.

Recall that in a direct mechanism, players report their type to the mechanism such that the mechanism can compute an outcome based on these types. In the maintenance game, the type of an agent $i$ is its maintenance cost function $c_i$ (network costs are publicly known)

and the social choice function of the mechanism, i.e. the optimal planning algorithm, will allocate activities to the time steps such that the sum of these maintenance costs and network costs is minimised. Consequentially, the outcome that is returned by the algorithm can be influenced by misreporting the maintenance cost function to the mechanism. For example, the optimal outcome of the game is $o_{2,1}$ where agent $A$ is assigned time step t2 and agent $B$ time step t1, and this allocation will indeed be returned by $f$ if both agents report their maintenance costs truthfully. But, although this outcome is *jointly* optimal, it is not optimal from agent $A$'s perspective. The valuation that the agent has for this outcome is $-6$ whereas it has a valuation of $-2$ for outcome $o_{1,2}$. In essence, agent $A$'s utility, i.e. the actual gain or loss it has after mechanism execution (which is equal to its valuation in this mechanism because no payments are used), is sacrificed by the planning algorithm in favour of a globally optimal outcome. If the agent is very selfish, it might not accept such a utility loss and instead falsely report maintenance costs $c_A(\mathtt{a}) = \langle 2, 20\rangle$. Given this false report, the planning algorithm mistakenly finds that outcome $o_{2,1}$ has a joint utility of $-20 + -3 = -23$ and therefore now returns outcome $o_{1,2}$, that has a joint utility of $-12$, as the optimal allocation. The valuations reported by agent $A$ (and $B$) in both cases are shown as payoff matrices in Figure E.3.

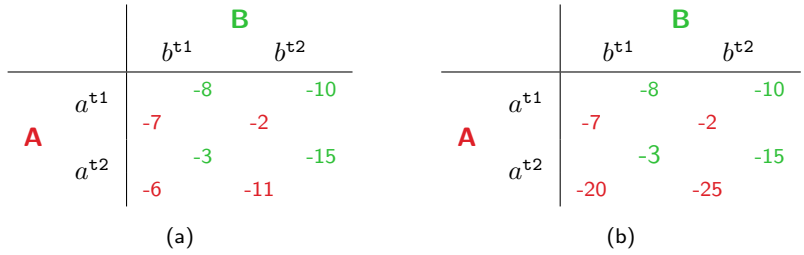



**Figure E.3** *Payoff matrices under both cost reports where the valuations of agent $A$ and $B$ are shown in red and green respectively: (a) agents report their maintenance costs truthfully as $c_A(\mathtt{a}) = \langle 2, 6\rangle$ and $c_B(\mathtt{b}) = \langle 3, 10\rangle$ and (b) agent $A$ lies about its costs by reporting $c_A(\mathtt{a}) = \langle 2, 20\rangle$ (remember that each agent is penalised $-5$ whenever their activities are performed concurrently).*

To discourage such strategic behaviour, the agents now turn to mechanisms that do use monetary transfers. Intuitively such a mechanism should reward the agents for reporting truthfully and/or penalise them when they do not. They propose a mechanism that still computes the optimal outcome using the reported maintenance costs but splits the 'valuation loss' over both agents. The payment rule computes the valuation for both agents using the reported maintenance costs (it knows only these costs and not the true costs) and charges half their difference to the agent that profits the most in the jointly optimal outcome, which is then paid to the other agent. Let $\hat{\boldsymbol{\theta}}$ denote the reported type of both agents, i.e. their reported maintenance costs, then the payment for both agents is defined as:

$$p_A(\hat{\boldsymbol{\theta}}) = p_B(\hat{\boldsymbol{\theta}}) = -0.5\Big(v_A(\hat{\theta}_A, f(\hat{\boldsymbol{\theta}})) - v_B(\hat{\theta}_B, f(\hat{\boldsymbol{\theta}}))\Big)$$

where $f(\hat{\boldsymbol{\theta}})$ is the optimal outcome of the planning algorithm given the reported types $\hat{\boldsymbol{\theta}}$ describing their maintenance costs.

If both agents report their maintenance costs truthfully under this mechanism in joint report $\hat{\boldsymbol{\theta}}$, the jointly optimal outcome $f(\hat{\boldsymbol{\theta}})$ is again $o_{2,1}$. The payments will be $p_A(\hat{\boldsymbol{\theta}}) =$

$-0.5(v_A(\hat{\theta}_A, o_{2,1}) - v_B(\hat{\theta}_B, o_{2,1})) = -0.5(-6 - -3) = 1.5$ and thus $p_B(\hat{\boldsymbol{\theta}}) = -1.5$. This leads to utilities $u_A(\theta_A, o_{2,1}) = -6 + 1.5 = -4.5$ and $u_B(\theta_B, o_{2,1}) = -3 + -1.5 = -4.5$, in which $\theta_A$ and $\theta_B$ are the true types of both agents. The utility an agent really obtains depends on its realised type and not (directly) on its reported type, although it may be able to influence its utility in some mechanisms by misreporting and thereby changing the outcome or affecting the payment.

Now imagine that agent $A$ misreports its maintenance costs like before as $c_A(\mathtt{a}) = \langle 2, 20\rangle$, leading to joint reported type $\hat{\boldsymbol{\theta}}'$ in which $B$ remains truthful, then the jointly optimal outcome becomes $f(\hat{\boldsymbol{\theta}}') = o_{1,2}$ and the resulting payments are $p_A(\hat{\boldsymbol{\theta}}') = -0.5(v_A(\hat{\theta}'_A, o_{1,2}) - v_B(\hat{\theta}'_B, o_{1,2})) = -0.5(-2 - -10) = -4$ and $p_B(\hat{\boldsymbol{\theta}}') = 4$. The utilities for the jointly optimal outcome now become $u_A(\theta_A, o_{1,2}) = -2 + -4 = -6$ and $u_B(\theta_B, o_{1,2}) = -10 + 4 = -6$. At first glance, this mechanism seem fair to both agents and it prevents lying because now agent $A$ obtains a lower utility for the outcome that results from its misreport and, as a consequence, it is better of by reporting its true maintenance costs. Moreover, this mechanism is strictly budget balanced as the sum of payments is always exactly zero. Nevertheless, even this mechanism is not strategyproof. The problem is that the agents can still influence their own payment and, in this specific mechanism, they can increase their payment by (slightly) overreporting costs in a way that the jointly optimal outcome does not change.

Consider for instance that agent $A$ has obtained information that agent $B$ will report its costs truthfully. In this case, $A$ knows that the optimal allocation will have a joint utility of $-9$ whereas the second-best allocation has joint utility $-12$. Now, $A$ can report maintenance costs $c_A = \langle 2, 6 + (3 - \epsilon)\rangle$ where $\epsilon \in (0, 6)$ and know that outcome $o_{2,1}$ will remain jointly optimal and it will not be the agent that profits most from the joint outcome (otherwise $A$ will have to pay $B$). The payment $A$ will receive becomes $p_A(\hat{\boldsymbol{\theta}}) = 0.5(v_A(\hat{\theta}_A, o_{2,1}) - v_B(\hat{\theta}_B, o_{2,1})) = -0.5(-(6 + (3 - \epsilon)) - -3) = 3 - 0.5\epsilon$ and thus by lying about its costs, it can increase its utility at the cost of agent $B$ (to a maximum of $-6 + 3 = -3$ when $\epsilon = 0$). Therefore, even this mechanism is vulnerable to strategically misreporting costs.

After some more research, the agents decide to use a Vickrey-Clarke-Groves mechanism. Where before the agents split the difference in valuation evenly, the VCG mechanism charges each agent the 'harm' it causes to the other. An important difference with the previous mechanism is that the payment for agent $i$ is *independent* of its declared maintenance costs, thus preventing it from affecting its own utility. Each agent pays the difference between the valuation of the other agent in the optimal allocation and the valuation of the other agent in the allocation that is optimal without it participating. For the maintenance planning mechanism with two agents, the payment for agent $A$ is hence defined as

$$p_A(\hat{\boldsymbol{\theta}}) = v_B(\hat{\theta}_B, o) - v_B(\hat{\theta}_B, o_{-A})$$

such that $o = f(\hat{\boldsymbol{\theta}})$ is the optimal outcome where both $A$ and $B$ are participating and $o_{-A} = f(\hat{\boldsymbol{\theta}}_{-A}) = f(\hat{\theta}_B)$ is the optimal outcome for agent $B$ when $A$ is not present. The payment for $B$ is defined similarly but uses the valuation of agent $A$ and the outcome $o_{-B}$ in which $B$ is not participating. Notice that slightly overreporting, i.e. reporting higher costs without changing the outcome, has no effect under this mechanism because the payment for each agent is completely independent of its own reported valuation.

To illustrate why the VCG mechanism is strategyproof, the utilities for the agents are computed when they both report truthfully and when agent $A$ misreports $c_A(\mathtt{a}) = \langle 2, 20\rangle$. When both agents are truthful, the optimal outcome is $o_{2,1}$, which is also the optimal outcome for agent $B$ when agent $A$ is not participating. The payment for agent $A$ will

always be zero as the valuation $B$ has for $f(\hat{\boldsymbol{\theta}})$ is equal to the valuation it has for $f(\hat{\boldsymbol{\theta}}_{-A})$, the optimal outcome without $A$'s presence. Agent $B$, however, has to pay for the valuation loss of agent $A$. Without agent $B$, the optimal outcome for agent $A$ is $o_{1,2}$ with valuation $-2$. The value loss of agent $A$ due to agent $B$'s presence is therefore equal to $-6 - -2 = -4$, which is exactly the payment for agent $B$. The utilities of $A$ and $B$ are then $-6$ and $-7$ respectively.

It was already established that no agent can profit from misreporting its valuation by a small amount; only when the optimal outcome changes as a consequence of the reported costs, the payment is affected. Consider again the joint report $\hat{\boldsymbol{\theta}}'$ in which $A$ misreports $c_A(\mathsf{a}) = \langle 2, 20\rangle$ and $B$ is truthful. Now the optimal joint outcome becomes $o_{1,2}$, and this is also optimal for $A$ without $B$'s presence as its value is maximised in both cases. Agent $B$ however prefers outcome $o_{2,1}$ when $A$ is not participating and therefore now agent $A$ pays the valuation difference of $B$ such that $p_A(\hat{\boldsymbol{\theta}}') = v_B(\hat{\theta}'_B, o_{1,2}) - v_B(\hat{\theta}'_B, o_{2,1}) = -10 - -3 = -7$. The utility of agent $A$ given the false type report $\hat{\boldsymbol{\theta}}'$ and its real type $\theta_A$ then becomes $u_A(\hat{\boldsymbol{\theta}}', o_{1,2}) = v_A(\theta_A, o_{1,2}) + p_A(\hat{\boldsymbol{\theta}}') = -2 + -7 = -9$, and its utility is lower than when it declares its true costs ($-6$). Through a formal proof, roughly along the lines of this example, it can be shown that agents participating in the VCG mechanism maximise their utility by always reporting truthfully. Consequentially, the dominant strategy for every agent is to tell the truth and hence a game that uses VCG has a dominant strategy equilibrium: it is strategyproof.

A final remark about this example is that the VCG mechanism here is not individually rational. Here, it is assumed that agents will participate in the mechanism to coordinate the maintenance activities for which they have been contracted, but their utility will always be below zero. In practice, the contracted agents will typically receive a fixed reward for their work, thus offsetting their costs by some positive amount. If these fixed profits are included in the mechanism, it can be made individually rational as well.

TRAIL

## Summary

This thesis explores the potential of self-regulation in collective decision making to align interests and optimise joint performance. Demonstrated in the domain of road maintenance planning, this research contributes novel incentive mechanisms and algorithmic techniques to incite self-regulation and coordinate agent interactions, paired with a practical validation of the concept through serious gaming. The learnings of this work guide the design and implementation of future performance-based partnerships and advance the current state-of-the-art in sequential decision making.

## About the Author

Joris Scharpff is an algorithmics researcher with a Master's degree in Computer Science who prefers to be on the front-line of innovation, implementing novel ideas to solve real-world problems. Joris is a strong advocate of self-regulation, whether it is applied to joint research, professional endeavours or the interactions of daily life.

**TRAIL Research School** ISBN 978-90-5584-274-2