TU Delft

Delft University of Technology

Autonomous Onboard Mission Planning for Multiple Satellite Systems

Zheng, Zixuan

DOI
10.4233/uuid:ef99fd03-4713-493f-b3c3-1f741193eefd

Publication date
2019

Document Version
Proof

Citation (APA)
Zheng, Z. (2019). *Autonomous Onboard Mission Planning for Multiple Satellite Systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:ef99fd03-4713-493f-b3c3-1f741193eefd

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

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

AUTONOMOUS ONBOARD MISSION PLANNING FOR MULTIPLE SATELLITE SYSTEMS

AUTONOMOUS ONBOARD MISSION PLANNING FOR MULTIPLE SATELLITE SYSTEMS

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 3 May 2019 at 10:00 am

by

Zixuan ZHENG

Master of Science in Flight Vehicles Design,
Northwestern Polytechnical University, Xi'an, China,
born in Xi'an, Shaanxi Province, China.

This dissertation has been approved by the promotors:

promotor: Prof. Dr. E. K. A. Gill
copromotor: Dr. J. Guo

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. Dr. E. K. A. Gill,	Delft University of Technology, promotor
Dr. J. Guo,	Delft University of Technology, copromotor
Independent members:	
Prof. Dr. A. Tsourdos	Cranfield university, United Kingdom
Prof. Dr. J. P. How	Massachusetts Institute of Technology, USA
Dr. Z. Wang	Tsinghua University, China
Prof. Dr. ir. P. H. A. J. M. van Gelder	Delft University of Technology
Prof. Dr. R. Curran	Delft University of Technology

This research was funded by the China Scholarship Council (CSC), and also supported by the Delft University of Technology.





Keywords: Artificial Intelligence, Onboard Autonomy, Mission Planning, Multiple Satellite Systems, Team Negotiation Mechanisms, Centralized and Distributed optimization.

Printed by: Ipskamp Printing

Cover by: Zixuan Zheng

Copyright © 2019 by Zixuan Zheng

ISBN: 978-94-028-1480-4

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

To my family and all my friends

CONTENTS

Summary xi

Samenvatting xv

1 Introduction 1
- 1.1 Background . . . 2
- 1.2 State of the art . . . 5
- 1.3 Research questions . . . 11
- 1.4 Methodology . . . 12
- 1.5 Thesis outline . . . 13

2 Status Review and setting 15
- 2.1 Background Mission . . . 16
- 2.2 Literature Review . . . 18
  - 2.2.1 Classical approaches . . . 19
  - 2.2.2 Heuristic approaches . . . 23
  - 2.2.3 Techniques related to distributed systems . . . 29
- 2.3 Preliminary concept selection . . . 31
- 2.4 Summary . . . 34

3 Centralized onboard initial planning 37
- 3.1 Introduction . . . 38
- 3.2 Problem Formulation . . . 40
  - 3.2.1 Notations and variables . . . 40
  - 3.2.2 Blocked area . . . 41
  - 3.2.3 Constraints . . . 41
  - 3.2.4 Objective function . . . 43
- 3.3 Hybrid Dynamic Mutation GA . . . 43
  - 3.3.1 Basic GA . . . 44
  - 3.3.2 HDMGA principle . . . 45
  - 3.3.3 Variables determination . . . 47
  - 3.3.4 Performance evaluation . . . 50
- 3.4 Initial planning procedures for the DSL mission . . . 54
  - 3.4.1 Simulation scenario assumptions . . . 54
  - 3.4.2 HDMGA results . . . 55
  - 3.4.3 Evaluation and analysis . . . 57
- 3.5 Summary . . . 58

4 Centralized onboard re-planning **59**
4.1 Introduction . . . 60
4.2 Architecture of Onboard planning & re-planning system . . . 61
4.2.1 General mission planner . . . 61
4.2.2 DS mission executor and monitor . . . 63
4.2.3 Decision-maker . . . 64
4.2.4 Re-planner . . . 66
4.3 Re-planning methods . . . 67
4.3.1 Core algorithm . . . 67
4.3.2 Cyclically re-planning method . . . 67
4.3.3 Near real-time re-planning method . . . 68
4.4 Simulations and analysis . . . 70
4.4.1 Simulation assumptions . . . 71
4.4.2 CRM simulation . . . 71
4.4.3 NRRM simulation . . . 75
4.4.4 Performance comparison . . . 79
4.5 Summary . . . 82

5 Team negotiation **83**
5.1 Introduction . . . 84
5.2 Multi-satellite Mission Allocation problem . . . 85
5.2.1 Organizational architectures . . . 85
5.2.2 Game-theoretical formulation for an MSMA problem . . . 87
5.2.3 Utilities design . . . 88
5.3 Negotiation and cooperation mechanisms . . . 90
5.3.1 Agent communication languages . . . 90
5.3.2 Distributed negotiation mechanism . . . 90
5.3.3 Decentralized negotiation mechanisms . . . 93
5.4 Simulation results and analysis . . . 97
5.4.1 Simulation environment . . . 97
5.4.2 Simulation results of the distributed architecture . . . 98
5.4.3 Simulation results of the decentralized architecture . . . 101
5.5 Summary . . . 106

6 Distributed Onboard Mission Planning Approach **107**
6.1 Introduction . . . 108
6.2 Problem statement . . . 108
6.2.1 Notations and assumptions . . . 110
6.2.2 Variables . . . 111
6.2.3 Constraints . . . 111
6.2.4 Evaluation functions . . . 112
6.3 Distributed mission planning approach . . . 113
6.3.1 Local constraint satisfaction module . . . 113
6.3.2 Global distributed optimization . . . 115

6.4 Verification . . . 117
6.5 Simulation and analysis . . . 118
6.5.1 Study cases . . . 119
6.5.2 Centralized vs Distributed . . . 121
6.5.3 Distributed vs Distributed . . . 123
6.6 Conclusion . . . 125

7 Conclusions **127**
7.1 Summary . . . 128
7.2 Conclusions and Innovations . . . 131
7.3 Outlook . . . 133

References **135**

Acknowledgements **157**

Curriculum Vitæ **159**

List of Publications **161**

SUMMARY

With the rising demands from customers and users and the development of ever advanced technologies, many space missions nowadays require more than one satellite to fulfill their mission objectives. Although replacing single satellite systems (SSSs) by multiple satellite systems (MSSs) offers advantages, such as enhanced spatial and temporal coverage as well as high robustness and multifunctional purposes, it also introduces new challenges. There is no doubt that as the number of satellites in a mission grows, the complexity and operation cost of controlling and coordinating these satellites only by human (or ground based) operators will increase dramatically. In addition, for some deep space missions or complex operational tasks, due to the long signal transmission time between the spacecraft and ground-based antennas or short communication windows, there will not be enough time or resources for operators to sufficiently and efficiently control all of the required onboard functions from mission control centers. Therefore, to enhance the efficiency of operating an MSS, and to reduce the cost of human resources and ground infrastructure, an onboard autonomous system (OAS) for MSS is a promising solution. For specific missions, the use of an OAS may even be a mission enabler. One important function of an OAS is to provide planning and re-planning services based on different mission requirements. The objective of this research is to develop and characterize onboard autonomous mission planning and re-planning approaches for MSSs.

Traditional planning approaches have been reviewed and proven to be inappropriate and inefficient for complex planning problems in the harsh space environment when severe system constraints are enforced and a large number of vehicles constitutes the MSS. To overcome these deficiencies, engineers and researchers have started to develop OAS with the help of Artificial Intelligence (AI) techniques to allow for more complex space missions. Based on the relevance of this problem, the following research questions (RQs) have been formulated and will be answered in this thesis.

RQ1: What are the strengths of using AI in space missions? How to use a centralized AI algorithm in a multi-satellite system to decompose mission objectives and perform mission planning for the entire system?

RQ2: How to define emergency situations which may occur during mission operations? How to use AI algorithms to handle mission re-planning and re-scheduling problems?

RQ3: How to design cooperation and negotiation approaches for an MSS to reach an agreement? How to improve AI algorithms for distributed onboard mission planning problems?

To define potential scenarios, a reference mission is introduced in this thesis, called *Discovering the Sky at the Longest Wavelength (DSL)*. The mission is assumed to comprise one Mother Satellite (MS) and eight Daughter Satellites (DSs) in a lunar orbit. Its scientific objective is to observe the universe in the hitherto-unexplored very low frequency

(below 30 MHz) electromagnetic spectrum. The DSs collect scientific data only in those parts of the orbit which is shielded from radio frequencies emitted by the Earth. These DSs can only transmit collected data to the MS when they are outside of this shielded orbit sections, to prevent interferences caused by communication. This renders mission operations of DSL very complex. The existing body of knowledge on mission planning problems for multi-satellite systems is reviewed. It comprises three categories: classical approaches, heuristic approaches, and advanced techniques (e.g., team negotiation mechanisms, evaluation algorithms). Targeting the complexity of foreseeable DSL planning problems, nine representative optimization algorithms are applied to fourteen test functions. The results indicate that Evolutionary Algorithms (EAs) have a broader adaptability than classical approaches. They are also more efficient than other heuristic approaches. Therefore, EAs family is selected as suitable candidate for the reference MSS.

The operations concept of the DSL mission foresees that the initial mission planning is performed by the MS, while the eight DSs are preliminary executing data collection and transmission tasks. During this phase, the MSS implements a centralized architecture and the MS conducts a centralized planning approach. By comparing basic Genetic Algorithm (GA) with several state-of-the-art improved GAs, its weaknesses are revealed. In this thesis, to overcome early and slow convergence problems, the need to develop a new mutation strategy for GA is motivated. The proposed novel mutation strategy is called Hybrid Dynamic Mutation (HDM), which contains a standard mutation operator and an escape mutation operator. While the standard mutation operator uses a small mutation rate for approaching the global optimum, the escape mutation operator uses a larger mutation rate to allow an escape from local optima. The simulation results indicate that the proposed HDM can improve the basic GA (which turns into the HDMGA) leading to a superior performance on correctness and effectiveness as compared to alternative GAs. Based on these findings, AI related methods are considered a promising category as compared to classical methods due to their flexibility and effectiveness to support the onboard planning for an MSS. In addition, the proposed HDMGA also provides a satisfying result for the considered initial mission planning problems.

Internal or external causes, e.g. an actuator failure or the challenging space environment, can lead to a satellite malfunction during mission operations. This thesis considers the two most important behaviors of the DSL mission, observation and communication, and proposes potential emergency scenarios to handle possible system failures on DSs. Two re-planning methods, one called the Cyclically Re-planning Method (CRM), the other one the Near Real-time Re-planning Method (NRRM), are established and compared. The CRM performs re-planning at the beginning of each orbit and only re-plans for one orbit. The NRRM performs re-planning in a near real-time setting when the emergency occurs. Its re-planning covers for the rest of the mission. Three simulation study cases are formulated based on assumed emergency scenarios. The proposed two methods are compared on three aspects: the total number of data observed from all DSs within a certain time frame, the total number of data the MS received from all DSs within a certain time frame, and the average computation time needed for re-planning. The results indicate that: (1) The NRRM allows to observe and transmit more data than the CRM within a specific operational lifetime. (2) The NRRM requires more computational time than the CRM for emergency situations, while it requires less time than the

CRM for nominal situations.

This research also covers a much more severe scenario, namely that the MS becomes fully non-functional in an emergency situation. This would render the MS unable to provide mission planning and re-planning services for the MSS. Without its main controller on the MS, all DSs now need to cooperate to jointly solve the mission planning problems. Due to the loss of the MS, both distributed and decentralized architectures, which the MSS could then use are introduced. In a distributed architecture, each DS is connected with all other DSs directly or through DS which acts as retranslator. In a decentralized architecture, each DS can only communicate with its neighbors. Considering that the mission allocation problems in different organizational architectures are similar to information games in game theory, a game-theoretical model of the Multi-Satellite Mission Allocation (MSMA) problem is formulated. The Utility-based Regret Play (URP) negotiation mechanism is proposed for an MSMA problem using a distributed architecture. It inherits the ability to evaluate individual utility at each negotiation step from the Utility-based Fictitious Play, and the ability to regret the current choice and for not proposing particular choices in the past negotiation steps from the Regret Matching Play. The Smoke Signal Play (SSP) and Broadcast-based Play (BBP) are developed for a decentralized architecture instead. The SSP is inspired by an old communication method called *Smoke Signal*, where each satellite is considered as a smoke tower, passing information of utility to its succeeding neighbor. The BBP uses broadcasting as the communication method, where each satellite can transmit information to all its neighbors. The simulation results show that the URP can outperform the other three state-of-the-art mechanisms (Action-based Fictitious Play, Utility-based Fictitious Play, and Regret Matching Play) with studied cases. For the decentralized architecture, the results reveal that both SSP and BBP can provide valid solutions for mission allocation problems. The BBP mechanism shows a superior performance on computation time as compared to SSP and a state-of-the-art approach called Market-based Auction. The SSP mechanism, on the other hand, shows the best performance with respect to power consumption.

To solve complex optimization problems in distributed mission planning scenarios, an approach, named Hybrid Distributed GA (HDGA) is proposed. This approach contains two modules: the Local Constraint Satisfaction module (LCS) and the Globally Distributed Optimization module (GDO). In the LCS module, the greedy best-first search algorithm is employed as the local search heuristic for helping each DS to find suitable solutions which can satisfy individual constraints. This module is designed to generate multiple solutions to form local populations for the GDO. The GDO module employs the HDMGA as the core optimization algorithm, while the individual populations are formed through the local populations exchange procedure between one participant and all other participants. For a standard planning case, the results indicate that HDGA can reduce the computation time while ensuring a higher success rate compared to the HDMGA. Comparing the HDGA with two other state-of-the-art distributed optimization algorithms, the Distributed Ant Colony Optimization (DACO) and the Coevolutionary Particle Swarm Optimization (CPSO), the statistical results reveal that HDGA is more stable and accurate to handle large-scale planning problems. The HDGA also shows the best performance on computation time among all tested distributed approaches.

SAMENVATTING

Met de stijgende vraag van klanten en gebruikers en de ontwikkeling van steeds meer geavanceerde technologieën worden veel ruimtemissies tegenwoordig uitgevoerd door systemen die uit meer dan één satelliet bestaan. Hoewel het vervangen van afzonderlijke satellietsystemen (SSS's) door multi-satellietsystemen (MSS's) voordelen biedt, zoals een verbeterde ruimtelijke en temporele dekking als ook een betere robuustheid en multifunctionaliteit, brengt het ook tot nieuwe uitdagingen. Het lijdt geen twijfel dat bij toenemend aantal satellieten in een missie de complexiteit en de operationele kosten van het controleren en coördineren van deze satellieten met uitsluitend grondstations, drastisch zullen toenemen. Voor sommige missies naar de verre ruimte of missies met uiterst complexe operationele taken, waarbij het communicatiesignaal een lange weg moet afleggen tussen MSS en grondstation en/of slechts gedurende een korte tijd gecommuniceerd kan worden met het MSS, maakt dat er voor het grondpersoneel onvoldoende tijd en/of middelen beschikbaar zijn om alle taken aan boord van het MSS optimaal te kunnen plannen en regelen. Om de efficiëntie van een MSS te verbeteren, en om de kosten van personele middelen en grondinfrastructuur te verlagen, is een ingebouwd autonoom systeem (OAS) voor MSS een veelbelovende oplossing. Een OAS kan daarnaast ook missies mogelijk maken die anderszins niet mogelijk zouden zijn. Een belangrijke taak van een OAS is om het autonoom (her)plannen van de missietaken binnen de missievereisten te realiseren. Het doel van dit onderzoek is om een dergelijke autonome aanpak voor MSSs te ontwikkelen en te karakteriseren. Traditionele methoden van missieplanning zijn geanalyseerd. Gevonden is dat de traditionele methoden niet geschikt of inefficiënt zijn voor het plannen van de meer complexe taken van een MSS in de meedogenloze ruimteomgeving met name als dit MSS bestaat uit een groot aantal satellieten en met inachtneming van alle beperkingen van zo een MSS. Om deze beperkingen voor complexe ruimtemissies te boven te komen, zijn ingenieurs en onderzoekers begonnen een OAS te ontwikkelen met gebruik making van Kunstmatige Intelligentie (KI) -technieken. Gebaseerd op de relevantie van dit probleem zijn een drietal onderzoeksvragen (OV'n) geformuleerd, die in dit proefschrift zullen worden beantwoord. OV1: Wat zijn de voordelen van het gebruik van KI in ruimtemissies? Hoe een gecentraliseerd KI-algoritme te gebruiken in een multi-satellietsysteem om missiedoelen te analyseren en op basis daarvan een missieplanning te genereren voor het gehele systeem? OV2: Hoe kunnen noodsituaties, die zich voordoen tijdens missieactiviteiten, worden onderkent? Hoe kunnen KI-algoritmen worden ingezet bij het (hernieuwd) plannen van de missieactiviteiten bij gerezen problemen? OV3: Hoe kan een wijze van aanpak voor samenwerking en afweging worden uitgewerkt, die een succesvolle missieplanning voor een MSS mogelijk maakt? Hoe kunnen KI-algoritmen voor gedistribueerde missieplanning aan boord van het ruimtevaartuig worden verbeterd? Om mogelijke scenario's op te stellen die in dit proefschrift nader worden onderzocht met betrekking tot missieplanning, is een referentiemissie geïntroduceerd, genaamd "Discovering the Sky at the

Longest Wavelength" (DSL). Deze missie heeft als wetenschappelijk doel om het universum te observeren in het lage frequentiegebied (minder dan 30 MHz), een tot nog toe niet eerder voor observatie gebruikt deel van het elektromagnetisch spectrum. De missie wordt uitgevoerd door een MSS bestaande uit een moeder-satelliet (MS) en acht dochter-satellieten (DS's) die zich allen in een baan om de Maan bevinden. De DS's verzamelen alleen wetenschappelijke gegevens in die delen van de baan, waarin zij door de Maan afgeschermd worden van door de Aarde uitgezonden straling in het betreffende frequentiegebied. Verzamelde data worden door deze DS's alleen naar de MS verzonden als ze zich buiten deze afgeschermde baandelen bevinden, dit om storingen van de metingen veroorzaakt door het communicatiesignaal te voorkomen. Dit maakt missie operaties van DSL erg complex. De bestaande kennis over aanpak van missieplanning voor MSS is onderzocht. Drie verschillende categorieën van aanpak worden onderscheiden: klassieke aanpak, en heuristische aanpak met en zonder gebruikmaking van geavanceerde technieken (bijv. team onderhandelings-mechanismen, evaluatie algoritmen). Gericht op de complexiteit van te verwachten planningsproblemen in de DSL-missie, zijn negen representatieve optimalisatie-algoritmen uitgetest op veertien testfuncties. De resultaten tonen dat "Evolutionary Algorithms" (EA's) zich gemakkelijker laten aanpassen voor verschillende toepassingen dan bij een klassieke aanpak. Ze zijn ook efficiënter dan andere heuristische benaderingen. Het is om deze redenen dat de EA-familie is geselecteerd als geschikte kandidaat voor toepassing in de referentie MSS. In het operationele concept van de DSL-missie wordt de initiële planning uitgevoerd door de MS, terwijl de acht DS's zich voornamelijk bezig houden met het vergaren van de observatiegegevens en het overzenden van deze gegevens naar de MS. Tijdens deze fase hanteert de MSS een gecentraliseerde architectuur en voert de MS een gecentraliseerde planningsaanpak uit. Uitgaande van een standaard Genetisch Algoritme (GA) zijn de zwakke punten van deze standaard geïdentificeerd. Om trage convergentie en problemen met convergentie in de vroege optimalisatiefasen te voorkomen wordt in dit proefschrift de noodzaak om een nieuwe mutatiestrategie voor GA te ontwikkelen gemotiveerd. De voorgestelde nieuwe mutatie-strategie is genaamd "Hybrid Dynamic Mutation" (HDM) en omvat een standaard mutatie-operator en een escape-mutatie-operator. Terwijl de standaard mutatie-operator een kleine mutatiesnelheid gebruikt voor de bepaling van het globale optimum, gebruikt de escape-mutatie-operator een grotere mutatiesnelheid om te vermijden dat de oplossing blijft steken in lokale optima. De simulatieresultaten tonen dat de voorgestelde HDM in combinatie met GA (kortweg aangeduid als HDMGA) superieur is op gebied van correctheid en effectiviteit in vergelijking met andere GA's. Op basis van deze bevindingen worden KI-gerelateerde methoden als een veelbelovende categorie beschouwd in vergelijking met meer klassieke methoden ter flexibilisering en een meer effectieve ondersteuning van de gecentraliseerde missie planningsaanpak aan boord van de MS. Bovendien biedt de voorgestelde HDMGA ook een bevredigend resultaat voor de problemen zoals meegenomen bij de initiële missieplanning. Interne of externe oorzaken, b.v. een actuatorstoring of een weerbarstige ruimteomgeving, kunnen tijdens missie-operaties leiden tot een storing in een satelliet. In dit proefschrift worden de twee belangrijkste taken van de DSL-missie, observatie en gegevensoverdracht (communicatie) nader beschouwd en worden mogelijke scenario's gedefinieerd voor het afvangen van systeemstoringen aan boord

van één of meerdere DS's. Twee herplannings-methoden, de een genaamd de "Cyclic Re-planning Method" (CRM) en de andere de "Near Real-time Re-planning Method" (NRRM), zijn ontwikkeld en nader onderzocht. De CRM voert een nieuwe planning uit aan het begin van elke baanomloop en wel voor de duur van een enkele omloop. De NRRM vernieuwt de planning bijna instantaan als zich een noodsituatie voordoet. Deze vernieuwde planning omvat de gehele verdere missie. Voor het onderzoek zijn drie simulatiegevallen geformuleerd op basis van een aantal aangenomen noodscenario's. De beide methoden zijn vergeleken op drie aspecten: het totale aantal gegevens verzameld door alle DS's binnen een bepaald tijdsbestek, het totale aantal gegevens dat de MS van de DS's heeft ontvangen binnen een bepaald tijdsbestek, en de gemiddelde computertijd die nodig is voor het hernieuwen van de planning. De resultaten tonen dat: (1) De NRRM meer gegevens verzamelt en verzendt dan de CRM binnen een specifieke operationele levensduur. (2) De NRRM meer computer tijd vereist dan de CRM voor noodsituaties, en méér onder nominale omstandigheden. Dit onderzoek dekt ook een veel ernstiger noodscenario, namelijk dat de MS niet langer functioneert. Dit maakt de MS onbekwaam om missie(her)planningsdiensten uit te voeren voor de MSS. Zonder de hoofdcontroller op de MS moeten nu alle DS samenwerken om gezamenlijk de missie(her)planningsproblemen op te lossen. Hiertoe zijn zowel gedistribueerde als gedecentraliseerde organisatie-architecturen voor de MSS geïntroduceerd. In een gedistribueerde architectuur is elke DS direct verbonden met alle andere dochters of naar een enkele DS die dan weer als relais fungeert naar anderen. In een gedecentraliseerde architectuur daarentegen kan elke DS slechts communiceren met zijn directe buren. Daar de toewijzing van missietaken in verschillende organisatie-architecturen vergelijkbaar is met die in informatiespellen in speltheorie, is een spel-theoretisch model van het "Multi-Satellite Mission Allocation" (MSMA) probleem geformuleerd. Het "Utility-based Regret Play" (URP) onderhandelingsmechanisme wordt voorgesteld voor een MSMA-probleem in een gedistribueerde architectuur. Het combineert de mogelijkheid om het individuele nut te evalueren bij elke onderhandelingsstap zoals in "Utility-based Fictitious Play", en om de huidige keuze te betreuren en bepaalde voorstellen van keuzes in eerdere onderhandelingsstappen achterwege te laten zoals in Regret Matching Play. Voor een gedecentraliseerde architectuur zijn in plaats daarvan "Smoke Signal Play" (SSP) en "Broadcast-Based Play" (BBP) ontwikkeld. SSP is geïnspireerd op een oude communicatiemethode, "Smoke Signal" genoemd, waarbij elke satelliet wordt beschouwd als een rooktoren, die nuttige informatie doorgeeft aan zijn directe buren. Bij BBP kan elke satelliet informatie verzenden naar alle DS's. De simulatieresultaten tonen aan dat, voor de bestudeerde gevallen, URP beter kan presteren dan drie andere "state-of-the art" mechanismen ("Action-based Fictitious Play", "Utility-based Fictitious Play" en "Regret Matching Play"). Voor de gedecentraliseerde architectuur tonen de resultaten dat zowel SSP als BBP valide oplossingen bieden voor de verdeling van missietaken over de systeemelementen. Het BBP-mechanisme toont superieure prestaties qua computertijd in vergelijking met SSP en een andere state-of-the-art benadering genaamd "Market-based Auction". Het SSP-mechanisme, aan de andere kant, heeft als voordeel het laagste energieverbruik.

Om complexe optimalisatieproblemen in gedistribueerde missieplanningsscenario's op te lossen, wordt een hybride aanpak voorgesteld, genaamd "Hybrid Distributed" GA

(HDGA). In deze aanpak wordt gebruik gemaakt van een tweetal modules: de “Local Constraint Satisfaction”-module (LCS) en de “Globally Distributed Optimisation” module (GDO). In de LCS-module wordt het “Greedy Best-First” zoek-algoritme toegepast om voor elke DS geschikte oplossingen te vinden, die aan de individuele beperkingen van de DS tegemoet komen. Deze module is ontworpen om meerdere lokale oplossingen te genereren die dan worden doorgegeven aan de GDO-module. De GDO module maakt gebruik van het HDMGA optimalisatie-algoritme waarbij de individuele populaties worden gevormd via de ingestelde procedure voor uitwisseling van lokale populaties tussen de verschillende deelnemers. Voor een standaard planningsgeval geven de resultaten aan dat HDGA de rekentijd kan verkorten en tegelijkertijd tot een hoger slagingspercentage leidt dan HDMGA. Resultaten verkregen met HDGA in vergelijking met twee andere “state-of-the-art” gedistribueerde optimalisatie-algoritmen, de “Distributed Ant Colony Optimization” (DACO) en de “Coevolutionary Particle Swarm Optimisation” (CPSO) laten zien dat HDGA stabieler en nauwkeuriger is voor het oplossen van grootschalige planningsproblemen. HDGA toont ook de beste prestaties qua computer tijd van alle geteste alternatieve gedistribueerde aanpakken.

1

INTRODUCTION

1

THIS chapter provides an introduction to the content of this dissertation: Autonomous Onboard Mission Planning for Multiple Satellite Systems (MSSs). Firstly, the background on the need and development of various space missions is introduced. The growing requirements from users of space services which trigger the motivation to develop autonomous onboard systems is revealed. Then, the state of the art of representative space missions which use autonomous systems is briefly discussed. The concept and application area of MSSs are introduced. In the sequel, the control and coordinate approaches for MSSs are categorized to identify the current body of knowledge and to allow to identify innovative research areas. As a result, three research questions are formulated, followed by a description of the methodology used to answer these questions. Finally, a short description of the content of the individual chapters of this thesis is provided.

1.1. BACKGROUND

HISTORY OF SPACE MISSIONS

On October 4th, 1957, the Soviet Union launched their first artificial satellite Sputnik 1 (Figure 1.1) [Wikipedia, 2018l] into an elliptical low Earth orbit. This mission opened a new chapter in the human exploration of space. After this, many countries became space-faring nations by launching their satellites, such as the United States of America (Explorer 1, 1958 [Wikipedia, 2018d]), France (Astérix, 1965 [Wikipedia, 2018a]), Japan (Ohsumi, 1970 [Wikipedia, 2018i]), and China (Dong Fang Hong 1, 1970 [Wikipedia, 2018c]).

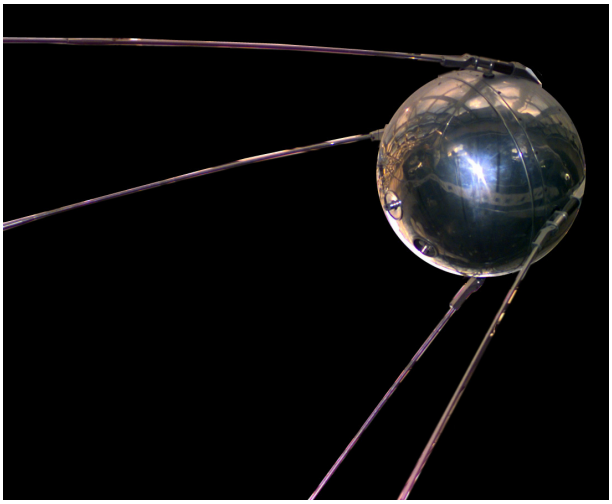


Figure 1.1: The first artificial satellite: Sputnik 1 (taken from [Wikipedia, 2018l])

Driven by the “Space Race" between the Soviet Union and the United States of America, many milestones have been achieved, such as the first automated soft landing on the Moon (Luna 9 [Dunham et al., 2002]), the first manned Moon landing (Apollo 11 mission [Berry, 1970]), and the first space station (Salyut 1 [Bluth and Helppie, 1986]). During

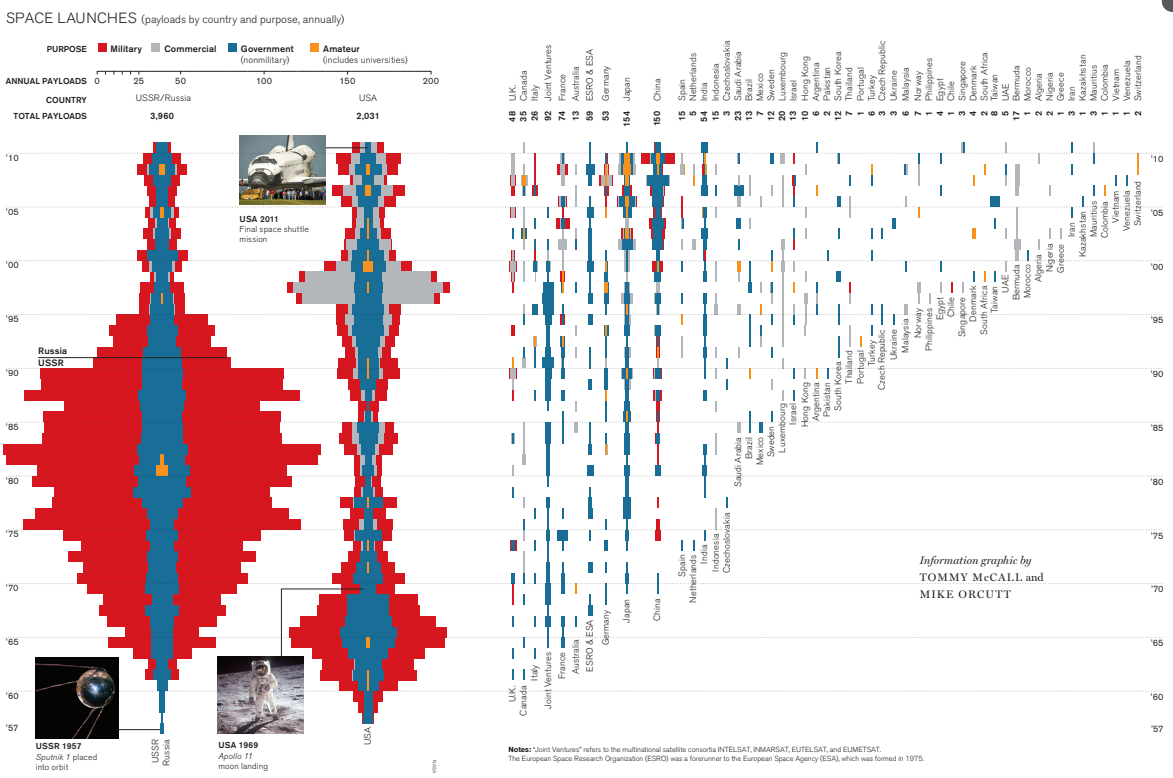


Figure 1.2: Space missions launched over the past 50 years (This figure is taken from [Orcutt, 2018])

this period, these space missions were more triggered by military and political purposes than scientific applications. Figure 1.2 shows the satellites that have been launched over the past 50 years.

After the 1970s, civil space missions began to dominate. With the increasing number of space applications, such as communications and people started to benefit from space technologies. For instance, the fixed satellite services could provide information transmission for terrestrial users, while mobile satellite services could help to connect mobile communication units. Space missions for scientific research could provide many types of applications, such as *Solar System Exploration* (Voyager 1 & 2, ICE, Galileo, etc.) and geodesy (LAGEOS 1 & 2, QuikSCAT, etc.). *Communication* and *navigation* as well as *Earth observation* became key application pillars of space flight (ATS-6, Ekran, Iridium, GPS, GLONASS, Galileo, etc.).

Later on, with the development of advanced technologies and the rising demands from customers and users, many space missions required more than one satellite to ful-

fill their mission objectives. The central concept of an MSS is that a group of spacecraft performs complex tasks where each individual satellite contributes to the overall mission goal. Many existing missions have implemented an MSS. For example, the Global Positioning System (GPS) constellation consists of 31 satellites in orbit to provide navigation services for military and civilian purposes; the Earth Observing-1 (EO-1) & LandSat-7 (2 satellites) monitor the lava flows from space; the Gravity Recovery and Climate Recovery (GRACE) (2 satellites) mission provided detailed measurement of the Earth's gravity field; the PRISMA (2 satellites) mission provided a radio frequency metrology system that enabled these two satellites to fly in close formation while autonomously avoiding collisions. The idea of the MSS becomes particularly interesting when combined with the trend towards miniaturization of future space systems.

MORE IS BETTER?

Since NASA's *Faster, Better, Cheaper (FBC)* [McCurdy, 2001] missions have achieved impressive accomplishments, many space researchers have began to search for ways to accomplish given tasks in less time and at less cost. Instead of using one large instrument on board a very expensive large spacecraft, using an MSS with a group of small, inexpensive spacecraft seemed a better approach. Compared with single-satellite systems (SSSs), MSSs show many advantages, such as low cost, high robustness, multifunctional purposes, and enhanced spatial and temporal coverage.

Take the Spektr-R space project as an example. The objective of this project is to study astronomical objects with an angular resolution up to a few millionths of an arcsecond. The total cost of this mission, which uses a large single spacecraft is more than 22 million €[Zak, 2018b]. This does not include the annual expense of mission operations and maintenance. Considering the low cost of a small satellite platform, if the Spektr-R satellite could be replaced by ten MicroSats or CubeSats, the overall cost might be reduced significantly.

Space is very challenging environment. Many external factors such as thermal conditions, radiation, solar wind, space debris, and vacuum environment can cause the failures of satellites. Therefore, using multiple small satellites can reduce the out-of-service risk caused by the space environment or internal system malfunction. This in turn enhances the robustness of the entire system. Furthermore, large satellites are customarily designed for specific missions. The unique payloads these satellites are carrying usually cannot be used for other space missions. This limits the efficiency of SSS. MSS, on the other hand, can provide diverse functions by augmenting an existing system with a new satellite hosting a desired payload.

Although using an MSS to replace the functionality of one large satellite can be beneficial, it also suffers many difficulties. Usually, space missions are strongly dependent on the ground segment and the operations performed by flight engineers who monitor the enormous amount of telemetry data sent back to Earth during operations, plan maneuvers, command the satellite for bus or payload activities. There is no doubt that as the number of satellite grows, the complexity and operations cost of controlling and coordinating these satellites only by operators will increase dramatically. In addition, for some deep space missions or complex operational tasks, due to a long signal transmission time or short communication windows, there is not enough time or resources for

operators to control all of the needed onboard behaviors from the control centers.

SMARTER IS BETTER!

With the increasing complexity of new space missions, it becomes inefficient for operators to control the entire satellite from the mission control center. To overcome these deficiencies, space engineers and researchers have started to develop onboard autonomous systems (OAS) with the help from various artificial intelligence (AI) techniques to enhance the satellite intelligence for more complex space missions.

As early as in 1998, the Deep Space 1 mission was the first space mission to demonstrate an onboard autonomous system by using the Remote Agent (RA) architecture [Muscettola et al., 1998]. Later on, in 2000, the EO-1 demonstrated the onboard fault diagnostics and recovery, as well as a considerable autonomy of the science instruments and downlink of the resulting imagery and data [Chien et al., 2004]. The DLR BIRD satellite had an onboard navigation system (ONS), which was able to compute the instantaneous nadir and flight direction for camera pointing, as well as precise positions for real-time geocoding of image data. It also could perform real-time estimation of SGP4 mean elements, allowing an onboard forecast of ground station contacts or eclipse times [Gill et al., 2000]. Other autonomous systems have also been used for various space missions, such as Orbital Express [Wikipedia, 2018j] and Mars Exploration Rovers [Zak, 2018a].

To implement an MSS as a platform for current (e.g. TDRS [Holmes, 1978], STEREO [Kaiser et al., 2008]) or upcoming space missions (e.g. ANTS [Truszkowski et al., 2004], MAIA [NASA, 2018b]), the OAS plays a more important role since it can assist the MSS in several ways:

1. Firstly, the OAS typically is indispensable when communications between the satellites and the ground are insufficient to operate the satellite. Then, the OAS can make decisions based on the system's current state and the mission objectives.

2. Secondly, the OAS is the only way to control a large number of satellites without requiring huge human and computational resources as well as infrastructure on the ground.

3. Thirdly, the OAS can provide onboard team coordination and negotiation services without any interference from human, which is more efficient.

Considering all the benefits the OAS can bring, the objective of this thesis is to develop concepts and approaches of intelligence systems for an MSS to perform different levels of onboard autonomy.

1.2. STATE OF THE ART

Many early studies have already implemented autonomous technologies to fulfill the different requirements for either single-satellite missions or multi-satellite missions. In this section, a survey of the research topics regarding onboard autonomy for space systems is presented. Related research in non-space domains is also discussed in this section.

ONBOARD AUTONOMY IMPLEMENTATIONS

An essential application for onboard autonomy is to facilitate the onboard navigation of satellites to solve the navigation problems, which have been investigated for the past four decades. In the last 20 years, with the rapid development of onboard computers (including their growing computational power and memory, and decreasing mass), onboard Autonomous Navigation Systems (ANS) have been designed for satellites to determine their orbit and to predict their motion parameters automatically. There are four main branches for measurement systems which are key to the development of ANS. The first branch uses radiometric systems based on radio frequencies. This is by far the most widely and mature technique that has been used for many satellites' ANS, such as BIRD [Gill et al., 2000], PRISMA [Gill et al., 2007], and X-SAT [Gill et al., 2004]. The second branch is based on magnetometers to sense the Earth's magnetic field, which has been revealed in [Shorshi and Bar-Itzhack, 1995], [Wiegand, 1996], and [Psiaki et al., 1993]. The third branch is based on star sensors using stellar refraction, such as the implementations in [Gounley et al., 1984], [Yunfeng and Renwei, 1995], and [Ning et al., 2013]. The fourth branch is based on pulsars as shown in [Sheikh et al., 2006], [Shuai et al., 2007], and [Zhang et al., 2017].

Rendezvous and docking (RVD) is another application which typically requires support from OASs. In 1998, the Engineering Test Satellite-VII (ETS-VII) [Kawano et al., 1999] successfully performed the first autonomous RVD between unmanned spacecraft. Afterwards, many studies have been done to further facilitate RVDs by using different techniques. Based on the characteristics of the target, autonomous RVD can be divided into two types: for collaborative targets [Romano et al., 2007] and for non-cooperative targets [Chen and Xu, 2006]. To fully operate RVD autonomously, a variety of research topics have also been investigated, including absolute and relative autonomous navigation technologies [Gill et al., 2001; Fehse, 2003; D'Amico et al., 2006], the autonomous RVD close-in sensors [Ruel et al., 2012], and autonomous RVD algorithms for different operation phases [Karr et al., 1990; Mukundan et al., 1994; Guglieri et al., 2014].

Implementing OASs for MSSs can reduce the operating cost on the ground segment and enhance the real-time reaction capabilities for unexpected situations. Several representative satellites which apply OASs are shown in Tab 1.1. The existing OASs can provide a convenient step-wise approach to initiate fully autonomous formation flying, relative navigation, guidance, and onboard control. Several ongoing space missions are also included to demonstrate multi-satellite systems such as PROBA mission [ESA, 2018b], JC2Sat-FF mission [ESA, 2018a], and the CanX program [Wikipedia, 2018b].

CONTROL MODELS FOR MULTI-SATELLITE SYSTEMS

A multi-satellite system contains two or more satellites working together to accomplish a common goal. So it can also be recognized as a distributed space system (DSS). The control models for MSSs can be diverse, depending on the organizational relationship between the participating satellites. Two prevalent types of control models exist in current DSS studies; the centralized control model (CCM), and the distributed control model (DCM).

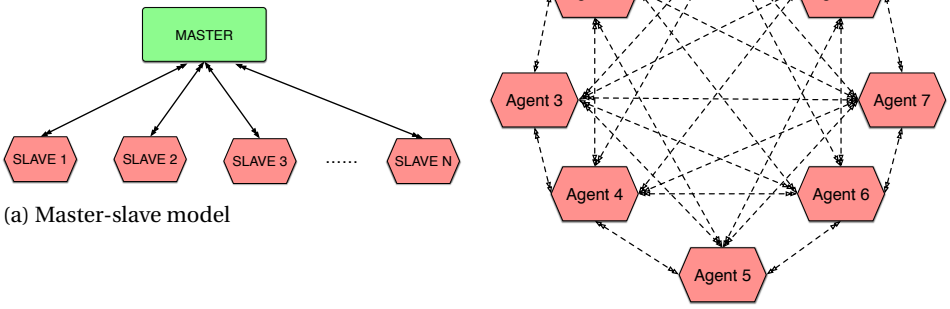
The CCM is also referred as the Master-Slave model [Kang et al., 2001] (Fig 1.3a), which relies on a central controller (Master) which has access to all the information

Table 1.1: Several representative multi-satellite systems: (1) The Morning and Afternoon Constellations [Wikipedia, 2018h] are a group of Earth orbiting satellites with synergistic science objectives in similar sun-synchronous orbits; (2) The Gravity Recovery and Climate Experiment (GRACE) mission [Wikipedia, 2018e] uses two satellites to perform detailed measurement of Earth's gravity field; (3) The TanDEM-X mission [Wikipedia, 2018m] uses Synthetic Aperture Radar to generate a consistent global Digital Elevation Model (DEM); (4) The Gravity Recovery and Interior Laboratory (GRAIL) mission [Wikipedia, 2018f] is a lunar science mission to map the gravitational field of the Moon; (5) The Magnetospheric Multi-scale Mission (MMS) [NASA, 2018a] is a unmanned space mission to study the Earth's magnetosphere using four satellites; (6) The PRISMA mission [Gill et al., 2007] contains two satellites to perform autonomous formation flying;

Name	Satellites	Purpose	Year	Owner	Altitude
Afternoon Constellation	Aqua	A synergistic instrument package measuring at visible, infrared, and microwave frequencies allows comprehensive studies of water in the Earth/atmosphere system.	2002	NASA/GSFC	705 km
	Aura	Observations from limb sounding and nadir imaging allow studies of the horizontal and vertical distribution of key atmospheric pollutants and greenhouse gases and how these distributions evolve and change with time.	2004	NASA/GSFC	
	CALIPSO	Observations from space-borne lidar, combine with passive imagery, will lead to improved understanding of the role aerosols and clouds play in regulating the Earth's climate.	2006	NASA/GFSC /LaRc	
	CloudSat	Cloud Profiling Radar allows the most detailed study of clouds to date and should better characterize the role clouds play in regulating the Earth's climate.	2006	NASA/GSFC /JPL	
	GCOM-W1	Observations of water circulation changes. Specifically it observes precipitation, vapor amounts, wind velocity above the ocean, sea water temperature, water levels on land areas, and snow depths.	2012	JAXA	
	OCO-2	Three grating spectrometers are used to make global, space-based observations of the column-integrated concentration of CO2, a critical greenhouse gas.	2014	NASA/GSFC /JPL	
Morning Constellation	LandSat-7	This mission provides global coverage, and spectral characteristics to allow comparisons for global and regional change detection and image data to various international users throughout the world during times of sudden global changes (e.g. earthquakes or floods).	1999	USGS/NASA	705 km
	Terra	Terra is a multi-national, multi-disciplinary mission that helps us to understand how the complex coupled Earth system of air, land, water and life is linked.	1999	NASA/GSFC	
	EO-1	This mission develops and validates a number of instrument and spacecraft bus breakthrough technologies. It is designed to enable the development of future earth imaging observatories that leads to a significant increase in performance while also having reduced cost and mass.	2000	NASA/GSFC	
	LandSat-8	This mission provides moderate-resolution measurements of the Earth's terrestrial and polar regions in the visible, near-infrared, short wave infrared, and thermal infrared. Landsat 8 provides continuity service with the 38-year lifetime.	2013	USGS/NASA	
GRACE	GRACE-1	GRACE is the first Earth-monitoring mission in the history of space flight. Twin satellites took detailed measurements of Earth's gravity field.	2002	NASA/DLR	500 km
	GRACE-2				
TanDEM-X Mission	TerraSAR-X	An imaging radar Earth observation satellite. It is a joint venture which is carried out under a public-private-partnership between the German Aerospace Center (DLR) and EADS Astrium.	2007	DLR/EADS	514 km
	TanDEM-X	TerraSAR-X's twin satellite, an observation satellite using SAR (Synthetic Aperture Radar) technology. It is an almost identical spacecraft to TerraSAR-X	2010		
GRAIL Mission	GRAIL A (Ebb)	Satellites transmit and receive telemetry from the other spacecraft and Earth -based facilities. To measure the gravity field and geological structure of the Moon.	2011	NASA/JPL	Periselene: 25 km Aposelene: 86 km
	GRAIL B (Flow)				
MMS Mission	No.1	Four identical spacecraft flying in a tetrahedral formation to study the Earth's magnetosphere.	2015	NASA	Perigee: 2,550 km Apogee: 70,080 km - 152,900 km
	No.2				
	No.3				
	No.4				
PRISMA Mission	Mango	Two satellites is used perform autonomous formation flying with autonomously avoiding collisions.	2010	SSC	Perigee: 668,3 km Apogee: 749 km
	Tango				

of the entire system. Based on the information sent from each participant (Slave), the controller needs to make all decisions regarding to specific mission objectives. Many distributed systems chose CCMs because of their relatively simple implementation (e.g. [Shao et al., 2000; Pei et al., 2004]). However, the CCM may become infeasible for distributed systems with a large number of participating satellites. Such systems can lead to an unacceptable complex communication environment where the jammed channels can make the master unable to access the data provided by each participant in time. These potential disabilities restrain the application of CCM for distributed systems. Meanwhile, the CCM may also become the performance bottleneck for an MSS due to its requirements on the computational power.

DCMs, in contrast, can be used for those distributed systems which have a large



(a) Master-slave model

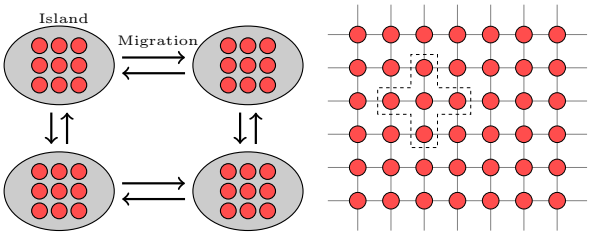
(b) Multi-agent model

Figure 1.3: Centralized master-slave model (left) and distributed multi-agent model (right)

number of participating satellites. In the terminology of distributed systems, participating satellites may also be called the agents of the systems. In a DCM, all the participants need to work through team negotiation and coordination since there is no central controller that makes decisions. One representative DCM is called the Multi-agent model (Fig 1.3b). This concept was first proposed for a computer system which consists of multiple interacting intelligent agents (also referred to a Multi-agent System (MAS) [Ferber and Weiss, 1999]). An automated intelligent agent in a MAS is commonly recognized to have several characteristics: (1) Decision-making ability. An agent can make decisions individually based on its status and mission requirements without any direct intervention by human operators; (2) Interaction ability. An agent needs to be able to communicate and establish interactions with other agents to achieve the global goal; (3) Reaction ability. An agent needs to detect the changes in either the external operating environment or internal system status and provides corresponding strategies. Judging by these characteristics, the participating satellites of an MSS can be considered as automated intelligent agents.

Considering the diversity of space missions using MSSs, the corresponding multi-agent models can have different topologies [Gong et al., 2015], such as the island topology (Fig 1.4a), the cellular topology (Fig 1.4b), and several hybrid topologies (Fig 1.5).

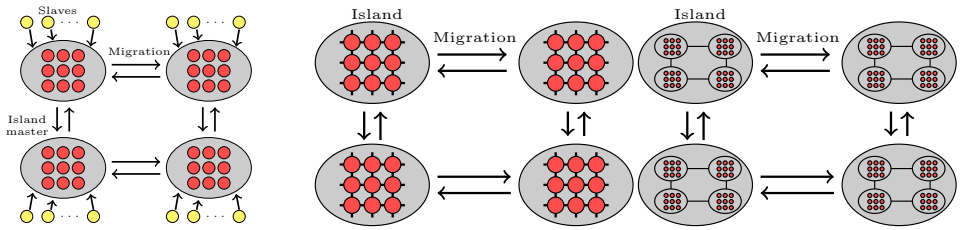
Fig 1.4a indicates the island topology while Fig 1.4b shows the cellular topology. The difference between an island topology and a cellular topology lies in the parallelization grain. The island model is coarse-grained where each island consists of several subpopulations, represented by the red circles in this figure. The inter-island communication is done by migrating from one individual to another individual in another island. Using the island topology can improve global search ability and save search time. The cellular topology, on the other hand, is a fine-grained structure, where all the individuals are placed in a network. Each individual is connected with its neighbors, which are the



(a) Island topology (b) Cellular topology

Figure 1.4: Sample topologies for multi-agent systems

only individuals it can communicate with and connect to. Cellular topology is particularly efficient for highly complex problems. Besides these two typical topologies, hybrid topologies combine two or more topologies hierarchically to absorb all their benefits, improving problem-solving capability and scalability. Fig 1.5 shows three representative hybrid topologies.



(a) Island-master topology (b) Island-cellular topology (c) Island-island topology

Figure 1.5: Sample hybrid topologies for multi-agent systems

COORDINATION FOR MULTI-SATELLITE SYSTEMS

Based on different mission requirements, the same MSS may required different coordination strategies. Team coordination for an MSS can be performed separately based on two kinds of scenarios: cooperation scenarios and competition scenarios, as shown in Fig 1.6.

For cooperation scenarios, all the satellites work together for a collective global goal. Due to the differences in topology and system architecture, the cooperation scenarios may use the centralized model or the distributed model. The centralized model uses a central controller which employs centralized planning approaches [Siciliano et al., 2010] for assigning tasks and generating control sequences for each team member. Using the distributed model, all satellites have to perform team collaboration to solve task allocation problems [Zlot and Stentz, 2005]. Then, based on the sub-tasks assigned to each satellite, the team needs to perform distributed planning [Siciliano et al., 2010] to generate feasible control sequences to meet all individual and global constraints.

For competition scenarios, each satellite is considered as a self-interested agent. For

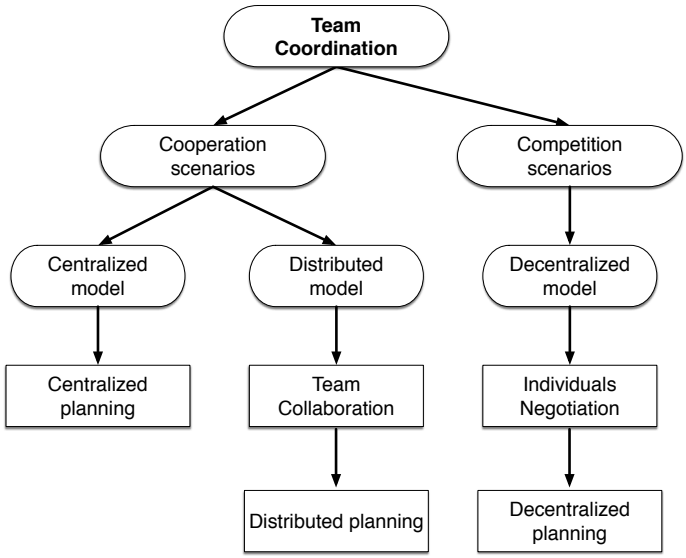


Figure 1.6: Concept discovery tree of team coordination approaches

these scenarios, the MSS could use the decentralized architecture. Here, the local objective and system status of each satellite is not fully transparent to other satellites. Therefore, in order to continue operating the mission, all individual satellites need to negotiate to reach a global agreement on task allocation problems [Shehory and Kraus, 1998]. Many decentralized planning approaches can be employed (e.g. in [Ponda et al., 2010; Moehlman et al., 1992; Siciliano et al., 2010]).

MISSION PLANNING FOR MULTI-SATELLITE SYSTEMS

For efficient and effective space-based operations with a large number of satellites, onboard autonomous systems are required. By employing team coordination, many simple activities can be solved onboard. The operators in the mission control center then can in principle, focus on non-nominal situations only. As shown in Fig 1.6, for onboard team coordination, one vital stage is to perform planning to generate valid control sequences for all participants. The basic idea of mission planning is that a computer or an operator generates feasible control sequences based on the mission requirements. These sequences will guide all the sub-systems to perform specific behaviors within the defined time or order. This procedure is very complex due to massive characteristics associated with communication capabilities, constraints from mission scientific requirements, onboard storage capacities, and upload and download time windows.

For different coordination scenarios, the corresponding planning approaches can be different. These planning approaches are mainly divided into traditional approaches and artificial intelligence (AI) approaches according to the core algorithm. The traditional approaches use classical methods and algorithms to provide planning service. Considering their excellent performance and reliable solutions on fully observable, de-

terministic, and static operating environment, many early studies have shown their contributions on this area (e.g. [Kwek, 1997; Richards et al., 2002; Flint et al., 2002]). However, traditional approaches have limitations on solving complex and large-scale planning problems. AI approaches, on the other hand, can handle partially observable, nondeterministic, and unknown operating environment by employing heuristics methods. AI approaches also show good adaptability and efficiency on finding valid solutions for large-scale problems, loosely coupled and highly competition problems, and dynamic problems (e.g. [Nissim and Brafman, 2012; Chbichib et al., 2012; Zidi et al., 2011; Xhafa et al., 2012; Zhang et al., 2014; Gong et al., 2011]).

1.3. RESEARCH QUESTIONS

This thesis is triggered by a mission called “Discovering the Sky at the Longest Wavelengths" (DSL) [Boonstra et al., 2016]. It is comprised of nine satellites to perform low radio frequency observation and it is designed to operate in a Lunar Orbit. The long signal transmission distance and challenging operating environment makes an OAS necessary for this mission. Considering autonomous mission planning as vital for realizing full onboard autonomy of an MSS, in this thesis, our research will focus on developing onboard autonomous mission planning and re-planning systems. The developed concepts are intended to be applicable for many MSSs, while DSL plays the role of a sample reference scenario. To sufficiently characterize and guide the research of this thesis, three specific Research Questions (RQs) have been formulated in this section.

RQ1: What are the strengths of using AI in space missions? How to use a centralized AI algorithm in a multi-satellite system to decompose mission objectives and perform mission planning for the entire system?

To be able to coordinate an MSS based on a mission objective, the first step is to decompose the primary mission objective, assign sub-tasks to each participating satellite, and generate a sequence of onboard commands as the initial plan. Traditional planning and scheduling approaches could solve specific problems with shorter computation time and higher accuracy than most heuristic approaches. However, they may have problems for dealing with complex constraints like linear, non-linear, convex and non-convex constraints. Real planning problems for an MSS could have more than one objective. They also can be constrained by multiple types of constraints which traditional methods cannot fulfill. Therefore, this RQ proposes to use Artificial Intelligence (AI) techniques for an MSS to solve onboard mission planning problems. However, during mission operations, both the changes on initial mission objectives and the system failures caused by the challenging space environment or spacecraft internal failures can lead to the invalidity of the original mission plans. To identify these specific emergency situations, the following research question related to re-planning has been formulated.

RQ2: How to define emergency situations which may occur during mission operations? How to use AI algorithms to handle mission re-planning and rescheduling problems?

This research question aims to detect and identify possible situations which could occur due to the certain mission requirements. Based on these emergency situations, appropriate reaction strategies need to be formulated. Meanwhile, regular AI techniques used in RQ1 need to be modified and adapted based on re-planning problems. For a real mis-

sion, the central controller may have a certain probability of failure as well. In this case, the organizational architecture of the MSS needs to be transferred from a centralized architecture to a distributed one, where all the participating satellites need to work together to share the responsibility that the central controller originally used to hold. This leads to the third research question of this thesis.

RQ3: How to design cooperation and negotiation approaches for an MSS to reach an agreement? How to improve AI algorithms for distributed onboard mission planning problems?
Since the distributed architecture does not have a central controller, the global mission objective needs to be decomposed and assigned through the cooperation and negotiation among all participants. Developing efficient and effective negotiation mechanisms are needed. The first sub-question requires to solve the team's cooperation and negotiation problems. Once all the satellites reach an agreement on the mission objective allocation results, the second part of this RQ requires to solve the planning problems for an MSS with a distributed architecture. A distributed onboard mission planning approach needs to be developed based on previous proposed approaches. Finally, comparison with other state-of-the-art approaches shall be made to ensure that the proposed approach is successful and efficient.

1.4. METHODOLOGY

SCENARIOS

To investigate the previously formulated research questions, various scenarios need to be formulated to represent different characteristics of difficulties in each research question. These scenarios can provide specific verification environments to test the feasibility and applicability of the proposed methods and approaches. All these scenarios are driven by one reference scenario mission, the DSL mission. The primary scientific objective of this mission is to observe the universe in the hitherto unexplored, very low frequency (below 30 MHz) electromagnetic spectrum range. To ensure filling of an entire three-dimensional aperture and allowing all-sky observations, DSL contains one Mother Satellite (MS) and eight Daughter Satellites (DSs) [Boonstra et al., 2016]. Therefore, the basic MSS used in this thesis consists of one main satellite and eight participating satellites. The mission objective is to collect radio signals from outer space as much as possible within a specific lifetime. Targeting possible scenarios, the organizational architecture of the MSS could use either a centralized or a distributed architecture. For our research questions, the first two RQs will employ the centralized architecture scenario where the MS acts as the central controller, while the distributed architecture scenario will be used for RQ3. The details about these scenarios will be introduced in the next chapter.

NUMERICAL SIMULATION

As previously mentioned, several scenarios are considered in this thesis. To develop suitable approaches for these scenarios, numerical simulation is used to characterize the real-world problems. In this thesis, the simulators are developed in Matlab and Java. For centralized approaches related in RQ1 and RQ2, all the simulations are assumed to be

performed within the MS which is represented by a laptop with a 3.1 GHz Intel® Core™ i5 7267U dual-processor, 8 GB of RAM. For RQ3, to construct the distributed architecture, we use Java as the development language to build the multi-agent system in a Java IDE called Eclipse. To test the proposed approach, besides the high-end laptop used in the previous RQs, several low-end computers are employed to represent the DSs with lower computational power. Each of them has a 2.4 GHz Intel® Core™ i5 6300U dual-processor with 8 GB of RAM.

Both the centralized and the distributed approaches use heuristic algorithms for optimization. To compare their performance, multiple runs will be used to test each proposed approach. The statistical results will be presented as box plots, pie charts, bar charts, and other statistical graphics.

VERIFICATION AND EVALUATION

To verify the correctness of the proposed approaches in this thesis, several benchmarks are employed to test the performance of the proposed methods. For example, thirteen representative test functions [Hedar, 2018b; Floudas and Pardalos, 1990] are employed to test the performance of various candidate algorithms. Several Travelling Salesman Problems (TSPs) [University, 2018] are used to test the feasibility of both centralized [Johnson, 1990] and distributed optimization [Colorni et al., 1991] approaches. Job-shop scheduling problems [Wikipedia, 2018g] are used to test the correctness of the proposed team negotiation mechanisms [Agnetis et al., 2004].

To characterize the performance of the proposed methods, we also employ several state-of-the-art methods for each RQ as competitors. Many centralized optimization algorithms such as Genetic Algorithm (GA), Differential Evolutionary Algorithm (DE), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) are used to compare with the proposed centralized algorithms. Several task allocation approaches such as Auction-based approaches and Utility-based approaches are used to compare with the proposed negotiation mechanisms. Some representative distributed optimization approaches which employ optimizers like distributed ACO (DACO), and coevolutionary PSO (CPSO) are used to compare with the distributed optimization approach proposed in this thesis.

1.5. THESIS OUTLINE

This thesis consists of seven chapters. The overview of this thesis is described in this section. Chapter 1, the current chapter, introduces the history of space missions. It covers the state-of-the-art of relevant topics like onboard autonomous systems of MSS, associated key primary control models, and team coordination approaches. Based on the required fundamentals and the telescript of the state of the art, three RQs are introduced. Finally, methods to handle the RQs are introduced.

Chapter 2 introduces the background mission named DSL. Its scientific requirements help to build the reference MSS which is used in this thesis. A literature review of existing mission planning algorithms and techniques is presented, including related topics like task allocation approaches, optimization algorithms, negotiation mechanisms, and search heuristics. This chapter also presents preliminary tests of comparing the tradi-

tional methods and the heuristic methods on their performance of solving several optimization problems, which can provide a basic insight for further research.

Chapter 3 introduces the development of an improved genetic algorithm (GA) for solving initial planning problems using a centralized control model. The problem formulation includes the notations, variables, constraints and the fitness function, based on the mission requirements. The explanation of the proposed algorithm includes the motivation for improvements, the basic principles and the determination of essential variables. The verification uses both selected test functions and three other typical GAs to verify the performance of the proposed algorithm. Then, the simulation results from the DSL mission planning scenario illustrate the performance of the proposed algorithm.

Chapter 4 considers potential emergency situations which may occur to the DS of the DSL mission and proposes corresponding potential solution scenarios. A mathematical model of the re-planning problem is designed based on these scenarios, followed by the architecture of the proposed re-planning system. Two re-planning methods are proposed, with an explanation of the core algorithm, principles, re-planning strategies, and corresponding pseudo codes. The simulation results show the performance of the proposed re-planning methods for three proposed scenarios.

Chapter 5 focuses on developing coordination mechanisms for DSs. The organization architecture of the reference MSS is reconstructed by following an assumed malfunction of the MS. The MSS architecture is formulated by adapting ideas and methods from game theory, based on the similarity between this distributed mission allocation problem and typical perfect and imperfect information games. Considering the corresponding applicable scenarios and existing negotiation mechanisms, two types of negotiation mechanisms are proposed. The comparison with other state-of-the-art mechanisms illustrates the performance of the proposed negotiation mechanisms. Recommendations are made throughout the statistic results from the simulation.

In Chapter 6, a distributed mission planning approach is proposed. It starts with the problem statements, including their notation, variables, local and global constraints, and local and global fitness functions. Then, the detailed design procedure of this approach is introduced. The verification is realized by employing six instances from the Travelling Salesman Problem (TSP) database. The comparison is made between the proposed distributed approach with the previously proposed centralized approach. Finally, simulations show the performance of the proposed approach as compared with other distributed approaches.

Chapter 7 provides answers to each research question based on the results achieved on each chapter. The strengths and weaknesses of each proposed method are highlighted. Higher level conclusions from the answers to the RQs are drawn. Future research directions are proposed, along with preliminary thoughts on their methods, applications, new research fields and critical recommendations.

2

STATUS REVIEW AND SETTING

2

This chapter includes a brief introduction of the background mission which drives this research, the literature review of the state-of-the-art on mission planning approaches and preliminary tests. The background mission of this thesis, called “Discovering the Sky at the Longest Wavelengths” (DSL), is introduced first. Based on its scientific requirements, basic assumptions related to the research in this thesis can be made. The configurations and basic formulations of the multi-satellite system (MSS) used in this thesis are investigated. Based on the research questions in the previous chapter and the reference MSS, the literature review on related research topics is revealed. Tests on various algorithms which have been reviewed are presented to examine their performance. The preliminary results provide insight into suitable algorithms for this thesis.

2.1. BACKGROUND MISSION

As mentioned in Chapter 1, the background mission of this thesis is called the DSL mission [Boonstra et al., 2016]. The scientific objective of this mission is to observe the universe in the hitherto unexplored, very low frequency (below 30 MHz) electromagnetic spectrum range. In this frequency range, terrestrial observations are severely hampered by the strong ionospheric distortion and absorption, also known as man-made radio frequency interference (RFI). To overcome this problem, space-borne observations are needed.

Based on experiences from the RAE2-B explorer [Alexander et al., 1974], even at lunar distance, interfering signals from strong terrestrial radio transmitters can still be detected. However, the Moon can act as a natural shield to block any RFI from the Earth. Fig 2.1 illustrates that the conical zone is suitable for the DSL mission to perform the observations since the radio emission from the Earth will be blocked by the Moon in this area, called the “RFI-free zone" or “Blocked area”.

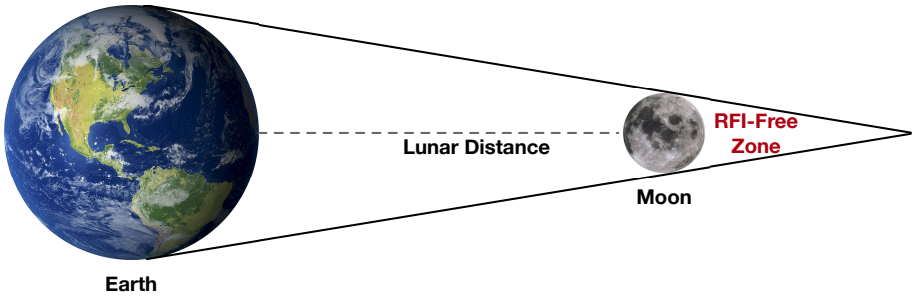


Figure 2.1: RFI-free zone

Single radio antennas have limited directivity. In order to map the celestial sky for tomography, the system requires large baselines between antennas to overcome the weak astronomical signals. Considering this requirement, the DSL mission chooses to use an MSS with a total of nine satellites, consisting of one Mother Satellite (MS) and eight Daughter Satellites (DSs).

The primary function of the MS is to carry all the DSs during the launch, and to release them after having reached the destination orbit. Once all the DSs have been de-

ployed, another function of the MS is to act as the central controller and information hub for gathering information from all DSs and providing control sequences. The total size of the MS (shown in Fig 2.2) is 1300 × 1300 × 950 mm, including yet to be deployed DSs, deployable solar panels, high gain antenna (for ground communication), inter-satellite antenna (to receive and send information to DSs), attitude control system, and electrical propulsion module.

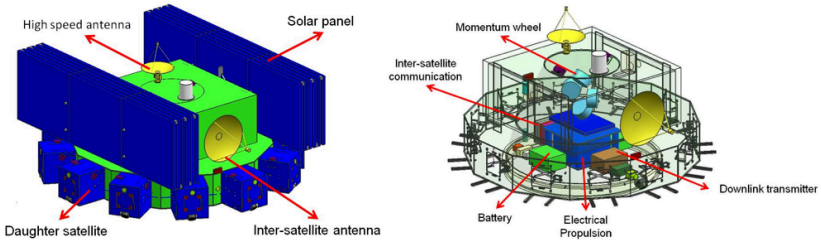


Figure 2.2: Mother satellite with examples of parts of onboard sub-systems, including stowed antennas, daughter satellites, solar panels, momentum wheel, battery, and propulsion system (these two figures are taken from [Boonstra et al., 2016]).

All eight DSs are identical on system functionalities and payloads, and act as data collection nodes. Fig 2.3 shows the preliminary design of a DS, which is a small satellite with the dimension of 300 × 300 × 300 mm and the mass of 12 kg. Its onboard memory unit has a capacity of 128 Gbits. The inter-satellite communication for both DS-to-MS and DS-to-DS is performed using a two-way radio frequency (RF) link, which enables a 6 Mbps data transmission rate.

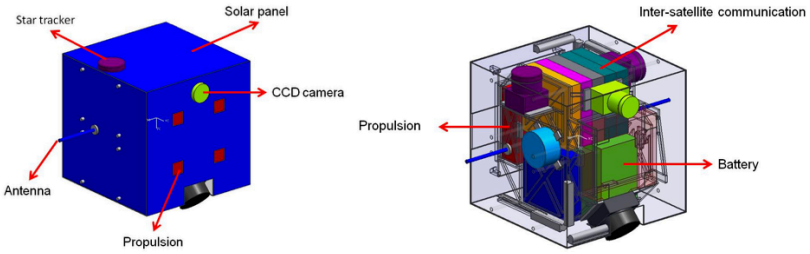
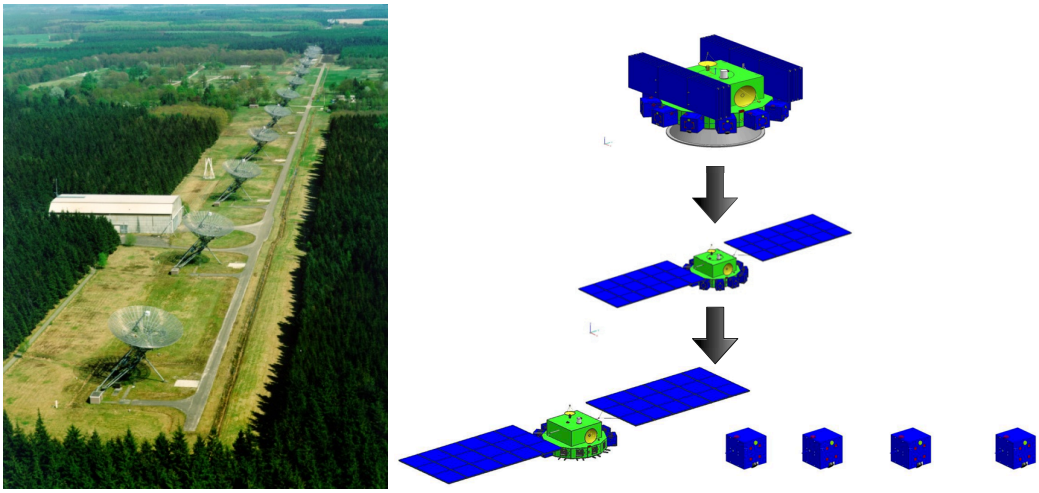


Figure 2.3: Daughter satellite with examples of parts of onboard sub-systems: star trackers, propulsion system, inter-satellite antenna, solar cells, and battery (these two figures are taken from [Boonstra et al., 2016]).

The DSL mission is inspired by the Westerbork Synthesis Radio Telescope which forms a distributed antenna system for higher frequencies than DSL [Hogbom and Brouw, 1974] (Fig 2.4a). This MSS will be configured in a quasi-linear array with adjustable baselines for different scientific observation strategies. All DSs are docked on the MS during launch. They will be in a similar orbit to the MS after deployment. This MSS is designed to be reconfigurable in a lunar orbit. It can change the baseline of adjacent DS from 100 m to 100 km to provide interferometric observations and perform polarimetry with radio signals received by the antennas of each DS. The deployment procedures are shown in Fig 2.4b. Based on this brief introduction of the DSL mission, some related scientific



(a) Westerbrok Synthesis Radio Telescope. Westerbork, the Netherlands (figure taken from [ASTRON, 2018])

(b) Daughter satellites deployment schematic (figure taken from [Boonstra et al., 2016])

Figure 2.4: Westerbrok telescope as the inspiration for the DSL configuration

requirements are provided in Tab 2.1.

2.2. LITERATURE REVIEW

The basic idea of mission planning is that an operator or the computer generates adequate control sequences based on the mission requirements. These sequences will allow the system to perform specific activities within the defined time window or sequence. This procedure is very complex due to the specific characteristics associated with communication capabilities, constraints from scientific mission requirements, onboard storage capacities, uplink and downlink time windows, orbital characteristics, and other requirements or constraints. Considering the reference MSS described in the previous section, to solve the mission planning problems onboard, there are two types of organizational architectures. One is the centralized architecture where the MS is responsible for controlling the entire MSS. Another type is a non-centralized architecture, which could be a distributed or a decentralized architecture based on the system topology. In a distributed architecture, no MS is available and each DS is connected with other DSs directly or through DS which acts as retranslator, as shown in Fig 2.5. The decentralized architecture is like the distributed architecture but with more restrictions on the communication environment, where each DS can only communicate with its neighbors. For non-centralized architectures, the participating satellites need to implement different distributed approaches to solve the mission planning problems.

As described in Chapter 1, when an MSS uses the centralized control model, it employs a central controller to oversee the entire team. Based on the information gathered from each participating agent, the controller uses centralized problem-solving techniques.

Table 2.1: DSL mission requirements and constraints [Boonstra et al., 2016]

MSS configuration	(1) 1 Mother Satellite; (2) Minimally 5 (baseline 8) Daughter Satellites (For calibration and redundancy. Also for higher reliability and longer lifetime;)
Orbit-related	(1) 350 km above Moon surface; (2) Absolute orbit knowledge: 30 m; (3) Relative orbit knowledge: 3 m;
Interferometry	(1) Formation flying with linear array; (2) Adjustable baseline between 100 m - 100 km;
Payload	(1) Three orthogonal dipoles for each DS; (2) Raw data downlink to MS with 6 Mbps through ISL; (3) Raw data observation rate through antenna can reach 48 Mbps; (4) Three channels receiver with a frequency of 0.1 MHz to 50 MHz;
Observation	(1) All observations need to be done within the observation window; (2) Onboard memory cannot be overwritten during observation;
Communication	(1) All satellites need to keep radio silence within the observation window; (2) MS can only communicate with one DS at the same time;

These problems are solved when the controller generates feasible solutions that can satisfy all the mission constraints for the entire team. All these problems can be considered as Constraint Satisfaction Problems (CSP) [Tsang, 2014]. For some other missions, the solution needs to be optimal on specific criteria (e.g. time-optimal, energy-optimal). These planning problems are considered as centralized optimization problems where the objective function is formulated based on mission criteria. Many search algorithms and optimization algorithms have been investigated and implemented in previous studies [Betts, 1998; Graham et al., 1979]. In this thesis, we first divide optimization and search approaches into two categories: classical approaches and heuristic approaches [Russell and Norvig, 2016]. Then, targeting the transmission architecture of the MSS, for a distributed architecture, task allocation, and distributed optimization approaches will be introduced.

2.2.1. CLASSICAL APPROACHES

BLIND SEARCH ALGORITHMS

As explained in [Russell and Norvig, 2016], blind search algorithms are a class of general purpose search algorithms that operate in a brute-force way (also known as generate and test method). These algorithms are also called uninformed algorithms because they do not need specific knowledge about the problem for searching. Many classical approaches use blind search algorithms as solvers for constraints satisfaction and optimization problems. Several well-known blind search algorithms are:

(i) *Breadth-first search (BFS)* [Moore, 1959] is a simple strategy for searching trees or graph data structures. This algorithm always expands the root node first, then comes to all the successors in the next lower level, and so on. Thus, BFS always has the shallowest path to every node on the frontier. If there is more than one solution, then the algorithm will find a solution with the minimal number of steps. However, the main drawback of this algorithm is its memory requirements. Also, if the solution is far off the beginning, then this algorithm is very time-consuming.



Figure 2.5: Three organizational architectures: The one in the middle is the centralized architecture, which has a single controller to control all the nodes; The one on the left is the distributed architecture, where all the nodes are equivalent and are connected; The one on the right is the decentralized architecture, where each node is only connected with its neighbors. (This figure is adopted from [Quantalysus, 2018])

(ii) *Uniform-cost search (UCS)* [Nilsson, 1971] follows the same idea as the BFS to expand the root node first in each level. Instead of expanding the shallowest node in the BFS, the UCS expands the node with the lowest path cost. This algorithm has the same weakness on memory requirement as the BFS.

(iii) *Depth-first search (DFS)* [Tarjan, 1972] is similar to the BFS except the priority of the search order. The DFS searches each branch to the last one. Then it can move to the second branch which has the same weight as the first branch. The memory requirement has a linear relationship with the scale of the search graph. However, the DFS has the possibility that it may go down to the most-left path forever. It cannot guarantee to find a solution. Meanwhile, if more than one solution exists, the DFS cannot guarantee to find the optimal one.

(iv) *Depth-limited search (DLS)* [Korf, 1990] is alleviated from DFS by adding a predetermined depth limit l, which represents that any node at depth l is treated as if they have no successors. This algorithm can solve infinite-path problems. However, if the chosen depth l is less than the solution depth (which means the shallowest goal is beyond the depth limit), using DLS may return incomplete solutions.

(v) *Dijkstra's algorithm* [Dijkstra, 1959] runs on a weighted graph, starting with an initial and goal node. This algorithm aims to find a shortest path between nodes in a graph, such as implementations on Google Maps. However, the computational cost can be expensive since it performs a blind search. Meanwhile, it cannot handle negative edges, which may lead to acyclic graphs and make the algorithm fail to obtain the proper shortest path.

(vi) *Bi-directional search (BS)* [Pohl, 1971], as the name implies, searches from two directions at the same time: one pushes forward from the beginning and the other

one pushes backward from the goal. This algorithm can guarantee optimal solutions with less memory requirement. Also, its search speed is much faster than other search algorithms since it searches from two directions. However, it is hard to implement BS because additional logic must be included to decide which search tree needs to be extended at each step. Meanwhile, the BS requires to know the goal state in advance, which could be not always possible to search backward through possible states.

Besides the previous algorithms, there are many other derivative algorithms belonging to blind search algorithms, such as *depth-first iterative-deepening search (DFIDS)* [Korf, 1985], *combinatorial explosion (CE)* [Edwards and Glass, 2000], and *backwards chaining (BC)* [Russell and Norvig, 2016].

Many planning approaches have been designed based on the principle of uninformed search algorithms. For example, the BFS has been used for motion planning [Lengyel et al., 1990] and path planning [Sabra et al., 2006]. The DFS is used together with a dynamic programming algorithm for nonlinear integer programming problems in [Ng and Sancho, 2001] and helps to explore unknown graphs in [Kwek, 1997]. In [Korf, 1990], the DLS is combined with the DFIDS for solving real-time problems in a dynamic environment. Due to the weaknesses of long computation time and large memory requirements, blind search algorithms are mostly used as a part of planning approaches to enhance the partial search ability.

LINEAR PROGRAMMING

Linear Programming (LP) [Beasley, 1996] was first used in computer modeling to find the best possible solution in a constrained environment to achieve best outcome (such as maximum profit or lowest cost). However, LP is applicable only when all the constraints are linear. It can only accommodate a limited class of objective functions. To implement LP for a linear optimization problem, four components need to be formulated:

(i) Decision variables
Any optimization problem requires defined decision variables. These variables are used to describe the different attributes of the problem.

(ii) Objective function
The objective function defines the purpose of the mission. Whether to find a solution to get the most or least, best or worst, can be described by the objective function. The objective function needs to be formulated as an algebraic expression to show the relationship between the output and the decision variables.

(iii) Constraints
The constraints in LP represent the restrictions or limitations on the decision variables. They are formulated based on the mission requirements. The constraints usually limit the search domain of the decision variables. For LP, the constraints can be formulated either as equality or as inequality.

(iv) Boundaries
Boundaries can help to narrow the search domain for decision variables. Bound-

aries are determined by the mission requirements. However, LP is still operable without boundaries, though it needs a longer computation time.

Due to the unique structure of the LP, it is one of the most widely used techniques of decision-making in business, industry and various other fields. Many problems for multi-agent systems (e.g. planning & scheduling, modeling, and control, decision-making) can be formulated as optimization problems with linear equality and inequality constraints. Therefore, many problems involving multi-agent systems are solved based on the theory of LP. For example, in [Schouwenaars et al., 2001][Richards and How, 2002][Yilmaz et al., 2008][Richards et al., 2002], mixed-integer LP is used to solve multi-agent planning problems. In [Earl and D'Andrea, 2002][Matossian, 1996], LP is used for modeling a multi-agent system. LP can also be used for controlling a multi-agent system [Shamma, 2008][Baker, 1998][Reinl and Von Stryk, 2007].

To conclude, linear programming is a compelling, efficient approach for solving linear problems. However, LP also has obvious limitations. Firstly, it is hard to formulate a real-world application as a mathematical problem. Secondly, the constraints of many problems cannot be directly specified by linear equations. This limits the scope of using LP. Thirdly, for large-scale problems, using LP can be computationally expensive due to its mathematical techniques. Because of the above limitations, it is difficult to find optimal solutions for various problems.

DYNAMIC PROGRAMMING

Dynamic Programming (DP) was first developed by Richard Bellman [Bellman, 1954]. It is a mathematical optimization method to solve a complex problem by breaking it down into a collection of simpler subproblems. Typically, all problems that require to maximize or minimize certain quantities or counting problems can be solved through DP. To implement DP for an optimization problem, four steps need to be followed:

(i) Characterize the structure of an optimization problem and break down the main problem into simpler subproblems. Then formulate the decision variables, objective functions and boundaries based on the subproblems. Express how to use the sub-problems to solve the entire problem.

(ii) Recursively define the value of an optimal solution by comparing it with previous solutions through the evaluation of the objective function.

(iii) Compute the value of an optimal solution in a bottom-up fashion.

(iv) Construct the optimal solution of the entire problem from computed information of the sub-problems.

There are many advantages of using DP. Firstly, the DP utilizes memorization techniques to store results of sub-problems such that the same sub-problems will not be solved again. Secondly, DP is well suited for a multi-stage or multi-point or sequential decision process. Thirdly, DP is applicable for both linear or non-linear problems, discrete or continuous variables, and deterministic problems.

Due to the various advantages of DP, many MAS planning problems chose to use DP as the solver. For example, in [Patek et al., 1999] the authors use a Neuro-Dynamic Programming approach for multi-platform path planning problems. In [Flint et al., 2002],

the cooperative path planning problem for a multi-vehicle system is modeled using a stochastic DP framework. In [Ganguly et al., 2013], a DP based planning algorithm is proposed for electrical distribution systems to optimize the feeder routes and branch conductors sizes. In [Yuejin, 2005], DP is used to solve the optimization problem for mission planning problems in multiple ground stations. Many other applications using DP have been reviewed in [Bellman and Dreyfus, 2015].

While producing the optimal solution, the computational time of the DP is proportional to the number of nodes in the graph. This depends on the griding (finer, coarser) of the solution space and increases geometrically with the dimension of the solution space. For the case of a large number of participants, DP may not be computationally feasible. Meanwhile, since DP relies on the performance of subproblems, for some complex planning problems, dividing a problem into subproblems and sorting intermediate results consumes system memory, which may not be suitable for mobile devices (e.g. CubeSat, UAVs).

2.2.2. HEURISTIC APPROACHES

Since all intelligent agents of an MAS are working in a common workspace, finding the optimal control sequences for such MAS composed of a large number of participants is a Non-deterministic Polynomial-time hard (NP-hard) problem. Therefore, many heuristic algorithms have been implemented for centralized approaches. Several representative categories will be illustrated here.

INFORMED SEARCH

Unlike the blind search mentioned in classical approaches, the informed search can find solutions more efficiently using problem-specific knowledge beyond the definition of the problem itself. Informed search tries to reduce the amount of search activities by making intelligent choices for the nodes that are selected for expansion. This implies the existence of some way of evaluating the likelihood that a given node is on the solution path. In general, this is done using a heuristic function. Two famous search algorithms belong to this catalogue:

(i) *Greedy Best-first search (GBFS)* [Russell and Norvig, 2016] is the general name for search and optimization algorithms which always make the greedy choice at each selecting point based on the heuristic function $g(n)$. It makes a locally-optimally choice, hoping it can lead to a globally optimal solution. Greedy searches do not require to be completed, and they do not necessarily detect the shortest path. The complexity of selection can be reduced substantially with a good heuristic function. The amount of the reduction is affected by the particular problem and the quality of the heuristic. The worst case performance of GBFS is no better than the BFS. Therefore, many other approaches combine the GS to enhance the local search ability. [Churchill and Buro, 2013] propose a novel GBFS called Portfolio greedy search for large-scale StarCraft combat scenarios. In [Barer et al., 2014], the authors compare GBFS with two other algorithms for multi-agent path-finding problems and propose several improvements. [Borrajo, 2013] implements lazy GBFS with actions costs to support the private planning part of the proposed approach.

(ii) A^* *search* [Hart et al., 1968] is an extension of *Dijkstra's algorithm*. This algorithm chooses the node on the search space for which its heuristic function is minimal. Instead of employing the heuristic function $g(n)$ which is used to estimate the cost to reach the goal from a current node in the GS, the heuristic function combines $h(n)$, the cost to reach the node and $g(n)$. For complex state spaces, A^* often becomes impractical to find an optimal solution. Researchers either use variants of A^* to find suboptimal solutions quickly or use it to design heuristics that are more accurate but not strictly admissible. A well designed A^* heuristic can provide enormous benefits compared with any blind search algorithms.
Many planning approaches have implemented the A^* algorithm. For example, in [Nissim and Brafman, 2012], a parallel version of the A^* algorithm is designed for multi-agent planning benchmark problems. [AlShawi et al., 2012] proposes a fuzzy approach with A^* to enhance the lifetime for wireless sensor networks. [Li and Sun, 2008] proposes an improved A^* algorithm to solve the route planning problem for multiple UAVs.

LOCAL SEARCH

The former search algorithms are all designed to explore the search space systematically. To achieve the systematicity, these algorithms usually keep one or more paths in memory, and record alternatives which have been explored along the path. This path is the solution to the problem when the goal is found. However, in many problems, the path to the goal is irrelevant. To avoid this, a different class of algorithm called *Local Search (LS)* can be considered. LS is a heuristic method for solving the hard optimization problems, which operate using a single node and move only to its neighbors. Although LS is not systematic, it has two key advantages: (1) since it uses only one node at each step, the memory requirement is minimal; (2) it can find reasonable solutions for problems with a vast or infinite search space, which systematic algorithms are unable to find. Due to its advantages, there are many derivatives for LS. Several representative algorithms are introduced here:

(i) *Hill Climbing (HC)* [Russell and Norvig, 2016] search algorithm is an iterative algorithm that starts at an arbitrary point and continually only moves in the direction of increasing value (i.e. climbing a hill). It terminates when it reaches the "best" solution where all the neighbors' value are not better than the current value. The HC is also called greed local search since it always chooses the best neighbor. This characteristic allows HC to improve a bad state and lead to rapid convergence. However, HC can also get stuck due to local maximum, ridges or plateaus. Variants of HC have been invented due to these weaknesses, such as *Stochastic HC* [Juels and Wattenberg, 1996], *First-choice HC* [Karypis and Kumar, 2000], and *Random-restart HC* [Hoos, 1999].
For mission planning problems, many researchers propose to use HC as a solver. For example, [Gerkey et al., 2005] proposed to use a parallel stochastic HC for a small size multi-robot system to solve the actions coordination problems. [Rocha et al., 2008] proposed an autonomous map building approach that uses occupancy grid maps as representation model and employed HC for exploring the unknown environment. This approach can be implemented for either a single robot system

or multi-robot systems. In [Chbichib et al., 2012], the authors combined HC with a descent algorithm to solve large-size multi-vehicle routing problems.

(ii) *Tabu Search (TS)* [Glover, 1989, 1990] was created by Fred W. Glover. It uses a neighborhood search to iteratively move from one potential solution to a more promising solution in the reachable search area (neighborhood). To avoid the pitfalls existing in local search procedures, TS uses three types of memory structures, which are short-term, intermediate-term, and long-term for guiding purposes. These flexible memory structures are also known as Tabu lists, which are sets of rules to filter which solutions will be admitted to a certain neighborhood that is to be explored next. By employing the adaptive memory feature, TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. In addition, TS can also combine with other meta-heuristics to create hybrid methods.
Since TS can overcome the local optima problems which could happen for ordinary LS, many studies have been done using various TS derivative algorithms for mission planning problems. For example, for multi-vehicle static Dial-a-Ride problem (DARP), [Cordeau and Laporte, 2003] design a TS heuristic to minimize route duration and ride times. This method has also been implemented in another article [Attanasio et al., 2004] for dynamic DARP. [Alonso et al., 2008] propose an algorithm based on TS for site-dependent multi-trip periodic vehicle routing problems. In [Gendreau et al., 1999], the authors proposed an elaborate TS which embedded an Adaptive Memory Procedure (AMP) for solving the heterogeneous fleet vehicle routing problem. The same idea was also used for solving the Team Orienteering Problem (TOP) in [Tang and Miller-Hooks, 2005].

(iii) *Simulated Annealing (SA)* was created by N. Metropolis in 1953 [Metropolis et al., 1953]. As described before, the HC always goes "uphill" and never will choose for lower value at the current node. This greedy characteristic makes HC incomplete, since it can get stuck with the local maximum. Simulated Annealing (SA) combines the HC with a random selection so that it can be both efficient and complete. The name of SA stems from metallurgy, where annealing is the process to heat up the materials like metals and glass to a high temperature and then slowly cool them down, eventually making them reach a low-energy crystalline state. Following the same idea in the real world annealing process, SA uses a temperature variable T to adjust the moving probability. The temperature T starts with a high value and then slowly decreases to simulate the cooling process. While this temperature variable is high, the algorithm has a higher probability to accept solutions that are worse than its current solution. This gives the algorithm the ability to escape from any local optimal. When the temperature reduces, the probability of accepting worse solutions decreases, which allows the algorithm to gradually focus on an area of the search space where the approximately optimum solutions can be found.
The adjustment on moving probability makes SA remarkably effective at finding an approximately optimum solution when dealing with large-scale problems which contain numerous local optima. Also, this algorithm is suitable for parallel processing and can handle very complex nonlinear optimization problems. However,

its performance strongly depends on the initial parameters, which are not easy to formulate for some complex problems. In [Thangiah et al., 1994][Chiang and Russell, 1996][Baños et al., 2013], SA is used to solve the vehicle routing problem with time windows. [Aydin and Fogarty, 2004] proposes to use parallel SA for multi-agent job-shop scheduling problems. In [Zidi et al., 2011], the authors propose a multi-objective SA for solving the multi-agent DARP (Dial A Ride Problem). [Keinänen, 2009] indicates that SA can also be used for multi-agent coalition structure generation problems.

EVOLUTIONARY ALGORITHMS

Inspired by natural selection in Darwin's theory of biological evolution, Evolutionary Algorithms (EAs) use mechanisms in nature, such as reproduction, selection, mutation, recombination, and termination. There are many different variants of EAs. The common underlying idea of all these algorithms is the same. Given a population of individuals, the environment will perform natural selection based on the performance of these individuals, and eventually, the best individual will survive. These characteristics make EAs the generic population-based meta-heuristic optimization algorithms. EAs are commonly used to solve problems that cannot be easily solved in polynomial time, such as NP-hard problems. They also perform well-approximating solutions for all kinds of optimization problems. Two popular types of EA will be introduced in this thesis, along with the application of these algorithms.

(i) *Genetic Algorithm (GA)* [Whitley, 1994] is a search heuristic that mimics the process of natural selection, which is the most widely used AI algorithm. It can be used to find optima in many fields, such as path planning [Ponda et al., 2015; Li et al., 2000], resource allocation [Kanury and Song, 2006], collision avoidance [Duan et al., 2013], mission scheduling [Xhafa et al., 2012], and so forth. One GA requires several steps: initialization, selection, genetic operators, and termination. Using GAs has several advantages. Firstly, the GA is easy to implement for any kind of optimization problem. Secondly, it can find quality solutions in a short time. Thirdly, the genetic operators can help GAs escape from local maxima. However, GAs also have some limitations. Firstly, it is hard to formulate the mathematical model which can reflect the problem to be solved. Secondly, for a complex problem, GA does not scale well with complexity. When the number of elements which are exposed to mutation is large, there is often an exponential increase in the size of search space. Furthermore, GA cannot solve problems which their fitness functions are only represent as right or wrong. This means that GA cannot solve decision-making problems.
Considering the advantages of GAs, many researchers have implemented various GAs for different multi-agent planning problems. For example, [Liu et al., 2004] uses based-knowledge genetic operators to enhance the performance of GA for multi-mobile-robotic system path planning problems. In [Pehlivanoglu, 2012], a new GA called multi-frequency vibrational GA is proposed to solve the path planning problems for multiple UAVs. [Chilan and Conway, 2007] uses a binary GA for the outer loop planning and a real-valued GA for the inner loop. In [Gad and Abdelkhalik, 2011; Abdelkhalik and Gad, 2012], the authors develop a hidden genes method and a dynamic population size method to modify GA for finding flyby se-

quences and optimal trajectories. In [Xhafa et al., 2012], the authors present a suitable formulation of satellite scheduling and various forms of optimization objectives, mainly using GA for satellite ground station scheduling. [Tangpattanakul et al., 2015a] and [Tangpattanakul et al., 2015b] presented a biased random-key genetic algorithm, which has been used for multi-user, Earth observation scheduling problems.

(ii) *Differential Evolutionary (DE)* [Storn and Price, 1997] is another meta-heuristics under the EA family. Similar with GA, DE is also a population-based, combinatorial optimization algorithm. Although DE has the same theory foundation as GA, the correspondences are not as exact as GA. The significant difference is that DE is using actual real numbers. As a result, the ideas of mutation and crossover are substantially different. However, a benefit is that there are handful of results showing that DEs are often more effective and/or more efficient than GAs. Moreover, when working on numerical optimization, DE is able to represent problems as actual real numbers instead of squeezing them into a chromosomal representation.
In [Chakraborty et al., 2009], the authors propose an alternative approach to solve both centralized and distributed realizations for multi-robot path planning problems by employing a parallel DE. [Nikolos and Brintaki, 2005] provides an off-line path planner for multiple UAVs coordinated navigation and collision avoidance in known static maritime environments by using DE techniques. In [Rakshit et al., 2013], the authors combine DE with Q-learning to realize an adaptive memetic algorithm for a real-time multi-robot path planning problem. [Mingyong and Erbao, 2010] construct a general mixed integer programming model which uses an improved DE to solve the vehicle routing problems with simultaneous pickups, deliveries, and time windows.

SWARM INTELLIGENCE

Mostly inspired by biological systems, a swarm is a large number of homogeneous agents interacting locally among themselves. In a swarm, there is no central controller to oversee the entire team. *Swarm Intelligence (SI)* is a relatively new branch of AI inspired by many social swarms in nature, such as ant colonies, fireflies, and flocks of birds. Many research papers are reporting the successful application of SI algorithms in various optimization problems. The basic principles of SI make them suitable for a variety of problem domains including function optimization problems, scheduling and planning, structural optimization, and data and image analysis. In this thesis, two of the most popular SI algorithms will be introduced.

(i) *Ant Colony Optimization (ACO)* [Colorni et al., 1991] is a cooperative population-based optimization algorithm which draws inspiration from the social behavior of ant colonies. As each ant constructs a route from the nest to food by stochastically following the quantities of pheromone level, the intensity of laying pheromone would bias the path-choosing, decision-making of subsequent ants. The basic principle of ACO is to model the optimization problem to be solved as a search for an optimal path in a construction graph and to use artificial ants to search for feasible solutions. Artificial ants simulate the behavior of real ants in several aspects: (1)

The ants deposit pheromone trails on each search node to indicate the quality of the current node; (2) The solutions are constructed through the choices the ants made through the construction graph. These choices depend on the pheromone trails; (3) The decreasing trend of these pheromone trails relies on the quality of the current choice of heading direction. Several advantages of the ACO are proposed through previous studies: (1) ACO has an inherent parallelism, which can be directly implemented for parallel computing; (2) Compared with other heuristics, in terms of solving performance, it shows a strong robustness and the ability to search for better solutions; (3) ACO can be used in dynamic environments, coping with changes on objectives or settings.
Considering its advantages, ACO has been used to solve various optimization problems, such as planning & scheduling, sequential ordering, assembly line balancing, TSP, etc. For MASs, ACO also provides powerful capabilities for many planning scenarios. For example, [Leung et al., 2010] identify that using ACO incorporated with MAS can effectively solve the integrated process planning and shop-floor scheduling problems. In [Garcia et al., 2009], a new method based on ACO and a simple tuning algorithm is proposed for mobile robot path planning problems. For multi-agent team collaboration problems, ACO can also provide a good performance, such as in [Yingying et al., 2003] and [LIU et al., 2010]. [Zhang et al., 2014][Gao et al., 2013][Wang et al., 2009] illustrate that ACO techniques could also be implemented for multi-satellite system's optimization, planning, and scheduling.

(ii) *Particle Swarm Optimization (PSO)* [Kennedy, 2011] is another population-based optimization algorithm which simulates the swarm behaviors of bird flocking. PSO is based on many individuals. Each individual is called a particle, which represents a potential solution. Each particle has a fitness value which indicates the current cost to get to the target, and a velocity value which indicates the flight direction and is calculated according to the distance between a particle's current position and its target position. The position of a particle is influenced by both the best position visited by it, and the best position visited by its neighbors. When one particle's neighbors are the entire swarm, the neighbor's best position can be referred to as the global optimal. All the particles iteratively evaluate the fitness of the particle solutions and remember the location of their best fitness value. Once the termination conditions have been reached, the algorithm stops. Several advantages of using PSO have been revealed by researchers: (1) PSO can be widely implemented for both engineering and scientific research; (2) Only the most optimal particle can transmit information to other particles. With the development of generations, the computation speed can be much faster; (3) PSO uses real number for coding, which is directly influenced by the solution. The variables' dimension is equal to the constant of the solution. Same as ACO, PSO is also implemented in many MAS applications due to its advantages. For example, in [Saska et al., 2006], PSO is used for robot path planning problems because of its relatively fast convergence and global search character. [Pugh and Martinoli, 2007] presents a multi-search algorithm inspired by PSO to mimic the multi-robot search process. [Gong et al., 2011] implements a multi-objective PSO for robot path planning problems, where the operating environment is dynamic due to unexpected sources of danger. In [Duan et al.,

2013], the authors proposed to combine PSO with GA to solve multi-UAV formation reconfiguration problems.

2.2.3. TECHNIQUES RELATED TO DISTRIBUTED SYSTEMS

TEAM NEGOTIATION & COOPERATION

Many of the centralized approaches for MASs have focused on the analysis of the mission allocation problem under a stable operating environment. However, in this thesis, the mission platform is an MSS, which will operate at a lunar orbit. Unpredictable situations may occur to every member of the system, such as system failures caused by meteorite, or a failure of onboard electronic components caused by cosmic radiation, etc. Centralized approaches for the MSS are unreliable under these scenarios. Therefore, the MSS needs to solve task allocation problems through team negotiation and cooperation mechanisms. Kose [Kose et al., 2004] uses the reinforcement learning approach for solving self-interested robot collaboration problems. However, due to limited communication, the system could converge to a sub-optimal solution. Zheng and Koenig in [Zheng and Koenig, 2009] propose a K-swaps mechanism to help the distributed algorithms exchange assigned tasks among agents to reduce the team cost. However, it is difficult to find a suitable value for K which can provide excellent performance both on team cost and on the computation time. Lang and Fink [Lang and Fink, 2012] present a quota-based negotiation mechanism for minimizing the total cost for a multi-machine scheduling problem. Later in [Lang et al., 2016], they re-design this mechanism by implementing a simulated annealing algorithm.

Raffard, Tomlin, and Boyd [Raffard et al., 2004] introduce an optimal distributed approach that uses a dual decomposition technique for a group of cooperative agents. However, the negotiation procedure can only start when the entire team is willing to cooperate. This assumption is not suitable for our case since satellites cannot establish the communication link with others on a continuous basis. In [Jin and Li, 2016], the authors propose a k-winner-take-all (k-WTA) negotiation mechanism for a multiple robots system in a limited communication environment. However, in a real-world application, the single point of failure on the winner could cause the algorithm to pause and go backward to perform k-WTA again. [Choi et al., 2009] propose two decentralized auction-based approaches, namely, the consensus-based auction algorithm and consensus-based bundle algorithm, for a fleet of autonomous mobile robots. Gao in [Gao et al., 2009] proposes an evolutionary computation decentralized approach using the genetic algorithm for solving MRTA (Multi-robot Task Allocation) problems. However, it is time-consuming for all agents to agree to regret the current assignment if a task has been bidden. The main advantage of non-centralized approaches is the strong robustness that can tolerant low-level system failures. The limitation of these approaches lies in the message communication between agents. If communication links are not sufficiently reliable, the outcome may degrade.

DISTRIBUTED OPTIMIZATION APPROACHES

Most space applications apply centralized approaches due to their various advantages, such as easy implementation, accurate solutions, less communication required than with distributed approaches. However, centralized approaches cannot support some

distributed systems when no spacecraft can take over the responsibility of being the central controller because of hardware or software constraints. Therefore, scientists have developed interests to explore distributed mission planning approaches.

Izzo [Izzo and Pettazzi, 2007] develops a behavior-based path planning algorithm using an equilibrium shaping technique for a set of identical spacecraft to reach a given configuration. This approach can autonomously assign the final configuration to all the satellites with a minimum amount of inter-satellite communication. Giovanini [Giovanini et al., 2007] formalizes a multiple UUVs (Unmanned Underwater Vehicles) autonomous mission planning problem as a receding horizon mixed-integer constrained quadratic optimal control problem. They propose a distributed Nash-based game approach aimed to divide the planning problem into a set of subproblems which can be distributed among the mission agents. Torreño [Torreño et al., 2012] proposes a cooperative refinement planning approach which builds upon the multi-agent partial-order planning paradigm. The comparison with a distributed Constraint Satisfaction Problem (CSP) based MAS planning approach under different levels of coupling between the agents is made to show the general capability of this system. Gao [Gao et al., 2016; Zhen et al., 2018] proposes to use the improved distributed Ant Colony Optimization (ACO) algorithm following the bottom-top structure for solving search-attack mission self-organization problems. The authors have implemented this approach in both two-dimensional and three-dimensional space. Evers [Evers et al., 2014] introduces the Maximum Coverage Stochastic Orienteering Problem with Time Windows (MCS-OPTW) and develops a fast heuristic algorithm which could cope with the uncertainty of the mission operation and could re-plan with the appearance of time-sensitive targets. Wu [Wu et al., 2018] adopts a newly developed distributed genetic algorithm (DGA) for multiple UAVs to cooperate in multiple task assignment problems. The proposed DGA is used in each UAV to generate feasible task assignments, and Dubin's paths are used for trajectory generation.

Besides the previously mentioned methods, there are many other distributed approaches for different kinds of MAS [Han et al., 2014; Das et al., 2015; Bhandari et al., 2014]. However, two key points still need further investigations when implementing distributed planning approaches (DPAs) for an MSS.

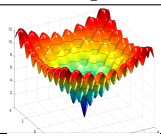
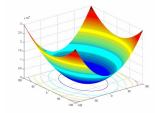
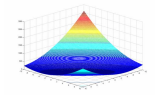
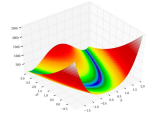
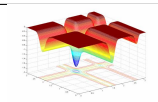
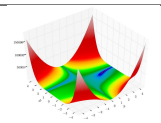
1. Most DPAs, during each iteration, usually choose to peel away the local regeneration procedure from the group information exchange procedure. This strategy can speed up the computation time for specific scenarios. However, if constraints on different agents are highly-coupled, employing those DPAs can be computationally expensive.

2. Many studies have failed to cope with unexpected changes on operating agents [Jin and Li, 2016]. The proposed DPAs are either especially suited for stable operational environments where all agents stay safe during the mission, or can only perform mission re-planning once the previous planning procedures are completed. Few algorithms can handle unexpected changes while the current planning procedures are running onboard.

2.3. PRELIMINARY CONCEPT SELECTION

In the previous section, many algorithms have been introduced, which are able to solve various optimization problems. Considering the complexity of our reference mission DSL, the mission planning problems could be solved by either centralized or distributed approaches. For different scenarios, the number of objective functions can be one or more, while the type of constraints can be consistent or mixed. To efficiently solve various planning problems, the MSS needs a robustness solver which can handle different types of optimization problems. Hence, in this section, a preliminary selection of algorithms has been made to draw basic conclusions for our research. These candidate algorithms are selected based on their categories and characteristics, basically covering both classical approaches and heuristic approaches. These algorithms are *Breadth-first*

Table 2.2: Six unconstrained test functions [Surjanovic and Bingham, 2018a]: (1)The Ackley function, with a two-dimensional form, characterized by a nearly flat outer regions and a large "hole" at the center; (2)Bohachevsky functions all have the same similar bowl shape. The shown shape is the first Bohachevsky function; (3)The Booth function shows a plate shape; (4)Rosenbrock functions are also two-dimensional forms, with a valley shape; (5)Michalewicz functions have a two-dimensional form with steep ridge shape; (6)Beale functions have sharp peaks at the corners of the input domain.

Function Name	Variables Number	Global minimum	Function Shape
Ackley Function	n	$f(x^*) = 0,$ at $x^* = (0, 0, ..., 0)$	
Bohachevsky Function	2	$f(x^*) = 0,$ at $x^* = (0, 0)$	
Booth Function	2	$f(x^*) = 0,$ at $x^* = (1, 3)$	
Rosenbrock Function	n	$f(x^*) = 0,$ at $x^* = (1, 1, ..., 1)$	
Michalewicz Function	2	$f(x^*) = -1.8013,$ at $x^* = (2.20, 1.57)$	
Beale Function	2	$f(x^*) = 0,$ at $x^* = (3, 0.5)$	

Search (BFS), Linear Programming (LP), Dynamic Programming (DP), A Search (A*), Simulated Annealing (SA), Genetic Algorithm (GA), Differential Evolution Algorithm (DE), Ant Colony Optimization (ACO)*, and *Particle Swarm Optimization (PSO)*.

Comparing the performance of different optimization algorithms is usually done by using common standard benchmark test functions, which are also known as artificial landscapes. Since planning problems are considered as CSP and optimization problems, the test functions used here need to cover all kinds of problem types ranging from unconstrained optimization problems to constrained problems. For unconstrained optimization problems, we employ six representative test functions, as shown in Tab 2.2. These six test functions cover six shapes of unconstrained optimization problems as explained in the previous table. The Ackley function has many local minima which can trap algorithms from finding the global minimum. The Bohachevsky functions have three forms, all of them continuously convex and showing bowl shapes. The Booth function is a continuous convex test function which shows a plate shape. The Rosenbrock function is defined in an n-dimensional space, which is also a continuous convex function. The Michalewicz function is also defined on an n-dimensional space, which has a total of $n!$ local minima. It has three global minima corresponding to three variable dimensions, for $n = 2$, $f(x^*) = -1.8013$, for $n = 5$, $f(x^*) = -4.687658$, and for $n = 10$, $f(x^*) = -9.66015$. The Beale function is a continuous non-convex function, which is also multimodal. The idea of using these test functions is to test whether the candidate algorithms can handle a simple unconstrained optimization problem. For the sake of simplicity, these six unconstrained test functions are labeled as TF1 to TF6, respectively.

Besides the unconstrained test functions, to further test the candidate algorithms' performance on different constrained optimization problems, some constrained test functions are employed here, shown in Tab 2.3.

The constrained test functions are named directly by following the unconstrained test function numbers, starting from TF7. To ensure a variety of test functions, these eight functions contain three types of objective functions which are quadratic, nonlinear, and linear. For different functions, the number of variables and constraints are also different. The constraints column in Tab 2.3 contains four kinds of constraints: linear inequality constraints (LI), nonlinear inequality constraints (NI), linear equality

Table 2.3: Eight constrained test functions gathered from [Hedar, 2018a]

Function Number	Variables Number	Function Type	Constraints				Global minimum
			LI	NI	LE	NE	
TF7	13	Quadratic	9	0	0	0	$f(x^*) = -15$
TF8	20	Nonlinear	2	0	0	0	$f(x^*) = 0.803619$
TF9	10	Nonlinear	0	0	0	1	$f(x^*) = 1$
TF10	5	Quadratic	4	2	0	0	$f(x^*) = -30665.539$
TF11	4	Nonlinear	2	0	0	3	$f(x^*) = 5126.4981$
TF12	2	Nonlinear	0	2	0	0	$f(x^*) = -6961.81388$
TF13	10	Quadratic	3	5	0	0	$f(x^*) = 24.3062091$
TF14	8	Linear	6	0	0	0	$f(x^*) = 7049.3307$

constraints (LE), and nonlinear equality constraints (NE). The specific number of constraints for each function is also revealed here, along with the global minimum of the corresponding functions.

Table 2.4: Test results on 14 test functions from nine candidate algorithms

Test function number	Algorithm candidates																	
	BFS		LP		DP		A*		SA		GA		DE		ACO		PSO	
TF1	Yes	S	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	M	Yes	M	Yes	Q	Yes	Q
TF2	Yes	S	Yes	M	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	S	Yes	S
TF3	Yes	M	Yes	S	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	S	Yes	S
TF4	Yes	M	Yes	M	Yes	M	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	S	Yes	S
TF5	Yes	S	Yes	Q	Yes	M	Yes	Q	Yes	Q	Yes	M	Yes	M	Yes	Q	Yes	Q
TF6	Yes	M	Yes	Q	Yes	Q	Yes	Q	Yes	Q	Yes	M	Yes	M	Yes	S	Yes	S
TF7	Yes	VS	No		Yes	M	Yes	VS	Yes	VS	Yes	S	Yes	M	Yes	S	Yes	S
TF8	No		No		Yes	M	Yes	VS	Yes	VS	Yes	S	Yes	S	Yes	VS	Yes	VS
TF9	Yes	VS	No		No		Yes	VS	Yes	VS	Yes	M	Yes	S	Yes	S	Yes	S
TF10	Yes	VS	No		No		Yes	VS	Yes	VS	Yes	M	Yes	S	Yes	M	Yes	M
TF11	No		No		No		Yes	VS	Yes	VS	Yes	M	Yes	S	Yes	M	Yes	M
TF12	Yes	VS	No		No		Yes	VS	Yes	VS	Yes	M	Yes	M	Yes	M	Yes	M
TF13	Yes	VS	No		No		Yes	VS	Yes	VS	Yes	S	Yes	S	Yes	VS	Yes	VS
TF14	Yes	VS	Yes	M	Yes	M	Yes	Q	Yes	VS	Yes	M	Yes	M	Yes	S	Yes	S

Tab 2.4 illustrates the test results of nine candidate algorithms on 14 test functions. For each test function, we have implemented all candidate algorithms, and re-run each algorithm 10 times to gather statistical results. TF1 to TF6 are all unconstrained test functions, while TF7 to TF14 are constrained functions. For each algorithm, the column on the left represents the feasibility of using this algorithm to get the correct solution for the corresponding test function. Considering that all the test functions have the global minima, to be more direct, a binary evaluation system is designed. "Yes" stands for the positive conclusion that this algorithm can find the global minimum or a near-global minimum where the difference to the global minimum is less than 5% for this function. "No" stands for the failure when the algorithm is unable to find the feasible solution, or the difference between the solution and the global minimum is larger than 5%. The right column stands for the convergence effectiveness of reaching the global minimum. "Q" stands for quick, "M" means medium, "S" represents slow, and "VS" is very slow. All these symbols represent the statistical convergence time of solving the corresponding test functions. This evaluation system is employed to simplify the comparison among all the candidate algorithms. For each test function, the convergence time of all the capable algorithms are represented by one symbol, which is concluded based on the computation time ranking list.

Based on the test results, the BFS can solve most of the test functions, except for those who do not have a globally optimal solution. This forces BFS unable to terminate. Meanwhile, the shortsighted feature makes BFS slow on finding the optimal solutions. Due to the unique requirements on LP, it can only solve the functions with linear objective function and constraints. Therefore, most of the constrained test functions are unsolvable by LP because of the nonlinear constraints. DP shows similar results as LP, except its convergence time is faster as compared to the LP. Although classical approaches can solve some of the test functions in a short time, for complex constrained functions, the efficiency of these algorithms decreases significantly.

Heuristic approaches can solve all the test functions. The A^* search algorithm can

correctly deal with the unconstrained and linear constrained problems. However, when facing nonlinearly constrained problems, A^* algorithm needs more computation nodes, which makes it less efficient. SA can solve the unconstrained problems in a short time, but for the constrained problems it is much slower. This is because SA needs to set a specific time for the algorithm to search the whole search space. It means the algorithm needs to have a constant long search period to find an optimal solution. Therefore, it is difficult to determine this constant time. If it is set to be too long, it can waste time, but if it set to be too short, the algorithm may not find the best solution. This defect severely affects the adaptability of this algorithm.

GA and DE both belong to the same algorithm family (Evolutionary Algorithm). There are several differences between GA and DE. GA has the advantage of being able to deal with constraints. It can directly use them to limit the search space. DE, on the other hand, needs to transfer constraints into penalty functions, and then add them to the objective function. This could make it difficult to choose the proper values for weight numbers. The crossover operator of a DE algorithm will randomly choose two candidates to perform crossover, which is faster than the crossover process in GA. Although these differences will cause differences on computational efficiency and accuracy, their processing procedures still follow the same fundamentals. These two algorithms do not use the gradient to optimize a problem. This means they do not require to be differentiable as it is required by classical optimization methods, such as gradient descent and the Quasi-Newton method. Therefore, GA and DE can be used on optimization problems that are not continuous, noisy, and that can change over time. These two algorithms have great characteristics for dealing with complex space missions. ACO and PSO are swarm intelligence methods. One simulates ants, and another one simulates birds. These two algorithms do not have crossover and mutation operators, which means that each particle or ant in these algorithms has to decide its behaviors based on the reflection from the environment and other particles or ants. These two algorithms have already been widely used in discrete optimization problems as shown in the previous literature review. In conclusion, GA, DE, ACO, and PSO all have great potential to become the main algorithm for the research in this thesis. Considering the overall relative convergence time for these test functions, optimization algorithms in the EA family (GA and DE) become our preferred core algorithms as problem solvers.

2.4. SUMMARY

This chapter introduces the background mission of this thesis, the “Discovering the Sky at Longest wavelength” (DSL) mission. The DSL will have multiple satellites in a circular lunar orbit to detect low frequency radio signals below 30 MHz. Based on the science requirements of the DSL mission, the reference MSS for this thesis is constructed. Considering that the fundamental objective of this thesis is to provide onboard mission planning & scheduling capability for the reference MSS, an in-depth literature review has been made. In this review, representative optimization algorithms are classified into two categories: classical approaches and heuristic approaches. Some innovative and hybrid approaches related to distributed system coordination and negotiation are also revealed. Finally, targeting the high complexity of foreseeable scenarios, tests on nine algorithms are made to provide a preliminary selection for the reference MSS. The chosen test func-

tions cover both six unconstrained and eight constrained optimization problems. These fourteen test functions contain different types of objective functions and constraints to guarantee the diversity of the preliminary selection. In the next chapter, we will explore how to solve the MSS onboard mission planning problems using various approaches.

3

CENTRALIZED ONBOARD INITIAL PLANNING

Parts of this chapter have been published in Z. Zheng, J. Guo, E. Gill, *Swarm Satellite mission scheduling & planning using Hybrid Dynamic Mutation Genetic Algorithm*, Acta Astronautica **137**, 243-253 (2017) and Z. Zheng, J. Guo, E. Gill, *Multi-Satellite Onboard Behaviour Planning Using Adaptive Genetic Algorithm*, 67th International Astronautical Congress, IAC-16-B4.3.9, (2016)

3.1. INTRODUCTION

In Chapter 1, the motivation of this thesis has been addressed, which is to develop concepts for an onboard autonomous system that can be used by a Multi-Satellite System (MSS) for mission planning and scheduling. In the previous chapter, through the background mission DSL, the reference MSS has been constructed. The nominal organizational architecture of this reference MSS is a centralized structure with one main controller called the mother satellite (MS), and eight participants called daughter satellites (DSs). Fig 3.1 illustrates the general architecture for a centralized control of this MSS, where the MS contains the general planner, decision-maker, and re-planner, represented by gray boxes. Each DS is equipped with a plan executor and a status monitor, represented by blue boxes. After the MS has received the mission objective from the con-

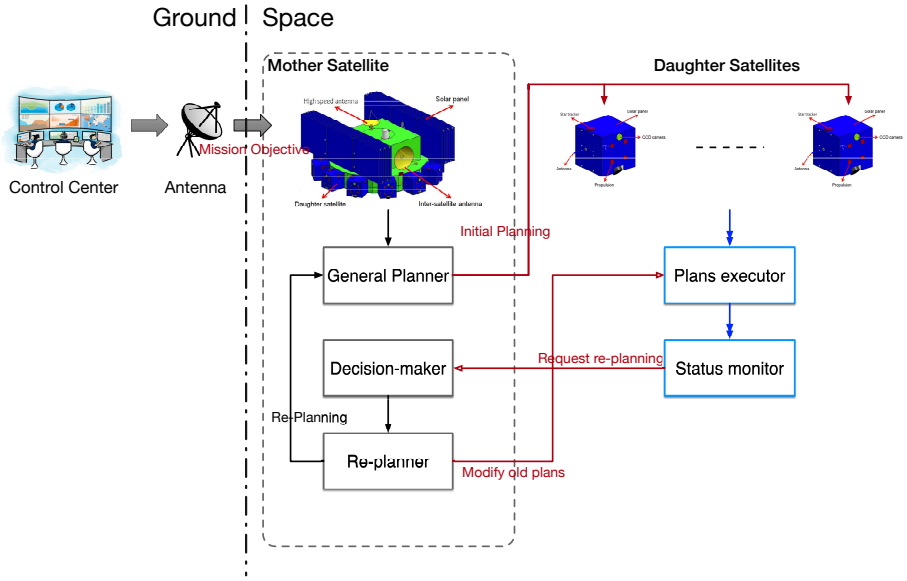


Figure 3.1: General architecture of controlling the centralized MSS

trol center, it will start the initial planning procedure using the general planner. It transmits generated control sequences to each DS as initial plans. Once a DS has received the initial plans, it will execute them off-line and constantly provide feedback to the MS for regular checks using the status monitor. During the plan execution procedure, each DS operates individually. The quality of these initial plans will directly affect the final performance of the entire team. Therefore, in this chapter, the centralized planning approach for providing the initial planning service will be developed in detail.

The scientific objective of the DSL mission is to detect low frequency RF signals from deep space using multiple satellites. Coordinating all the participating satellites to collect observations and communicate through a centralized controller is a very complex task. The MS needs to determine which DS can perform which task at what time such that the performance of the entire team can be optimal, which is illustrated by the schematic diagram shown in Fig 3.2. The initial planning problems that the MS is fac-

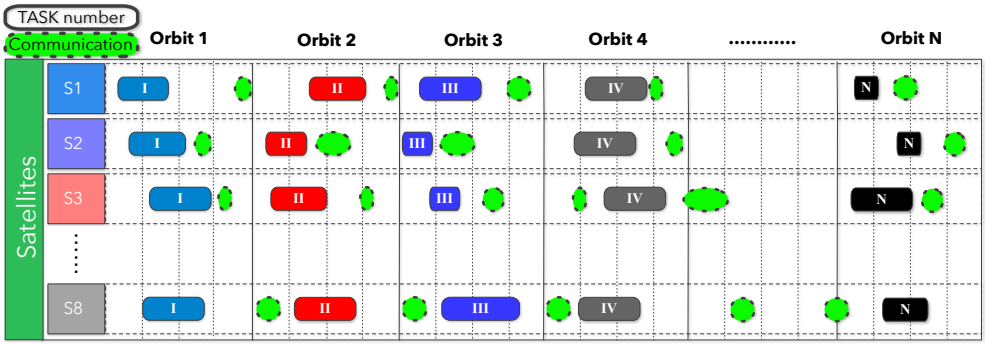


Figure 3.2: Example of initial plans generated by the MS. The orbit number represents the current orbit that every satellite is in since the start of the mission. The task number represent the assigned observation tasks by the MS. The com symbols represent the communication time assigned to each DS. The length of each box stands for the duration of corresponding behavior.

ing can be considered as NP-hard problems, where meta-heuristic algorithms are normally considered as efficient methods by many researchers. In the simulation done in Chapter 2, the EA family has been chosen as candidate algorithms to design the problem solver. Both the GAs and DEs are population-based optimization algorithms. The main difference is the encoding, where basic GAs use binary encodings, while DEs use real-valued encodings. Although DEs are more effective and more efficient than basic GAs, DEs are very complicated to formalize for a real-world problem and they cannot deal with mixed-integer variables (e.g. [Fleetwood, 2004]). Therefore, most existing research has implemented GA techniques to solve space mission planning and scheduling problems. Shi uses GA in [Shi et al., 1996] to solve the job-shop scheduling problem. Shtub in [Shtub et al., 1996] and Dorn in [Dorn et al., 1996] also propose to use GA as solvers for general scheduling problems. In [Sun et al., 2010], Sun uses GA to formalize a real-time algorithmic approach for determining a near-optimal sequence for a satellite payload. In [Globus et al., 2002][Globus et al., 2003], the authors have implemented GA to effectively schedule a cooperated fleet of Earth imaging satellites, and compare its performance with other EA algorithms to justify the advantages of GA. In [Soma et al., 2004], GA is used to perform optimal resolution of visibility clashes for a fleet of Indian remote sensing satellites, and the authors have developed an intelligent multi-objective priority activated task optimizer to generate a weekly optimal schedule for spacecraft operations. Chen proposes a hybrid optimization method based on GA and PSO in [Chen et al., 2012], which greatly improves the efficiency of the optimization for multi-satellite observation scheduling problems. All these articles show excellent performance of using GA as a scheduling tool for a space mission. However, most of the scheduling problems are focused on trajectory, resources, and antenna allocation. Few of them concern the behavior planning problems within a group of satellites. This chapter will use an improved GA as a tool which allows the MS to generate feasible initial plans for all the participating DSs such that the overall team performance can be maximized onboard.

3.2. PROBLEM FORMULATION

This section presents the details of how to transfer the initial planning procedure of the DSL mission into a behavior scheduling optimization problem.

3.2.1. NOTATIONS AND VARIABLES

The notations and the variables used in this chapter are defined as follows:

NOTATIONS

- H: Satellite operating orbital altitude. According to the original design of the DSL mission, this altitude is flexible to fulfill different scientific objectives. It ranges from 300 to 450 km.
- $S_i, (i = 1, 2, ..., N)$: Daughter Satellite set. In this thesis, the planning problems rely on a set of satellites, where N is the total number of participating satellites. For the DSL mission $N = 8$.
- B: Baseline. This parameter represents the geometric distance between two satellites in along-track direction. The baselines are flexible and depend on the scientific requirement, and range from 100 m to 100 km.
- T_O: Satellite orbital period. This notation represents the absolute time that a satellite orbits the Moon when operating at a specific orbit.
- $Time$: Mission lifetime. This variable represents the expected mission operating lifetime.
- $n_k, (k = 1, 2, ..., M, M = Time/T_O)$: Orbit number. It represents the current orbit since the start of the mission.
- $TW_{S_i} = \langle OW_{S_i}, CW_{S_i} \rangle$: Time window set. OW_{S_i} represents the *observation window* when satellite CW_{S_i} is able to obtain data. CW means the *communication window* when this satellite can communicate with other satellites.
- $Initial_{S_i} = \left\langle Obs^0_{S_i}, Com^0_{S_i}, M^0_{S_i} \right\rangle$: Initial sub-system parameter set . This set represents the initial value of key parameters for the three most relevant satellite aspects, which includes the initial observation data rate $Obs^0_{S_i}$, communication data rate $Com^0_{S_i}$, and onboard storage capacity $M^0_{S_i}$.

VARIABLES

- Observation time variable set $\mathbf{X_{S_i}}(n_k) = \left\langle x^S_{S_i}(n_k), x^E_{S_i}(n_k) \right\rangle$, where $x^S_{S_i}(n_k)$ represents the start time of the observation during orbit n_k, and $x^E_{S_i}(n_k)$ stand for the end time of the observation.
- Communication time variable set $\mathbf{Y_{S_i}}(n_k) = \left\langle y^S_{S_i}(n_k), y^E_{S_i}(n_k) \right\rangle$, where these two variables have the same meaning as observation variables, only these variables are for communication time.
- $M_{S_i}(n_k)$: Onboard memory. This represents available onboard memory for S_i at the beginning of the orbit n_k.

3.2.2. BLOCKED AREA

As mentioned in Chapter 2, to be able to detect the low-frequency radio spectrum without the RFI from the Earth, the DSL mission will use the Moon as a natural shield. The particular area of the Moon that is blocked from Earth interference is called "Blocked area", where observations can be performed. The distance from the Earth's center to

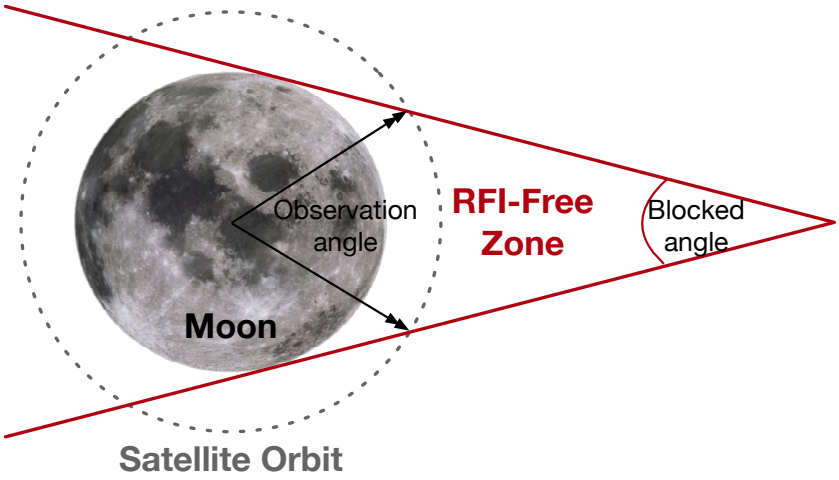


Figure 3.3: Observation window and blocked area of RFI

the Moon's center is called the Lunar Distance. The average value is 384402 km. Based on gravitation and classical physics, we can calculate the blocked angle from the moon's radius and distance. The blocked angle α_{blk} and the observation angle α_{obs} can be calculated by the following equations:

$$\alpha_{blk} = \arcsin \frac{r_E - r_M}{D} \tag{3.1}$$

$$\alpha_{obs} = \pi - \alpha_{blk} - 2 * \arccos \frac{r_M}{r_M + H} \tag{3.2}$$

In Eq 3.1, r_E and r_M represent the radii of the Earth and the Moon, respectively, and D stands for the average lunar distance. To match the observation frequency with the coherent radio emissions from Ultrahigh Energy Cosmic Rays (UHECR) and Neutrinos (UHEN) ([Gusev et al., 2006]), the DSL will operate in a particular circular Moon orbit at 300 km altitude. As the result of this assumption, the feasible observation window for this mission is obtained, as shown in Fig 3.3.

3.2.3. CONSTRAINTS

After providing the notations and the variables for the initial planning problem, boundaries and constraints need to be addressed based on the mission requirements.

CONSTRAINTS

The constraints formulated here represent scientific requirements, hardware limitations, and temporal and logical order.

- Variable boundaries
  Considering the hard constraints on the observations, the observation and communication window of each DS have strict boundary constraints. In each orbit, the observation behavior can only happen inside of the corresponding observation window OW^{n_k}.

$$\mathbf{X_{S_i}}(n_k) \in OW_{S_i}^{n_k} \tag{3.3}$$

  Meanwhile, all communication behavior can only proceed when each satellite is inside of the communication window $CW_{S_i}^{n_k}$. This is to avoid the radio interference which is caused by satellite communications and affects the observation quality.

$$\mathbf{Y_{S_i}}(n_k) \in CW_{S_i}^{n_k} \tag{3.4}$$

- Observation
  Besides the above boundaries, there are two types of constraints that need to be formulated. One type relates to the observation behavior, and another type relates to the communication behavior. One of the observation related constraints is the chronological order in every orbit. The observation start time shall always be ahead or at least equal to the end time.

$$x_{S_i}^S(n_k) \leqslant x_{S_i}^E(n_k) \tag{3.5}$$

  The time span of each observation behavior is another constraint. The span between the start and the end time in each orbit n_k shall be less than the time that satellite travels through the observation window. The full observation time is defined as TO.

$$x_{S_i}^E(n_k) - x_{S_i}^S(n_k) \leqslant TO \tag{3.6}$$

- Communication
  The communication behavior has the same constraint in the chronological order as the observation behaviors.

$$y_{S_i}^S(n_k) \leqslant y_{S_i}^E(n_k) \tag{3.7}$$

  Another important constraint of communication is that the amount of data transferred in a communication window in one orbit shall be less than the raw data it contains. For a long lifetime mission, the amount of data transfered in the orbit n_k shall be less than the raw data that onboard memory has at the start of this circle, plus the amount of data that it has gathered in this orbit.

$$\left[y_{S_i}^E(n_k) - y_{S_i}^S(n_k)\right] * Com_{S_i} \leqslant M_{S_i}^0 - M_{S_i}(n_k) + \left[x_{S_i}^E(n_k) - x_{S_i}^S(n_k)\right] * Obs_{S_i} \tag{3.8}$$

- Onboard storage
  In the DSL mission, to secure the integrity of the raw data sent by one DS, no other DS shall communicate at any time within the observation window. The MS only receives data from one DS at one time. Therefore, when a satellite re-enters an

observation window, the observation time span shall be limited by the accessible onboard storage.

$$\left[x^E_{S_i}(n_k) - x^S_{S_i}(n_k)\right] * Obs_{S_i} \leqslant M_{S_i}(n_k) \tag{3.9}$$

$$M_{S_i}(n_k) = M_{S_i}(n_k - 1) - \left[x^E_{S_i}(n_k) - x^S_{S_i}(n_k)\right] * Obs_{S_i} + \left[y^E_{S_i}(n_k) - y^S_{S_i}(n_k)\right] * Com_{S_i} \tag{3.10}$$

3.2.4. OBJECTIVE FUNCTION

The main objective of the DSL mission is to collect the maximum of raw data within a certain operation lifetime. To generate the best time sequence for each DS, the general planner needs to consider all the constraints from hardware and software in a realistic scenario as shown in before. In this chapter, the initial planning problem is simplified to only the observation and communication period for each DS. The purpose of this initial planning process is to generate initial plans, which help each DS to schedule its observation and communication activities. In this chapter, these plans are each DS's time windows for the observation and communication behaviors. Therefore, the first objective function is to maximize the total raw data that DSs can gather within a certain lifetime. The second objective function is to guarantee that the communication gap between each DS is minimized to maximize the communication efficiency of the MS.

$$OF1 : \max_{\mathbf{X}} \sum_{k=1}^{M} \sum_{i=1}^{8} \left[x^E_{S_i}(n_k) - x^S_{S_i}(n_k)\right] \tag{3.11}$$

$$OF2 : \min_{\mathbf{Y}} \sum_{k=1}^{M} \sum_{i=1}^{8} \left[y^S_{S_i+1}(n_k) - y^E_{S_i}(n_k)\right] \tag{3.12}$$

3.3. HYBRID DYNAMIC MUTATION GA

As mentioned before, the entire onboard mission planning problem can be considered as an NP-hard problem. Based on the DSL mission requirements, we incorporate observation and communication behavior together in one model (as shown in Eq 3.11 and Eq 3.12). The requirement of maintaining the onboard storage level as non-negative for every DS causes nonlinear constraints in this model. The combination of linear and nonlinear constraints increases the difficulty to solve the problem. Some preliminary experiments on algorithm selection have been done in Chapter 2. Due to the intractability and the large scale of the original problem, exact optimization algorithms such as the linear programming algorithm have been abandoned because of their poor adaptability. Meanwhile, for real applications, the oversubscribed feature is a common situation, and the constrained programming algorithm can not provide solutions for most oversubscribed situations. The nonlinear constraints in the model also cannot be handled by this type of algorithms. Therefore, algorithms such as mixed-integer programming, greedy search, local search or dynamic programming are all not feasible for this problem.

Instead, the planning algorithm should be flexible and scalable to adapt every possible scenario. The genetic algorithm (GA) is a meta-heuristic algorithm inspired by the process of natural selection, which has been shown to have great potential on both sides.

Therefore, we intent to use GA as the fundamental algorithm to solve initial mission planning problems.

3.3.1. BASIC GA

GA represents a series of methods based on heuristic random search technique, which belong to the larger class of EAs. The basic GA constructs a population by a set of solutions called chromosomes. The population is responses for evaluating its fitness value through the objective function at each generation. After the algorithm performs the *Evaluation* procedures, certain chromosomes are chosen from the current population through the *Selection* procedure. For this, it will use two genetic operators called: *Crossover* and *Mutation*. The purpose of these operators is to evolve the population for the next generation.

The crossover operator combines two chromosomes and randomly switches a part of them. This operator can produce more offspring with better (or worse) fitness values for the next generation. The mutation operator also changes the search space and creates new offspring, with sudden changes on one or few values of a chromosome. However, both crossover and mutation need a trigger to become active. For the crossover operator, this trigger is called the crossover rate, and for the mutation operator, it is called the mutation rate. After performing these two actions, one generation cycle is considered to be completed. During each cycle, all chromosomes need to be tested by evaluating their fitness function. When a GA reaches the stopping criterion which may contain the generation number, or a total calculation time, or other terms, the algorithm will stop.

The strength of GAs lies in that they can deal with multi-modal problems and are able to find the global optimum. However, the success rate of using GAs to solve these problems remains unstable and unreliable. For many problems, GAs may have a tendency to converge towards a local optimum instead of a global optimum. This is because of the diversity of different initial populations. Some populations may be closer to the global optimum, while others may be further. There are also other problems like low efficiency and operating on dynamic data sets. Among all these problems, the success rate is the most critical weakness when using GA as the mission scheduling algorithm for a deep space mission. One single mistake could cause the total failure of the mission. Many studies have been done to increase this probability of success. Two ideas have been proposed for this problem. The first idea is to create the data structure and genetic operators for each problem. However, this idea requires a huge modeling effort for each purpose, which is not practical. The second idea is to improve the efficiency by changing the strategies of each operator, e.g., one can use different rules to perform selection, crossover and mutation.

To target these weaknesses, in this thesis, we aim to enhance the performance of basic GAs by improving their mutation strategy. Many improvements have been made for mutation strategies. Still, these improved GAs have different weaknesses in different aspects. Tab 3.1 illustrates that two main weak points are still the early convergence and long computation time. However, for the DSL mission, the MS needs to provide initial plans to all the DSs as soon as possible once it has received the mission objective from the control center. This requires both high accuracy and short computation time. The Hybrid Dynamic Mutation Genetic Algorithm (HDMGA) is proposed for this

Table 3.1: Comparison of five kinds of mutation strategies: dynamic mutation strategy [Hong and Wang, 1996], schema mutation strategy [Li and Zhang, 2009], compound mutation strategy [Jin et al., 2014], adaptive mutation strategy [Srinivas and Patnaik, 1994a], and hyper mutation strategy Cheng [2012].

Name	Features	Advantages	Disadvantages
Dynamic Mutation	Use several mutation operators, change with generation grows	1) Help to determine appropriate mutation rate; 2) Increase algorithm convergence speed when come to the end;	1) It use too many mutation operators, which will slow down the algorithm at each generation; 2) This method only can help to increase the good fitness values' offspring mutation rate, which cause premature convergence problems;
Schema Mutation	Deal with premature convergence problem	1)Good wide range of constitutive property and maneuverability; 2)Effectively protect the excellent individuals; 3)Good efficiency and convergence stability	Easy to fall into local optimal
Compound Mutation	Big mutation rate for global search and small mutation rate for local search	1)Help to expend the search area and increase the diversity of individuals; 2)Assure the further evolution of population and avoid local convergence;	1)Each step needs to compare individuals, which may slow down the algorithm; 2)Not possible to produce a suitable mutation number;
Adaptive Mutation	Produces suitable mutation number adaptively; guaranteed the higher survival rate of good offspring	Reduce computation time and maintain the population variety for preventing premature convergence	Problem of random search
Hyper Mutation	Use two types of schemes to adjust the population diversity	Use two types mutation schemes to enhance the performance with dynamic environment	1)Hard to set proper boundaries for high and low mutation rates; 2)When using the second scheme, at each generation computer needs to recalculate the new mutation rate, costing time at each iteration;

requirement.

3.3.2. HDMGA PRINCIPLE

After comparing different mutation strategies, the efficient performance and improvements of the success rates of the GA is key in this research. This chapter proposes a new mutation strategy called Hybrid Dynamic Mutation (HDM). This strategy is inspired by the Dynamic Mutation [Hong and Wang, 1996] and the Adaptive Mutation [Srinivas and Patnaik, 1994a]. These mutation strategies have common advantages on adjusting their mutation operators to fit in different generations. However, due to their numerous mutation operators, at each iteration, the algorithm needs to decide which operator and which mutation rate is needed for this iteration. Meanwhile, their mutation rates will adjust when their fitness value approaches the current best fitness value. This is the reason why these two mutation strategies get easily stuck at a local optimum and cause early convergence.

Instead of using multiple mutation operators, the HDM only uses two mutation operators. These two operators work independently, one for the normal mode, and another one for the escape mode. The reason why HDM can overcome early convergence problems is because of these two operators. For a basic form GA, there exists a termination generation number T_G. This means that if the average value of the fitness function has not changed over several generations, the algorithm will stop. Within HDM, there is a switch criteria labeled as T_N. It will trigger the switch between the normal mode and the

escape mode before the termination criteria of the current generation is satisfied. Until the switch criteria has been reached, the algorithm uses the normal mutation rate R_{nor}. After this criteria has been met, the algorithm employs the escape mutation rate R_{esc} to try to jump out of the local minimum. To expand the search area, the mutation rate in the escape mode is larger than the one in the normal mode. The details about how to determine the values of these two important variables will be presented in next section.

Based on the previous idea, the general procedure will be described by the pseudo code shown in Algorithm 1.

Algorithm 1 Hybrid Dynamic Mutation Operator

Input: ε_S is the tolerant error,R_{nor} is the mutation rate used for normal mode, R_{esc} is the mutation rate used for escape mode, T_N is the trigger number
Output: Mutation rate M_R

function $mutationChildren = mutationHDM(PopulationSize, nvars, R_{nor}, R_{esc}, T_N, \varepsilon_S)$
  for $i = 1,2...PopulationSize$ **do**
    $Fitness_value(i,1) \leftarrow fitness_value(i)$
    for $j = 1,2...i$ **do**
      if $Fitness_value(j+1) - Fitness_value(j) \leqslant \varepsilon_S$ **then**
        $t_N = t_N + 1$
      else
        $t_N = t_N$
    if $t_N \geqslant T_N$ **then**
      $M_R = R_{esc}$
    else
      $M_R = R_{nor}$
  return M_R

Using this mutation strategy with the normal GA leads to the HDMGA which is described by the pseudo code in Algorithm 2. As shown in this pseudocode, it is clear

Algorithm 2 HDM Genetic Algorithm

Input: PopulationSiza pop_size, parameter number $nvars$, Termination error tolerant ε_T, Max stall generation SG_max, Max generation G_max, fitness function Fit_Fcn, R_{nor} is the mutation rate used for normal mode, R_{esc} is the mutation rate used for escape mode, T_N is the trigger number.
Output: Solution X

function $X = HDMGeneticAlgorithm(\ pop_size, nvars, \varepsilon_T, SG_max, G_max, Fit_Fcn)$
  $population \leftarrow Initial(pop_size, nvars)$
  while $counter \leqslant G_max$ **do**
    for $i = 1,2...pop_size$ **do**
      $Best_individual(i) = Cal(Fit_Fcn, population)$
      $Crossover(two\ random\ individuals)$
      $M_r \leftarrow mutationHDM(PopulationSize, nvars, R_{nor}, R_{esc}, T_N)$
      $pop = population$
      if $rand \leqslant M_R$ **then**
        $M_pos = round(nvars * rand)$
        if $M_pos = 0$ **then**
          $pop(i, M_pos) = 1 - pop(i, M_pos)$
       $population \leftarrow pop$
      if $Best_individual(i+1) - Best_individual(i) \leqslant \varepsilon_T$ **then**
        $T = T + 1$
      else
        $T = 0$
      if $T \geqslant SG_max$ **then**
        break
    $X \leftarrow bets_individual$
  return X

that the HDMGA still uses the basic GA frame, in each iteration it also includes the se-

lection, crossover, mutation and evaluation procedures. The HDMGA implements the HDM strategy to perform mutation procedure. This strategy provides the HDMGA the ability to fast escape from local optima, while it also provide stable mutation for regular situation. Once the termination terms of the HDMGA (number of generation, computation time, best error, etc.) have been met, the best individual will be the initial planning result.

3.3.3. VARIABLES DETERMINATION

From the last section, it is obvious that there are two main variables R_{nor} and R_{esc} which need to be determined. This section focuses on how to determine the best value for each of those variables.

NORMAL MODE

Table 3.2: Normal mode simulation results

	Computation time [s]				
Simulation number	rate=0.01	rate=0.05	rate=0.1	rate=0.3	rate=0.5
1	13.3369	14.9003	16.6729	11.0716	16.4441
2	20.5683	16.9102	14.5214	30	30
3	30	5.7414	15.7078	17.8776	16.8626
4	15.2022	7.3488	16.4792	15.8848	30
5	13.412	30	14.2844	17.853	14.8617
6	8.6736	16.1859	16.6208	30	17.2854
7	4.0094	14.209	30	30	20.3043
8	8.5721	18.2125	18.1646	19.4098	13.2203
9	30	6.841	16.6854	14.1032	16.1795
10	15.7552	15.082	15.6729	19.1987	30
11	17.8689	7.1522	18.7301	14.4185	17.885
12	15.1472	16.2313	16.1971	30	16.8645
13	30	30	19.219	16.9326	16.7691
14	6.2299	8.0894	17.498	30	30
15	14.924	7.5297	15.3424	15.1416	17.0021
16	14.9844	17.5528	6.1339	16.9371	16.9507
17	7.4624	18.1506	7.8213	16.3075	16.3346
18	16.2285	6.5755	12.5373	18.4377	18.6071
19	9.1894	6.3589	12.4471	30	16.6194
20	9.7759	18.384	7.7527	30	15.7848
21	14.6014	17.924	9.9632	18.9031	17.6374
22	13.4476	18.527	17.4132	11.3463	11.4826
23	10.0539	16.9837	16.5443	13.2492	30
24	16.5549	16.6369	15.5118	20.0295	16.8116
25	17.9073	10.7449	13.1734	30	13.7317
26	16.5192	16.7456	15.5657	17.8253	30
27	19.3513	17.5698	9.1163	20.5392	30
28	16.981	16.725	6.2711	16.961	30
29	14.1365	16.6542	7.3295	30	6.8656
30	13.8373	15.9048	13.2052	11.2499	13.1831
31	17.6754	15.7125	7.2497	18.4475	16.8372
32	17.955	7.5653	13.94	16.8063	15.8152
33	15.1276	9.2673	15.0834	16.7693	12.0196
34	17.5905	16.2885	15.8931	13.9612	14.1809
35	16.6031	8.0178	30	13.4521	9.0731
36	13.5552	7.8383	30	30	15.5831
37	16.0187	30	12.7102	18.5325	30
38	14.7639	15.5005	16.0046	17.9351	16.1537
39	14.4946	20.6298	17.0179	30	16.5179
40	17.7025	4.8683	15.262	30	17.0405
41	16.8998	8.527	13.3989	14.1631	18.5036
42	13.6083	11.2134	18.179	16.2882	30
43	12.2632	13.9158	16.978	19.1731	17.9894
44	22.166	12.0699	30	19.1034	30
45	14.2131	30	30	17.4174	15.8066
46	17.1661	12.2435	14.0398	16.4872	30
47	18.8417	10.4718	9.386	16.3488	13.9887
48	30	13.2519	16.6295	30	16.1584
49	14.2785	16.9294	15.2647	17.5944	30
50	17.2692	14.5639	30	16.9189	30
Mean	15.858462	14.494926	15.992376	20.061514	19.667102

Based on HDM principles, most of the time the HDMGA stays in the normal mode. This means that the performance of the normal mode directly affects the convergence process. The principle of the normal mode is the same as the mutation strategy used with uniform mutation. To determine the suitable value of this mutation rate for initial planning problems, we implement previous representative constrained problems as shown in Chapter 2 to test uniform mutation strategy (considering the page limitation, in this section, we only present results for TF9). The mutation rate ranges from 0.01 – 0.5 based on the empirical parameter value of an uniform mutation GA [Williams and Crossley, 1998]. Each mutation strategy is executed 50 times. During the simulation, the termination criteria for computation time is set to 30 to prevent the algorithm getting stuck at a local optimum for too long. For those results which reach the computation time of 30, it means that the algorithm is unable to find the global optimum. Therefore, it is considered as a failure. Based on test results, the following table (Tab 3.2) shows the detailed results of 50 simulations. The second row stands for five different mutation rates for the normal mode. The first column represents simulation number.

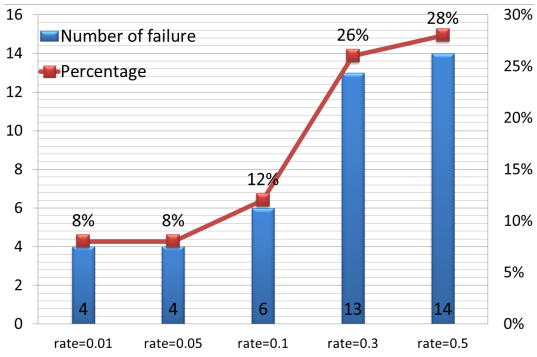


Figure 3.4: Failure number and percentage based on 50 simulations of various normal rates R_{nor}

Fig 3.4 is abstracted from Tab 3.2, which contains the number of failures and failure percentage of each mutation rate over 50 simulation. The failure percentage for the 50 simulations is growing with increased value of the normal mutation rate. A rate of 0.01 and 0.05 has the same failure number (4) and same failure percentage (8%). Meanwhile, the average computation time for each rate is summarized based on the valid data (i.e. values obtained in less than 30 seconds) over the 50 simulations, and the results are illustrated in the last row of this table. It is clear that when the mutation rate is set to be 0.05, the algorithm has a better performance on computation time than for the other four values. This result means that for HDM, the default mutation rate for normal mode should be set as 0.05.

ESCAPE MODE

After determining the mutation rate for the normal mode, the mutation rate for the escape mode R_{esc} needs to be determined. The methodology and test function remain the same as for the normal mode. The mutation rate for the normal mode is set as 0.05. The termination time for the algorithm is still set to 30. The escape mode is used to jump out

of the local optimal, which requires a much higher probability to trigger the chromosome mutation to expand the search space. Therefore, the escape mutation rate is set to a range from 0.55 – 0.95. Tab 3.3 and Fig 3.5 illustrate the performance and analysis from 50 simulations.

Table 3.3: Escape mode simulation results

	Computation time [s]				
Simulation number	rate=0.55	rate=0.65	rate=0.75	rate=0.85	rate=0.95
1	5.642354	7.440349	5.7341	16.5992	4.6345
2	4.936087	11.812699	7.0087	3.3413	6.4737
3	6.323403	7.113951	6.1287	3.9814	7.1643
4	8.780604	4.51954	6.6002	9.0021	13.8092
5	7.072866	5.372019	5.2113	14.1823	4.9098
6	13.841103	6.723661	30	10.3199	6.8936
7	7.47665	5.468952	4.2704	10.0142	9.3156
8	10.515967	5.134882	2.8074	5.3177	30
9	5.735285	7.781025	9.4067	2.9981	5.5717
10	9.110012	8.670706	8.0344	4.3601	5.002
11	6.0516	5.379014	5.1271	13.5266	3.0743
12	14.0878	9.542031	2.9281	23.357	5.9755
13	6.5088	30	5.7274	6.2589	14.5376
14	6.8534	14.589169	6.9153	7.4975	5.5834
15	5.1959	6.102298	7.5397	7.2002	9.4667
16	16.2939	6.708615	8.2937	8.0407	10.9731
17	4.6471	30	30	9.6205	4.7791
18	6.0429	5.709645	5.6858	6.2342	5.8722
19	2.1908	4.100325	8.1799	6.0911	10.911
20	4.0836	5.196435	4.765	6.205	7.9407
21	9.5936	8.074185	4.123	12.9654	4.9632
22	5.4417	3.041382	5.3204	6.835	30
23	3.1602	8.743459	5.1109	3.9543	5.0241
24	4.8365	4.316142	6.3088	5.0527	5.0109
25	4.246	3.072595	5.5983	6.9999	6.2391
26	7.9222	10.040585	30	8.2596	7.8247
27	6.8372	5.89488	5.0119	5.3122	30
28	12.3173	30	6.241	10.6712	30
29	14.5906	5.029452	30	3.7847	30
30	5.4981	7.987153	11.5877	9.3973	14.9428
31	6.6018	9.5442	5.8375	4.069	30
32	18.6335	8.3932	10.242	8.404	30
33	7.2245	8.8391	7.4204	15.1347	5.9507
34	25.683	7.4665	15.6282	7.6166	18.512
35	14.6313	30	7.2938	6.7527	17.8559
36	10.918	10.7536	2.4621	30	6.9455
37	14.7833	30	19.0555	10.9562	19.7611
38	9.9066	8.9235	30	30	7.3542
39	7.4032	7.5076	6.9454	30	17.4248
40	6.6318	11.2011	5.8806	30	9.8284
41	8.9551	30	2.94836	8.4162	30
42	8.5912	4.6674	7.7191	10.19	6.81
43	7.3302	7.17	4.0772	7.3788	18.8292
44	4.7189	5.2881	8.6315	6.3605	14.3623
45	30	9.7524	8.6531	30	17.1568
46	3.0655	5.0447	30	7.8641	30
47	6.0603	9.0608	26.009	4.9506	18.4234
48	11.9477	4.5465	14.7857	7.5732	6.5844
49	4.5254	12.0713	13.9623	30	30
50	7.6653	8.0116	30	6.6859	27.7874
Mean	8.82220262	10.03613498	10.7443532	10.794656	14.009578

Fig 3.5 shows the results of the failure number and percentage over the 50 simulations. It is clear that a mutation rate 0.55 has best performance among five mutation rates on both success rate and average computation time. The failure number is at least 80% less than other mutation rates. Tab 3.3 shows the detailed results of every simulation, and the average computation time from valid simulations (the last row of Tab 3.3). A rate of 0.55 has the lowest average computation time compared to other mutation rates. Therefore, $R_{nor} = 0.05$ and $R_{esc} = 0.55$ are the best choices for these two variables.

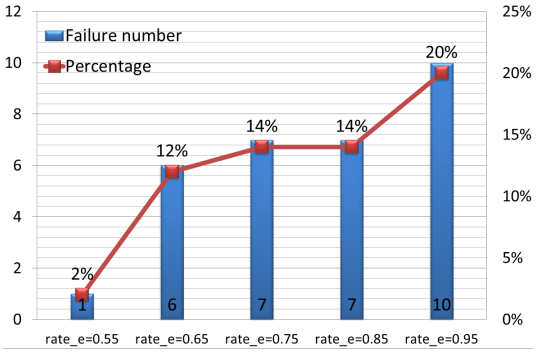


Figure 3.5: Failure number and percentage based on 50 simulations of various escape rates R_{esc}

3.3.4. PERFORMANCE EVALUATION

Once the main variables of HDM have been determined, it still needs to be evaluated whether this new mutation strategy has a better performance than other popular mutation strategies. All test functions mentioned in the previous chapter have been employed to evaluate the performance of the proposed method. This section presents part of the evaluation where two test functions are used to represent an unconstrained problem and a constrained problem. Four different mutation strategies are tested, including HDM, Gaussian mutation, adaptive feasibility mutation, and dynamic mutation [Srinivas and Patnaik, 1994b].

UNCONSTRAINED PROBLEM

For the unconstrained problem, a benchmark test function named "Beale function" is used [Surjanovic and Bingham, 2018b]. This function is defined as:

$$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2 \tag{3.13}$$

This test function has two variables, subject to $-4.5 \leqslant x_i \leqslant 4.5$. The global minimum is located at $\mathbf{X}^* = (3, 0.5)$, $f(\mathbf{X}^*) = 0$. The simulation results of 50 runs for each mutation strategy are shown in Tab 3.4.

Table 3.4: Computation time of 50 runs for the Beale function using four mutation strategies

	Computation time [sec]			
Experiment number	HDM	Gaussian	Adapt-feasible	Dynamic
1	5.642354	7.9	8.9263	14.9003
2	4.936087	7.1282	9.4191	16.9102
3	6.323403	17.5438	13.6213	5.7414
4	8.780604	8.3849	25.648	7.3488
5	7.072866	11.5475	5.2753	30
6	13.841103	30	8.6611	16.1859
7	7.47665	7.4122	14.3264	14.209
8	10.515967	16.7689	15.2937	18.2125
9	5.735285	15.7596	8.2695	6.841
10	9.110012	13.4733	5.8645	15.082
11	6.0516	6.2795	8.0252	7.1522
12	14.0878	7.6856	30	16.2313
13	6.5088	8.4806	30	30
14	6.8534	15.5095	4.5189	8.0894
15	5.1959	7.6407	30	7.5297
16	16.2939	8.6355	8.7017	17.5528
17	4.6471	16.4661	14.822	18.1506

18	6.0429	12.7039	13.9411	6.5755
19	2.1908	7.1996	30	6.3589
20	4.0836	7.2144	10.2225	18.384
21	9.5936	14.4803	13.5252	17.924
22	5.4417	16.969	6.8481	18.527
23	3.1602	14.9105	10.2479	16.9837
24	4.8365	8.3607	8.0006	16.6369
25	4.246	15.9953	4.678	10.7449
26	7.9222	15.8916	15.2829	16.7456
27	6.8372	16.9592	19.7967	17.5698
28	12.3173	15.2293	13.819	16.725
29	14.5906	8.4368	13.9595	16.6542
30	5.4981	15.1831	13.3313	15.9048
31	6.6018	15.3953	10.0195	15.7125
32	18.6335	16.4629	30	7.5653
33	7.2245	18.8614	5.9227	9.2673
34	25.683	30	9.0193	16.2885
35	14.6313	15.8133	14.5835	8.0178
36	10.918	13.9048	3.7788	7.8383
37	14.7833	7.5642	20.6044	30
38	9.9066	15.2451	7.2171	15.5005
39	7.4032	15.7064	16.4063	20.6298
40	6.6318	13.1919	13.3444	4.8683
41	8.9551	12.4203	5.5747	8.527
42	8.5912	14.9775	7.7419	11.2134
43	7.3302	13.5201	12.945	13.9158
44	4.7189	13.3626	6.2364	12.0699
45	30	16.5882	7.1536	30
46	3.0655	9.7365	30	12.2435
47	6.0603	30	5.8492	10.4718
48	11.9477	30	13.6836	13.2519
49	4.5254	18.413	30	16.9294
50	7.6653	7.2313	19.6281	14.5639

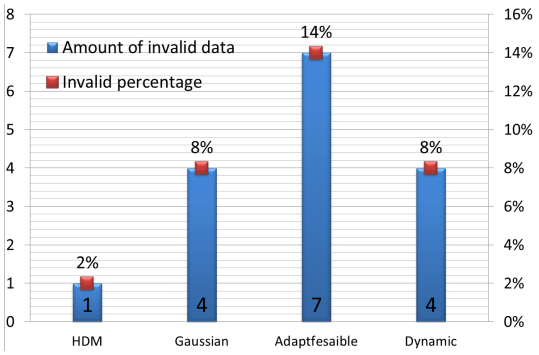


Figure 3.6: Failure number and percentage results for four mutation strategies based on 50 simulation runs

Fig 3.6 shows the failure number and percentage for four mutation strategies over 50 simulation runs. From this figure, among the four strategies, the adaptive-feasible mutation has the highest percentage of failure (14%). This number is almost as twice large as results of Gaussian mutation and dynamic mutation (8%). In contrast, HDM has a significantly lower rate of failure. During 50 simulation runs, this strategy only failed once. So, when considering the chance of non-failure, HDM has a competitive advantage compared with the other strategies.

To analyze these results, the computation time in Tab 3.4 is divided into seven levels. The first level represents computing times less than 5 seconds, each level increasing by 5 from the previous level. For computation times beyond 30 seconds, the test is considered as a failure. Fig 3.7 shows the pie chart for each mutation strategy, illustrating their performance on computing time. It can be seen that the computation time that

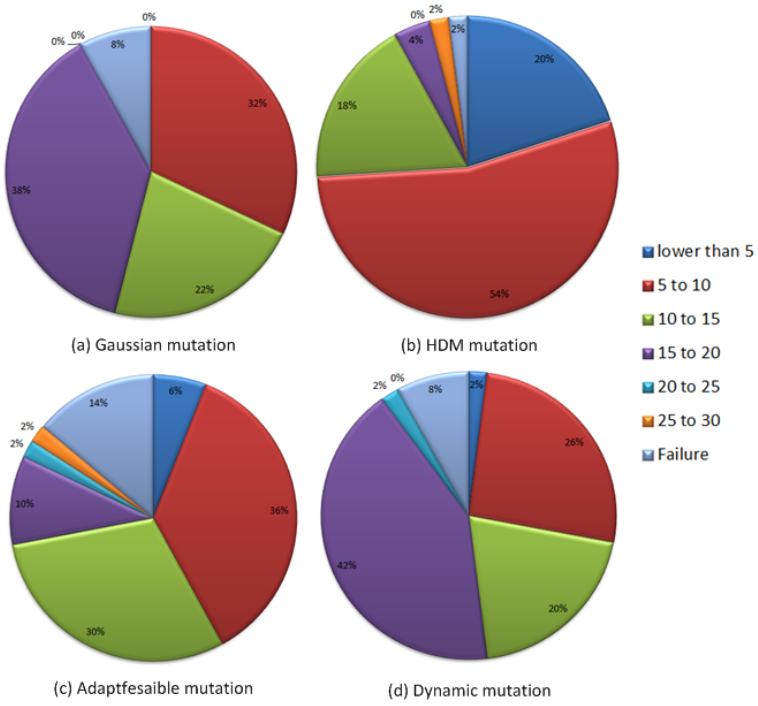


Figure 3.7: Data distribution based on computation time over the 50 simulations

HDM requires is mostly (54%) between 5 and 10 s, which is almost 20% higher than the Gaussian mutation and the adapt-feasible mutation, and 30% higher than the dynamic mutation. This means that HDM has the best performance on computation time among these four mutation strategies.

CONSTRAINED PROBLEM

As shown above, the HDM has the best performance in success rate and computation time among the four mutation strategies in solving unconstrained problems. It still needs to be tested whether HDM has a good performance for a constrained problem. The test function used here is adopted from [Hedar, 2018a].

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{3.14}$$

This problem has nine constraints, as shown in Eq 3.15:

$$
\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leqslant 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leqslant 0 \\
g_4(x) &= -8x_1 + x_{10} \leqslant 0 \\
g_5(x) &= -8x_2 + x_{11} \leqslant 0 \\
g_6(x) &= -8x_3 + x_{12} \leqslant 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \leqslant 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \leqslant 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \leqslant 0
\end{aligned}
\tag{3.15}
$$

The global minimum is located at $\mathbf{X}^* = (1,1,\cdots,3,3,3,1)$, $f(\mathbf{X}^*) = -15$. Considering the high time consumption for solving this problem, the function tolerance is set to be at a relatively large number (in this case 100s), so the algorithm can be terminated by this criteria when it gets stuck into a local optimum. Instead of trying to find the global minimum of this problem, this test is designed to verify the ability of escaping from a local optimum for four different mutation strategies. Therefore, the results focus on fitness values rather than on the computation time as shown in Tab 3.5.

Table 3.5: Fitness values of 50 simulation runs for the constrained function using four mutation strategies

	Fitness Value			
Experiment number	HDM	Gaussian	Adapt-feasible	Dynamic
1	-14.8886	-14.9995	-12.5129	-12.0806
2	-12.8721	-12.6512	-10.2718	-10.021
3	-14.9951	-5.4091	-12.1139	-9.1822
4	-9.3308	-11.9645	-9.3669	-10.1325
5	-14.9987	-6.7161	-12.4444	-11.0347
6	-14.9994	-11.9105	-14.9996	-12.6723
7	-12.962	-11.9048	-10.481	-10.2732
8	-14.999	-12.9687	-10.199	-10.5115
9	-11.8476	-10.4551	-12.0078	-14.9996
10	-14.9995	-12.5684	-14.9994	-14.9995
11	-13.2548	-6.2969	-14.9964	-9.461
12	-9.0221	-14.9995	-9.9649	-9.4534
13	-14.9478	-10.4354	-10.6301	-12.4483
14	-11.2541	-12.6691	-12.996	-12.033
15	-9.0518	-6.5441	-9.9694	-7.1963
16	-14.9995	-10.6224	-10.6301	-14.9997
17	-10.2648	-13.3414	-12.996	-9.2907
18	-13.2182	-12.047	-9.4347	-9.4844
19	-14.9997	-10.1834	-14.9995	-14.9992
20	-14.8477	-12.3226	-11.909	-14.9995
21	-10.2648	-9.0559	-14.9954	-14.9997
22	-14.2254	-9.3062	-7.6785	-10.1263
23	-14.9995	-10.6902	-11.7587	-12.0459
24	-13.2548	-9.6137	-14.9997	-14.9995
25	-14.9993	-12.6394	-8.115	-14.9994
26	-13.9874	-9.9774	-9.8515	-9.1938
27	-14.9995	-12.7101	-10.3731	-14.9995
28	-14.5841	-10.6558	-14.9957	-12.0019
29	-14.5971	-12.0025	-14.9957	-12.9981
30	-14.9997	-14.9995	-12.8629	-11.9388
31	-9.1848	-11.9235	-12.4082	-13.0478
32	-13.5841	-7.8327	-9.1286	-9.2957
33	-14.9999	-12.1134	-11.9813	-8.2301
34	-14.9958	-14.9996	-9.8558	-14.9997
35	-14.9584	-14.9989	-11.7356	-12.54
36	-14.6587	-11.9999	-12.8975	-10.5332
37	-9.0654	-12.5833	-14.9991	-11.5641
38	-14.9123	-10.3102	-14.9992	-9.0249
39	-8.5058	-12.3805	-8.94466	-14.9998
40	-9.2518	-9.3845	-9.7331	-12.2317
41	-14.9995	-9.5104	-14.9995	-12.1852

42	-14.9874	-7.8454	-14.9996	-12.9736
43	-14.9002	-12.8049	-12.6029	-11.7341
44	-13.9584	-12.0088	-12.4467	-9.7371
45	-9.2051	-12.916	-10.6195	-14.9995
46	-10.6584	-12.3419	-11.3241	-12.2826
47	-14.9995	-12.9693	-10.6704	-9.2024
48	-14.9996	-14.9999	-9.5477	-14.9995
49	-14.9987	-12.5327	-9.3519	-7.7376
50	-12.9284	-14.9997	-14.9997	-9.8823

Tab 3.5 shows the fitness values for each mutation strategy during each test. Comparing these fitness values with the global optimum of the test function, the success rate of each mutation strategy is shown in Fig 3.8 . Based on this figure, it is obvious that even

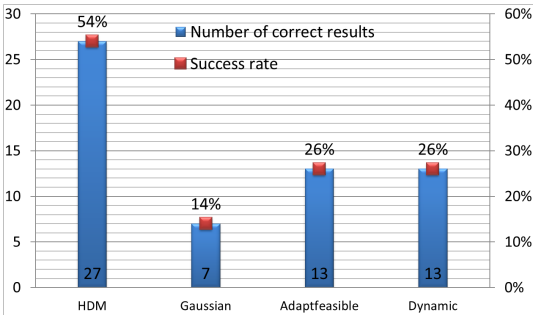


Figure 3.8: Success rates of four mutation strategies

with a high function tolerance, HDM shows the best performance on success rate compared with other strategies. Dynamic mutation and adapt-feasible mutation have the same percentage of success, while the Gaussian mutation has the lowest success rate. Therefore, HDM has the best performance in terms of escaping from a local optimum among four mutation strategies for the problem in Eqs 3.14 - 3.15.

After comparing the performance of solving the unconstrained and the constrained problems, HDM shows a great potential in saving computation time and escaping from local optima. The next section will apply HDMGA for the DSL initial planning problems.

3.4. INITIAL PLANNING PROCEDURES FOR THE DSL MISSION

3.4.1. SIMULATION SCENARIO ASSUMPTIONS

In the DSL mission, there are many payloads and subsystems which need to be considered. However, this chapter only intends to provide the initial planning sequences for the observation and communication behaviors of the eight DSs. Based on this constraint, further assumptions are made to simplify the simulation environment.

As mentioned in Chapter 2, the MSS will be operated in a lunar orbit with a linear array configuration. The MS is assumed to be positioned in the middle of the fleet, which is the 5th place. The initial positions of all the simulations are set to be the same, where the MS is placed at one of the conjunction points of the operating orbit and the Earth - Moon center connection line. To simplify this initial planning scenario, a circular orbit is assumed with an altitude of $Orbit$ = 300 km. Each DS's initial memory has the same setting: M_{S_i} = 128 GB. The raw data rate from the payload is identical for every

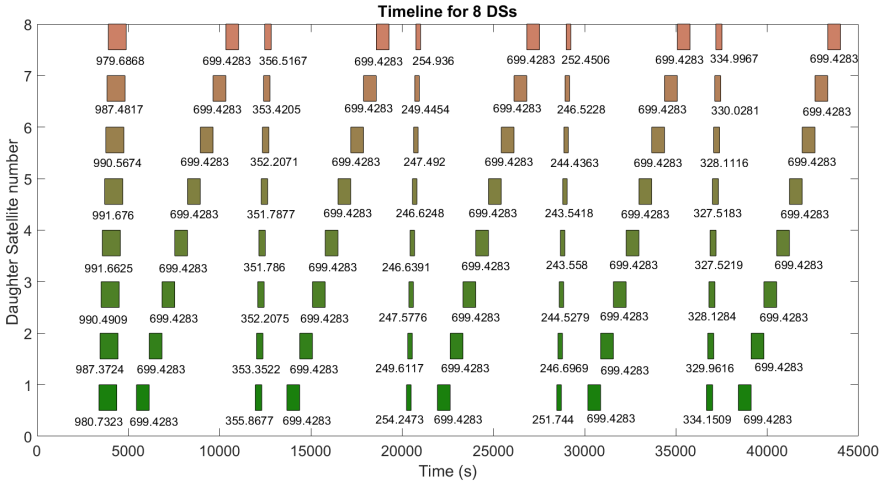


Figure 3.9: Initial planning sequences for eight DSs over five orbits. Each DS in each orbit has two behaviors, the frontier one represents the observation and the later one stands for the communication window. The parameters under each box represent the corresponding duration of each behavior.

DS: Obs_{S_i} = 48 Mbps. The data transmission rate is Com_{S_i} = 6 Mbps for each DS. The baseline (i.e. the distance between adjacent satellites) is set to be B = 100 km. The parameters of mutation operators in the HDMGA are follow the same setting as it been designed.

To avoid interference between the data transmission from the DS to the MS, in this phase, the communication strategy is chosen as point-to-point (P2P), i.e., at each instance, the MS can only communicate with one DS. With these assumptions, the general planner can provide valid initial plans automatically for different observation objectives.

3.4.2. HDMGA RESULTS

Based on the previous assumptions, Fig 3.9 represents a study case which illustrates detailed initial plans for all eight DSs with the total orbit number set to be M = 5. The Y-axis stands for the satellite number, while the X-axis represents the elapsed time. In this figure, eight DSs have received their own time sequences from the MS for their observation and communication behaviors. The observation windows of different DSs may have overlaps, but there is no overlap in communication time based on the P2P communication protocol. Fig 3.10 shows four different locations of the MS and DSs in the first orbit.

These four pictures show the different positions of DSs and the MS in the first orbit. Fig 3.10(a) shows all the satellites out of the observation window. Therefore, all the communication and observation indicators for each DS are switched off. When some DSs enter the observation window, the onboard executor will trigger the observation equipment to start gathering data. The example shown in Fig 3.10(b) illustrates the situation

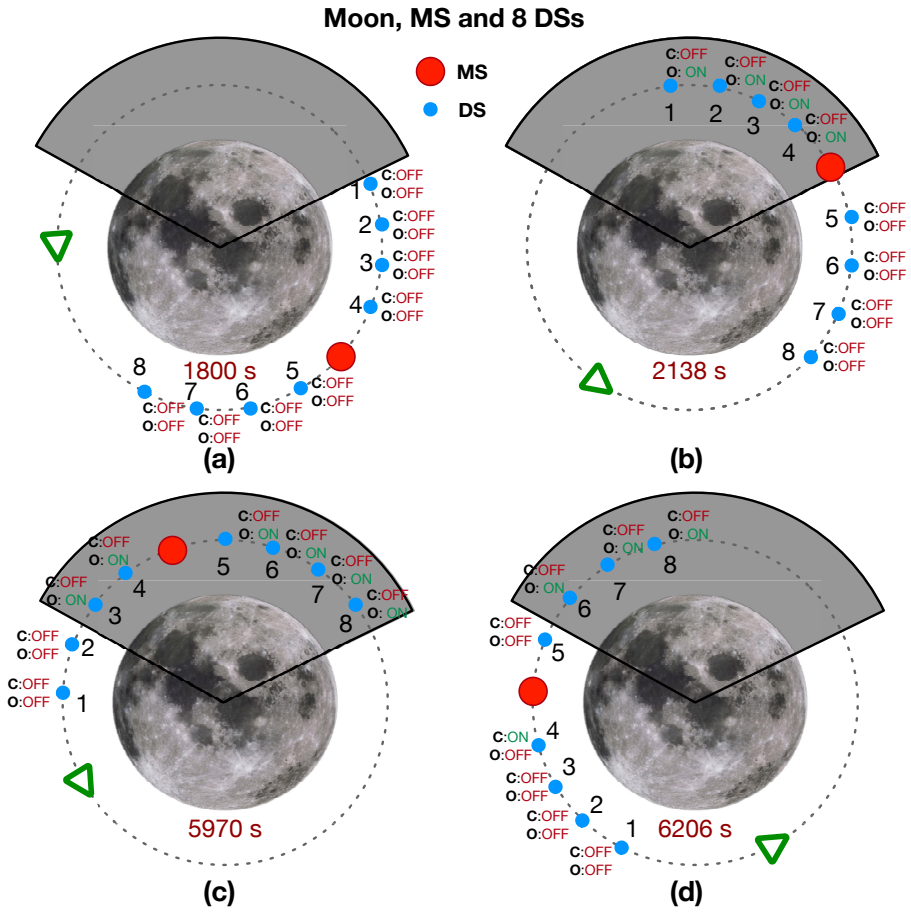


Figure 3.10: DSs' observation and communication plans for the first orbit, "C" stands for communication and "O" stands for observation. (a) stands for the location where all the satellites are outside of the OW; (b) stands for the DS 1-4 and the MS are inside of the OW, while other DSs are out of the OW; (c) indicates that the DS 2-8 and the MS are inside of the OW, DS1 and DS2 are out now; (d) indicates that DS4 is communicating with the MS now, while the DS 6-8 are still observing data inside of the OW.

2138 s after the simulation started. There are four DSs in the observation window, with the same observation indicators "ON". The DSs which are outside of the observation window remain with their payload "OFF". Fig 3.10(c) shows another location where DS1 and DS2 are out of the observation window, and their observation indicators show "OFF". Here, DS3 - DS8 are inside of the observation window, with their observation indicators "ON". Fig 3.10(d) shows that DS4 is communicating with the MS at time 6206 s. Based on the simulation results in Fig 3.9 and Fig 3.10, it is clear that HDMGA can provide proper initial plans for the reference MSS based on specific mission requirements.

3.4.3. EVALUATION AND ANALYSIS

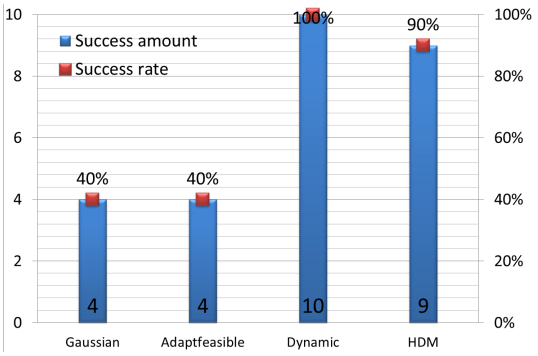


Figure 3.11: Success rate of four GA mutation strategies

The previous results show that HDMGA can solve the initial planning problem for the MSS reference scenario. However, for a real planning scenario, the computation efficiency and success rates are also important. To verify the performance of the proposed HDM, comparisons with three other popular mutation strategies are carried out. The details of the comparison procedures draw the same assumptions used in Chapter 3.3.4. However, considering the long computation time for this scenario, the simulation time is reduced by a factor of 10 for each strategy. The success rate and the computation time are two aspects to be compared in the sequel.

Fig 3.11 illustrates the success rate of each mutation strategy during this simulation. The dynamic mutation has the highest success rate which is 100%, and HDM reaches 90%. In comparison, the other two strategies, Gaussian mutation and adapt-feasible mutation, perform worse with only 40% success rate each. In Fig 3.8, HDM has a better performance than Dynamic mutation, because a larger termination error tolerant ε_T has been used when evaluating the constrained problem (Eq 3.14). However, for real planning problems, this termination error tolerance ε_T will be much smaller such that the algorithm can escape from local optima. The dynamic mutation strategy with its plenty mutation operators (depends on the scale of the problem), can provide more choices when the algorithm gets stuck at different local optima. In conclusion, the dynamic mutation strategy performs better than HDM in terms of success rate.

The average computation time and average convergence generations are shown in Fig 3.12. It is clear that HDM is faster in finding the results than the dynamic mutation strategy. Based on this figure, the proposed mutation strategy HDM saves almost 25% of computation time and 29% of convergence generation when compared with the dynamic mutation strategy. Considering the success rate and computation time, the HDM proofs itself a better mutation strategy than other considered mutation strategies for this specific planning problem.

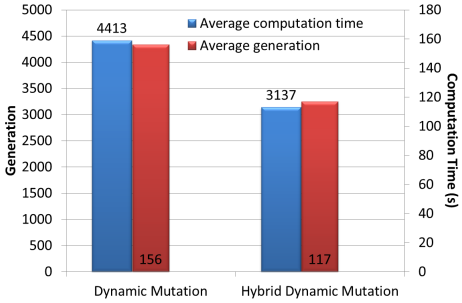


Figure 3.12: Results of average computation time and average convergence generation of the HDM and the dynamic mutation strategies

3.5. SUMMARY

In this chapter, a new mutation strategy for GA is proposed to solve initial planning problems for the DSL mission. Based on the reference MSS, the centralized onboard autonomous planning architecture is introduced in the first section, together with a review on GA implementations for related planning problems. Then, based on the DSL mission requirements, the initial planning problem is formulated as an optimization problem. The mathematical model of this problem is described as well. Considering the weaknesses of several representative mutation strategies, a new strategy, called HDM, is proposed. The test results illustrate that the proposed HDM strategy has a superior performance for standard fitness functions of both constrained and unconstrained problems. The average convergence time and success rate both indicate that the HDM is superior to its competitors namely, Gaussian mutation, Adaptive mutation, and Dynamic mutation.

Based on simplified assumptions for the DSL mission, the HDMGA is implemented as the solver for a general planner to generate the initial plans for observation and communication behaviors of the entire team. The simulation results on test functions have verified the proposed HDMGA. The performance of the proposed strategy is compared with three other state-of-the-art mutation strategies. The comparison shows that HDMGA has the best convergence time and a higher success rate. This conclusion indicates that HDMGA can be used as a general planner in the MSS architecture, and that it can generate suitable initial plans for the entire team.

With the initial plans, the DSs can start to execute these plans without any interference from ground operators. However, the space environment is very harsh, and the satellites are vulnerable. So, the planning system shall be robust and show graceful degradation against failures of operating satellites. In the next chapter, we will consider several emergency situations which may occur during operations. The corresponding re-planning methods will be investigated there.

4

CENTRALIZED ONBOARD RE-PLANNING

Parts of this chapter have been published in Z.Zheng, J.Guo, E.Gill, *Onboard Autonomous Mission Re-planning for Multi-Satellite System*, Acta Astronautica **145**, 28-43 (2018)

4.1. INTRODUCTION

IN the previous chapter, a mutation strategy called Hybrid Dynamic Mutation (HDM) was proposed to improve the Genetic Algorithm (GA) to solve the initial planning problems for a multi-satellite system (MSS). It is assumed that when the general planner on the MS performs the initial planning for the entire team, all the DSs remain healthy and the system status is consistent. Therefore, the initial planning procedure is considered as an off-line optimization problem. However, during the mission, many factors can change the conditions of the operating team. These factors can either be new mission objectives which have been uploaded from the control center, or they can be malfunctions on the operating satellites which are caused by external space environment (e.g. space debris, meteorites) or internal system failures (e.g. failures on sensors, processors or actuators). No matter what is the reason that causes the changes on the operating satellites, these influences invalidate the initial plans. Under these circumstances, the mother satellite (MS) can either choose to terminate the current mission plans and perform the initial planning processes again based on the new conditions, or it can employ a re-planner to modify and adjust the current plans to fit the new situation.

However, to solve a real-world re-planning problem, several questions need to be considered first:

(1) When to perform the re-planning?

(2) How much computation resources should be given to the re-planner for its task?

(3) How to minimize the effect of executing old plans during the re-planning procedure?

(4) How to ensure the new plans are not in conflict with the current executed activities?

By answering these questions, many re-planning approaches are proposed for specific mission re-planning problems. One widely used re-planning method is called Iterative Repair (IR), which was first proposed to be ready to continually modify the initial plans for the DS-1 mission. It has been used for different re-planning problems, such as [Rabideau et al., 1999b], [Chien et al., 2000], [Zweben et al., 1993], and [Rabideau et al., 1999a]. The basic idea of IR is that at each iteration, the re-planner will first identify the old plans for the current planning horizons, then update the goals and system states, and generate new planning horizons. By employing IR, the onboard reaction time to respond to changes in the operating environment or mission objectives can be reduced, and system conflicts caused by these changes can be avoided. Later on, the idea of IR has been expanded and combined with other re-planning techniques to formalize different re-planning approaches. The main purpose of these re-planning approaches is to provide suitable, approximate optimal new plans for the system within a short reaction time. These approaches normally focus on improving the quality of the new plans by targeting one or a few perspectives. For example, [Brock and Khatib, 2000] [Boella and Damiano, 2002] [Grecu and Gonsalves, 2000] [Tompkins et al., 2006] all aim to provide feasible re-planning architectures for specific applications, which can decrease the re-planning reaction time for these specific applications. [Likhachev et al., 2005] [Stentz et al., 1995] [Duan et al., 2009] [Das et al., 2016] implement different heuristic algorithms

to improve the efficiency and effectiveness of the optimization procedures for the re-planner. [Jiang et al., 2008] [Chamseddine et al., 2015] [SalarKaleji and Dayyani, 2013] propose several unique fault-detection mechanisms to reduce the reaction time of the re-planning methods.

Although many re-planning approaches have shown good performances for different scenarios, few of these scenarios are related to multiple subjects under an unpredictable operating environment. Therefore, considering the DSL mission as the reference, unexpected situations that can occur during the operations need to be addressed. This chapter aims to solve the mission re-planning problems for the reference MSS under three specific emergency situations (failures related to onboard memory, observation, and communication), and also provide corresponding re-planning approaches.

4.2. ARCHITECTURE OF ONBOARD PLANNING & RE-PLANNING SYSTEM

As described in Chapter 3, the DSL mission employs a centralized control architecture where the MS acts as the main controller to perform the initial planning. Fig 3.1 reveals the general structure of the centralized mission planning approach for the reference MSS. Based on this, the architecture of the planning & re-planning system is constructed, as shown in Fig 4.1. This architecture includes the general planner (GP), the decision-maker (DM), and the re-planner (RP), which are all executed on the MS, while the mission executor and status monitor (MEM) is running on each DS. Following this flowchart, once a DS receives the initial plan, it will execute this plan and constantly provides feedback to the MS. The MS will make decisions about whether the re-planner needs to be activated on the basis of these status feedbacks. With the combination of all these components, the MS has the ability to plan or re-plan for the entire MSS under certain scenarios. This section will introduce the functionality of these components, provides the basic assumptions, and formulates the entire architecture for the centralized onboard planning and re-planning system.

4.2.1. GENERAL MISSION PLANNER

The general mission planner (GP) is the key component for solving the onboard initial mission planning problem. Its flowchart is shown in Fig.4.2. This module has been designed to receive high-level mission requirements from the control center and provide suitable initial control sequences for every DS in the system. Considering the centralized control model, the MS receives the commands from the control center and considers them as high-level goals (e.g. which area to observe, or the total observation time, etc.). Based on these high-level goals and the current available resources on all DSs, the GP on the MS can establish the initial planning problem. Then, the GP will proceed according to the following steps:

a. Sampling the operation area
   As described in Chapter 2, the RFI-blocked area provides an Observation Window for the DSL mission to observe low frequency radio signals from deep space. This step uses the pre-designed orbit altitude and the Earth-Moon distance, sampling the

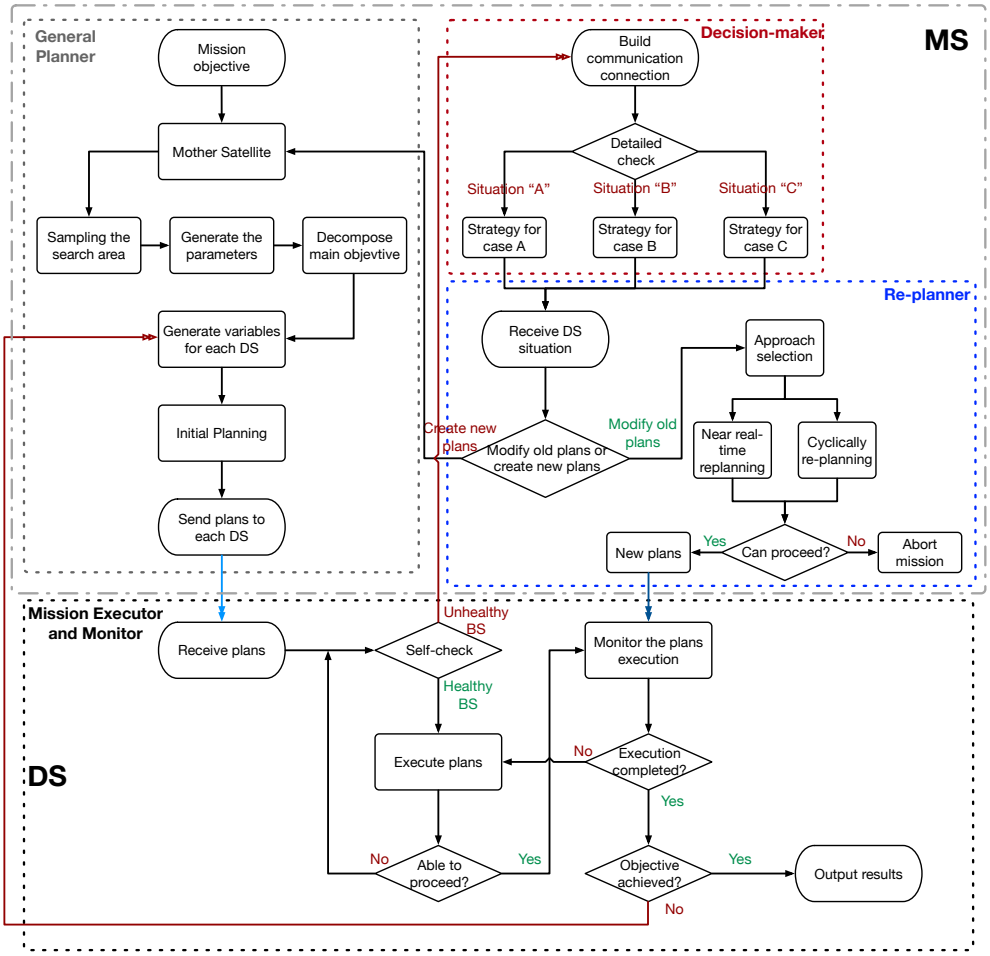


Figure 4.1: General flowchart of the planning and re-planning system for the reference MSS. It consists of four modules, the general planner (GP), the decision-maker (DM), the re-planner (RP), and DS mission executor and monitor (MEM). The GP, DM and RP are located on the MS, and the MEM is located on each DS.

operation area for the mission assignment. The blocked angle and observation angle can be calculated using Eq 3.1 and Eq 3.2.

b. Mission decomposition

In this step, the high-level goal is expanded and transformed into a mathematical form which the GP is able to handle. Based on the system status of all DSs, the planner can decompose the main mission into several sub-missions. For each DS, input parameters can be calculated based on these sub-mission requirements. These input parameters include the observation and communication windows, the mission lifetime, the satellite orbital period, and other planning parameters. Therefore, the high-level goal can be decomposed and the entire planning problem can be formu-

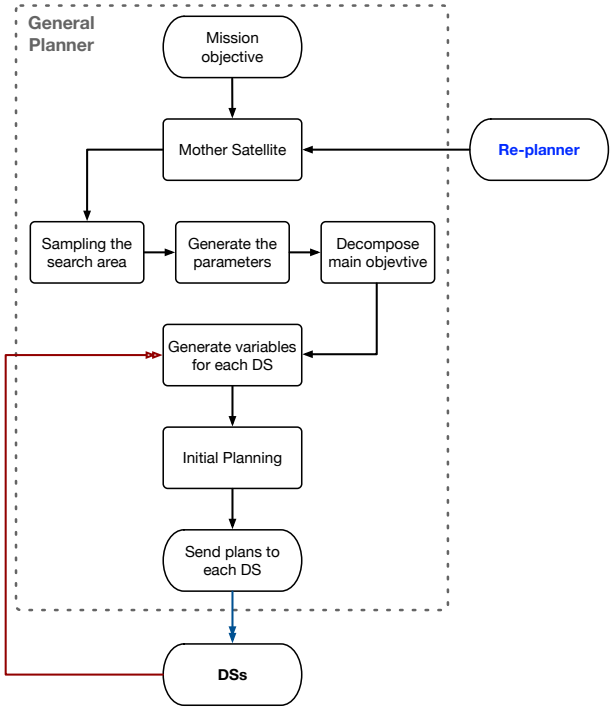


Figure 4.2: Flowchart of the general mission planner

lated as a multi-objective optimization problem.

c. Initial planning
After the decomposition in the previous step, the MS needs to generate feasible initial plans for all DSs. In this chapter, the system status of all DSs is assumed to be identical. Then, the GP implements the proposed HDMGA from the previous chapter to perform initial planning and generate optimal control sequences for all DSs.

4.2.2. DS MISSION EXECUTOR AND MONITOR

The flowchart of these two modules is illustrated in Fig 4.3. Based on the control sequences received from the MS, each DS can perform these preplanned tasks independently. To ensure safety, a self-check procedure is required to define the current status of each agent. A Beacon Signal (BS) is used during self-check, which carries the basic information about the health status of the corresponding DS. If the MS identifies the BS which contains the unhealthy condition from any DS, the onboard planning system will request the decision-maker to perform a detailed check with this DS. The executor can proceed based on the BS signals from DS with healthy conditions. The monitor has the ability to monitor the performance of the plan execution. When any error is detected by onboard sensors, the monitor can stop the current process and request for a re-do of the current scheme. Once all plans have been executed, the monitor needs to check whether

the sub-goal assigned to it has been reached or not. If necessary, the monitor can request the general mission planner to provide new plans according to new objectives.

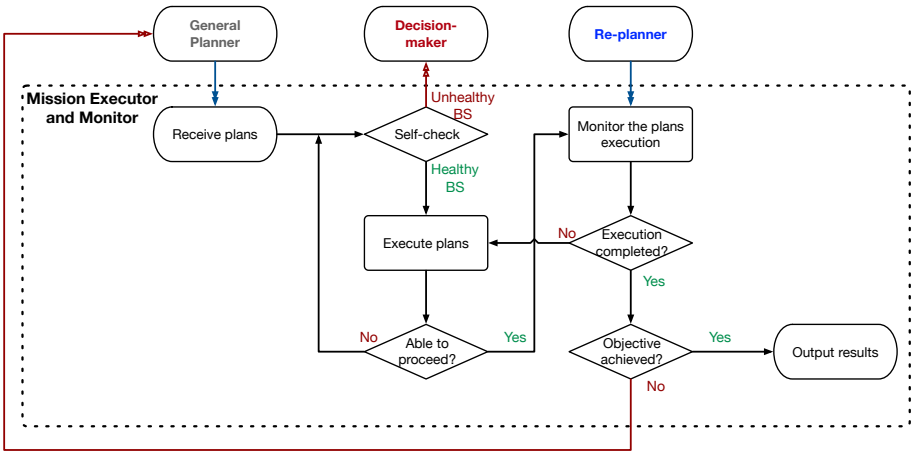


Figure 4.3: Flowchart of the mission executor and monitor for each DS

4.2.3. DECISION-MAKER

The decision-making process is activated by the BS signal which contains the unhealthy content sent from a DS. The BS only contains simple information about whether this DS is healthy or not. Therefore, to fully analyze the current status of this DS, an emergency communication channel between this unhealthy DS and the MS needs to be established immediately, while other DSs continue their work. Through this channel, the MS can receive detailed information from the target DS to allocate the system error of this DS.

To design this decision-maker, the priority is to define the possible emergency scenarios which the system can handle. During mission operations, either situation whether it is the initiative to change of a mission objective, or a DS failure can render the current mission plans obsolete. For these reasons, in this chapter, we take three emergency situations into account, expressed by **E**. This set consists of four elements: (1) Time when an emergency occurs T_e, its unit is second. (2) The emergency scenario set $es = \{A, B, C, ...\}$ represents the scenarios the decision-maker is facing. It can be one of emergency scenarios, or a combination of different scenarios. (3) The position array for dysfunctional satellites **n**. This array stands for the dysfunctional satellites' positions in the MSS. For example, $\mathbf{n} = [1, 2, 5, 7]$ means that DS1, DS2, DS5, and DS7 are damaged. (4) The functional array **p**. Each number represents the percentage of the functionality situation for the corresponding dysfunctional DS, where 100% means perfectly functional and 0% means totally dysfunctional. The same idea has been used in [Barua and Khorasani, 2011]. Combining all four elements, an emergency situation is characterized as: $\mathbf{E} = \{T_e, es, \mathbf{n}, \mathbf{p}\}$. For the DSL mission, observing from lunar orbit to collect raw data and send these data back to Earth are the two most important aspects that need to be considered. Therefore, the emergency situations we handle in this chapter are failures in the observation and communication functionalities. Based on these, three possible

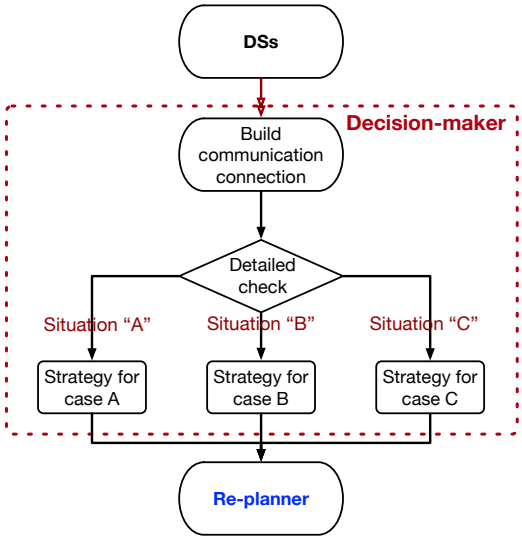


Figure 4.4: Flowchart of the decision-maker

scenarios are assumed:

(i) Scenario *A*
In space, there are plenty of ways a spacecraft can get dysfunctional. Although in lunar orbit, there are much less man-made space debris objects than in earth orbit, other influences such as radiation or micro-meteoroids can cause a damage to the fleet. Therefore, the first scenario we need to consider is facing the downsizing of the fleet. The number of the dysfunctional DSs can range from one to eight.

(ii) Scenarios *B* and *C*
Besides for external influences which can cause fatal failures of satellites, the internal errors can also cause malfunctions for different sub-systems. Here, observation and communication malfunctions are two main aspects we consider. Therefore, we assume that during operations, the observation equipment and/or the communication equipment may face partial-failures and become semi-functional due to a system error such as circuit aging or other factors. In these two scenarios, the number of reduced-functional satellites can range from one DS to eight DSs. The functional percentage can range from $0\% - 100\%$ to represent different levels of the damage. Scenario *B* represents partial-failure of the observation system, and *C* represents partial-failure of the communication system.

Although here only 3 scenarios have been defined, the concept and architecture of this planning and re-planning system is generic and can be used for other scenarios. Based on these assumptions for emergency situations, the decision-maker has to react to these three sample scenarios. When the MS builds the communication link with the target DS, it will trigger the self-check. The flowchart of the decision-maker is illustrated in Fig 4.4.

4.2.4. RE-PLANNER

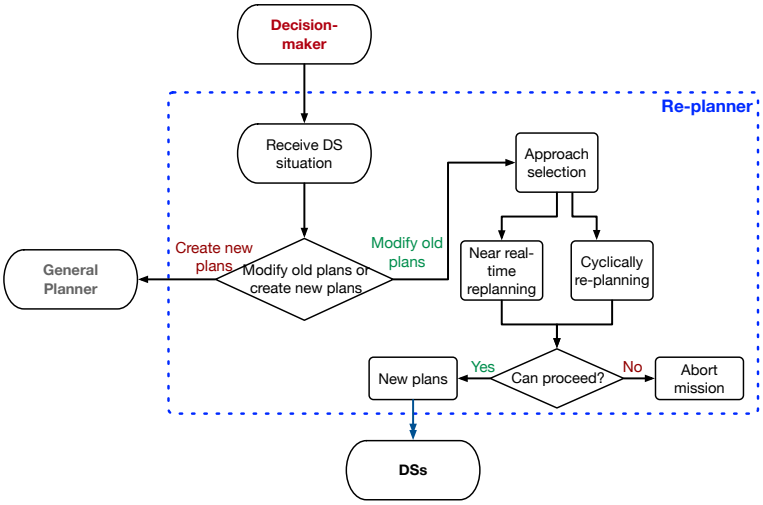


Figure 4.5: Flowchart of the re-planner

Based on the decision made by the decision-maker from the previous step, the re-planner will receive a signal from the decision module about which situation it is facing. Then the first step is to evaluate whether this request is within the cognitive database. This database is pre-designed to target possible emergency situations. In this thesis, we only consider the three scenarios previously introduced. Any case beyond those three cannot activate the re-planning procedure due to the knowledge limitation of the MS. Meanwhile, during operations, new objectives can also trigger the re-planner. Different emergency scenarios combined with different numbers of dysfunctional satellites can lead to very diverse re-planning cases. For some cases, by changing the old plans' order, or inserting and deleting plans, new valid plans can be establish. For other situations, based on new objectives or a change of operation environments, the re-planner needs to create new plans without consulting the initial plans. Therefore, during this evaluation, the re-planner needs to choose whether to modify old plans or generate new plans.

The differences between initial planning and re-planning are the total number of plans that need to be handled and how to manage them. In initial planning, the system needs to generate all mission plans for each DS according to the main mission objective. In contrast, in re-planning, the first option is always trying to modify the current plans instead of creating new plans. For the DSL mission, considering its unique observation constraints, all the satellites need to keep communication silent when satellites are within the observation window. This causes traditional re-planning methods unable to handle this situation. Based on this unique constraint, we propose two re-planning methods, each with its own principles, and each approach corresponding to one preference. The flowchart of the re-planner is shown in Fig 4.5. The details of these two methods will be explained in the next section.

4.3. RE-PLANNING METHODS

Considering the unique requirements for the DSL mission, along with emergency situations which can occur during mission operations, effective re-planning methods which can plan for common tasks and re-planing for emergency situations are needed.

This section proposes two re-planning methods, called the Cyclically Re-planning Method (CRM) and the Near Real-time Re-planning Method (NRRM), respectively. Both methods consist of a core planning algorithm which has been used in Chapter 3 to perform the onboard initial planning. The differences between these two methods are the strategies of selecting the re-planning time point and choosing a number of unfinished tasks during the re-planning process. The efficiency and accuracy of these two approaches are different when targeting different scenarios under different time constraints. Each method has its own strengths and weaknesses, so the specific choice of the method depends on the user requirements.

4.3.1. CORE ALGORITHM

In this chapter, we employ the HDMGA as the core algorithm for our re-planning methods. The HDMGA consists of two mutation operators. These two operators work independently from each other during the mutation procedure. One operator is for the normal mode, the other is for the escape mode. For the normal GA, there exists a termination generation number T_G, which means that if the average value of fitness function has not changed over several generations T_G, the algorithm will stop. Within HDM, there is another condition T_N which is the trigger to switch the algorithm between the normal mode and the escape mode. Until the switch criteria has been reached, the algorithm uses the normal mutation rate R_{nor}. Once T_N satisfies the switch criteria, the algorithm will use the escape mutation rate R_{esc} to try to jump out of the local minima. To expand the search area, the mutation rate in escape mode is larger than the rate in normal mode.

4.3.2. CYCLICALLY RE-PLANNING METHOD

PRINCIPLE

As the name implies, this cyclically re-planning method (CRM) is executed once per-orbit to perform mission planning and re-planning for the entire team. Instead of using GP to perform initial planning for the entire operation lifetime at the beginning of the mission, the CRM will repeat the single orbit planning procedure at the beginning of each orbit using the latest system status of all DSs and the old plans from the previous orbit. There are several advantages for using this method: Firstly, for some observation missions, the entire operation lifetime can be a very extended period. During each orbit, each DS requires four variables to control observation and communication tasks (as explained in Chapter 3.2), which are the start and end time points for each of the task. With the long operation time, the total number of variables of the planning algorithm will be huge. Using CRM at the beginning of each orbit can reduce the planning variable number and make it a constant due to the fixed number of satellites. Secondly, considering the large scale of variables, the dimension of the constraint matrix for the re-planning problem of all orbits will be much larger than the one for the single orbit re-planning problem. Therefore, employing CRM can reduce the complexity of solving the optimiza-

tion problem at each orbit, which can reduce the computation time. Thirdly, when facing different emergency scenarios, the CRM needs to establish the communication link between the MS and each DS at the first possible communication time (both MS and DS are out of the communication window). This can eliminate the problem that initial time sequences in the MS are occupied by establishing the emergency link with the damaged DSs. Finally, under the normal scenario, the old plans for previous orbits can be directly used by the CRM while only changing the variables' boundaries. When the re-planner faces emergency scenarios, the new model can be modified based on the previous normal situation model and it can perform re-planning based on the current status of the functional DSs. All these advantages render the CRM a fast and easy re-planning method for operations.

During the re-planning produce, the steps of using this method are as follows:

(1) At the end of each orbit, the MS needs to activate the re-planning procedure by evaluating the status of each DS through the BS. During operations, the BS is sent every certain period which is defined by the user. However, due to the observation window (OW) constraints, all the DSs need to keep communication silent during that time, including the BS. Therefore, when the MS receives an unhealthy BS from the DS which just exits the OW, this signal only indicates that some abnormal events have happened during this period. It cannot provide the MS the exact time when this abnormal event occurred. In this case, the MS has corresponding strategies for different time intervals at which the emergency situations may occur.

(2) Once the emergency scenario has been determined from the last step, the MS has to evaluate how many DSs are still fully or partially functional. Meanwhile, several important parameters need to be updated according to re-planning time, such as the onboard memory $M_{S_i}(n_k)$ for the S_i at the orbit n_k, data observation rate $Obs_{S_i}(n_k)$, and the communication rate $Com_{S_i}(n_k)$.

(3) Based on the DSL mission requirements, basic parameters such as satellite number, baseline, orbit altitude, initial settings for observation and communication systems can be determined. The new array of variables and new constraint matrix need to be re-assigned according to the new model. Then, based on the specific scenarios, the algorithm will follow predefined strategies to perform a re-planning for the next orbit.

The pseudo-code of the CRM is shown in Algorithm 3.

4.3.3. NEAR REAL-TIME RE-PLANNING METHOD

PRINCIPLE

Unlike CRM, the near real-time re-planning method (NRRM) follows a different strategy to perform mission re-planning. In the previous method, in order to reduce the total number of variables and the complexity of the constraint matrix, we choose to plan the satellite tasks at the beginning of each orbit. The NRRM, on the other hand, uses the strategy which only reacts to the emergency scenarios once they occurred, and perform re-planning for the unfinished mission objective. However, the NRRM has the same hard constraint as the CRM, which is, during the observation all satellites need to keep silent.

Algorithm 3 Cyclically re-planning method

Input: daughter satellite number N, objective of the orbit number M, cyclically operation period T_O, observation window **OW**, communication window **CW**, emergency situation **E**, observation rate Obs_{S_i}, and communication rate Com_{S_i}
Output: Task plans **X**

function $\mathbf{X} \leftarrow CRM(N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i})$
  counter = 1
  while $counter \leqslant M$ **do**
    for $i = 1 : N$ **do**
      if $BS_i = healthy$ **then**
        $\mathbf{E}_{S_i} \leftarrow$ **null**
      else
        $\mathbf{E}_{S_i} \leftarrow \{T_e^{S_i}, es, \mathbf{n}_{S_i}, \mathbf{p}_{S_i}\}$
    switch es **do**
      case $es = 0$ ▷ *Normal situation*
        $\mathbf{x} \leftarrow function(HDMGA\{N, M, T_{Orbit}, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i}\})$
      case $es = 1$ ▷ *Scenario A*
        $[Num, 1] = size[find(E \neq 0)]$
        $N \leftarrow N - Num$
        $\mathbf{x} \leftarrow function(HDMGA\{N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i}\})$
      case $es = 2$ ▷ *Scenario B*
        $\mathbf{Obs}(n_{counter}) = \mathbf{Obs}(n_{counter} - 1) * \mathbf{p}$
        $\mathbf{x} \leftarrow function(HDMGA\{N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, \mathbf{Obs}(n_{counter}), \mathbf{Com}(n_{counter})\})$
      case $es = 3$ ▷ *Scenario C*
        $\mathbf{Com}(n_{counter}) = \mathbf{Com}(n_{counter} - 1) * \mathbf{p}$
        $\mathbf{x} \leftarrow function(HDMGA\{N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, \mathbf{Obs}(n_{counter}), \mathbf{Com}(n_{counter})\})$
    $N_{counter} \leftarrow N - f(\mathbf{X})$
    $\mathbf{x}_{counter} = \mathbf{x}$
    $counter = counter + 1$
  $\mathbf{X} = [\mathbf{x}_1; ..., \mathbf{x}_M]$
  return X

Therefore, the reaction to the emergency scenarios may not occur in real-time. This is the reason why this method is called near real-time re-planning method.

The strengths of the NRRM are as follows: Firstly, although the re-planning procedure of the NRRM requires more computation resources than the CRM because the number of variables for the NRRM is larger than the CRM, the overall re-planning times for the NRRM is much lower than the CRM. This is because the CRM will re-plan for each orbit, while the NRRM only performs re-planning if necessary. Therefore, the overall computational resource consumption of the NRRM can be much less than the CRM when facing a small number of emergencies. Secondly, the NRRM can react to emergency scenarios much quicker than the CRM. Although based on the main scientific requirement that during the OW all the satellites need to keep silent, the NRRM can rely on the positions of dysfunctional satellites and their situation parameters to decide when to activate the re-planning procedure. Therefore, this method is called a near real-time method. This means that for specify scenarios, the MS can react immediately, while for other scenarios, the MS needs to determine when to perform re-planning without jeopardizing the observation tasks. Thirdly, since the whole mission task sequences for every DS have been planned during initial planning, under any emergency scenario, the NRRM can react based on the initial plans. During the re-planning, some re-plan techniques such as deleting, inserting, and modifying can be employed by the NRRM, which will accelerate the re-planning procedure.

The steps of using the NRRM for re-planning are as follows:

Algorithm 4 Near real-time re-planning method

Input: daughter satellite number N, objective of the orbit number M, cyclically operation period T_O, observation window **OW**, communication window **CW**, emergency situation **E**, observation rate Obs_{S_i}, and communication rate Com_{S_i}
Output: Task plans **X**

function $X \leftarrow NRRM(N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i})$
  $\mathbf{E} \leftarrow$ **null**
  $\mathbf{X}_{initial} \leftarrow function(HDMGA\{N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i}\})$ ▷ *Initial planning*
  for $i = 1 : T_O * M$ **do**
    if $E_i \neq 0$ **then**
      $\mathbf{E}_{S_i} \leftarrow \{T_e^{S_i}, es, \mathbf{n}_{S_i}, \mathbf{p}_{S_i}\}$
    $[a, b] = size(\mathbf{n}_{S_i})$
    if $Pos_{MS} < \mathbf{n}_{S_i}(a)$ **then** ▷ *Error Detection*
      $r_t = Outtime(S_a)$
    else $Pos_{MS} > n_m(a)$
      $r_t = Outtime(MS)$
  switch es **do**
    case $es = 0$ ▷ *Normal situation*
      $\mathbf{X} = \mathbf{X}_{initial}$
    case $es = 1$ ▷ *Scenario A*
      $\mathbf{X}_{initial} \leftarrow deleting\{es, \mathbf{n}\}$
      $\mathbf{X} \leftarrow function(HDMGA\{\mathbf{X}_{initial}, N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i}, r_t\})$
    case $es = 2$ ▷ *Scenario B*
      $\mathbf{X}_{initial} \leftarrow modifying\{es, \mathbf{n}, \mathbf{p}\}$
      $\mathbf{X}_{initial} \leftarrow inserting\{T_e^{S_i}, es, \mathbf{n}, \mathbf{p}\}$
      $\mathbf{X} \leftarrow function(HDMGA\{\mathbf{X}_{initial}, N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i} * p_{S_i}, Com_{S_i}, r_t\})$
    case $es = 3$ ▷ *Scenario C*
      $\mathbf{X}_{initial} \leftarrow modifying\{e, \mathbf{n}, \mathbf{p}\}$
      $\mathbf{X}_{initial} \leftarrow inserting\{r_t, e, \mathbf{n}, \mathbf{p}\}$
      $\mathbf{X} \leftarrow function(HDMGA\{\mathbf{X}_{initial}, N, M, T_O, \mathbf{OW}, \mathbf{CW}, \mathbf{E}, Obs_{S_i}, Com_{S_i} * p_{S_i}, r_t\})$
  return X

(1) After the initial planning, the NRRM keeps low power operations, which only monitors the health condition of each DS.

(2) Once an emergency occurred, the NRRM needs to react differently based on the specific situation, determining the key parameters which include the start time of the re-planning procedure, the current system status of each DS, and the achieved mission status.

(3) Then, targeting different scenarios, the NRRM will save the part of current plans for healthy DSs to reduce unnecessary time consumption on these satellites. Meanwhile, the core algorithm will be based on the start time of the re-planning and other parameters to re-plan all the tasks for the rest of the mission.

The corresponding pseudo-code is shown in Algorithm 4.

4.4. SIMULATIONS AND ANALYSIS

To analyze the performance of the two re-planning methods, multiple simulation cases have been designed based on the emergency scenarios introduced before. Each case represents a unique emergency scenario. For each case, we apply two re-planning methods (CRM & NRRM) to perform mission re-planning separately. Furthermore, we analyze their performance based on several criteria such as computation time, completion of the mission objective, and efficiency value.

4.4.1. SIMULATION ASSUMPTIONS

SIMULATION ENVIRONMENT

The DSL mission will operate in a lunar orbit to prevent RFI from the Earth. Based on the scientific mission requirements, some basic parameters about the simulation environment are following the same assumptions as in Chapter 3.4. Meanwhile, since the purpose of this section is to test whether the proposed re-planning methods can be qualified when facing various emergency situations, the total number of operational orbits is set to be $M = 4$.

SIMULATION CASES

The simulation cases are adopted from three different scenarios introduced before. For each case, we need to provide the emergency situation set $\mathbf{E} = \{T_e, es, \mathbf{n}, \mathbf{p}\}$ for the re-planner to describe an emergency case. In order to test the performance of each method, the simulation environment for the same case will stay the same. Meanwhile, to verify the diversity of the real situation, each case contains different information.

- Case A
  This case aims to simulate an example of the scenario A, which represents a complete malfunction of several DSs during the mission. To simulate this situation, we assume that the emergency occurred at the time when the MS was inside the OW. Based on the orbit altitude, the orbit period T_O can be calculated by Eq 3.1 and Eq 3.2. The emergency time T_e is set to be 12000 seconds after mission started. Meanwhile, to increase the diversity of this case, we assume that three DSs are dysfunctional, two of them are ahead of the MS, and another is behind the MS. The emergency situation set can be written as $\mathbf{E}_A = \{T_e = 12000, es = 1, \mathbf{n} = [1,3,5], \mathbf{p} = [0\%, 0\%, 0\%]\}$, where $es = 1$ stands for the case A, $\mathbf{n} = [1,3,5]$ stands for the number of the dysfunctional DSs (DS1, DS3, and DS5), $\mathbf{p} = [0\%, 0\%, 0\%]$ indicates that all of the damaged DSs are totally dysfunctional.

- Cases B & C
  These two cases are two examples to simulate the scenario B and C, which represent a partially dysfunctional situation related to either their observation or communication functionality. Following the same approaches as in case A, the emergency time is set as same as the previous case. The number of partial-failure DSs are four, two in front of the MS and two behind the MS. The functional percentage of each DS is different. We assume the emergency sets for cases B and C are $\mathbf{E}_B = \{T_e = 12000, es = 2, \mathbf{n} = [2,4,6,8], \mathbf{p} = [20\%, 40\%, 60\%, 80\%]\}$ and $\mathbf{E}_C = \{T_e = 12000, es = 3, \mathbf{n} = [1,3,5,7], \mathbf{p} = [20\%, 40\%, 60\%, 80\%]\}$, respectively. All representations are the same as explained in case A.

4.4.2. CRM SIMULATION

Based on previous assumptions, we first employ CRM to solve all three cases.

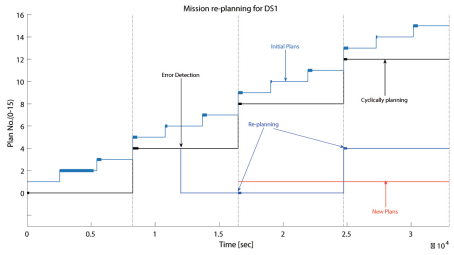
- Case A
  The simulation results when using CRM to solve case A are shown in Fig 4.6. This figure contains eight sub-figures representing task sequences for all eight DSs. The

blue line with blocks on the top represents initial plans made by the MS based on the mission objective. Each block stands for a task operation period. The black line represents the mission cyclically re-planning procedure under the normal situation. The red line with blocks stands for new plans made by the CRM, each block represents a new task operation period. The bottom blue line is the re-planning procedure line when the emergency has been detected. The vertical gray dash line separates four orbits of the operation lifetime. The X-axis stands for the mission elapsed time which starts from 0. The Y-axis represents the plan number with the range from 0 to 16, which helps to distinguish each task.

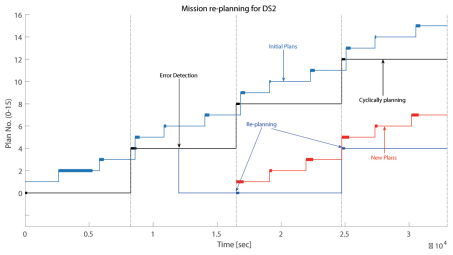
To explain the results in this figure, several aspects need to be mentioned. The black block line illustrates the strategy of the CRM to re-plan at the beginning of each orbit. All the sub-figures contain this line to show that every DS needs the CRM to provide plans during each orbit. Based on the assumptions, this emergency will occur at 12000 seconds after the mission has started. This time point has been marked on the black line. However, due to the core concept of the CRM, the re-planning procedure after this emergency is triggered at the beginning of next orbit only, represented by the blue line at the bottom. This case assumes that only three DSs are dysfunctional. The dysfunctional set is $\mathbf{n} = [1,3,5]$. These three satellites share the same condition, of which re-planning results are illustrated in Fig 4.6a, Fig 4.6c, and Fig 4.6e, respectively. These figures show that after the emergence occurred, all the plans for DS1 have been deleted, as well as for DS3 and DS5. The total number of operable satellites is decreased due to the damage. Therefore, all operable DSs (in this case are: DS2, DS4, DS6, DS7 and DS8) have longer communication times with the MS, as shown in Fig 4.6b, 4.6d, 4.6f, 4.6g, and 4.6h, respectively. The increased communication time for these DSs also provides them extra storage capacity for a longer observation time. The new plans in the figures of these DSs' show that the periods for both communication tasks and observation tasks are indeed increased.

- Case B
  Following the assumptions made for case B, the simulation results are shown in Fig 4.7. The meaning of each line is the same as in case A, along with the explanation of the X- and Y-axes. The partial non-functionality on observation tasks only influences the observation duration of certain DSs, while the old plans of observation tasks for other healthy DSs are not influenced since they already reach their maximum capability. Therefore, this figure only contains the re-planning information about the satellites (DS2, DS4, DS6, and DS8) showing partial-failure. The following aspects explain the simulation results shown in this figure.

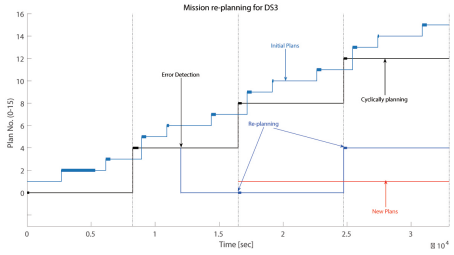
  This case is about the reduced functionality of the observation tasks. It leads to a direct impact on the data observation rate Obs_{S_i} of each DS. With the same amount of onboard storage capacity, lower observation rate means longer observation time if it is inside of the OW. In this figure, the representations of the boxes and the lines are the same as for case A. Comparing the new plans (red line) with the initial plans (blue line) in each sub-figure, the re-planned observation task blocks are longer than the initial blocks. The functional matrix **p** in this case influ-



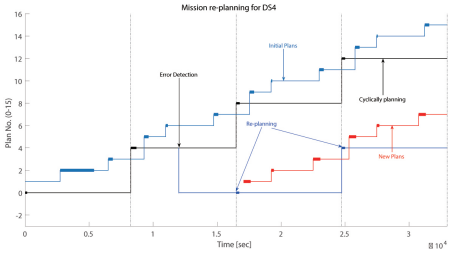
(a) DS1 in Case A



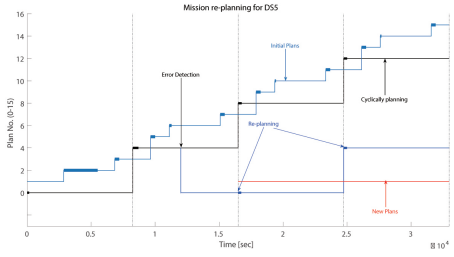
(b) DS2 in Case A



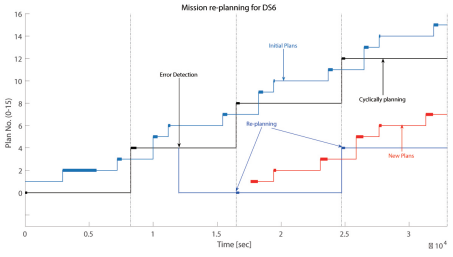
(c) DS3 in Case A



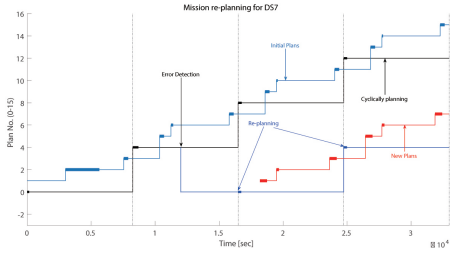
(d) DS4 in Case A



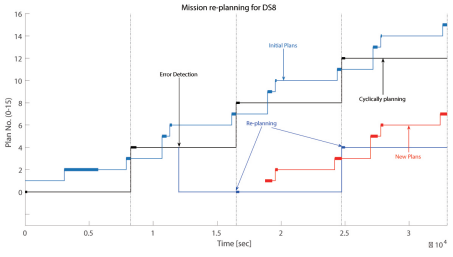
(e) DS5 in Case A



(f) DS6 in Case A

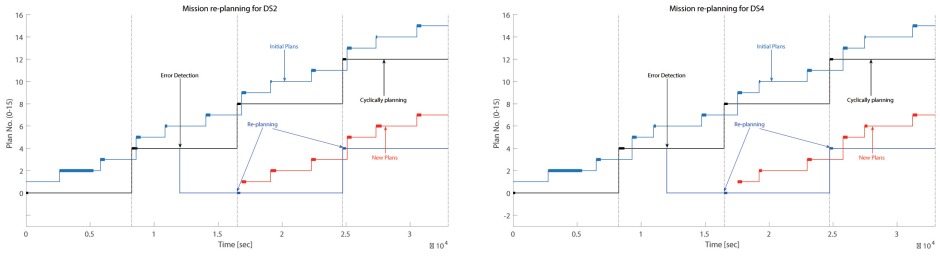


(g) DS7 in Case A



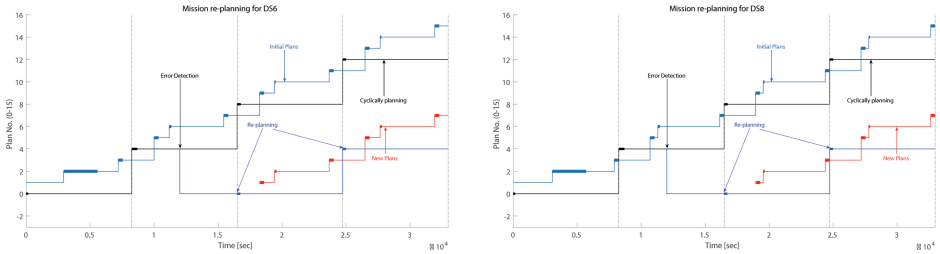
(h) DS8 in Case A

Figure 4.6: Re-planning results for Case A using the CRM. For all figures, at each orbit, the middle box stands for the observation task, and the earlier and later boxes stand for the communication tasks. Gray dash lines divide operation time into four orbits. For the first orbit, since there is no data to communicate, there is no communication task in front of the observation task.



(a) DS2 in Case B

(b) DS4 in Case B



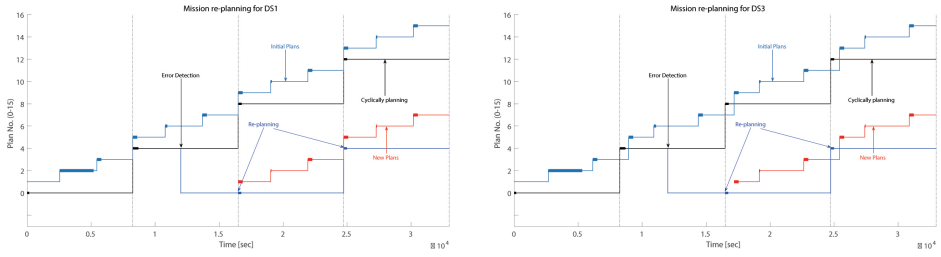
(c) DS6 in Case B

(d) DS8 in Case B

Figure 4.7: Re-planning results for Case B using the CRM. Only the partially non-functional DSs are shown.

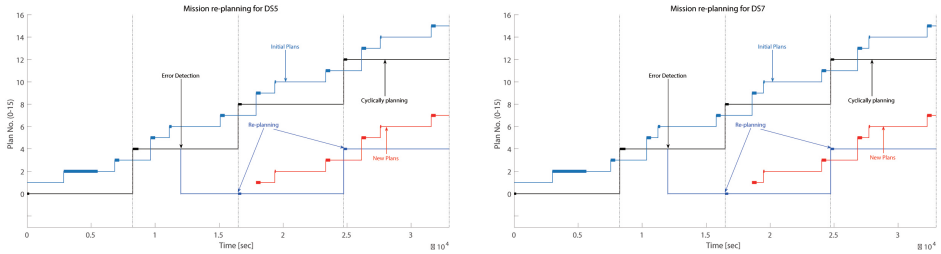
ences the observation functionality of each dysfunctional DS. The percentage in the matrix denotes the current observation functionality of the corresponding DS. For example, for DS2, the functional percentage is 20%, which means that the new observe rate $NewObs_{S_2}$ is only 20% of the normal rate. The observation blocks of DS2 are the longest among all four damaged DSs (as shown in Fig 4.7a), and corresponding blocks for DS8 are the shortest due to its smallest damage (in Fig 4.7d).

- Case C
  In this case, the dysfunctionality on communication tasks of four DSs cannot affect the remaining DSs since the communication window has been equally distributed to every DSs by the MS. With the same amount of DSs, the communication time for the dysfunctional DSs cannot change, which means the healthy DSs cannot take over the unhealthy DSs' communication time. Therefore, we only focus on the four dysfunctional DSs. The results are shown in Fig.4.8, which contains four DSs (DS1, DS3, DS5 and DS7). The results shown in each sub-figure illustrate that communication task blocks in the new plans (red lines) have the same lengths as blocks in the initial plans (blue lines). Due to the malfunction on the communication tasks, with the same communication time, the dysfunctional DSs can transmit less data than in the initial plans. This causes less storage recovery for the next observation orbit. Comparing with the initial plans, observation blocks of the new plans in each DS are much shorter. This indicates less observation time due to the communication malfunction. The re-planning details of dysfunctional DSs are shown



(a) DS1 in Case C

(b) DS3 in Case C



(c) DS5 in Case C

(d) DS7 in Case C

Figure 4.8: Re-planning results for Case C using the CRM. Only partial non-functional DSs are shown.

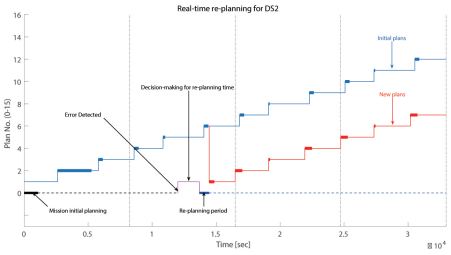
in each sub-figure.

To summarize, it is verified that the concept of the proposed CRM works for every scenario introduced before and can provide valid plans for further mission operations. Meanwhile, with a constant or even fewer DSs for different cases, the CRM can re-plan the mission cyclically in a short computation time due to a smaller number of variables and a smaller dimension of the constrains matrix.

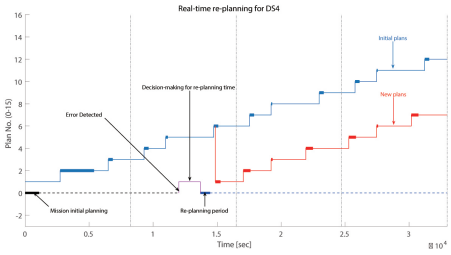
4.4.3. NRRM SIMULATION

In this part, we employ the NRRM to solve all three simulation cases. To compare the performance of CRM and NRRM, we use the same case assumptions. Every simulation shown in this part has the same settings as the previous simulations.

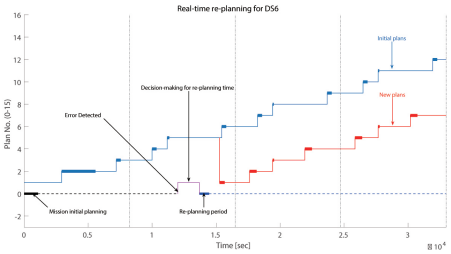
- Case A
  The NRRM has a different re-planning strategy as compared to the CRM strategy. The simulation result for case A is shown in Fig 4.9. In each sub-figure, the NRRM overall planning procedure is represented by the black line at the beginning. The dashed line stands for the system standby. Once an emergency is detected, the NRRM needs to wait to reach the start time of the re-planning, which is indicated by the pink line in each sub-figure. This waiting period is related to the dysfunctional satellite's position within the fleet. The blue block stands for the re-planning procedure and the dashed line after that represents the system standby. The rest of the line follows the same connotation as in the previous figures.



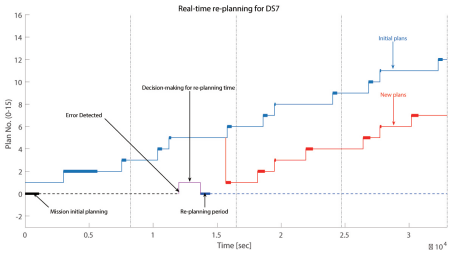
(a) DS2 in Case A



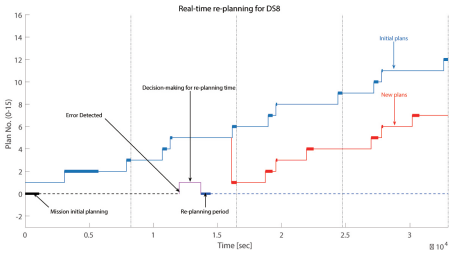
(b) DS4 in Case A



(c) DS6 in Case A



(d) DS7 in Case A



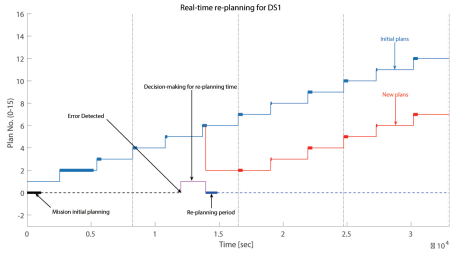
(e) DS8 in Case A

Figure 4.9: Re-planning results for Case A using the NRRM

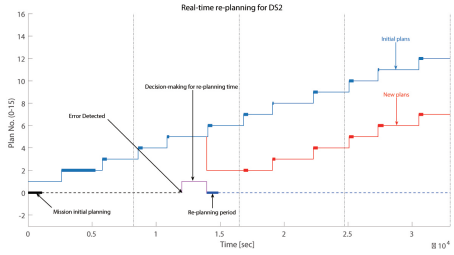
To fully explain this figure, we discuss here the following aspects: (1) Due to the case A, DS1, DS3, and DS5 are eliminated from the fleet after the emergency. Therefore, in this figure, we only present re-planning results of the five healthy DSs. (2) Fig 4.9a shows the results of DS2. The third box of the blue line in the second orbit represents the second communication task in this orbit, which has an overlap with the re-planning period. Based on NRRM, all tasks after the emergency should be suspended and re-evaluated. At the time the NRRM finishes the re-planning, the time window for this communication task has passed. However, after eliminating all dysfunctional satellites, DS2 is re-ranked to the first position within the fleet. Therefore, after the re-planning, this communication task has been replaced and re-calculated according to the number of operable DS. (3) For the DS4 (Fig 4.9b), the same box also represents a communication task in the second orbit, of which operation period in initial plans is behind the re-planning period in the MS and

should be unchanged. However, the total number of healthy DSs in the fleet is decreased, and this communication time for each operable DS is correspondingly increased. The new plan for this communication task indicates that DS4 has to wait for DS2 to finish its communication task first. Only then it can start its own communication task. (4) The results of DS6, DS7, and DS8, are shown in Fig 4.9c, Fig 4.9d and Fig 4.9e, respectively. Due to the team downsizing, all the tasks shown in initial plans have been moved ahead accordingly. (5) For all operable DSs, due to the increasing communication time, the observation time in each orbit after the emergency is also increased. This also happens when using the CRM.

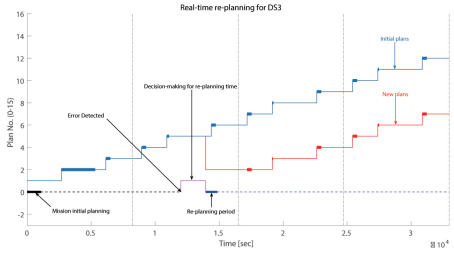
- Case B
  The simulation results are shown in Fig 4.10. Considering the potential influences of the NRRM during the re-planning procedure, we show four dysfunctional DSs



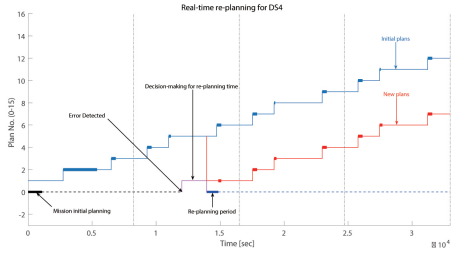
(a) DS1 in Case B



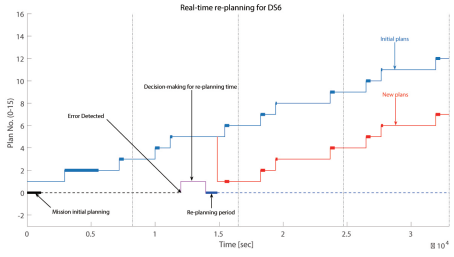
(b) DS2 in Case B



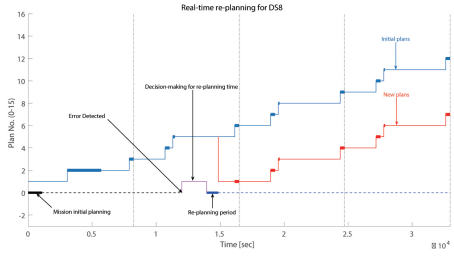
(c) DS3 in Case B



(d) DS4 in Case B



(e) DS6 in Case B



(f) DS8 in Case B

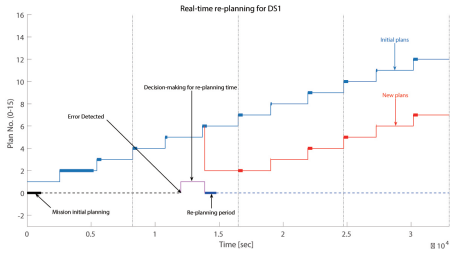
Figure 4.10: Re-planning results for Case B using the NRRM. Only the relevant DSs are shown.

(DS2, DS4, DS6, DS8) and two unharmed DSs (DS1, DS3).

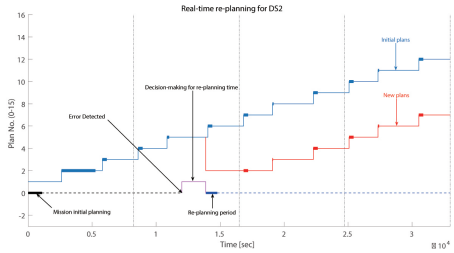
This figure shows the following: (1) Although DS1 and DS3 are unharmed, we can see from Fig 4.10a and Fig 4.10c that the re-planning procedure is occupying the MS when the communication task of these two DSs should occur. These communication tasks require the corresponding DS and the MS to be available for this task at the same time. Therefore, for DS1, the communication task during the re-planning procedure cannot be fully finished due to the occupation of the MS. For DS3, the communication task happens after the re-planning procedure needs to be deleted. (2) Meanwhile, one communication task in this orbit for DS2 is also canceled due to the same reason. For DS4, the initial plan about its communication task in this orbit is delayed and decreased due to the re-planning end time. The sub-figures for DS2 and DS4 show the corresponding phenomena by red lines, as shown in Fig 4.10b and Fig 4.10d, respectively. (3) The reason for increasing the observation time for each damaged DS is the same as the reason when using the CRM. In this figure, new plans for DS2 and DS4 show the increased length of the observation block in the new plan lines, which are the same for DS6 and DS8.

- Case C
  In this case, for the same reasons as in case B, we present six DSs in Fig 4.11. From this figure, several conclusions can be drawn: (1) For the same reason as in case B, the MS is occupied due to the re-planning procedure. Therefore, the communication tasks in the second orbit of DS1, DS2, and DS3 need to be deleted or decreased. The new plans (red line) in Fig 4.11a illustrate that the communication task box is stopped because of this reason. For DS2 and DS3, the results are similar. (2) The new observation tasks of DS1, DS3, DS5, DS7 are shorter than the ones in the initial plans because of the malfunction on the communication equipment of these DSs. The reason causing this phenomenon is the same as explained in case C using the CRM as shown in Fig 4.11c, Fig 4.11e, and Fig 4.11f.

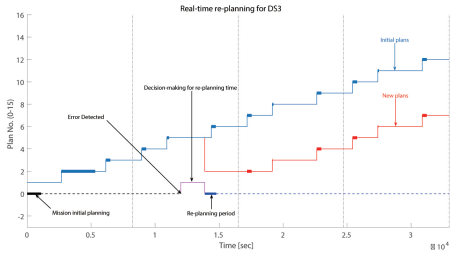
Using these simulations, the concept of this method has been successfully demonstrated to solve the emergency scenarios introduced before. The NRRM has a different re-planning strategy compared to the CRM. More variables and larger dimensions of the constraint matrix, using the NRRM for initial planning lead to longer time than what the CRM spends on planning time in each orbit. When an emergency occurs, the NRRM has to re-plan the rest of the mission in one step, which results in increases on the variable number and constraint matrix. These aspects all lead to a more complex programming process and a longer computation time. However, the NRRM has the unique ability that it can react to the emergency in near real-time, which is faster than waiting for the next orbit as for the CRM. Using the NRRM to solve mission re-planning problems also has the benefit that it can use the initial plans as baselines to implement the re-planning. This can help the re-planner to reduce unnecessary computations. Other performance comparisons between the CRM and the NRRM will be described in the following section.



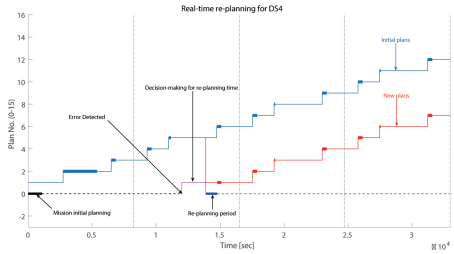
(a) DS1 in Case C



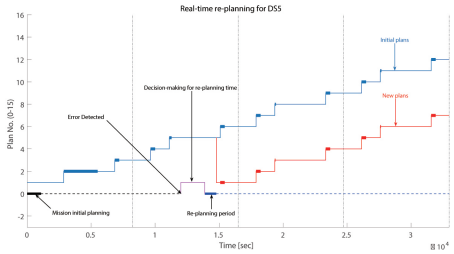
(b) DS2 in Case C



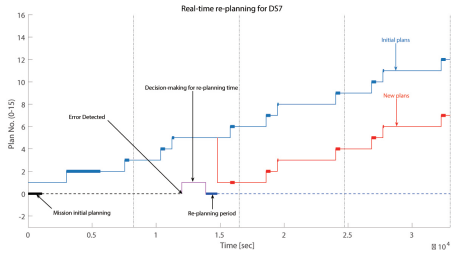
(c) DS3 in Case C



(d) DS4 in Case C



(e) DS5 in Case C



(f) DS7 in Case C

Figure 4.11: Re-planning results for Case C using the NRRM

4.4.4. PERFORMANCE COMPARISON

To compare the performance between CRM and NRRM, we focus on three aspects: the total number of data observed from all DSs within a certain time period, the total number of data the MS received from all DSs within a certain time period, and the average computation time for re-planning in each case.

The histogram Fig 4.12 shows four categories. The first case is a control group representing the normal situation without emergencies. The other three cases stand for simulation cases introduced before. From the normal situation, the NRRM has received 1122 Gbits of raw data, which is 6 Gbits more than the CRM. The reason for this difference lies in their different re-planning strategies. The CRM does the re-planning at the start of each orbit, which occupies the time of a communication task for the DS1. The difference for the normal case between the two methods is 0.53%. This indicates that using the NRRM allows to collect more data than using the CRM. This difference effects all

three cases, making the NRRM to observe more data than the CRM in the same period. The figure also shows that for cases B and C there are similarly small differences. The differences on these two cases are 0.72% and 0.74%, respectively. In Case A, the difference is 3.15%. This is much higher than other cases, which is because when using the NRRM, the re-planner can react within this orbit, which can extend the operable communication time in this orbit.

Fig 4.13 shows the comparison of the number of data received by the MS when using the two proposed methods for each simulation case. The figure illustrates the total data received from all DSs during the mission. From the differences, we can observe: (1) Due to the different re-planning strategies, using the NRRM helps the MS to obtain 3.91% more data than using the CRM. This is because for the normal case, the NRRM can transmit more data than the CRM, for the same reason as what influences the data observation. Each case shows that using the NRRM makes the MS to receive more data than using the CRM. (2) Based on the results, the difference of two methods in case B is 2.66% lower than the difference in the normal case. This is because the NRRM needs time to re-plan the mission, which occupies the time that the MS is supposed to communicate with a DS. Therefore, the number of data received by the MS in case B using the CRM is 108.96 Gbits, which is close to the number under the normal case (110.44 Gbits). (3) The NRRM shows steady performance on both cases A and B. The MS can receive almost the same amount of data (110.28 and 110.33 Gbits). For case C, the amount of data received by the MS drops significantly because of the malfunction on communication equipment of four DSs.

In Fig 4.14, the differences between total computation time of these two methods for re-planning are shown. For the normal case, the NRRM has a shorter computation time due to the one-time planning policy, which is 26.63% shorter than using the CRM. The CRM has to perform the re-planning procedure at each orbit, whether an emergency occurred or not. The NRRM only needs to perform the initial planning at the beginning of the mission. However, besides the normal case, all other simulation cases show that the average computation time spent by the NRRM is longer than the time by the CRM. The

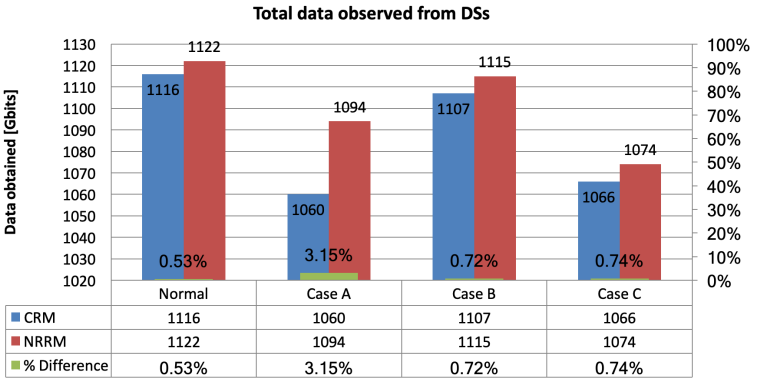


Figure 4.12: The number of data observed for the normal case and the three emergency cases within a certain lifetime

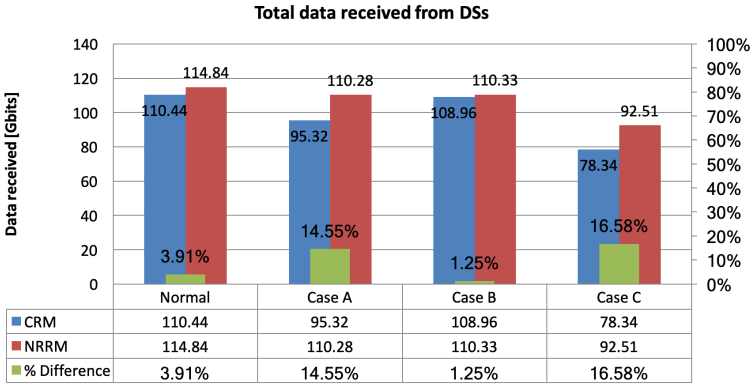


Figure 4.13: The number of data received by the MS for the normal case, and the three emergency cases within a certain lifetime

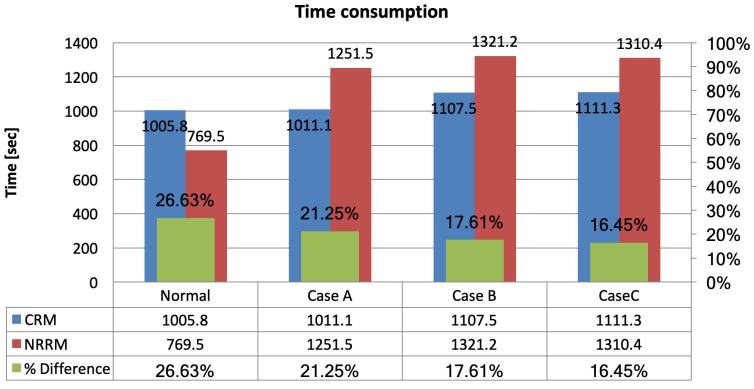


Figure 4.14: Time consumption information in three cases

reason is that the NRRM has more variables and a larger constraint matrix than the CRM. Although the CRM has to perform re-planning many times, for each time the number of variables and constraints is small. This leads to a shorter computation time in total. The differences show that for cases B and C, the differences are at the same level (17.61% and 16.45%). For case A this difference is larger (21.25%) because the NRRM has less variables and constraints due to the loss of three DSs.

From Fig 4.15, we can draw conclusions on the differences in the performance of these two methods: (1) Based on the amount of data observed on DSs and the total amount of data received by the MS, the red and blue lines reveal that NRRM supports observation and transmission of more data than the CRM in a certain lifetime. For each emergency case, the differences follow this pattern. The differences in the differences for each case are caused by the unique situation formed by corresponding simulation cases, such as cases A and C. (2) For all emergency cases, the time consumption reveals that the NRRM requires at least 16.45% more computational time than the CRM. Therefore,

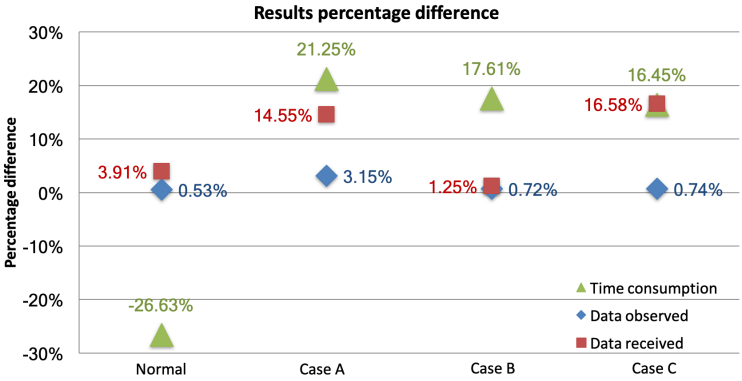


Figure 4.15: Percentage differences of CRM and NRRM on three aspects for various cases

for emergency situations, the CRM can provide re-planning sequences faster than the NRRM. The comparison shows that the NRRM has a superior performance on data observation and transmission and the CRM has a superior performance on the consumption of the computation time for re-planning in these emergency cases.

4.5. SUMMARY

In this chapter, we first develop the model of the onboard mission planning and re-planning problem for a multi-satellite system. Then, on top of the nominal scenario, three emergency scenarios covering partial or complete failures of spacecraft in the fleet are introduced. This is followed by the general architecture of the planning and re-planning system, which helps to realize functions of four sub-systems, including the general planner, the decision-maker, the re-planner, and the mission executor and monitor. To handle the introduced three emergency scenarios, two re-planning methods are proposed. These two methods use the same core optimization algorithm which has been proposed in the previous chapter. The difference of two methods lies in their re-planning strategies. The CRM performs re-planning at the beginning of each orbit, and it only re-plans for one orbit. The NRRM performs re-planning in a near real-time setting when the emergency occurs. Its re-planning is for the rest of the mission. The comparison between these two methods on three study cases indicates that the CRM has the advantage of saving computation resources, while the NRRM has the advantage that it allows to observe and transmit more data within the same lifetime. Both methods show their adaptability and flexibility when dealing with different cases. Both of them can successfully provide feasible new plans for various emergency scenarios.

In the next chapter, we aim to conduct further simulations for another realistic situation, which is the malfunction on the MS. Under this circumstance, the MSS will lose the main controller and the proposed centralized planning and the re-planning methods presented in this chapter are invalid. In such a scenario, all operable DSs have to cooperate to takeover the responsibility of the MS. Therefore, team cooperation and negotiation will be discussed in the next chapter.

5

TEAM NEGOTIATION

Parts of this chapter have been published in Z.Zheng, J.Guo, E.Gill, *Onboard Mission Allocation for Multi-Satellite System in Limited Communication Environment*, Aerospace Science and Technology **79**, 174-186 (2018)

5.1. INTRODUCTION

In Chapter 3 and 4, the MSS onboard mission planning & re-planning problems were investigated through a centralized control model where the MS acted as the main controller and all the DSs were only participants in this system. When the MS is functional, it obtains essential information (e.g. healthy condition, sub-system status) from each DS and performs onboard planning & re-planning by using centralized optimization algorithms. Due to its omniscient ability, the MS can decompose the main mission objective into several sub-goals and allocate them to the suitable DSs. This mission decomposition problem is called Multi-satellite Mission Allocation (MSMA) problem. It can be seen as an optimal assignment problem where the objective is to optimally assign a set of subgoals to a set of satellites in a way that optimizes the overall system performance. Many researchers have done studies on similar fields such as the Multi-robot Task Allocation (MRTA) problem [Lerman et al., 2006; Tang and Parker, 2007], the Sensor Networks (SN) deployment problem [Howard et al., 2002; Mini et al., 2014], the weapon-target assignment problem [Murphey, 2000], and many other problems related to unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned underwater vehicles (UUVs).

Basically, three types of approaches have been used to solve the previous problems. The *Centralized approaches* [Koes et al., 2006; Semsar and Khorasani, 2007] are the most widely used methods, which require one central controller to determine the mission assignment for each team member. The centralized approaches have the main advantage of performing optimization based on the overall objective function, which leads to a solution to be optimal or near-optimal. However, the centralized approaches also suffer from several weaknesses. Firstly, centralized approaches are strongly depending on the main controller, which makes them very vulnerable to its failure. Secondly, they require steady communication between the main controller and the participants, which causes limitations on communication and mission coverage. Thirdly, the demand on computational power of the main controller is high. As mentioned in Chapter 2.1, this thesis focuses on providing the onboard mission planning and re-planning capability for the DSL mission. The original design of this mission is to employ the centralized architecture by using the MS as the main controller. This controller responds to mission objectives received from the ground station, gathers the system status from all daughter satellites, and generates the sequential plans for each DS. In Chapter 4, we took three emergency situations of the DSs into account and formulated several potential cases for the proposed re-planning methods. Under these emergency situations, the DSs suffer from different kinds of system failures, such as nonfunctional or partially functional observation or communication capabilities. However, in a real-world scenario, the MS as well faces risks on being completely or partially nonfunctional. In this case, the MSS will lose the main controller, while any of the daughter satellites cannot perform the centralized planning for the whole fleet due to the design limitations. That is, DSs were never designed to support the MS failure. Under this circumstance, the MSS will lose the central controller, which makes centralized approaches unusable. To overcome this situation, daughter satellites are forced to break the original centralized topology and construct a new topology. It can be either a distributed or a decentralized architecture based on the baseline between the daughter satellites and the efficient communication

range.

Therefore, from this chapter on words, we investigate non-centralized approaches for onboard mission planning. Unlike centralized approaches, non-centralized approaches require cooperation and negotiation between participant DSs, which are treated as agents in the network. Many of the earlier works that have been reviewed in Chapter 2 are focused on the analysis of the mission allocation problem under a stable operating environment. However, in our study, we also cover unpredictable situations during operations, where the MSS needs to adjust cooperation and negotiation mechanisms based on different conditions. In this chapter, we propose several negotiation and cooperation mechanisms for the reference MSS to accomplish onboard mission allocation problems.

5.2. MULTI-SATELLITE MISSION ALLOCATION PROBLEM

An MSMA problem can be divided into three sub-problems: firstly, how to decompose the global goal into several sub-goals; secondly, how to construct the organizational architecture based on different mission requirements; and thirdly, how to assign the sub-goals to each satellite through the negotiation and cooperation mechanisms. Solving a mission allocation problem is a dynamic decision making procedure. It should be solved iteratively over time considering the changes of self-status or mission environment. Thus, choosing a suitable problem model and an adequate organizational architecture can lead to a more realistic solution.

5.2.1. ORGANIZATIONAL ARCHITECTURES

The organizational architecture provides a framework for activities and interactions between participant agents through the definition of characteristics, authority relationships and communication links [Durfee et al., 1999]. In principle, the topology of the MSS usually follows one of three organizational architectures: the centralized architecture, the distributed architecture, and the decentralized architecture.

- Centralized architecture
  As shown in Fig 5.1, all sub-tasks need to be determined by the main controller.

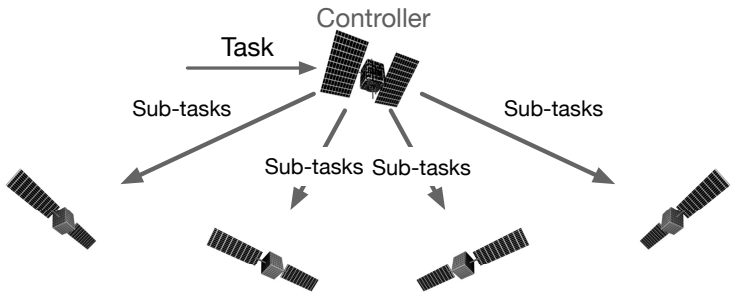


Figure 5.1: The centralized architecture

Its advantage is the controller can observe and gather information from all team

members and then generate suitable solutions for them. However, lacking of robustness is one of the most severe drawbacks of the centralized architecture. Once the main controller is dysfunctional due to either internal system errors or external environment changes, the whole system will fail immediately. Meanwhile, the scalability of the problem is restricted because all the agents are connected to the main controller. This is considered as a bottleneck for the whole system regarding the high demands from the main controller on both computation and communication capability. Thus, for an MSMA problem with a few participants and easy operating environment, the centralized architecture is the most-suited [Al-Yafi et al., 2009].

- Distributed architecture
  As shown in Fig.5.2, all the agents in the distributed architecture have to interact

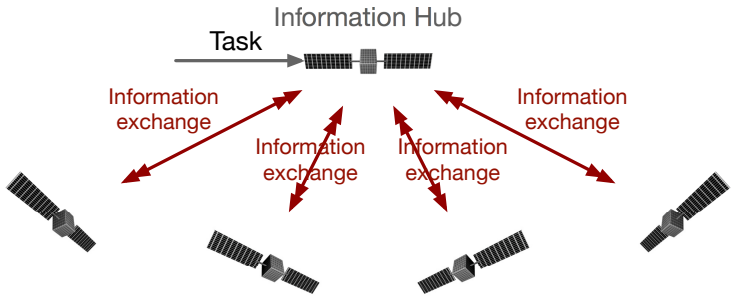


Figure 5.2: The distributed architecture

with others to achieve a common goal. There exist either several stable information-exchange channels or an information hub to help connect all agents. Using this architecture, the computational load for the main agent can be shared with other agents in the system. Moreover, the strong robustness of the distributed architecture guarantees the system can still operate with the loss of few agents. However, time delay during the communication needs to be handled properly. Thus, for an MSMA problem with sufficient communication links but no single agent which can take over the total computational load, the distributed architecture is the best solution.

- Decentralized architecture
  In the decentralized architecture shown in Fig.5.3, there is no hub or channels for agents to exchange their information. Each agent can only connect with their neighbors which determined by the range of the communication system and their relative positions. The agents have to make decisions based on the limited information gained from their neighbors. If an agent cannot find any neighbor, it may work on its own. Therefore, the decentralized architecture has high robustness. Once any damage occurs to a decentralized architecture, the whole system will break into many decentralized systems. However, for a decentralized architecture, a good local solution for each agent may not lead to a good global solution. It

also faces the allocation redundancy problems once it breaks into several smaller decentralized systems. In general, when applying the decentralized architecture onto the MSS, restrictions on communication will affect the global performance.

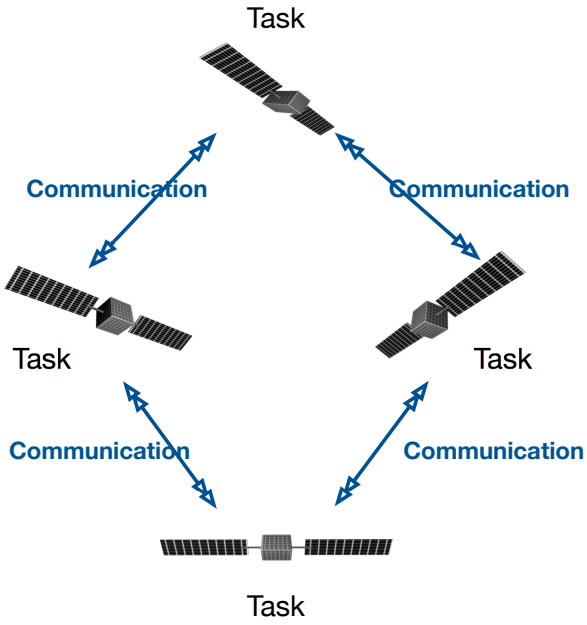


Figure 5.3: The decentralized architecture

5.2.2. GAME-THEORETICAL FORMULATION FOR AN MSMA PROBLEM

Considering the distributed or the decentralized architecture the MSS could form, in this chapter, we adopt concepts and methods from game theory to formulate negotiation procedures between agents in the MSMA problem. Game theory is a study on mathematically modeling the conflicts and cooperation between several intelligent rational decision-makers. In general, it studies how cooperation and negotiation mechanisms can be designed and how to maximize the global welfare from the perspectives of each agent.

In our MSMA problem, the DSs are considered as players and the goal is to minimize the DS decision time that has the longest decision time in all DSs. Different architectures lead to different kinds of games. For the distributed architecture, there exists an information hub or few information exchange channels. All the players know the moves or decisions previously made by all the other players, which makes this MSMA problem a *Perfect Information Game*. For the decentralized architecture, each player is only familiar with its neighbors' information and decisions. This feature sorts this MSMA problem as an *Imperfect Information Game*. Despite the differences in game types between the distributed and the decentralized architectures for an MSMA problem, the objective is to find the optimal task assignments and the resulting collective payoffs, which makes

both games the so-called Cooperative Multi-player Game.

For this MSMA problem, we make the following assumptions:

1. Participant agents
   A team of satellite S, each agent has individual needs, which is labeled as $S_i, i = 1,2,3,...,N$;

2. Main mission goal and sub-mission goals
   The mission objective MO is a high-level target for the whole group. During the mission, this global goal needs to be decomposed into several sub-goals for each satellite. The set of sub-goals is formulated as $\{mo_{S_1}, ...mo_{S_N}\}$;

3. Task assignment profile
   As mentioned previously, the main mission needs to be decomposed and assigned to each participant. Therefore, for all participants, the collection of task assignments are called an assignment profile, or sometimes an action set: $\mathbf{a}^j = \{a_1^j, ..., a_N^j\}$. Each profile represents a possible combination of task assignments for all the participant agents, and together all the assignment profiles form the profile search space $\mathscr{A}$;

4. Utility functions
   The utility function for each satellite $U_{S_i}(a_i^j)$ can be either different or indifferent depending on mission requirements. Each assignment profile $a_i^j \in \mathscr{A}$ corresponds to a global utility $U_G(\mathbf{a}^j)$.

The task assignment profile which is agreeable to every player is well-known as the Nash equilibrium. A Nash equilibrium is expressed as $\mathbf{a}^* = (a_1^*, ..., a_N^*)$ such that no player could improve its utility by choosing a different solution. For any satellite S_i in the system, we use a_{-i} to represent the collection of the task assignments of the other satellites beside S_i, i.e.,

$$\mathbf{a}_{-i}^j = (a_1^j, ..., a_{i-1}^j, a_{i+1}^j, ..., a_n^j). \tag{5.1}$$

With this notation, if all the satellites $S_i \in S$ are satisfied with the following condition,

$$U_{S_i}(a_i^*, \mathbf{a}_{-i}^*) = \max_{a_i^j \in \mathscr{A}} U_{S_i}(a_i^j, \mathbf{a}_{-i}^*) \tag{5.2}$$

we call the assignment profile $\mathbf{a}^*$ a pure Nash equilibrium (or to be Pareto efficiency). In an MSMA problem, the system objective is to find out a pure Nash equilibrium for all participants. Normally, a pure Nash equilibrium may not exist in every cooperative multi-player game. Therefore, the utilities for the participant satellites in our MSMA problem should be tailored so that at least one pure Nash equilibrium exists.

5.2.3. UTILITIES DESIGN

In game theory, utility represents the motivations of players. Well-designed utility functions for every player can lead all players reach a more preferred global outcome (global

utility). It is important to distinguish the global utility $U_G(\mathbf{a}^j)$ and the individual utility $U_{S_i}(a_i^j)$. The global utility represents the overall outcome under certain assignment profile $\mathscr{A}$, whereas an individual utility partly reflects the payoff of a single agent.

The MSMA problem considered in this chapter is an important input for our next phase research. The solution of the MSMA problem will be used as the initial assignment profile which represents the sub-goals for all the DSs. The quality of this solution will directly affect the performance of the follow-up planning algorithm. Therefore, the global utility and individual utilities for this MSMA problem are designed based on the mission requirements and computational needs.

Since there is no sequential requirement for operations, all daughter satellites can work simultaneously. Therefore, the global utility aims to search for an assignment profile which is a pure Nash equilibrium among all the assignment profiles. It means that the performance of this Nash equilibrium (chosen profile) is better or at least the same compared to other profiles. The equation for the global utility is shown as follows:

$$U_G(\mathbf{a}^*) = \min_{\mathbf{a}^j \in \mathscr{A}} \{\max_{i \in N} \{U_{S_1}(a_1^j), ..., U_{S_i}(a_i^j), ..., U_{S_n}(a_n^j)\}\} \tag{5.3}$$

where the objective of this global utility is to minimize the utility of the satellite that has the largest utility in the team.

The individual utilities follow the same rules as the global utility. Based on the scientific assumptions of the DSL mission, all eight daughter satellites have identical hardware and software settings. Although some system errors can change the parameters of the key components, the self-interest for each satellite is the same. Therefore, we treat these individual utilities as identical interest utilities. As introduced in previous chapters, in this thesis, we only concern about observation and communication behaviors. Therefore, in this chapter, for each DS there are three parameters related to these two behaviors, and they are considered to have influences on the individual utility of each DS. These three parameters stand for the functionality situation of the onboard memory HM, the observation capability HO, and the communication capability HC. For each satellite, once an assignment profile comes in, the On-board Computer (OBC) needs to estimate the time consumption based on the profile a_i^j and the three above condition parameters. The individual utility for each satellite is to find the assignment profile $\mathbf{a}^*$ which minimizes the time consumption, shown as:

$$U_{S_i}(\mathbf{a}^*) = \min_{\mathbf{a}^j \in \mathscr{A}} \{\mathscr{T}_{S_i}(a_i^1, HM, HO, HC), ..., \mathscr{T}_{S_i}(a_i^j, HM, HO, HC)\} \tag{5.4}$$

where $\mathscr{T}_{S_i}$ is the equation to calculate the time consumption for different inputs of the satellite S_i.

The self-interest of each satellite is to get an assignment which can minimize their time consumption. However, for this cooperative multi-player game, each player has to cooperate with others to improve the global utility. They need to negotiate with other players and reach a compromised solution which every player can accept. To efficiently accomplish the negotiation and cooperation procedures, the next section will introduce three negotiation mechanisms for different organizational architectures.

5.3. NEGOTIATION AND COOPERATION MECHANISMS

As mentioned before, in order to solve the mission allocation problem for an MAS, cooperation and negotiation between the agents are very important. To develop a good mechanism means to design the protocols for governing all the agents' interactions. Such mechanisms have certain desirable properties, including *Simplicity*, *Success rate*, *Maximizing welfare*, *Individual intelligence*, and *Stability* [Zlotkin and Rosenschein, 1996]. Considering all these aspects, in this section, we will discuss the equilibrium selection under distributed and decentralized organizational architectures, and design the suitable cooperation and negotiation mechanisms.

5.3.1. AGENT COMMUNICATION LANGUAGES

Before diving into the details about how to design the cooperation and negotiation mechanisms, first we need to introduce how the agents can interact with each other. For an agent, these interactions include how to enlist the support of other agents to achieve its goals, how to provide information to other agents, how to report its own status to other agents, and how to request something from other agents. These special communication languages are called the Agent Communication Languages (ACL). These languages rely on speech act theory [Searle, 1969] and provide a separation between the communication act and the content ontology. The first ACL was KQML (Knowledge Query and Manipulation Language) [Mayfield et al., 1995], which was developed in the early 1990s. By using the ACL, agents can interact with other agents by providing information to other agents, reporting their status, requesting from other agents, and so forth.

Currently, the most widely used ACL is called the FIPA (Foundation for Intelligent Physical Agents) [IEEE, 2018], which incorporates many features from KQML. The advantages of the FIPA ACL are the predefined interaction protocols which help to manage conversations using different language contents. An ACL message contains a set of message parameters. One of the mandatory parameters in all ACL messages represents the communicative acts (i.e. the purpose of this message), known as *Performatives*. Using this parameter to identify the priority and the need of the sender will increase the efficiency of the receiver's reaction. For example, if an agent receives two messages with the performative to be *INFORM* and *REQUEST*, it will react to the *REQUEST* earlier than the *INFORM* by defined protocols. The most commonly used performatives can be found in Tab 5.1.

5.3.2. DISTRIBUTED NEGOTIATION MECHANISM

APPLICABLE SCENARIO ASSUMPTIONS

When the MS fails, the MSS system needs to reform its architecture to a distributed or a decentralized architecture. In our study case, we firstly assume that no obstacle can block the communication between DS, and the individual communication power can provide reliable connections with other DSs. This assumption allows each DS to be able to communicate with other DSs during operations, which makes this MSS fit in the characteristic of a distributed architecture (as shown in Fig 5.2). When the communication between agents can be guaranteed, this assumption is also valid for other multi-agent system applications such as multiple drones or multiple intelligent vehicles.

Table 5.1: FIPA ACL Performatives (taken from [IEEE, 2018])

	Explanation
ACCEPT-PROPOSAL	This is a general-purpose acceptance of a proposal (at some point in the future) the receiving agent il perform the action, once the given precondition is, or becomes, true.
CALL-FOR-PROPOSAL(CFP)	This is a general-purpose action to initiate a negotiation process by making a call for proposals to perform the given action. The actual protocol under which the negotiation process is established is known either by prior agreement or is explicitly stated in the protocol parameter of the message.
CONFIRM	The sender confirms to the receiver the truth of the content. The sender initially believed that the receiver was unsure about it.
DISCONFIRM	This indicates that the sending agent believes that some proposition is false, intends that the receiving agent also comes to believe that the proposition is false, or believes that the receiver either believes the proposition, or is uncertain of the proposition.
INFORM(-IF, -REF)	INFORM is the most used performative. The single INFORM is to tell another agent something, and the sender must hold that proposition to be true. Also intends that the receiving agent also comes to believe that the proposition is true; For INFORM-IF, it is used as content of REQUEST to ask another agent to tell the sender is a proposition to be true or false; For INFORM-REF, it is similar as INFORM-IF, only this action asks for the value of the proposition;
PROPOSE	This is a general-purpose act to make a proposal or respond to an existing proposal during a negotiation process by proposing to perform a given action subject to certain conditions being true. The sender informs the receiver that the proposer will adopt the intention to perform the action once the given precondition is met, and the receiver notifies the proposer of the receiver's intention that the proposer performs the action.
QUERY(-IF, -REF)	QUERY-IF is the act of asking another agent whether (it believes that) a given proposition is true. The sending agent is requesting the receiver to inform it of the truth of the proposition; QUERY-REF is the act of asking another agent to inform the requester of the object identified by a descriptor. The sending agent is requesting the receiver to perform an inform act, containing the object that corresponds to the descriptor.
REFUSE	The action of refusing to perform a given action, and explaining the reason for the refusal. This act is performed when the agent cannot meet all of the preconditions for the action to be carried out, both implicit and explicit.
REJECT-PROPOSAL	This is a general-purpose rejection to a previously submitted proposal. The agent sending the rejection informs the receiver that it has no intention that the recipient performs the given action under the given preconditions.
REQUEST(-WHEN, -WHENEVER)	The sender is requesting the receiver to perform some actions by using REQUEST; REQUEST-WHEN allows an agent to inform another agent that a certain action should be performed as soon as a given precondition, expressed as a proposition, becomes true; REQUEST-WHENEVER allows an agent to do the same as the REQUEST-WHEN, furthermore, if the proposition should subsequently become false, the action will be repeated as soon as it once more becomes true.

UTILITY-BASED REGRET PLAY

For the MSMA problem using the distributed architecture, we propose a new negotiation mechanism called Utility-based Regret Play (URP). In our case, under the distributed architecture, each DS has to negotiate with other DSs without any knowledge about previous utilities of others. Therefore, the new negotiation mechanism requires each DS to only know, at most, the proposals made by itself, the proposals made by other DSs, and its own utility function. Considering these requirements, the new mechanism is inspired by two negotiation mechanisms, namely, the Utility-based Fictitious Play (UFP) [Fudenberg and Levine, 1995] and the Regret Matching Play (RMP) [Hart and Mas-Colell, 2000]. The UFP has the advantage of low computation burden on each participant, but it also suffers from slow convergence because it does not require each participant to compare their results. The RMP, on the other hand, has the advantage of regret ability for not playing the target that has been chosen in current negotiation step. However, the RMP cannot guarantee to converge to a pure equilibrium assignment profile. The URP inherits the ability to evaluate individual utility at each negotiation step from the UFP, and the ability to regret current choice and to choose another option in the past negotiation steps from the RMP. To ensure the steady communication connection within the distributed MSS, one of the DSs is selected as the information hub. The selection procedure is done by evaluating the relative positions of the totally functional DSs. The principle of this selection is to chose a DS which is in the middle of the fleet and has the highest communication capability. Considering that one DS is selected to be the information hub, we name this negotiation mode as "Hub Mode", the Fig 5.4 shows the procedures of this mode in the FIPA representation.

As shown in the figure, after the user sends the mission objectives through the ground station to the selected information hub, this *CFP* message will trigger the start of the group negotiations. The first action for the hub is to send self-check requests to every agent. Afterwards, the hub can determine the total number of team members and then

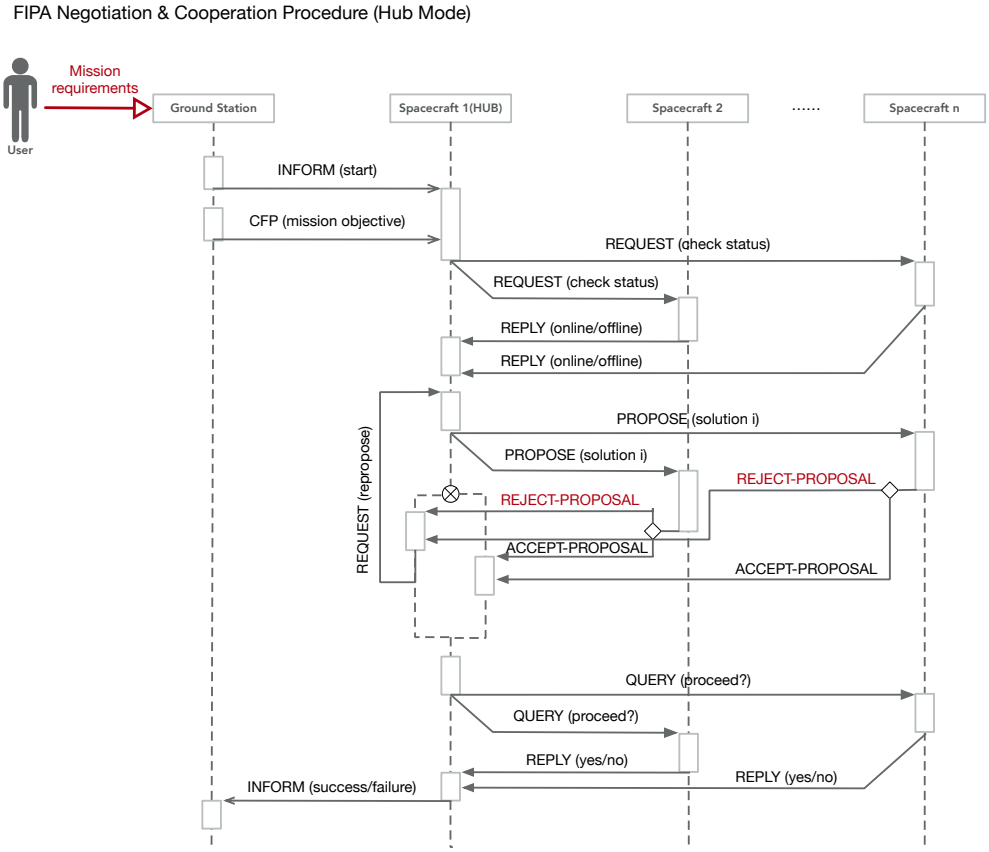


Figure 5.4: Utility-based Regret Play

generate the initial assignment profile for each participant. The agents which received this request will perform a self-check on three aspects, which have been introduced before (HM, HO, HC). If one agent's condition is nonfunctional on either of these aspects, it will reply to the hub with a *DISCONFIRM* message to indicate that it is unable to join the current mission (Then, the content of the message is "OFFLINE"). If one agent's condition is totally or partially functional on all these aspects, it will reply to the hub with a *CONFIRM* message to indicate its presence for this missions (the content of this message is "ONLINE"). Once the hub receives the status information from potential participants, it considers all the agents who have replied with "ONLINE" are ready for the team negotiation.

When the negotiation procedure starts, the hub randomly generates the initial assignment profile $\mathbf{a}^0 = [a^0_{S_1}, a^0_{S_2}, ...a^0_{S_n}]$ based on the global goal and the total number of agents and then sends messages as *REQUEST* to every agent. All the non-hub agents will receive the initial local task, which triggers their utility evaluation process. The utility value is then sent back to the hub and shared with other agents.

At each negotiation step g, each agent can only access to other agents' utility values through the hub. Through this information, the team can define the agent with the best utility and the agent with the poorest utility. These two agents can establish a unique communication link to share their personal information and local tasks. The information they share depends on the restrictions of their communication. A message with more detailed content about one agent's personal information, local tasks and the utility value can help the other agents to have a better understanding of its decision, but this message will be massive, and therefore, requires longer time to deliver this message. A simplified message, on the other hand, can save transmission time but it also loses the detailed descriptions about agent's information. In our work, to expand the adaptability of the URP, it is designed to compromise between the detailed message and the simplified message. It means these two agents will come to an agreement on new assignments which will be used for the next negotiation step. Assume these two agents to be S_i and S_j, the utilities of step g are $U_{S_i}(a^g_{S_i})$ and $U_{S_j}(a^g_{S_j})$, respectively. The new assignments for both agents can be predicted as follows:

$$[a^{g+1}_{S_i}, a^{g+1}_{S_j}] \leftarrow \mathscr{R}(a^g_{S_i}, a^g_{S_j}, HC_{S_i}, HC_{S_j}, HM_{S_i}, HM_{S_j}, HO_{S_i}, HO_{S_j}) \tag{5.5}$$

$$\begin{cases} a^{g+1}_{S_i} = [a^g_{S_i} + a^g_{S_j}] * \frac{U_{S_i}(a^g_{S_i})}{U_{S_i}(a^g_{S_i}) + U_{S_j}(a^g_{S_j})} \\ a^{g+1}_{S_j} = [a^g_{S_i} + a^g_{S_j}] * \frac{U_{S_j}(a^g_{S_j})}{U_{S_i}(a^g_{S_i}) + U_{S_j}(a^g_{S_j})} \end{cases} \tag{5.6}$$

Eq 5.5 is used for the situation where the detailed information is exchanged between S_i and S_j, including all the relative parameters (HS, HC, HO, a^g). With this information, two agents can reallocate the total assignment based on their conditions. Eq 5.6 is used for the situation where only the simplified information is exchanged (the utility value of $U_{S_i}(a^g_{S_i})$ and $U_{S_j}(a^g_{S_j})$). Under this situation, these two agents can only reallocate the total assignment based on their utilities.

The hub also needs to generate the task assignment profile for other agents besides these two agents, and sends it to each agent. During each negotiation step, every agent has the power to "regret" its current assignment based on the current utility value and the previous utility values. The agent who regrets about the assignment can use *QUERY* to ask the hub to reassign the previous profile so that it can have a better utility value. Before all the agents reach an agreement on one assignment profile, they will keep sending *REJECT-PROPOSE* back to the hub, and the hub has to continue the negotiation procedure. Once a proposal of assignment profile has been accepted by all agents, the hub will send *QUERY* to reconfirm the availability of each agent. The final assignment profile is sent back to ground through the hub to inform the final decision made by the group.

5.3.3. DECENTRALIZED NEGOTIATION MECHANISMS

SCENARIO ASSUMPTIONS

Compared with the distributed architecture, the decentralized architecture has less requirements on system-level information-sharing abilities [Tanenbaum and Van Steen, 2007]. In our case, due to the long distance between two DSs or hardware limitations,

the MSS may not be able to establish communication channels which can connect every DS in the group. Therefore, the decentralized architecture is a preferred choice for such an MSMA problem (as shown in Fig 5.3). For terrestrial multi-agent systems or under water (e.g. the multi-agent system using UAV or other kinds of intelligent vehicles), this architecture is also applicable on the system which has restrictions on communication (e.g. geographical obstacles, low communication distance). These communication restrictions make that the agents in such MAS can only connect with its neighbors. For this architecture, we propose two mechanisms to help the participants to negotiate based on their self-interests and the global mission objectives.

Before we jump into the negotiation procedures, a neighbor map needs to be established based on the mission requirements and other instructions. For a given agent S_i, the set of agents connected to it via communication links is called its neighboring set Ne_{S_i}. The existence of a communication link between two agents in general may refer to the availability information of one agent to the other one. For this situation, we introduce the neighbor map matrix $Map = [L(S_i, S_j)]_{N*N}$ to describe the graph associated with information exchanging in a network of agents, and to provide an initial statement for using the decentralized negotiation mechanisms.

$$L(S_i, S_j) = \begin{cases} 1 & |i-j| = 1 \\ -1 & |i-j| \neq 1 \\ 0 & |i-j| = 0 \end{cases} \tag{5.7}$$

SMOKE SIGNAL PLAY

We firstly propose the Smoke Signal Play (SSP) as the most intuitive communication and negotiation mechanism for a decentralized architecture. This negotiation mechanism is inspired by one of the oldest forms for long-distance communication called *Smoke Signal*. It was used in the ancient China to alert soldiers in other stations about the station of impending enemy attack by signaling from tower to tower [Wikipedia, 2018k]. The general idea of this negotiation mechanism is to use each agent as a smoke tower, then pass the information of utility value to the one after it. Once the negotiation procedure starts, each agent only needs to react to particular messages with the *REQUEST*performative, and responds with the *INFORM* performative message which only contains the utility values, while other status information remains hidden.

The FIPA representation of this mechanism is shown in Fig 5.5. Different from the distributed architecture, for the decentralized architecture, the mission objective is sent to every agent at the beginning of the negotiation procedures to make sure that each agent can make its decisions. With the neighbors map, each agent can determine which position it holds in the MSS and how to configure the communication sequences with its neighbors. Once the negotiation procedure starts, the SSP will trigger the first agent in the system to send utility value to the second one. Then, the second agent will send its own utility values and the preceding agent's together to its successor, and so forth. These steps belong to the forward transmission, which ends at the last agent who only has one neighbor in front. At this point, the last agent has the entire knowledge about the utilities of all the other agents, which enables it to evaluate the local goal. Then, the backward transmission procedure is activated by this fully knowledge agent. The

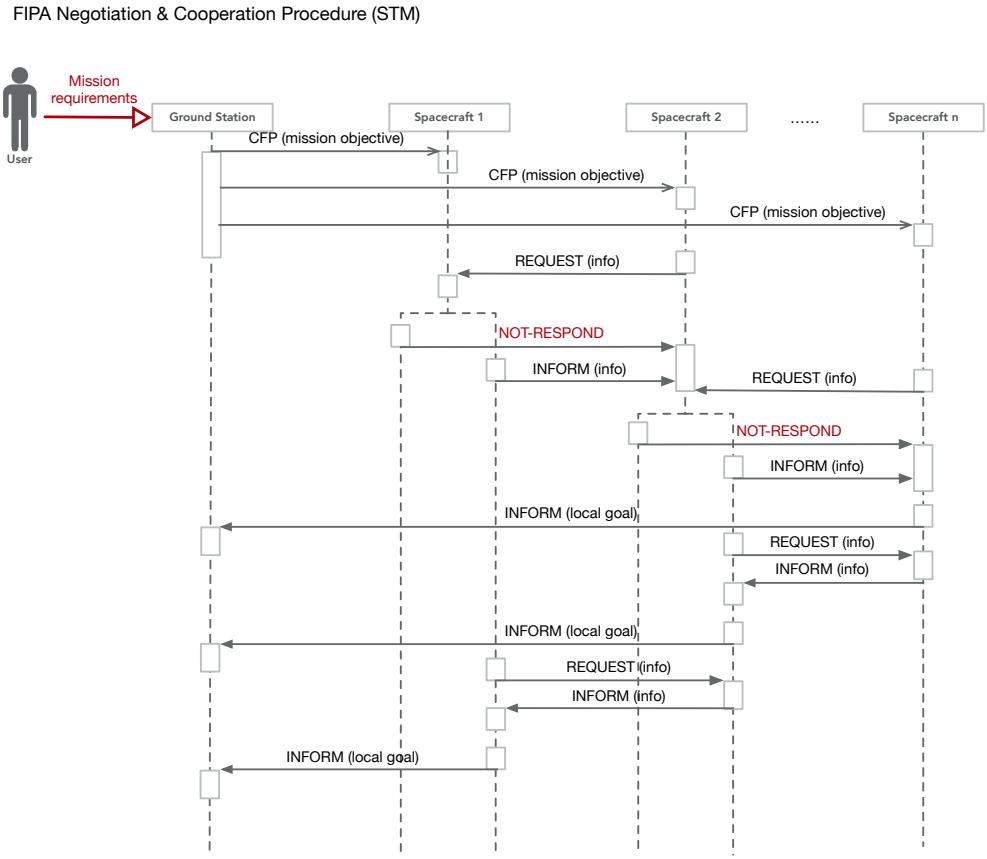


Figure 5.5: Smoke Signal Play

backward transmission follows the same protocols as the forward transmission, where this last agent is the start point.

Unlike the URP of which the local goals for each agent have to wait until all the participants agree with current assignment profile, the SSP ensures that any agent with complete knowledge of the group can define its own local goal based on the following expression:

$$\widetilde{a_i^*} = MO * \frac{U_{S_i}(MO)}{\sum_{j=1}^{N} U_{S_j}(MO)} \tag{5.8}$$

where $\widetilde{a_i^*}$ is the proximate result for satellite S_i since the utilities of all agents are determined by the global goal instead of a suitable local goal. This result is only a temporary solution and will improve as the negotiation iteration increases.

BROADCAST-BASED PLAY

Besides the SSP, we propose another mechanism named the Broadcast-based Play (BBP). Unlike the point-to-point (P2P) communication method used in the SSP, the BBP, as its

name suggests, uses broadcasting as the communication method. In our case, if any daughter satellite is damaged, the fleet needs to reconfigure or re-established the satellites order to fit the SSP requirements. The BBP, on the other hand, can enhance the robustness of the system by continuously broadcasting information to all the neighbors. The FIPA representation of this mechanism is shown in Fig 5.6.

Same as the SSP, the BBP also requires the ground station to send the global goal to every agent so that they can make decisions even without cooperation. Based on the payload requirements, the efficient transmission distance can be determined, and each agent will know how many neighbors it can reach and how many can reach it based on the neighbor map matrix. Once the negotiation procedure starts, each agent will start to broadcast the *REQUEST* message. Every time an agent receives a *REQUEST* message from another agent, it will store this message into its message database. Then, this agent will check whether it has replied to the sender in previous steps or not. If any information has changed, this agent will send a REPLY message with the new information, otherwise it will send a *REQUEST* message to decline this request. If not, this agent will send *RE-*

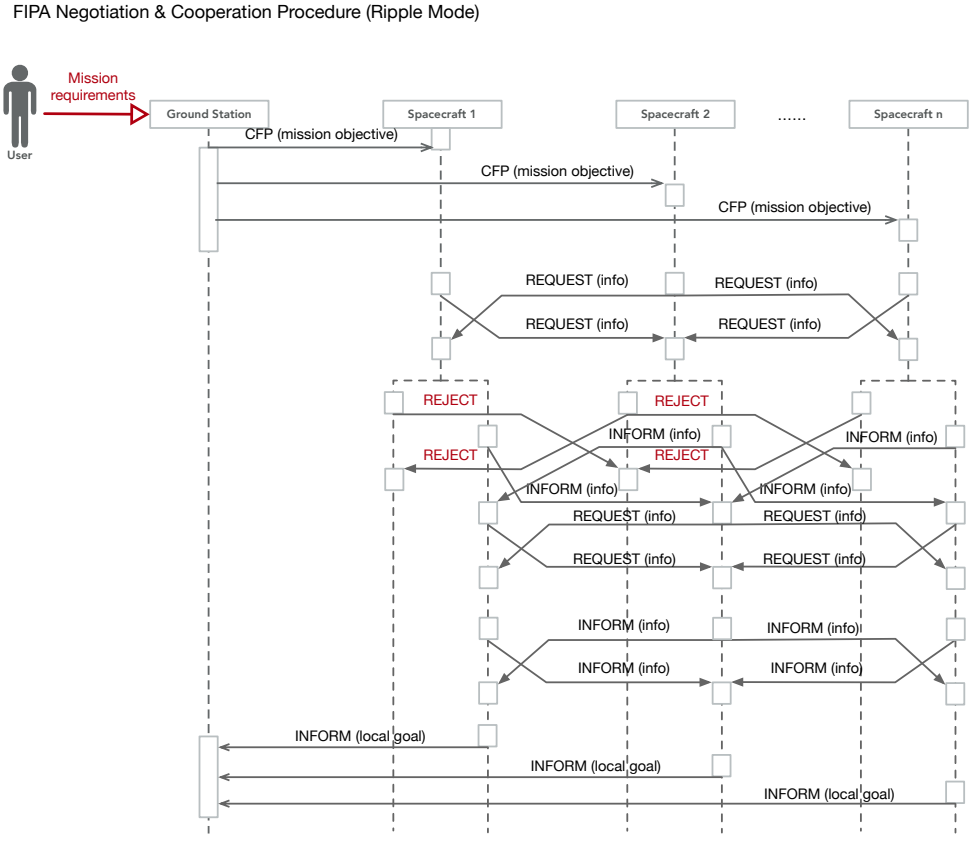


Figure 5.6: Broadcast-based Play

JECT message to decline this request. If this agent has never replied to the sender, it will send a *REPLY* message which contains all the information it currently contained. This strategy ensures that no useless information will be shared twice, which helps the receiver to avoid duplication. Moreover, when sending a message, the sender needs to address the ascription of this message along with its time tag. This additional content on every message helps each agent to expand its message database. Once the database reaches the predefined condition (e.g., receive all the information related to every agent), the local goal can be determined using the same expression shown in Eq 5.8.

COMPARISON

In this section, we have introduced two negotiation mechanisms for the decentralized architecture: the SSP and the BBP. These two mechanisms have many similarities. However, from an application point of view, there exist several differences when implementing these mechanisms in real missions. Firstly, due to the differences between P2P and broadcast, their power consumption is different. The SSP uses the P2P protocol to establish the communication links. The total communication slots depend on the total number of the participant agents. If there is no message lost during transmission, the number of communication slots N_P can be determined by $N_P = 2 * (N-1)$, where N is the total number of agents. The BBP uses the broadcast protocol. The total number of the broadcast slots N_B can be estimated by $N_B = N * (N-1)$. However, since the agents are assumed to broadcast simultaneously, the consumption of negotiation time the BBP will be much lower as compared to the SSP. Secondly, comparing the communication slots of the SSP and the BBP, it is evident that the BBP needs more times than the SSP. Considering the power consumption of all message transmissions, the total power consumption of BBP is expected to be much larger than SSP's. Moreover, the SSP strictly relies on the communication sequence, where as the BBP is more flexible about the agents' configuration.

5.4. SIMULATION RESULTS AND ANALYSIS

In this section, we will present the assumptions in our simulation study cases, and the results based on the proposed mechanisms for different scenarios. This allows to evaluate the performance of the proposed mechanisms, along with the comparisons with the state-of-the-art mechanisms to verify the superiority of the proposed mechanisms.

5.4.1. SIMULATION ENVIRONMENT

As in previous chapters, the simulations in this chapter will follow the same scientific requirements of the DSL mission as before. There are in total nine satellites including one mother satellite (MS) and eight daughter satellites (DSs). In this case we consider the situation that the MS, for whatever reasons, gets completely dysfunction. Then in order to continue the mission the eight DSs must implement either a distributed or a decentralized architecture.

To simulate different architectures, we propose two simulation scenarios. For both scenarios, we assume that all eight daughter satellites participate in the MSMA problem. For each DS, we use a functionality status set $Health_{S_i} = \langle HM_{S_i}, HC_{S_i}, HS_{S_i} \rangle$ to represent the healthy condition of this DS, where HM stands for the onboard memory

condition, *HO* stands for the condition of the observation capability, and *HC* stands for the condition of communication capability. Each variable represents the percentage of the functionality situation of corresponding sub-system, 0% meaning this sub-system is entirely dysfunctional and 100% meaning it is fully functional. The same idea has been used in [Barua and Khorasani, 2011]. The operating orbit and other basic parameters are the same as in previous chapters.

Scenario 1. We assume that the distances between any two satellites are within the efficient transmission distance and there is no obstacle to block the communication. This means that any daughter satellite can establish the communication link with the rest of the satellites. Using this assumption, this scenario is considered as the study case that employs the distributed architecture. In this scenario, we do not consider the propagation delay of the inter-satellite communication since we only want to verify the correctness of the proposed mechanisms. Any daughter satellite who has fully functional communication equipment can be selected as the information hub. For the communication efficiency, we define that the satellite in the middle of the fleet will make a proposal for becoming the hub. The rest of the satellites can react to this proposal based on their status and position.

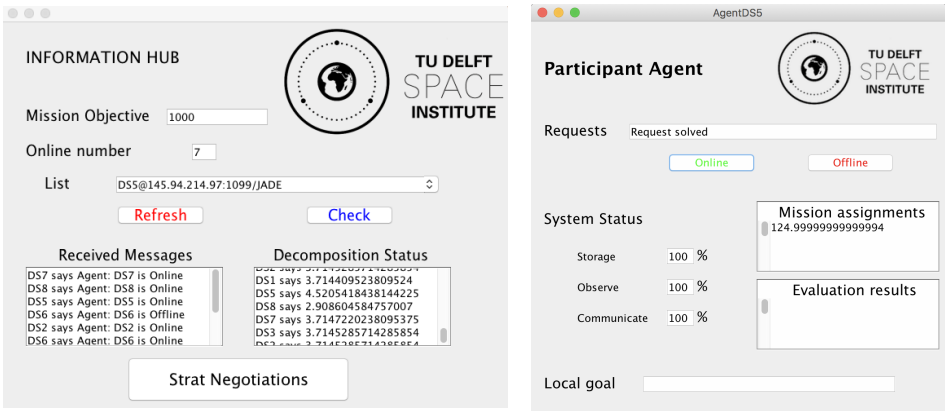
Scenario 2. Here, the communication between the satellites is restricted. There is no single satellite that can establish communication links with all the other satellites. Each satellite can only communicate with its neighbors. Based on this assumption, this scenario employs the communication-constrained decentralized architecture and aims to test the SSP and BBP mechanisms. We assume that each satellite's efficient "normalized" transmission distance is one, that is, it can only reach one satellite both in front and behind.

To have a fair comparison, we use the same computer for all the simulations, which has a 3.1 GHz Intel® Core™i5 processor, 8 GB of RAM, and MacOS V10.13.3 as the operation system. To construct the MSS, we use the JADE (Java Agent DEvelopment Framework) in Eclipse to build the multi-agent simulation environment for our MSMA problems.

5.4.2. SIMULATION RESULTS OF THE DISTRIBUTED ARCHITECTURE

INTERFACE AND SETUP

To test the URP performance, we establish the interface for information hub and participant satellites, as shown in Fig 5.7. From this figure, we can see that the users can customize the mission objective by changing the number in the corresponding box. By using the *Refresh* and *Check* buttons, the hub can send self-check request to each agent in the network. Every satellite has to indicate the hub whether it joins the team or not by the "Online" or "Offline" button in Fig 5.7b. The three healthy aspects related to onboard storage, observation, and communication are also inputs from users to simulate the onboard monitoring results. Once the hub begins the negotiation procedure, at each negotiation step, it will send assignments to every satellite and wait for the evaluation results.



(a) Panel for the information hub

(b) Panel for all participant agents

Figure 5.7: Multi-agent simulation interfaces in JADE for testing the URP

5

RESULTS AND ANALYSIS

For the DSL mission, there are eight DSs. To test the effectiveness of the URP under the different number of participants, we simulate five different team sizes which range

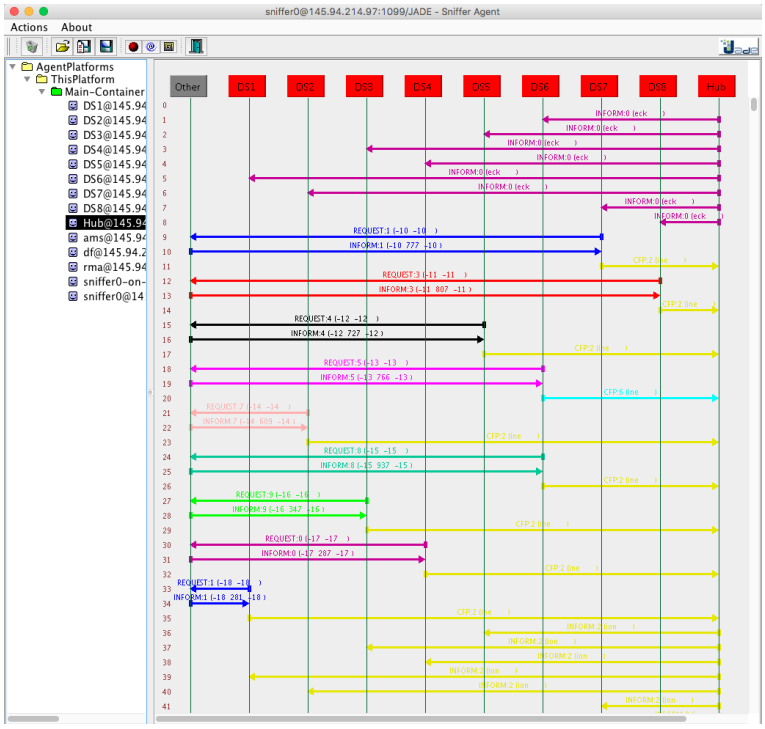
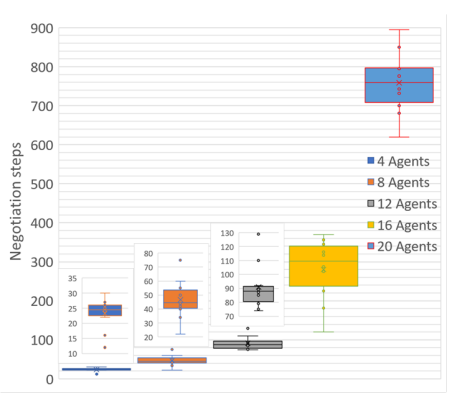


Figure 5.8: Single negotiation step shown in the Sniffer Agent in JADE

from 4 to 20 satellites with the increment of 4 satellites. We consider four satellites as a test unit, and each satellite has different system status. The status parameters for these four satellites (the test unit) are as follows: $DS1:\{HM=100\%, HO=100\%, HC=100\%\}$, $DS2:\{HM=50\%, HO=100\%, HC=100\%\}$, $DS3:\{HM=100\%, HO=50\%, HC=100\%\}$, $DS4:\{HM=100\%, HO=100\%, HC=50\%\}$. For each team size, we run the simulation under the same initial condition for ten times to collect the statistic results of the negotiation steps and the CPU time. In order to facilitate quantitative results, we set the mission objective for every test unit to be 500 Gbits. Therefore, along with the increasing number of agents, the mission objective is also increased. Fig 5.8 shows a single step of negotiation procedure for the eight DSs in JADE by a Sniffer Agent, which shows the same communication strategy as Fig 5.4. The hub sends the request messages to every DS at the beginning of this negotiation. Once it receives all the feedback, it will distribute the assignment profile for this negotiation step and wait for the evaluation results of each participant DS.

For each team size, we re-run the simulation with the same initial setting for ten times to get the results of the required negotiation steps and the required CPU time. The statistic results of this simulation are presented by box plots in Fig 5.9. Fig 5.9a shows the results of the negotiation steps and Fig 5.9b shows the total CPU time consumption and the average time consumption at each step. From Fig 5.9a we can see a clear trend that



(a) Number of negotiation steps for five different team sizes



(b) Total CPU time consumptions and average time consumption on each negotiation step for five different team sizes

Figure 5.9: Statistics results of five different team sizes using the URP mechanism, which are gathered from ten simulation runs for each size.

with increasing number of the agents, the number of negotiation steps increases significantly. Each box represents the statistic results for a certain number of agents, starting with 4 agents and ending with 20 agents. When the number of agents is four, the average negotiation steps are 25. After doubling the number of agents, the average required steps turn to 45, and tripling the number of agents to 12 leads the raise of the negotiation steps up to 88. The upward trend for agents below 12 is relatively flat considering the increased number of negotiation steps. However, comparing the yellow box which

represents 16 agents with the gray box which stands for 12 agents, the average steps increased almost four times from 88 to 302. Moreover, after the team size expands to 20, the needed steps go up to 760. Then, the hub needs more negotiation steps to evaluate the feedback received from each agent and re-assign a new assignment profile to them.

Fig 5.9b shows the CPU time consumption for different number of agents. Clearly, there is also an increasing trend. The boxes represent the statistic results from ten simulation runs. For four agents, the average time consumption is about 480 ms, and for other team sizes (8, 12, 16, 20 agents) are 2257 ms, 8132 ms, 46908 ms, and 225400 ms, respectively. The tendency is the same, but the amplitudes are much larger compared to the negotiation steps. To have a more detailed insight on this phenomenon, we combine the numerical data from all figures and calculate the average time consumption at each negotiation step, as the green line shown in Fig 5.9b. With the increasing number of the agents, the CPU time needed for each negotiation step is also increased, which begins from 20 to 301 ms/step. This is because with the increasing number of negotiation steps, the message database is also expanding. This makes the hub needing more time to find the suitable profile in the database and reply it to the agent who sends the regret message. Based on these simulation results, the URP shows good success rate on solving the MSMA problem. However, when the number of agents grows, the CPU time consumption is also facing a growth. Therefore, the URP mechanism is suitable for solving the MSMA problem with a relatively small team member.

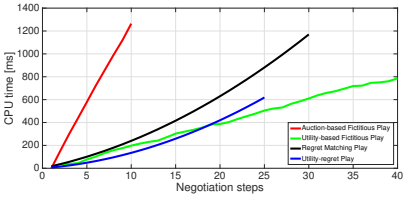
To illustrate the performance of the URP, we use the same computational resources and the same setup to test three other negotiation mechanisms: Action-based Fictitious Play (AFP) [Smyrnakis and Veres, 2016], UFP, and RMP. Fig 5.15 shows the simulation results for all those mechanisms. Compared to the AFP, although URP needs more negotiation steps, the consumption on CPU time is 50% less. This gap will increase with the number of agents increased. The results of the UFP show the similar CPU time consumption with the URP at all five team sizes. However, the URP requires almost 30% less number of negotiation steps than the UFP. For the RMP, the URP has better performance on both negotiation steps and CPU time consumption. Although the URP can outperform the other three mechanisms in our study, it has a weakness. The growth rate of the time consumption at each negotiation step is increasing with a growing number of agents. Therefore, the URP is best for solving the MSMA problem with a relatively small number of agents.

5.4.3. SIMULATION RESULTS OF THE DECENTRALIZED ARCHITECTURE

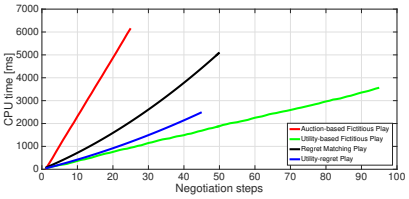
INTERFACE AND SETUP

Unlike the distributed architecture, for the decentralized architecture, there is no information hub. All the satellites have to negotiate with their neighbors by sharing their own utility values. The interface we have established for this architecture is shown in Fig 5.11.

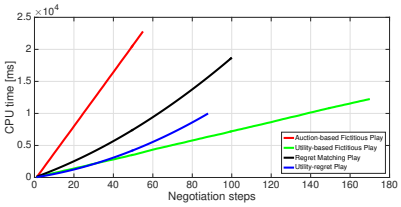
For the SSP and the BBP mechanisms, each participating satellite needs to determine who are its neighbors. The total number of neighbors is shown in the text field next to the list in Fig 5.11. Considering any dysfunction of participating agents will break one decentralized system into several independent decentralized systems. For a decentralized architecture, the global objective has to be sent to every agent at the beginning of



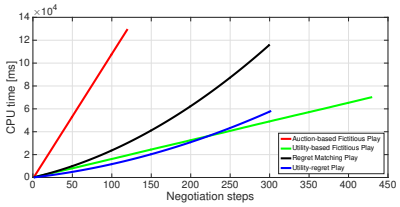
(a) Number of negotiation step and CPU time for 4 agents team



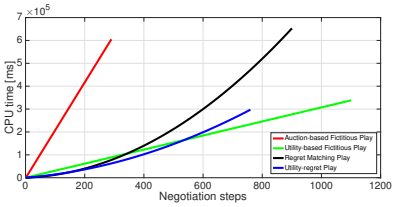
(b) Number of negotiation step and CPU time for 8 agents team



(c) Number of negotiation step and CPU time for 12 agents team



(d) Number of negotiation step and CPU time for 16 agents team



(e) Number of negotiation step and CPU time for 20 agents team

Figure 5.10: Comparison on the number of negotiation step and CPU time for five different team sizes

the negotiation procedure. During the mission allocation procedure, each satellite can only make decisions based on the information it has gathered from its neighbors. This leads to another parameter called *Response Time (RT)*. This parameter helps the agent to determine how long it needs to wait for its neighbors to react. If any neighbor fails to send a message within RT, it will be considered as a fully dysfunction agent and abandoned from this system. Here, we assume the $RT = 1000$ ms.

RESULTS AND ANALYSIS

To test the performance of the SSP and the BBP mechanisms, we also test five groups of agent numbers from 4 to 20. We take the Market-based Auction (MBA) mechanism [Walsh and Wellman, 1998] as the contrast mechanism. The system status of each test follows the same setting as in the distributed architecture. The numerical data gathered from ten simulation runs indicates statics number of the CPU time consumption and the total power consumption for each set of agents. Fig 5.12 and Fig 5.13 are two examples of the interaction between 8 agents during the simulation, which fits the FIPA representations in Sec.6.3.

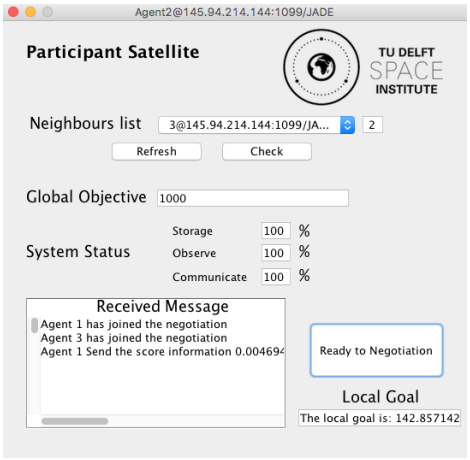


Figure 5.11: Multi-agent simulation interface in JADE for testing the BBP and the SSP

5

The statistical results of the CPU time consumption are shown in Fig 5.14. In this figure, the X-axis in each sub-figure stands for the participant number, and the Y-axis stands for the CPU time (with ms as the unit). We use box plots to show the results of ten simulation runs for each mechanism. From this figure we can see that BBP performs almost 4 times faster than MBA for all five team sizes. SSP can be twice as fast as MBA. For a direct comparison, we extract the average CPU time for each mechanism, shown in Fig 5.15a. The average time consumption for all three mechanisms shows the linear growth. The blue line stands for BBP, the orange line stands for SSP, and the black line stands for MBA. The gap between every mechanism on each size of agents becomes larger with the enlarged team member because of the differences on *Response Time*. The RT is a safety precaution to prevent any dysfunction agent affecting other agents. For all functional agents, during the negotiation procedure, they have to wait until the RT passes. Un-

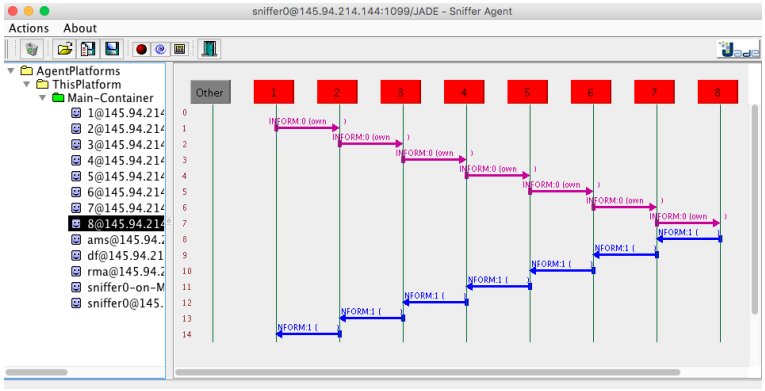


Figure 5.12: SSP negotiation

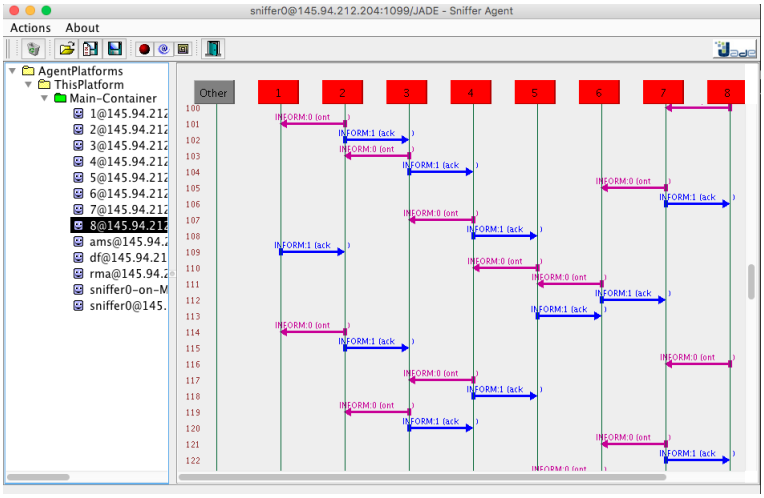


Figure 5.13: BBP negotiation

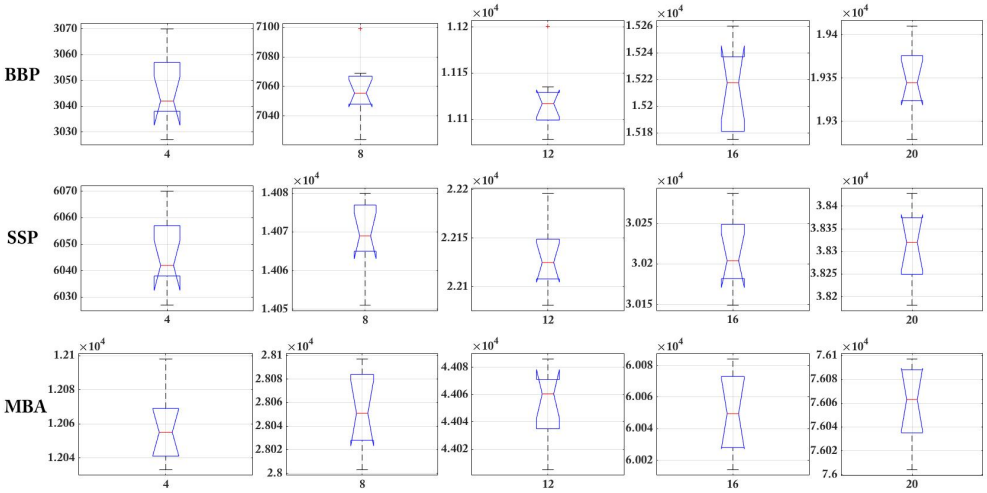
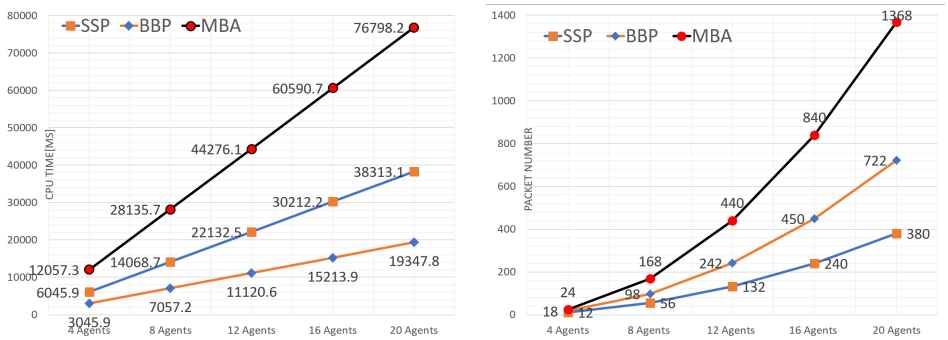


Figure 5.14: Statistical results of CPU time for three negotiation mechanisms (BBP, SSP and MBA)

der this assumption, comparing the SSP which uses a sequential P2P communication strategy with the BBP which uses the synchronized broadcast communication strategy, the BBP only requires half the time which the SSP requires. The MBA needs even more time because it requires agents to send bid messages regularly for checking the auction results. This regular behavior can extend the negotiation time by adding the redundant waiting time. The simulation results also justify that for all simulated cases, the SSP takes on average twice the time than the BBP, while four times of the MBA.



(a) Average CPU time

(b) Packet transmission number

Figure 5.15: Comparison on average CPU time consumption and packet transmission number for three negotiation mechanisms (BBP, SSP and MBA)

5

Although the BBP saves more time than the SSP, the SSP has an advantage from an energy consumption perspective. To simplify the energy consumption calculation process, we assume that each message equals to one packet and only contains one utility value. To transmit multiple information, the sender has to send the corresponding number of messages. Fig 5.15b shows the total packet transmission number for different sets of agents.

These three lines are depicted based on the communication strategy each mechanism choses. For SSP, during the forward information sharing procedure, the anterior agent needs to pass all the information it received from its preceding neighbor, along with its own information to the next agent, and vice versa for the backward procedure. The total number of packets can be calculated by $PN_{ssp} = N*(N-1)$, where N is the size of the team. For the BBP, since it uses the broadcast method, we assume that each agent only broadcast the latest information it received. To ensure the neighbors can receive the messages, each agent needs to continue broadcasting the same message until a new request has been received. In this case, for a team with N participating agents, the total number of packets transfered is $PN_{bbp} = 2*(N-1)^2$. For the MBA, due to its regular bidding behavior and the price comparison on the same bidder, the packet consumption in a worst scenario is $PN_{mba} = 4*(N-1)*(N-2)$. Comparing the results shown in Fig 5.15, we can conclude that both the BBP and the SSP require less packet transmissions than MBA during the negotiation, while SSP requires even less compared to BBP. The broadcasting method requires more power consumption than the P2P method. Even though in this case, to simplify the comparison procedure, for both mechanisms, we assumed that the power consumption for transmitting each packet is a constant value, the power consumption of BBP is still higher than SSP, and the difference of these two mechanisms will increase with the number of agents growing. In real applications, this difference may be even higher considering that more energy is needed for broadcasting than for P2P. From this comparison (Fig 5.15), we can conclude that SSP requires less energy consumption while the BBP requires less CPU time.

5.5. SUMMARY

In this chapter, we developed three negotiation and cooperation mechanisms for the on-board mission allocation problems in multi-satellite system. Considering the possible scenario where the MS may face a complete dysfunction during mission operations, we introduced two organizational architectures that the MSS may use when losing the MS, which were the distributed architecture and the decentralized architecture. Implementing these architectures, we adopted concepts from game theory to formulate the mathematical model for the multi-satellite mission allocation problem. For the distributed architecture, a mechanism called Utility-regret Play (URP) was proposed. This mechanism inhered advantages from the Utility-based Play (UBP) and the Regret Matching Play (RMP) negotiation mechanisms. The simulation results have shown that the URP has a better performance on the convergence negotiation steps and the average CPU time compared with three other mechanisms, namely, Action-based Fictitious Play (AFP), UBP, and RMP. For the decentralized architecture, two mechanisms named Smoke Signal Play (SSP) and Broadcast-based Play (BBP) were proposed. Comparing these two mechanisms in multiple simulation runs, the BBP on average required 50% less CPU time consumption than the SSP, while SSP required much less power consumption than the BBP. The comparison with another state-of-the-art mechanism called Market-based Auction (MBA) also indicated that the proposed mechanisms (BBP and SSP) outperformed MBA on both average CPU time and energy consumption.

In next chapter, we will focus on developing a distributed mission planning approach for the distributed architecture, to improve the convergence performance for the multi-satellite system with a large number of satellites.

6

DISTRIBUTED ONBOARD MISSION PLANNING APPROACH

Parts of this chapter are from Z.Zheng, J.Guo, E.Gill, *Distributed Onboard Mission Planning for Multi-Satellite Systems*, submitted to Aerospace Science and Technology

6.1. INTRODUCTION

In the previous chapter, the MS faced a completely operative dysfunction. The loss of the main controller would render of the centralized planning and re-planning approaches proposed in Chapter 3 and Chapter 4 useless. Based on this scenario, we proposed several negotiation mechanisms in Chapter 5 for different organizational architectures to solve the corresponding MSMA problems.

After the main mission objective has been decomposed and allocated to each participating DS, more detailed behavior planning procedures for each DS can be produced. Considering that the MSS will lose the MS, the rest of the team needs to work with an non-centralized architecture. This chapter aims to solve the onboard planning problems through collaboration among the DSs. Based on the literature review in Chapter 2, scientists have already developed distributed planning approaches (DPAs) for different types of multi-agent systems. However, there are two key points still require to be further investigated when implementing DPAs for a multi-satellite system (MSS). (1) Most of the DPAs, during each iteration, normally choose to peel away the local regeneration procedure from the group information exchange procedure [Shen and Norrie, 1999; Han et al., 2014]. This strategy can speed up the computation time for certain scenarios. However, if constraints on different agents are highly-coupled, employing those DPAs can be computationally expensive. (2) Many early studies (e.g. [Das et al., 2015; Bhandari et al., 2014; Schetter et al., 2003; Zlot et al., 2002]) did not consider scenarios where unexpected changes could happen to operating agents. These previously proposed DPAs are either especially suited for stable operational environments where all agents stay safe during the mission, or can only perform mission re-planning after the previous planning procedures have been completed. Few algorithms ([Torreño et al., 2012; Evers et al., 2014; Wu et al., 2018]) can handle unexpected changes while planning procedures are executed onboard.

In this chapter, a distributed onboard approach is proposed for distributed mission planning scenarios where multiple small satellites need to cooperate to solve a global task. The proposed approach consists of a Local Constraint Satisfaction (LCS) module and a Global Distributed Optimization (GDO) module. The LCS is performed individually on each satellite using a local search heuristic to provide feasible sub-solutions, while the GDO is running via the inter-satellite communication. During each iteration the proposed approach uses an improved distributed genetic algorithm to guarantee that highly-coupled global constraints can be met. Considering the combination of these two modules, this approach is named by Hybrid Distributed Genetic Algorithm (HDGA).

6.2. PROBLEM STATEMENT

In previous Chapters 3 and 4, the MS is responsible for gathering information from each DS before and during the planning procedures. It uses centralized planning approaches to generate suitable control sequences for each DS to maximize the overall observation performance. However, this architecture is extremely vulnerable to faults, degradation, and other non-normal situations, causing a reduction or even total loss of the functionality of the MS, as the scenario described in Chapter 5. Therefore, a distributed architecture is proposed to replace the centralized architecture, as shown in Fig 6.1. The upper

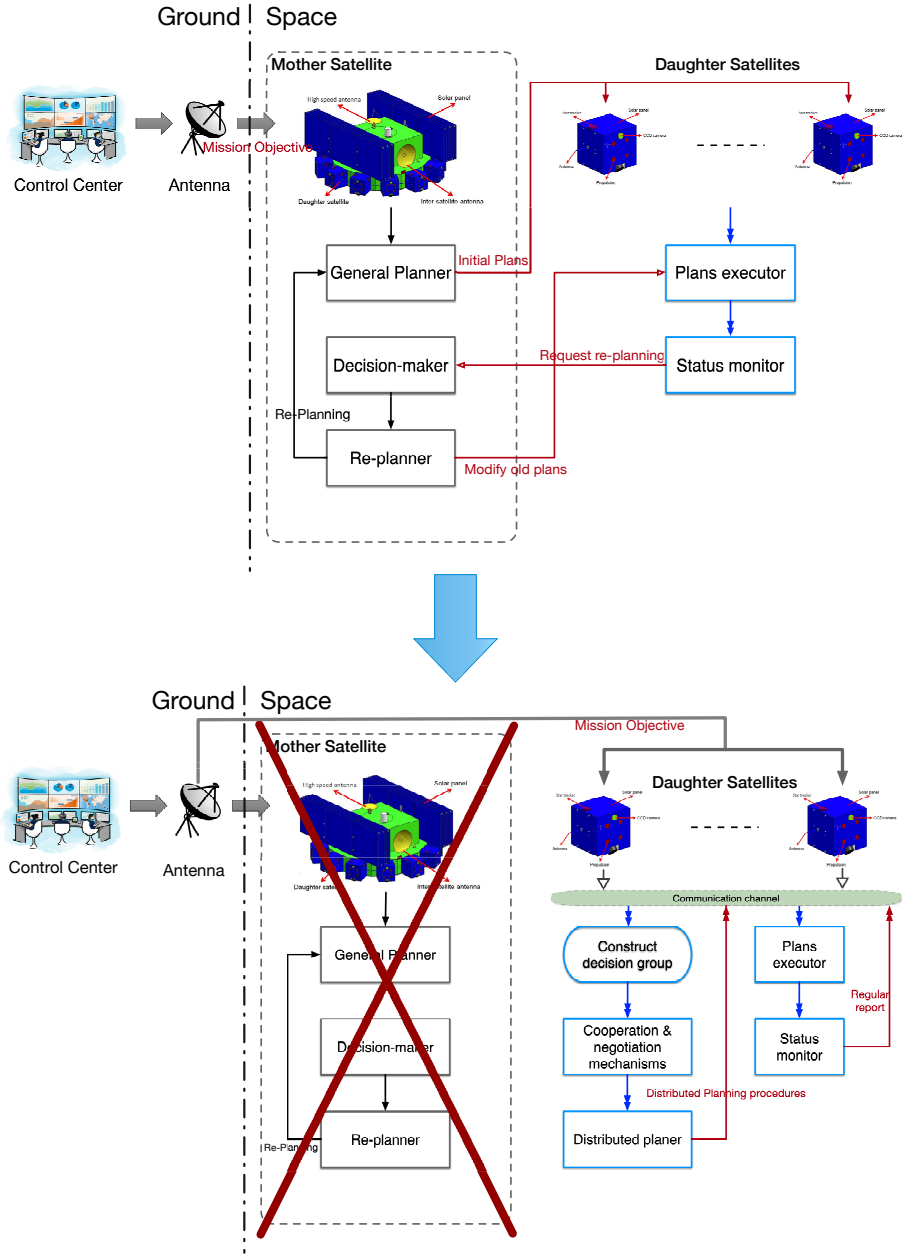


Figure 6.1: Organizational architecture change due to the loss of the main controller

figure shows the centralized architecture where the MS is controlling the entire team. All DSs are only responsible for executing plans and monitoring their self-status. The lower part in Fig 6.1 illustrates the change of the MSS architecture once the MS is dysfunctional

and the new architecture is a distributed architecture. From this figure, we can conclude that when the MS is eliminated from the team, the mission objective has to be sent to every DS at the beginning of the planning procedure. All the DSs are connected with each other through communication channels. Each DS is considered as a decision agent which is used to cooperate and negotiate with other agents to solve the multi-satellite mission allocation problems. Then, the distributed planner on each DS has to execute the distributed planning procedure with the information exchanged through the communication channels. Once every DS in the team comes to an agreement on a planning result, the plan executor and monitor on each DS will ensure the execution of these plans and keep track of the system status.

Due to the change of the system architecture as compared to the centralized architecture, the generic description for the planning problem is different from previous approaches and needs to be reconstructed for the new architecture. The planning problem considered here is more complicated than when using the distributed architecture. The primary mission objective remains the same as in previous chapters, which is to maximize observation data within the specific operational lifetime. The centralized optimization problem formulation is now invalid since no single DS has the capability to take over the functionality of the MS. Instead, the planning problem will be handled through the cooperation of all DSs. Therefore, the problem formulation in this chapter is involving all DSs.

6.2.1. NOTATIONS AND ASSUMPTIONS

- The planning problem relies on a set of N DSs. Here, we uses the same expression as in previous chapters, i.e. $S_i, (i = 1,2,...,N)$, where N is the total number of participating satellites in the mission.

- Mission lifetime $Time$. This variable represents the expected mission operating lifetime. Here, $n_k, (k = 1,2,...,M, M = Time/T)$ represents the current orbit number since the start of the mission, and T represents the orbit period of this operating orbit. This assumption is the same as in Chapter 3.

- Constraint set $C^j_{S_i}, (j = 1,2,...,Q)$, where Q is the total number of local constraints for DS S_i. For this case, since all satellites are identical, they face the same types and the same number of constraints. For other heterogeneous applications, constraints of each individual may be different.

- Initial sub-system parameter set $Initial_{S_i} = \left\langle Obs^0_{S_i}, Com^0_{S_i}, M^0_{S_i} \right\rangle$. This set represents the initial values characterizing the three most relevant sub-systems. The values include the initial observation data rate $Obs^0_{S_i}$, the communication data rate $Com^0_{S_i}$, and the onboard memory $M^0_{S_i}$.

- Functionality status set $Health^{n_k}_{S_i} = \left\langle HM^{n_k}_{S_i}, HC^{n_k}_{S_i}, HS^{n_k}_{S_i} \right\rangle$. This set stands for the healthy condition for DS S_i in a certain orbit n_k. Each variable represents the percentage of the functionality situation for each sub-system, where 100% means perfectly functional and 0% means completely dysfunction.

6.2.2. VARIABLES

- Sub-mission objective set $mo_{S_i}, (i = 1, 2, ..., N)$. This set is a predefined set based on the mission allocation results provided in the previous chapter. The main mission objective MO needs to be decomposed into sub-mission objectives. The objective decomposition procedure is performed by using the negotiation mechanisms proposed in the previous chapter. Once a DS receives a sub-mission objective, it will consider this objective as an input for the upcoming distributed planning procedures until its system status is changed due to system failures. In this case, the renegotiation protocol will be triggered and the main mission objective will be reallocated to each DS.
- Observation time variable set $\mathbf{X_{S_i}}(n_k) = \left\langle x^S_{S_i}(n_k), x^E_{S_i}(n_k) \right\rangle$, where $x^S_{S_i}(n_k)$ represents the start time of the observation window during orbit n_k, and the $x^E_{S_i}(n_k)$ stands for the end time of the observation.
- Communication time variable set $\mathbf{Y_{S_i}}(n_k) = \left\langle y^S_{S_i}(n_k), y^E_{S_i}(n_k) \right\rangle$, where these two variables also represent the start and end time of the communication time during orbit n_k.
- Onboard memory variable $M_{S_i}(n_k)$ represents the available onboard memory for S_i, which changes with orbit number n_k.

6.2.3. CONSTRAINTS

Based on the definition of the scenario with its notations and variables, in this section, the constraints will be formulated for the distributed architecture. These constraints comprise the onboard storage constraints C^{sto}, the behavioral synchronization constraints C^{syn}, and the communication constraints C^{com}.

(1) Individual storage constraints C^{sto}: The onboard storage for individual DS depends on the initial condition of the memory usage, the observation time and the communication time. This can be formulated as:

$$C^{sto}_{S_i} : \forall n_k, [x^E_{S_i}(n_k) - x^S_{S_i}(n_k)] * Obs^{n_k}_{S_i} \leqslant M_{S_i}(n_k - 1) + [y^E_{S_i}(n_k - 1) - y^S_{S_i}(n_k - 1)] * Com^{n_k - 1}_{S_i} \tag{6.1}$$

(2) Behavior synchronous constraints C^{syn}: Different mission requirements need different strategies for providing behavior synchronization. This means that certain behaviors on different satellites need to be executed at the same time. In this case, the data observing behavior for each DS needs to be correlated in time and bandwidth with other DSs for further signal processing, expressed by:

$$\forall n_k, \begin{cases} \xi^{S_{ij}}_{n_k} = 1, \ \mathbf{X_{S_i}} \cap \mathbf{X_{S_j}} \neq \varnothing \\ \xi^{S_{ij}}_{n_k} = 0, \ \mathbf{X_{S_i}} \cap \mathbf{X_{S_j}} = \varnothing \end{cases} \tag{6.2}$$

$$C^{syn}_{S_i} : \sum_{i=1}^{N} \sum_{j=1}^{N} (\xi^{S_{ij}}_{n_k}) \geqslant \vartheta_{MO} \tag{6.3}$$

where $\xi_{n_k}^{S_{ij}}$ represents the observation set relationship between two satellites S_i and S_j under orbit n_k. The variable ϑ_{MO} stands for the required number of temporally correlated observations to satisfy the signal processing requirements.

(3) Communication constraints: For the distributed architecture, the communication channels could be jammed due to bandwidth limitation. The constraints on communication can be expressed by:

$$\forall n_k, \begin{cases} \mu_{n_k}^{S_{ij}} = 1, \ \mathbf{Y_{S_i}}(n_k) \cap \mathbf{Y_{S_j}}(n_k) \neq \varnothing \\ \mu_{n_k}^{S_{ij}} = 0, \ \mathbf{Y_{S_i}}(n_k) \cap \mathbf{Y_{S_j}}(n_k) = \varnothing \end{cases} \tag{6.4}$$

$$C_{S_i}^{com_1} : \sum_{i=1}^{N} \sum_{j=1}^{N} (\mu_{n_k}^{S_{ij}}) \leqslant \kappa_{MO} \tag{6.5}$$

$$C_{S_i}^{com_2} : \forall n_k, y_{S_i}^{E}(n_k) - y_{S_i}^{S}(n_k) \geqslant TC_{min} \tag{6.6}$$

where $\mu_{n_k}^{S_{ij}}$ represents the set relation of communication. Here, κ_{MO} is the total communication number that can be handled by the communication channel due to the hardware requirements, and TC_{min} is a constant number which is determined by the hardware to represent the minimum time of each communication slot. For the DSL mission, the motivation of launching satellites in lunar orbit is to avoid the RFI. Therefore, RFI generated by other satellites needs to be avoided as well to ensure the data integrity, κ_{MO} is set to be 1.

6

6.2.4. EVALUATION FUNCTIONS

The evaluation functions for the distributed approach consist of two parts: one for solving the local constraint satisfaction problem (LCS), the other for solving the global distributed optimization problem (GDO).

For the local CSP, each DS needs to generate operable plans based on its own constraints, including the sequential constraints C^{seq} shown in Eq 3.5 and 3.7, data transmission constraints C^{tra} shown in Eq 3.8 and 3.9, and the onboard storage constraints shown in Eq 6.1. The local CSP can be expressed as a triple $\langle \mathbf{V}, \mathbf{D}, \mathbf{C} \rangle_{S_i}$, where $\mathbf{V}_{S_i} = \{\mathbf{X}_{S_i}, \mathbf{Y}_{S_i}\}$ is a set of variables for satellite S_i; $\mathbf{D}_{S_i}$ is the respective search domain for variables, which can be calculated based on the orbit altitude and satellite baseline (distance between two DSs). The set $\mathbf{C}_{S_i} = \{C^{seq}, C^{tra}, C^{sto}\}_{S_i}$ represents of constraints for satellite S_i. In our approach, the results from the local CSP are the candidates for further distributed optimization problem.

After solving the local CSP, each DS can formulate its own global variables set through exchange of the team information. This individual global variables contain all the variables from the entire team. The distributed evaluation function (Eq 6.7) is used for individual evaluation procedures during each generation. Once all satellites get their individual evaluation result through the global evaluation function (Eq 6.8), the best individual solution among the team can be selected as the best solution for the current generation. These two functions are formulated as follows:

$$L(\mathbf{X_{S_i}}) = \mathbf{argmax} \sum_{i=1}^{N} \sum_{j=1}^{M} [x_{S_i}^{E}(n_k)^* - x_{S_i}^{S}(n_k)^*] \tag{6.7}$$

$$G(\mathbf{X}) = \mathbf{argmax}\{L(\mathbf{X_{S_1}}), L(\mathbf{X_{S_2}}), ..., L(\mathbf{X_{S_N}})\} \tag{6.8}$$

where in Eq 6.7, $x^E_{S_i}(n_k)^*$ and $x^S_{S_i}(n_k)^*$ stand for the individual global variables randomly formed in satellite S_i by combining the information from all other satellites. In Eq 6.8, the global solution is given by the best of all the satellites' solutions. In the next section, we will introduce the local search heuristic and the distributed optimization algorithm in detail.

6.3. DISTRIBUTED MISSION PLANNING APPROACH

Based on the problem statement from the last section, the HDGA is designed to solve the onboard mission planning problem for an MSS with a distributed planning architecture. The flow chart of the HDGA for the distributed mission planning problem is shown in Fig 6.2. From this chart, it can be seen that besides the team negotiation, the HDGA is divided into two parts: the local constraint satisfaction and the globally distributed optimization, each of them is represented by a LCS module and a GDO module. In the LCS module, a heuristic is designed to generate local populations as candidates to form the global population in the next phase. In the GDO module, we transform the HDMGA (proposed in Chapter 3) into a distributed form to perform distributed optimization.

6.3.1. LOCAL CONSTRAINT SATISFACTION MODULE

Algorithm 5 Local constraints satisfaction module

Input: BP be the set of basic parameters. $Initial_{S_i}$ is the initial system status for this satellite, including the onboard storage, observation, and communication rate. $CurrentIte$ is the number of current iteration. LP_{size} is the size of the local population.

Output: Local population LC_{S_i}

1: **function** $Localpopulation = PopGeneration(BP, Initial_{S_i}, CurrentIte, LP_{size})$
2: $Constraints \leftarrow CalConstraints(BP, CurrentIte)$
3: **while** $LC_{S_i}() \neq LP_{size}$ **do**
4: $SubChromosome \leftarrow GBFS(CurrentIte, Initial_{S_i})$
5: **if** $SubChromosome$ is satisfied with $Constraints$ **then**
6: **for** $i = 1, 2, ...LC_{S_i}.size$ **do**
7: **if** $SubChromosome$ is not as same as anyone in LC_{S_i} **then**
8: add{$SubChromosome$} to LC_{S_i}
9: **else**
10: remove {$SubChromosome$} from candidate
11: **return** LC_{S_i}

In this module, each DS needs to determine the local population for the local variables by implementing a local search heuristic. Based on the planning architecture, this module generates suitable local solutions for the inter-satellite information exchange procedure. Considering the existing search heuristics, the greedy best-first search (GBFS) is chosen as the local search heuristic for one main reason: quick convergence speed

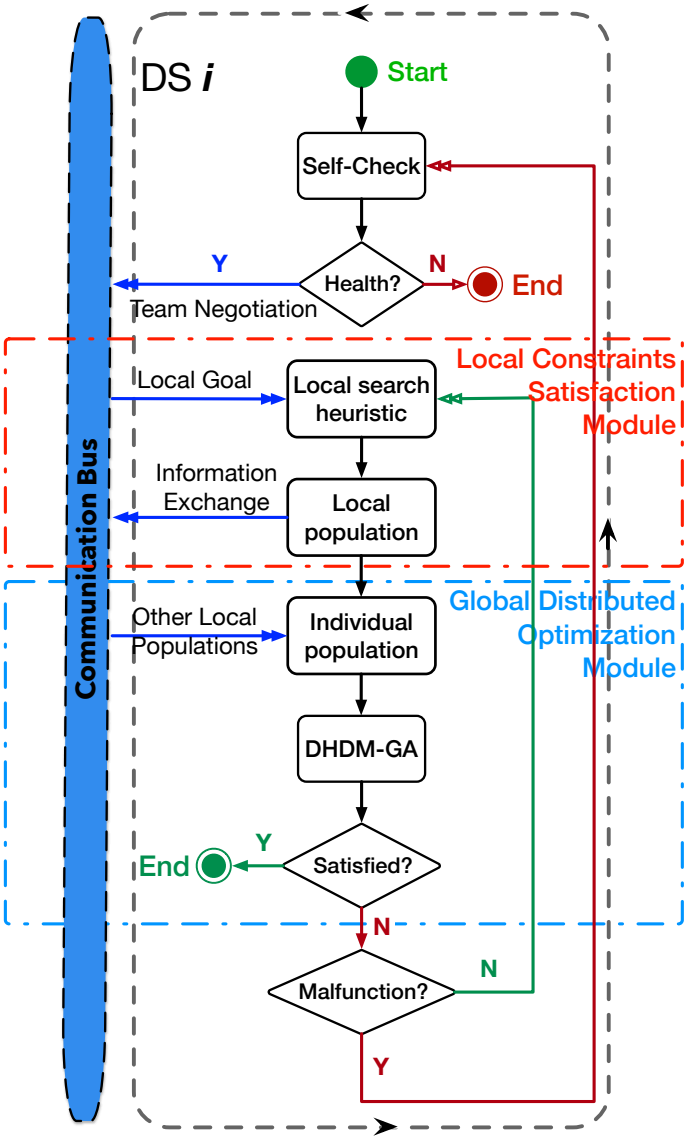


Figure 6.2: Flow chart of HDGA for each satellite

[Bonet and Geffner, 2001]. The "greedy" characteristic of the GBFS will make the algorithm blindfolded by immediate interests, leading to a sub-optimal solution. Unlike the standard CSPs which demands an optimal solution, in our case, we need the LCS module to provide several sub-optimal solutions within a short time frame. The GBFS can provide many solutions which satisfy local constraints within a short time frame. The

local population can also be initialized by this algorithm. The Pseudo code is shown in Algorithm 5 on Page 115.

6.3.2. GLOBAL DISTRIBUTED OPTIMIZATION

As described in Fig 6.2, the second module is the globally distributed optimization module. Considering the similarities in global constraints between the distributed planning problem and the centralized planning problem (e.g. observation constraints, communication constraints, and onboard memory constraints), we modify HDMGA into a distributed form to cope with the distributed architecture. However, different from the centralized optimization approaches, the distributed approach cannot perform a global evaluation for each DS based on its own variables. The constraints on synchronous observations requires that the evaluation procedure has full knowledge of other participating satellites. Therefore, the proposed distributed HDMGA (DHDMGA) needs the (strongly-coupled) global population at every iteration to ensure the correctness of the final solution. Through the communication bus of the MSS, all satellites can share their local populations with others. By combining the local populations from all others, each satellite can formulate its global population which contains all the variables from the entire team.

The generation procedures of this global population are shown in Fig 6.3. This figure consists of two parts; the upper part with several rectangles represents the local population for each DS. As explained in the previous chapter, each DS will use the GBFS to

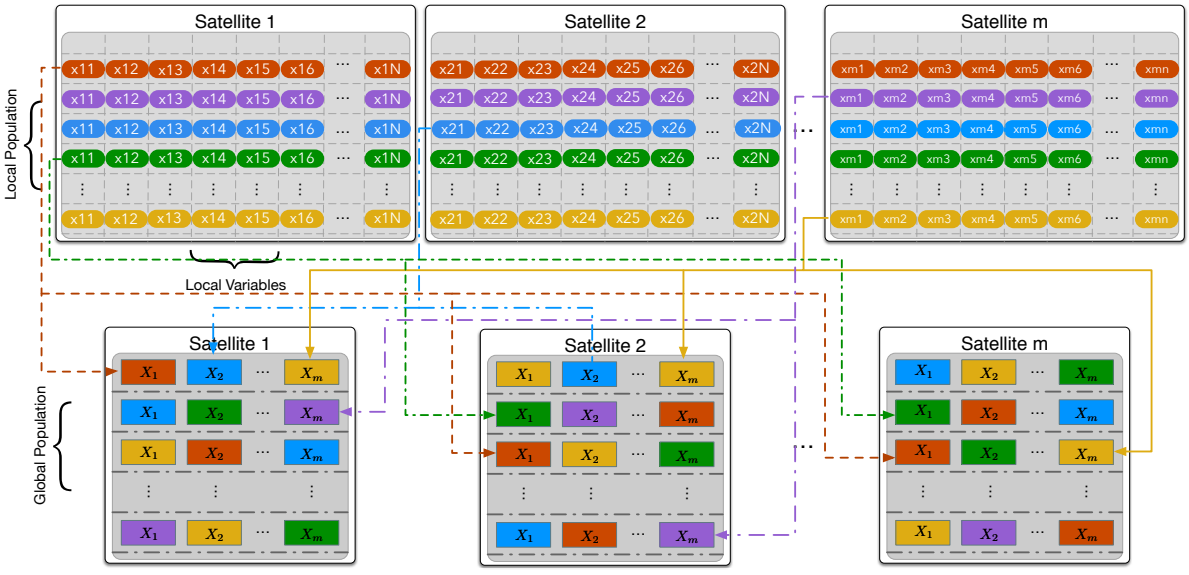


Figure 6.3: Global population generation procedure, each colored box in the global population stands for a local population from a satellite. For example, $\mathbf{X}_1^{RED} = \{x_{11}, x_{12}, x_{13}, ..., x_{1N}\}^{RED}$

Algorithm 6 Global distributed optimization module

Input: Local populations from other satellites $LC_{S_1},...,LC_{S_N}$; Satellite number S_i; global population size GP_{size}; Termination error tolerant ε_T; Max stall generation SG_{max}, Max generation G_{max}, fitness function $D(\mathbf{X}_{S_i})$

Output: Global solution $G(\mathbf{X})$

function $G(\mathbf{X})$ =*DHDM-GA*$(GP_{size}, LC_{S_i}, \varepsilon_T, SG_{max}, G_{max}, Fit_Fcn)$
  $Global^0_{pop} \leftarrow GPFormulation(GP_{size}, LC^0_{S_1},...LC^0_{S_N})$
  while $\kappa \leqslant G_{max}$ && $\varepsilon_t \leqslant \varepsilon_T$ **do**
    for $j = 1,2...GP_{size}$ **do**
      $best_individual(j) = Cal(D(\mathbf{X}_{S_i}), Global_{pop}(j))$
      $Crossover(two\ random\ individuals)$
      $HDMmutation(GP_{size}, R_{normal}, R_{escape}, T_N)$
    $Best_individual^\kappa \leftarrow$ **Max**$\{best_individual(1),...,best_individual(GP_{size})\}$
    Information exchange with other satellites to update:
    $G(\mathbf{X})^\kappa \leftarrow$ **Max** $\left\langle Best_individual^\kappa_{S_1},...,Best_individual^\kappa_{S_N} \right\rangle$
    if $Best_individual^\kappa == G(\mathbf{X})^\kappa$ **then** break;
    else
    $Global^\kappa_{pop} \leftarrow RefreshGP(Global^{\kappa-1}_{pop}, GP_{size}, LC^\kappa_{S_1},...LC^\kappa_{S_N})$
  $G(\mathbf{X}) \leftarrow G(\mathbf{X})^\kappa$
return $G(\mathbf{X})$

solve the local CSP and generate several solutions. These solutions are entirely independent of each other. In Fig 6.3, these local solutions are distinguished by different colors. Although in this figure, for different satellites, we used the same colors to represent the chromosomes in the local populations, the variables for different satellites are different. Each color within each satellite represents the combination of the values for their local variables. For instance, the first variable set in upper left rectangle $\{x_{11}, x_{12}, ..., x_{1N}\}$ is presented by red, which stands for a local chromosome (LC) in satellite S_1, while the same color in the upper right rectangle $\{x_{M1}, x_{M2}, ..., x_{MN}\}$ is the variable set for satellite S_M.

The lower part of the figure shows the global population of each DS. Once the local population has been generated, this information will be exchanged with other satellites through the communication bus. By combining all the local populations received from other satellites, each DS can formulate its individual global population by assembling the LCs from other satellites. To ensure the integrity and randomness of the global population in each satellite, the assembling procedures for the global chromosome (GC) in each satellite are performed by implementing the Roulette Wheel Selection (RWS) [Lipowski and Lipowska, 2012]. In this figure, the squares stand for the global population for each satellite. Each small rectangle within the global population represents the LC gathered from one satellite, which is selected through the RWS in each satellite. For example, the first global chromosome in satellite S_1 consists of the red LC $\mathbf{X}_1$ from the S_1, the blue LC $\mathbf{X}_2$ from the S_2, and in the end the yellow LC $\mathbf{X}_M$ from the S_M. The RWS will guarantee a non-repeatable selection for GCs in each satellite. Once the global pop-

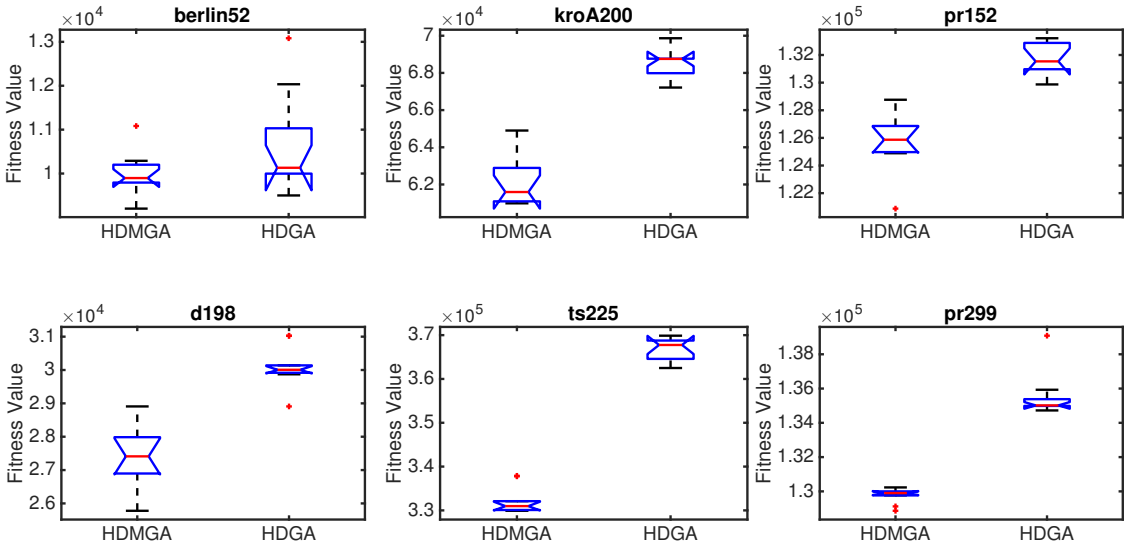


Figure 6.4: Fitness value on six study cases for HDMAGA and HDGA approaches

ulations have been formulated, each satellite will start individual evaluation through the evaluation function $G(\mathbf{X})$. The procedure of the globally distributed optimization module is expressed in Algorithm 6 on Page 118.

6.4. VERIFICATION

In the previous section the HDGA has been designed. To verify the proposed approach, in this section, several instances of the Travelling Salesmen Problem (TSP) will be implemented as study cases to compare the performance of the proposed HDGA and the previous centralized approach (HDMGA).

The typical TSP is a well-known NP-hard problem that exists in many real-world scenarios. The TSP is usually used to test the sufficiency and correctness of optimization approaches. The sufficiency of an approach means how quick one approach can find the right solution, and the correctness of an approach means whether this approach can find the optimal solution or the solution is within the acceptable range. In this chapter, we implement six instances gathered from the TSPLIB [Reinhelt, 2014], every instances have their optimal solution. For each instance, both the HDGA and the HDMGA are examined. Both the centralized and the distributed approaches use heuristic algorithms as optimizers. To compare the performance, for each instance we run ten simulations under the same initial conditions. Since our proposed approaches are designed for the DSL mission which has eight DSs, the distributed approach uses eight computing nodes. Each node utilizes a 2.4 GHz Intel® Core™i5 6300U dual-processor, 8 GB of RAM, and 1 Gb/s network interface. Only one processor core per node is used to simulate the dis-

Table 6.1: TSP instances results for both approaches

Approaches	Centralized Approach (HDMGA)			Distributed Approach (HDGA)		
Examples	Av.#iter	Av.time[s]	Av.FV	Av.#iter	Av.time[s]	Av.FV
berlin52	114	18.97	9890	147	27.88	10231
kroA200	101	29.88	62890	205	35.98	68903
pr152	136	24.87	125631	197	26.76	131928
d198	130	38.87	27986	160	40.15	30139
ts225	93	34.64	337895	124	43.74	368751
pr299	77	47.97	129821	102	62.82	134723

tributed environment for the HDGA. Each run is terminated when the maximum number of generations has been reached or the lowest error has been achieved. The results contain the average computation time, average iteration number, and average fitness value. The details of the statistic results on fitness values are shown in Fig 6.4, where the blue lines stand for upper, middle and lower quartiles, black lines represent upper and lower whiskers, and red markers are the abnormal results.

The statistical results of this verification are shown in Tab 6.1. It includes the average convergence iteration number (represents by Av.#iter), average computation time (the sum of computation times of all the participant nodes, represented by Av.time[s]), and average fitness value (represented by Av.FV) for both approaches. Although in this table, the centralized approach (HDMGA) shows that it can get much lower fitness value compared with the distributed approach (HDGA) on every instance, the average differences between two approaches on fitness values are all less than 5%. This makes the use of the proposed HDGA acceptable when dealing with different TSP instances. According to Fig 6.4 and Tab 6.1, the centralized approach presents a better performance than the distributed approach on every considered aspect. The verification made in this section reveals the usability of HDGA. However, to check the usability of the proposed approach for the DSL mission scenario, we need to compare it with other approaches for our unique study case. In next section, simulation results on different approaches will be shown to characterize the overall performance of the proposed HDGA.

6.5. SIMULATION AND ANALYSIS

This section will introduce the simulation results for the proposed distributed approach and its comparison with two other state-of-the-art approaches. These comparisons will help to analyze the characteristics of the proposed method.

We first evaluate the proposed approach by comparing it with a centralized approach. Then, we employ two other state-of-the-art distributed approaches as competitors to compare their performance with the proposed HDGA for a distributed mission planning scenario. One approach employs the distributed ant colony optimization (DACO) algorithm [Collings and Kim, 2014] as the solver, and the other one uses the coevolutionary particle swarm optimization (CPSO) algorithm [Krohling and dos Santos Coelho, 2006]

as the solver. Both solvers have been implemented for the distributed system optimization problems.

6.5.1. STUDY CASES

STANDARD PLANNING CASE

In previous chapters, the study case was simplified as a centralized optimization problem where the variables are the start and end time of observation and communication behaviors for each DS, and the mission objective is to maximize the collection of observation data within a certain lifetime. For this standard planning scenario, the centralized method uses the MS to perform the centralized planning algorithm (HDMGA) while the distributed approach needs all the eight DSs to perform the HDGA. In order to ensure the consistency of the comparisons, despite of differences in the planning architecture, both approaches need to follow the same assumptions.

To simplify the mission operation environment, the orbit is assumed to be a circular orbit with an altitude of 300 km above the moon surface. The initial status $Initial_{S_i}$ is assumed to be identical within the team, where the detail assumptions are $Obs^0_{S_i} = 48$ Mbps, $Com^0_{S_i} = 6$ Mbps, $M^0_{S_i} = 128$ GB. The MS position has the same relative position as in Chapter 3 and 4. Three key constraint parameters from Eq 6.3, Eq 6.5 and Eq 6.6 are assumed to be constant based on the DSL requirements, which are $\vartheta_{MO} = 5$, $\kappa_{MO} = 1$ and $TC_{min} = 5$. To test the capability of dealing with long lifetimes and more satellites, the mission objectives cover 10 to 50 orbital periods, and the number of DS ranges from 6 to 12.

DISTRIBUTED PLANNING CASE

The distributed planning case represents the abnormal situation that the MSS will encounter a malfunction of the MS. In this scenario, the MSS is unable to perform the centralized approach due to the loss of the central controller. For this distributed case, the proposed approach is compared with two other distributed approaches.

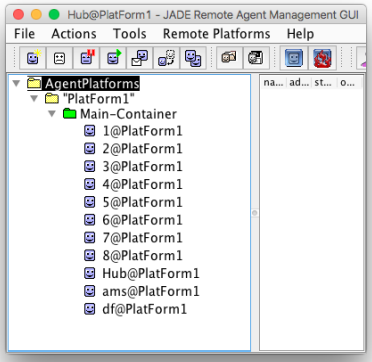
Basic settings including the orbit altitude (300 km), initial status ($Initial_{S_i}$), and the mission objectives (10 to 50 orbits) stay the same as in the normal case. The differences are: (1) There is no MS in the team. Therefore, the baseline between the 4th and the 5th DS is the same as the baseline of other DSs; (2) Besides the initial status, in this case, we add the health status $Health^{n_k}_{S_i}$ to represent the current health conditions of each DS. The details about the health status are shown in Tab 6.2; (3) Considering the functionality status, the total number of DSs for this case is assumed to be a constant number 8, which is equivalent to the real scenario in the DSL mission.

Table 6.2: Functionality status settings for participating satellites

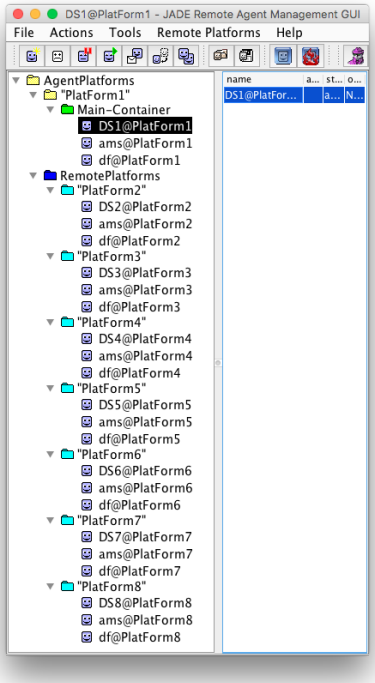
	Sat 1	Sat 2	Sat 3	Sat 4	Sat 5	Sat 6	Sat 7	Sat 8
HM (%)	100	80	100	100	100	50	25	75
HO (%)	100	100	60	100	75	100	25	50
HC (%)	100	100	100	40	75	50	100	25

SIMULATION ENVIRONMENT

To guarantee a fair comparison, the simulation environment needs to be addressed. As mentioned in Chapter 6.2, the system architecture is forced to switch from a centralized architecture to a distributed architecture due to the loss of the main controller (MS). The HDGA is performed by a group of DS where the single onboard computational power is much lower than the MS's. However, it is difficult to get realistic and reliable numbers about the onboard computers on the DSs and the MS. Therefore, to simulate the differences in the computational power, we employ two kinds of computers as a simple case. One is a high-end laptop (2017 MacBook Pro) to represent the MS for the centralized approach. This laptop has a 3.1 GHz Intel® Core™i5 7267U dual-processor, 8 GB of RAM, and MacOS V10.13.3 as operating system. The other hardware comprises four low-end laptops (2015 Surface Pro 4) where each core is assigned to represent one DS in the distributed approach. Each laptop has a 2.4 GHz Intel® Core™i5 6300U dual-processors, 8 GB of RAM and a Windows 10 operating system. Regarding the software, we have trans-



(a) GUI for centralized approach



(b) GUI for distributed approach

Figure 6.5: JADE software GUI for both approaches

ferred the previous MATLAB codes into the JAVA codes, in which the distributed architecture is programmed. The simulation environment is constructed by using the JADE (Java Agent DEvelopment Framework) in the *Eclipse*. The graphical user interfaces (GUI) for

both architectures are shown in the Fig 6.5. Fig 6.5a illustrates the centralized approach where all the satellites exist in the same platform called the "PlatForm1", while Fig 6.5b shows that each DS has its own platform and all the DSs are remotely connected with each other. This distributed system architecture is implemented for all the distributed approaches.

6.5.2. CENTRALIZED VS DISTRIBUTED

It is difficult to compare the centralized approach (CA) with the distributed approach (DA) considering the differences between a centralized planning problem and a distributed planning problem. Therefore, in this part, the statistical simulation results are only to show the capability of the proposed HDGA.

The first notable aspect is the success rate of the proposed approaches. Compared to previous studies, for artificial intelligence (AI) related approaches, usually their success rate cannot reach 100% [Ferber and Weiss, 1999; Russell and Norvig, 2016]. Both HDMGA and HDGA are based on the GA, which makes them inherit the uncertainty of the success rate from the normal GA. The second aspect is the accuracy rate of the proposed approaches. Both approaches have unique stop criteria to terminate the algorithm, which lead to different solutions for the same problem. To compare the success rate (SR) and accuracy rate (AR) of HDMGA and HDGA, we simulate multiple times to summarize statistical results on these two aspects. The SR stands for the success rate that this approach can provide a solution for the corresponding problem, while the AR is the percentage of accurate solutions in all solutions. Tab 6.3 shows SR on both approaches for five different objectives. All statistical results are gathered from 10 repetitive simulations with the same initial conditions. From this table, several conclusions can be drawn. Firstly,

Table 6.3: Statistical results of success rate and accuracy rate for HDMGA and HDGA

	Centralized Approach (HDMGA)								Distributed Approach (HDGA)							
	6 satellites		8 satellites		10 satellites		12 satellites		6 satellites		8 satellites		10 satellites		12 satellites	
	SR	AR	SR	AR	SR	AR	SR	AR	SR	AR	SR	AR	SR	AR	SR	AR
10 orbits	100%	100%	100%	90%	90%	90%	90%	80%	100%	90%	100%	90%	100%	80%	90%	70%
20 orbits	90%	90%	90%	90%	90%	90%	90%	70%	100%	90%	100%	80%	100%	80%	100%	80%
30 orbits	90%	80%	90%	80%	90%	80%	80%	80%	100%	80%	90%	70%	90%	70%	90%	70%
40 orbits	80%	80%	80%	70%	80%	70%	70%	70%	90%	70%	90%	70%	100%	70%	80%	60%
50 orbits	80%	70%	80%	70%	70%	70%	60%	50%	80%	70%	80%	60%	80%	60%	80%	50%

the overall success rates for DA are better than the CA. For CA, with increasing number of satellites and operation lifetime, the number of variables increases correspondingly. The larger number of variables makes the HDMGA much harder to generate suitable solutions. While DA relies on individual evaluation performed by each satellite, the chance to produce suitable solutions on each DS is much higher. Secondly, CA produces more accurate solutions compared to DA. The different system architectures lead to different procedures of convergence, where the DA chooses to employ a low standard on stopping criteria to make sure that the continuity of the distributed optimization procedures can be guaranteed.

Fig 6.6 shows the statistical results of one scenario from Tab 6.3. This scenario has 8 satellites as participants. The boxes represent the computation time for 10 runs. The upper part illustrates the results for the CA, while the lower part stands for the DA. All markers follow the same definition as in Fig 6.4. The average computation times for

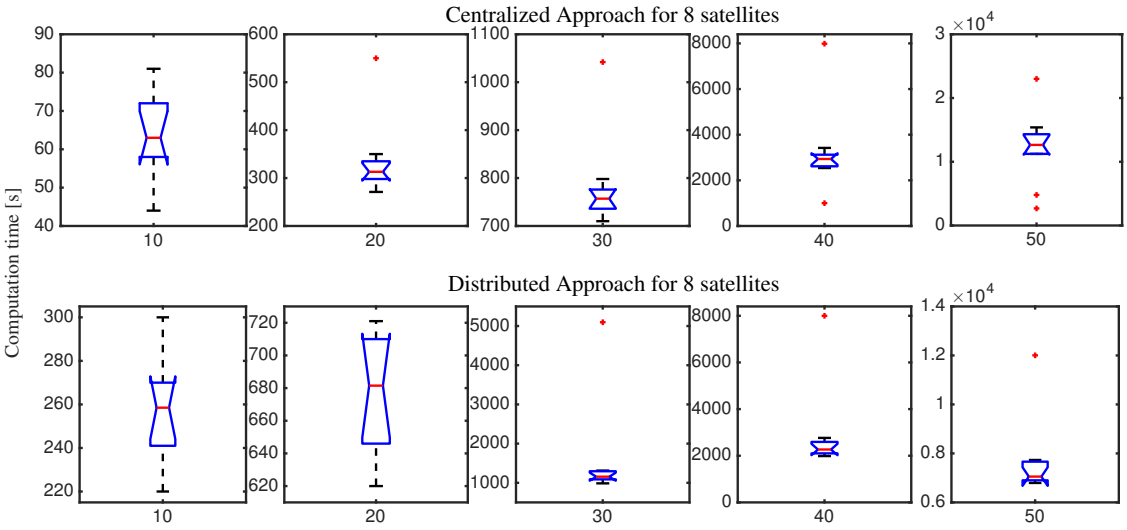
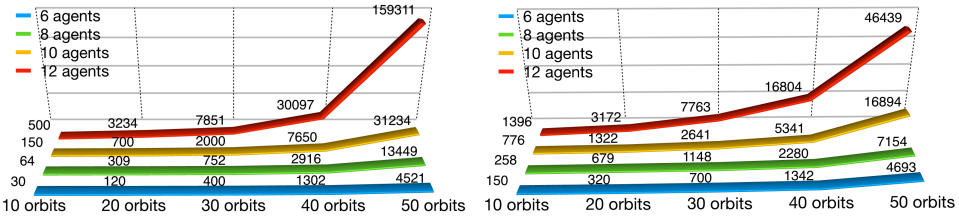


Figure 6.6: Computation time on 8 satellites for both approaches

both approaches are shown in Fig 6.7, where Fig 6.7a shows the CA results and Fig 6.7b represents results of DA.



(a) Results for the centralized approach

(b) Results for the distributed approach

Figure 6.7: Average computation time comparison for centralized and distributed approaches to solve five mission objectives using four different team sizes

From this figure, several conclusions can be drawn. Firstly, for short lifetime missions, the DA requires more computation time than the CA. This is because the DA requires team information exchange during each iteration, where some DSs have to wait for others. The CA can produce the solutions without any stop during the calculation. Secondly, with the increasing number of orbits, the DA requires less average time consumption compared with the CA. Meanwhile, for the same number of orbits, the differences between the two approaches are getting larger with the increasing number of agents. All these findings indicate that the DA can handle large-scale variable problems better than the CA. The time consumption of DA is larger than of CA when lifetime is lower than 30 orbits. However, the DA can achieve better or at least the same perfor-

Table 6.4: Success rates and accuracy rates for three distributed planning approaches

	HDGA		DACO		CPSO	
	SR	AR	SR	AR	SR	AR
10 orbits	100%	90%	100%	90%	100%	100%
20 orbits	100%	80%	90%	80%	100%	80%
30 orbits	90%	70%	90%	70%	90%	80%
40 orbits	90%	70%	80%	70%	80%	70%
50 orbits	80%	60%	70%	50%	60%	40%

mance on time consumption compared with the CA when the lifetime is 40 orbits, while the DA outperforms the CA for lifetime of 50 orbits due to the increasing number of the variables. The dimension of the constraints matrix for HDMGA is growing with a growing number of variables. This can increase the computation time exponentially. However, for DA, in the LCS module, each satellite only evaluates individual variables. This reduces the overall computation time. When using the HDGA, all satellites have to share the computation load within the team. Even with the same number of variables as CA, the number of iteration steps which each DS performs is much lower than for the CA. This leads to a lower computation time when compared with the CA for a larger number of variables. Fig 6.7 shows the average computation time for different team sizes. In the 12-agent scenario, CA consumes more time than DA when the lifetime exceeds 30 orbits, while for the 8-and 10-agent scenarios, this phenomenon happens for $30-40$ orbits. For the 6-agent scenario, the gap between the DA and the CA is becoming smaller when the lifetime enlarges.

Considering the comparison on success rate, accuracy rate, and average computation time, we are able to draw some conclusions on both approaches.

(1) When the mission requires the accuracy of the final solution as the priority, the centralized approach is always better than the distributed approach. The CA may, however, require more computation time than the DA.

(2) When the mission requires a successful operation as the first priority, the DA can provide better success rates than the CA. In this case, DS solutions may be less accurate.

(3) When the mission demands a quick reaction as first priority, for a small number of variables, CA should be chosen, and for large-scale problems, DA should be used.

6.5.3. DISTRIBUTED VS DISTRIBUTED

In this part, the distributed planning study case is employed to test the performance of the proposed HDGA against DACO and CPSO approaches. All three approaches use the same settings and the same distributed architecture. The statistical results are summarized from 10 simulation runs with the same initial setup. Tab 6.4 shows the SR and the AR of each approach on five different objectives. Based on this table, we observe that when the lifetime is below 30 orbits, the performance on success rates and accuracy for all three approaches is similar. For 10 and 30 orbits, the CPSO can provide more accurate

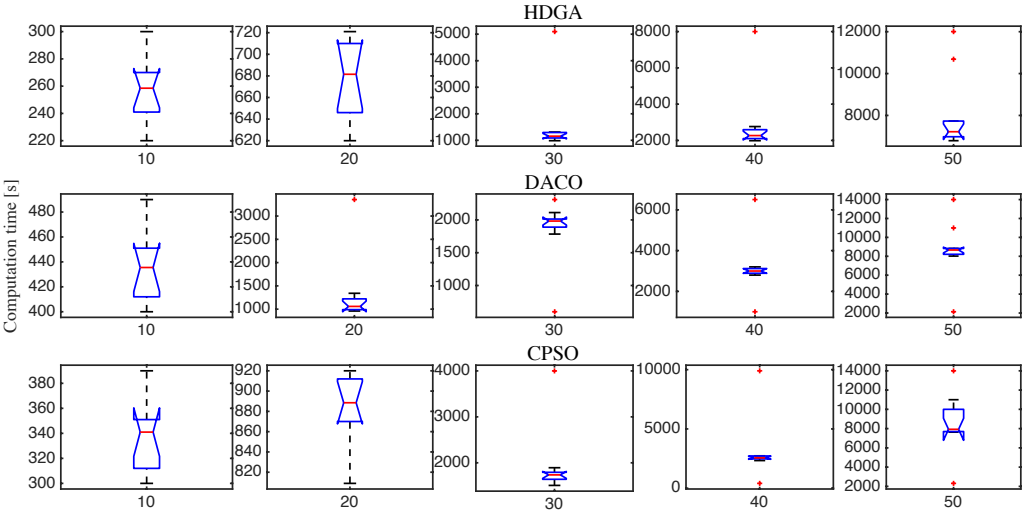


Figure 6.8: Statistic results on computation time for three approaches

solutions than the HDGA. However, when the lifetime exceeds 40 orbits, the SR starts to decrease for both DACO and CPSO, while the HDGA stays the same as the previous lifetime. When the lifetime reaches 50 orbits, the performance on both competitors is worse than the proposed method. This table also indicates that the proposed HDGA is more stable and accurate for dealing with large-scale problems.

Fig 6.8 illustrates the simulation results of the three approaches on computation time. The upper part is for HDGA, the middle part is for DACO, and the lower part is for CPSO. Each box represents the statistical results based on 10 simulation runs. The red pluses stand for the abnormal results which indicate the number of failures of each approach during the simulation. Based on the statistical results, the average computation time for each approach is shown in Fig 6.9.

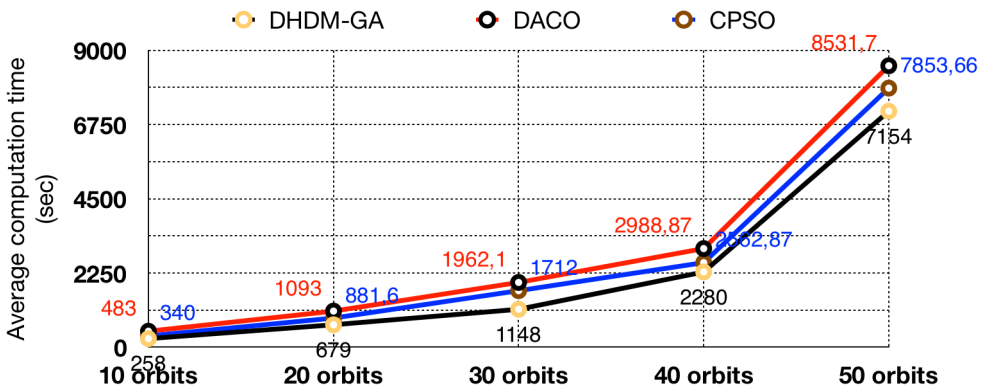


Figure 6.9: Average computation time for three approaches

In Fig 6.9, the black line represents the proposed HDGA, the blue line stands for the CPSO, and the red line denotes the DACO. The proposed HDGA presents the lowest average computation time among all three approaches, while the DACO is the highest. The main reason why HDGA is superior to the other two approaches is because it has a pre-selection procedure before the team evaluation procedure. This can help to avoid the global population of each iteration to generate infeasible solutions. For DACO and CPSO approaches, after the information sharing procedures are finished, each computation node can generate candidate solutions based on its current knowledge. For the distributed planning study case in this chapter, the global fitness function relies on strongly-coupled variables which come from different satellites. Independent evaluation approaches like the DACO and the CPSO may not provide sufficient solutions for other satellites. This causes these approaches to waste more time on retrieving the suitable candidates from previous generations. The proposed HDGA uses two level of information exchange procedures instead of one to improve the success rate of establishing the global population at each generation. This method can therefore avoid conflicts caused by individual constraints from other satellites. However, in this thesis, the time delay of the inter-satellite communication is not considered. With the current assumptions, then the HDGA can solve the same planning problem within a shorter time compared to the other two distributed approaches.

6.6. CONCLUSION

A novel hybrid distributed planning approach (HDGA) is presented for multiple-satellites systems. This approach consists of a local search heuristic and a distributed optimization algorithm. The local heuristic uses a greedy best-first search as the search algorithm for solving the local constraint satisfaction problems, while the HDMGA of Chapter 3 is used as a core algorithm for distributed optimization problems. The proposed approach is validated through similar results obtained for the centralized approach. Addressing several unique mission planning study cases derived from the DSL mission, the HDGA shows better performance for dealing with large-scale problems compared to the previous centralized approach. Compared with other state-of-the-art approaches for distributed planning problem, the HDGA shows a better performance on success rate, accuracy, as well as the average computation time. Therefore, the HDGA shows a good potential for solving distributed mission planning problems in future multi-satellite missions.

7

Conclusions

7.1. SUMMARY

WITH the increasing complexity of space applications, using multiple-satellite systems (MSSs) instead of single-satellite systems (SSSs) is becoming an inevitable trend due to its many advantages, such as higher robustness, increasing functionality and lower cost. However, implementing an MSS for a space mission will also bring many challenges, e.g. the operational complexity of MSS with a large number of satellites. A large number of operating satellites is a heavy burden for ground operators to control every sub-system of every satellite in the MSS. Meanwhile, for specific missions, such as deep space missions, the commands sent from the ground segment can be delayed or jammed due to a long communication distance, making it impossible to control an MSS from ground in real-time. Furthermore, for highly complex missions, the MSS may require a short reaction time to critical situations (e.g. collision avoidance). Also, the MSS may only be reached by a control center within a short communication window. These constraints will bring extra requirements on hardware and software capabilities on operations for both the MSS and the control center. Therefore, to enhance the efficiency of operating an MSS, and to reduce the cost of human resources and ground infrastructure, developing an onboard autonomous system (OAS) for MSS is a promising solution. One important function of an OAS is to provide the planning and re-planning services based on different mission requirements.

Traditional planning approaches have been reviewed and prove to be inefficient and inappropriate for complex planning problems in a harsh environment with severe system constraints and a large number of variables. In contrast, Artificial Intelligence (AI) approaches are more suitable for these complex problems due to their broad adaptability and their ability to deal with large-scale variables. Therefore, this thesis focuses on the development of onboard mission planning and re-planning approaches using AI techniques for supporting future highly capable OAS. The goal of developing these approaches is to enable the MSS to perform onboard autonomous mission planning and re-planning for various scenarios. In order to guide the research of this thesis, three research questions (RQs) were defined and formulated, as follows:

RQ1: What are the strengths of using AI in space missions? How to use a centralized AI algorithm in a multi-satellite system to decompose mission objectives and perform mission planning for the entire system?

RQ2: How to define emergency situations which may occur during mission operations? How to use AI algorithms to handle mission re-planning and rescheduling problems?

RQ3: How to design cooperation and negotiation approaches for an MSS to reach an agreement? How to improve the AI algorithm for distributed onboard mission planning problems?

Chapters 2 and 3 addressed and answered **RQ1**. In Chapter 2, we first introduced the reference mission of this thesis, called *Discovering the Sky at the Longest Wavelength (DSL)*, which scientific objective is to observe the universe in the hitherto-unexplored area, very low frequency (below 30 MHz) electromagnetic spectrum range in a lunar orbit. Then, existing work on mission planning problems for the multi-satellite system was reviewed in three categories: classical approaches, heuristic approaches, and advanced techniques (e.g. team negotiation mechanisms, distributed optimization approaches)

for distributed systems. Three classical approaches and six heuristic approaches were chosen from the related work as candidates for a preliminary selection. The goal of this selection was to identify a suitable approach for MSS to handle different types of optimization problems. Eight constrained and six unconstrained test functions were employed as benchmarks. The results of using these functions in simulations indicated that Evolutionary Algorithms (GA and DE) have a broader adaptability than classical approaches. They also have better efficiency than other heuristic approaches.

Following the preliminary selection of suitable algorithms in Chapter 2, Chapter 3 investigated how to use the selected AI algorithm to solve the mission initial planning problems under a centralized architecture, i.e. a Mother Satellite (MS) in charge of mission planning and re-planning and maintaining communication to several Daughter Satellite (DS), in our case 8 DSs. The mathematical model of the mission initial planning problem was formulated, including its notations, variables, constraints, and objective functions. By revealing the weaknesses of the basic Genetic Algorithm (GA), along with a comparison with several other improved GAs, the need of developing a new mutation strategy for GA has been motivated. The proposed mutation strategy is called Hybrid Dynamic Mutation (HDM) strategy, which contains two mutation operators, the normal mutation operator and the escape mutation operator. While the normal mutation operator uses a small mutation rate $R_{nor} = 0.05$ for approaching the global optimum, the escape mutation operator uses a larger mutation rate $R_{esc} = 0.55$ to allow an escape from local optima. The simulation results indicated that the proposed HDMGA has a superior performance on correctness and effectiveness than alternative GAs for the problem. Based on these findings, AI methods are considered as a promising category compared with traditional methods due to their flexibility and effectiveness to support onboard planning for an MSS. The proposed HDMGA has also been shown to provide a satisfying result for considered initial mission planning problems.

Chapter 4 addressed and answered **RQ2**. The architecture of the centralized mission planning and re-planning system was first described. This architecture includes four components, namely, the general mission planner, the DS plan executor and monitor, the decision-maker, and the re-planner. Three emergency scenarios were proposed based on the potential failures (e.g. space debris, meteorites, or failures on sensors, processors or actuators) which could occur to important sub-systems of each DS during operations. Two re-planning methods, one called the cyclically re-planning method (CRM), the other one the near real-time re-planning method (NRRM), were established and compared in fundamental terms. Both methods were tested and compared using the same simulation environment. Three simulation case studies were formulated based on the introduced three emergency scenarios. Each case was designed to represent a different level of failures on multiple DSs. Here, case A stood for a total operative dysfunctional of DS1, DS3 and DS5, case B stood for a partial failure of observation functionality on DS2, DS4, DS6 and DS8, and case C represented a partial failure of communication functionality on DS1, DS3, DS5 and DS7. To ensure the diversity of the simulation, the functionality status in each case was set differently. The performance of the proposed two methods was compared for three aspects: the total number of data observed from all DSs within a certain time frame, the total number of data the MS received from all DSs within a certain time frame, and the average computation time for re-planning. We

concluded that: (1) Both the total data obtained on DSs and the total data transmitted to the MS, the NRRM could observe and transmit more data than the CRM within a specific operation lifetime. (2) The NRRM required at least 16.5% more computational time than the CRM for emergency situations, while it required 26.6% less time than the CRM for normal situations. This indicates that for emergency situations, the CRM could provide re-planning sequences faster than the NRRM, while for the normal case, the NRRM is much faster than the CRM.

In Chapter 5, a much more severe scenario was treated, namely that the MS became non-functional in a emergency situation. This would render the MS unable to provide mission planning and re-planning services for the MSS. Without its main controller on the MS, a single DS cannot take over the responsibility of the MS due to its limited hardware. Instead, all DSs need to cooperate to jointly perform the distributed mission planning.

Chapter 5 also answered the first part of **RQ3**. Due to the loss of the MS, this chapter first introduced distributed and decentralized architectures which the MSS could encounter. These two architectures were different in terms of communication topologies. In the distributed architecture, each DS was connected to other DSs, while in the decentralized architecture, one DS was only linked with its neighbors. Considering that the mission allocation problems in different organizational architectures are highly similar to the information games in game theory, a game-theoretical model of the multi-satellite mission allocation (MSMA) problem was formulated. Three new negotiation mechanisms were introduced, compared and analyzed in theoretical terms. The Utility-based Regret Play (URP) negotiation mechanism was proposed for a MSMA problem using a distributed architecture. The URP inherited the ability to evaluate individual utility at each negotiation step from the Utility-based Fictitious Play, and the ability to regret the current choice and to choose another option in the previous negotiation steps from the Regret Matching Play. The Smoke Signal Play (SSP) and Broadcast-based Play (BBP) were developed for a decentralized architecture instead. The SSP was inspired by an old communication method called *Smoke Signal*, where each agent was considered as a smoke tower, passing information of utility to its neighbor after it. The BBP used broadcasting as the communication method, where each agent can transmit information to its neighbors. The numerical simulations were carried out through several connected computers, where the communication environments of these computers were designed to follow the corresponding organizational architecture. The simulation results showed that the URP could outperform the other three state-of-the-art mechanisms using the designed distributed study cases. For the decentralized architecture, the results revealed that both SSP and BBP could provide valid solutions for mission allocation problems. In addition, strengths and weaknesses of these methods were identified and compared with another state-of-the-art mechanism called Market-based Auction (MBA) through a statistic analyses. This analysis included the average number of negotiation steps, the average CPU time consumption, and the total number of transmitted packets. The BBP mechanism showed a better performance on saving the computation time compared to SSP and MBA. The SSP mechanism, on the other hand, showed the best performance with respect to power consumption.

Chapter 6 answered the second part of **RQ3**. Based on the allocation results gath-

ered from the previous chapter, this chapter focused on the development of a distributed mission planning approach, named hybrid distributed GA (HDGA). This approach contained two modules: the local constraints satisfaction module (LCS) and the globally distributed optimization module (GDO). In the LCS module, the greedy best-first search algorithm was employed as the local search heuristic to help each DS to find the suitable solutions which can satisfy individual constraints. This module was designed to generate multiple solutions to form local populations for the GDO. The GDO module employed the distributed HDMGA as the core optimization algorithm, while the individual populations were formed through the local populations exchanging procedure between one agent and all other agents. The proposed HDGA was tested by employing six instances of the TSP (Travelling Salesman Problem). Although test results indicated that the fitness value of HDGA was around 5% inferior than the centralized HDMGA, these results still justify the correctness and applicability of the proposed HDGA. When the HDGA was implemented for a standard planning case, the results indicated that it could reduce the computation time while ensuring a higher success rate compared to the centralized HDMGA. While comparing the HDGA with two other state-of-the-art distributed optimization algorithms, the distributed ant colony optimization (DACO) and the coevolutionary particle swarm optimization (CPSO), the statistical results indicated that HDGA was more stable and accurate to handle large-scale planning problems. Finally, the HDGA also showed the best performance on computation time among all tested distributed approaches.

7.2. CONCLUSIONS AND INNOVATIONS

This thesis was investigating the development of the onboard autonomous mission planning and re-planning system which can provide sufficient onboard control capabilities without any interference from ground operators. This section provides the conclusions and the main innovations of this thesis.

1. Evaluation of mission planning approaches for multi-satellite systems

Multi-satellite systems can be considered as multi-agent systems (MASs) with limitations on operation environment. To develop an onboard autonomous mission planning and re-planning system for an MSS, existing planning approaches need to be reviewed and compared thoroughly. The advantages and drawbacks of classical approaches and heuristic approaches have been, for the first time, reviewed from the perspective of space applications. The most relevant techniques and methods for distributed systems have been addressed as well, including the coordination mechanisms, the topology of MASs, and distributed optimization approaches. A careful selection methodology has been designed to sift through representative approaches to identify their strengths and the weaknesses. This preliminary selection covers almost all types of optimization problems. The specifically defined evaluation and selection criteria is used to find an approach with the best applicability and computational efficiency among the candidate approaches. The results of this selection have also provided a basic understanding of tested candidates for other research fields, such as selecting best-suited optimization algorithms for optimization problems of other multi-agent systems.

2. Onboard centralized autonomous planning

A new mutation strategy has been designed to overcome the shortcomings of the con-

ventional genetic algorithm (GA). This new algorithm called Hybrid Dynamic Mutation GA (HDMGA) helps the Mother Satellite (MS) to perform the onboard initial planning for the entire MSS. Considering the unique scientific requirements of our reference mission, the initial planning problem has been formulated as a mixed-constrained optimization problem. Based on the preliminary selection results, GA has been chosen as the solver for this mixed-constrained optimization problem. The goal of developing HDMGA was to compensate for shortcomings of the original algorithm in terms of computational efficiency and accuracy. The simulation results have indicated that the proposed HDMGA can handle complex optimization problems in a short computation time compared with other centralized planning approaches. The proposed algorithm can also be applied to other centralized planning scenarios which have a high number of participants, such as multi-robot and multi-UAV (Unmanned Aerial Vehicle) systems.

3. Onboard centralized autonomous re-planning
The architecture for the centralized mission planning and re-planning system has been developed, including four modules: general mission planner, plan executor and monitor, decision-maker, and re-planner. Considering the potential failures on DSs, three emergency situations have been developed and specified. These emergency situations are assumed to be critical representative scenarios due to the scientific requirements of the DSL mission. Targeting various mission priorities, two new re-planning methods have been designed. One is called the cyclically re-planning method (CRM), and the other one is called the near real-time re-planning method (NRRM). Both methods have been shown to be able to reproduce valid control sequences for designed emergency situations. The comparison has shown that the CRM has a better performance on the re-planning computation time, while the NRRM allows the MSS to collect more observations than the CRM. These re-planning methods are also applicable for other emergency situations and re-planning scenarios with the customized initial settings and constraints.

4. Multi-satellite negotiation mechanisms
The MS may face itself a fatal system failure. Under this circumstance, the organizational architecture of the MSS is forced to transfer from a centralized architecture to a non-centralized architecture due to the loss of the main controller on the MS. Considering the different communication topologies, the MSS could form, in this research, non-centralized architectures which can be divided into either a distributed architecture or a decentralized architecture. Both differ in the condition of the internal communication network. For the distributed architecture, a new negotiation mechanism called the Utility-based Regret Play (URP) has been developed. This new mechanism has been shown to be able to outperform most other state-of-the-art negotiation mechanisms under the same distributed negotiation scenario. For the decentralized architecture, two new negotiation mechanisms have been proposed, namely, the Smoke Signal Play (SSP) and the Broadcast-based Play (BBP). The SSP is designed to save total power consumption on inter-satellite communication, while the BBP is designed to provide negotiation results with a shorter computation time. Both mechanisms have been compared with a representative method (Market-based Auction) and have been proven to be superior in saving computation time and energy. Meanwhile, to implement the proposed mechanisms for the MSS using both the distributed and decentralized architectures, a vali-

dation platform has been developed and implemented based on the JADE (Java Agent DEvelopment Framework). The proposed negotiation mechanisms are also applicable for other multiple intelligent agent systems which require system-level negotiation and cooperation to accomplish one or more tasks.

5. Onboard distributed planning approach

A distributed planning approach called Hybrid Distributed GA (HDGA) has been developed in this thesis to solve MSS planning problems using a distributed architecture. The unique two-layer architecture of the HDGA consists of two modules: the local constraints satisfaction module (LCS) and the globally distributed optimization module (GDO). This innovate hybrid approach uses the LCS to simplify the impact of strongly-coupled constraints from the scientific requirements. It employs the GDO to increase the search space and accelerate the convergence time. Although the cross-verification results of the HDGA and the HDMGA have indicated that the accuracy performance of the distributed approach is inferior to the centralized approach, the HDGA can still provide solutions within an acceptable range (less than 5% difference). The advantage of the HDGA is that it has a broader range of applicability and can be used on problems that cannot be solved by a centralized approach. The comparison with other state-of-the-art distributed approaches also proves that the proposed method has the conspicuous advantages in terms of success ratio, accuracy, and computation time. With these advantages, the HDGA can be applied on other distributed systems with multiple independent agents (e.g. multi-robot systems, multi-UAV systems, multi-UUV (Unmanned Underwater Vehicle) systems).

7.3. OUTLOOK

In this thesis, several important conclusions have been formulated based on innovative contributions to develop the onboard mission planning and re-planning system for a multi-satellite system. Although some approaches and mechanisms have been thoroughly investigated and verified, some other assumptions and interesting research topics remain untouched for many reasons. Recommendations for future research are provided in the sequel.

1. **Influence of a more detailed orbital model**
   As described above, the DSL mission will be operated in a lunar orbit. All the planning and re-planning approaches developed in this thesis were modeled and constructed under the assumption of circular orbits. However, for real-world missions, the operating orbit will be elliptical to a certain extent and shaped by various gravitational and non-gravitational perturbations. A more detailed orbital model can alter the problem formulation and influence the performance of the proposed approaches. This needs to be further investigated to demonstrate the applicability of the proposed methods for non-circular orbits.

2. **Incorporation of more sub-systems related to DSL mission**
   In Chapter 3, the planning problem has been simplified to only concern two key sub-systems: the payload and the communication sub-system. The mathematical models and objective functions were all constructed based on these two sub-systems only. However, a real spacecraft requires many other sub-systems to fulfill

the mission objective. Under this circumstance, more variables and constraints need to be considered. The corresponding expansion of computational complexity needs to be further investigated to characterize the performance of the proposed approaches.

3. **More extensive and detailed database for emergency situation**
   According to Chapter 4, one key component for re-planning is the decision-maker. Its decision-making capability relies on the database it can access. In this thesis, to simplify the planning problems, only three emergency situations have been taken into consideration. To enhance the intelligence of the decision-maker, a more extended and detailed refined database should be established. With a comprehensive database, the proposed methods can react to more complex re-planning situations.

4. **Communication delay for negotiation**
   When performing team negotiation to decompose the main mission goal into subgoals and allocate them to each DS, the communication channels are assumed to be instantaneous, i.e. there is no time delay for transmitting messages between DSs. However, in real systems, it is unrealistic to ignore communication delays. Further investigation is required to identify the influence of latency on the proposed mechanisms.

5. **Development of ground test-bed**
   In this thesis, several approaches and methods have been developed to solve onboard mission planning problems. To verify the proposed methods, many simulation and cross-verifications have been designed and performed. However, actual operation will differ from simulations. A ground test-bed is therefore needed to test and verify the proposed methods under representative conditions. Some preliminary configurations, such a test-bed, have been done by employing several quadrotors (Matrix 100 from DJI company). Many other group maneuver capabilities are still needed or need to be further developed to enhance the applicability of this test-bed.

REFERENCES

Abdelkhalik, O. and A. Gad
2012. Dynamic-size multiple populations genetic algorithm for multigravity-assist trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 35(2):520–529.

Agnetis, A., P. B. Mirchandani, D. Pacciarelli, and A. Pacifici
2004. Scheduling problems with two competing agents. *Operations research*, 52(2):229–242.

Al-Yafi, K., H. Lee, and A. Mansouri
2009. Mtap-masim: a multi-agent simulator for the mobile task allocation problem. In *Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on*, Pp. 25–27. IEEE.

Alexander, J., M. Kaiser, J. Novaco, F. Grena, and R. Weber
1974. Scientific instrumentation of the radio-astronomy-explorer-2 satellite.

Alonso, F., M. J. Alvarez, and J. E. Beasley
2008. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59(7):963–976.

AlShawi, I. S., L. Yan, W. Pan, and B. Luo
2012. Lifetime enhancement in wireless sensor networks using fuzzy approach and a-star algorithm.

ASTRON
2018. Westerbrok synthesis radio telescope. https://www.astron.nl/radio-observatory/public/public-0. Accessed: 2018-07-23.

Attanasio, A., J.-F. Cordeau, G. Ghiani, and G. Laporte
2004. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387.

Aydin, M. E. and T. C. Fogarty
2004. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. *Journal of Intelligent Manufacturing*, 15(6):805–814.

Baker, A. D.
1998. A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: dispatching, scheduling, and pull. *Journal of Manufacturing Systems*, 17(4):297–320.

Baños, R., J. Ortega, C. Gil, A. FernáNdez, and F. De Toro
2013. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5):1696–1707.

Barer, M., G. Sharon, R. Stern, and A. Felner
2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search*.

Barua, A. and K. Khorasani
2011. Hierarchical fault diagnosis and health monitoring in satellites formation flight. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(2):223–239.

Beasley, J. E.
1996. *Advances in linear and integer programming*. Oxford University Press, Inc.

Bellman, R.
1954. The theory of dynamic programming. Technical report, RAND Corp Santa Monica CA.

Bellman, R. E. and S. E. Dreyfus
2015. *Applied dynamic programming*, volume 2050. Princeton university press.

Berry, R.
1970. Launch window and translunar, lunar orbit, and transearth trajectoryplanning and control for the apollo 11 lunar landing mission. In *8th Aerospace Sciences Meeting*, P. 24.

Betts, J. T.
1998. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207.

Bhandari, A. K., V. K. Singh, A. Kumar, and G. K. Singh
2014. Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using kapur's entropy. *Expert Systems with Applications*, 41(7):3538–3560.

Bluth, B. J. and M. Helppie
1986. Soviet space stations as analogs.

Boella, G. and R. Damiano
2002. A replanning algorithm for a reactive agent architecture. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Pp. 183–192. Springer.

Bonet, B. and H. Geffner
2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33.

Boonstra, A.-J., M. Garrett, G. Kruithof, M. Wise, A. van Ardenne, J. Yan, J. Wu, J. Zheng, E. K. Gill, J. Guo, et al.
2016. Discovering the sky at the longest wavelengths (dsl). In *Aerospace Conference, 2016 IEEE*, Pp. 1–20. IEEE.

Borrajo, D.
2013. Multi-agent planning by plan reuse. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, Pp. 1141–1142. International Foundation for Autonomous Agents and Multiagent Systems.

Brock, O. and O. Khatib
2000. Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, Pp. 550–555. IEEE.

Chakraborty, J., A. Konar, L. C. Jain, and U. K. Chakraborty
2009. Cooperative multi-robot path planning using differential evolution. *Journal of Intelligent & Fuzzy Systems*, 20(1, 2):13–27.

Chamseddine, A., D. Theilliol, Y. Zhang, C. Join, and C.-A. Rabbath
2015. Active fault-tolerant control system design with trajectory re-planning against actuator faults and saturation: Application to a quadrotor unmanned aerial vehicle. *International Journal of Adaptive Control and Signal Processing*, 29(1):1–23.

Chbichib, A., R. Mellouli, and H. Chabchoub
2012. Profitable vehicle routing problem with multiple trips: Modeling and variable neighborhood descent algorithm. *American Journal of Operational Research*, 2(6):104–119.

Chen, T. and S.-j. Xu
2006. A fuzzy controller for terminal approach of autonomous rendezvous and docking with non-cooperative target. *Journal of Astronautics*, 27(3):416–421.

Chen, Y., D. Zhang, M. Zhou, and H. Zou
2012. Multi-satellite observation scheduling algorithm based on hybrid genetic particle swarm optimization. In *Advances in Information Technology and Industry Applications*, Pp. 441–448. Springer.

Cheng, H.
2012. Genetic algorithms with hyper-mutation for dynamic load balanced clustering problem in mobile ad hoc networks. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, Pp. 1171–1176. IEEE.

Chiang, W.-C. and R. A. Russell
1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Chien, S., R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, et al.
2004. The eo-1 autonomous science agent. In *Proceedings of the Third International*

Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1, Pp. 420–427. IEEE Computer Society.

Chien, S. A., R. Knight, A. Stechert, R. Sherwood, and G. Rabideau
2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *AIPS*, Pp. 300–307.

Chilan, C. and B. Conway
2007. A space mission automaton using hybrid optimal control (aas 07-115). *Advances in the Astronautical Sciences*, 127(1):259.

Choi, H.-L., L. Brunet, and J. P. How
2009. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926.

Churchill, D. and M. Buro
2013. Portfolio greedy search and simulation for large-scale combat in starcraft. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, Pp. 1–8. IEEE.

Collings, J. and E. Kim
2014. A distributed and decentralized approach for ant colony optimization with fuzzy parameter adaptation in traveling salesman problem. In *Swarm Intelligence (SIS), 2014 IEEE Symposium on*, Pp. 1–9. IEEE.

Colorni, A., M. Dorigo, V. Maniezzo, et al.
1991. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, Pp. 134–142. Paris, France.

Cordeau, J.-F. and G. Laporte
2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

Das, G. P., T. M. McGinnity, S. A. Coleman, and L. Behera
2015. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *Journal of Intelligent & Robotic Systems*, 80(1):33–58.

Das, P. K., H. S. Behera, S. Das, H. K. Tripathy, B. K. Panigrahi, and S. Pradhan
2016. A hybrid improved pso-dv algorithm for multi-robot path planning in a clutter environment. *Neurocomputing*, 207:735–753.

Dijkstra, E. W.
1959. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dorn, J., M. Girsch, G. Skele, and W. Slany
1996. Comparison of iterative improvement techniques for schedule optimization. *European Journal of Operational Research*, 94(2):349–361.

Duan, H., Q. Luo, Y. Shi, and G. Ma
2013. Hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration. *IEEE Computational intelligence magazine*, 8(3):16–27.

Duan, H.-b., X.-y. Zhang, J. Wu, and G.-j. Ma
2009. Max-min adaptive ant colony optimization approach to multi-uavs coordinated trajectory replanning in dynamic and uncertain environments. *Journal of Bionic Engineering*, 6(2):161–173.

Dunham, D. W., R. W. Farquhar, J. V. McAdams, M. Holdridge, R. Nelson, K. Whittenburg, P. Antreasian, S. Chesley, C. Helfrich, W. M. Owen, et al.
2002. Implementation of the first asteroid landing. *Icarus*, 159(2):433–438.

Durfee, E. H., C. L. Ortiz Jr, M. J. Wolverton, et al.
1999. A survey of research in distributed, continual planning. *Ai magazine*, 20(4):13.

D'Amico, S., E. Gill, M. Garcia, and O. Montenbruck
2006. Gps-based real-time navigation for the prisma formation flying mission. In *3rd ESA workshop on satellite navigation user equipment technologies, NAVITEC*, Pp. 11–13. Citeseer.

Earl, M. G. and R. D'Andrea
2002. Modeling and control of a multi-agent system using mixed integer linear programming. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, Pp. 107–111. IEEE.

Edwards, R. and L. Glass
2000. Combinatorial explosion in model gene networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 10(3):691–704.

ESA
2018a. Jc2sat-ff (japan canada joint collaboration satellites – formation flying). https://directory.eoportal.org/web/eoportal/satellite-missions/j/jc2sat-ff. Accessed: 2018-07-19.

ESA
2018b. Proba (project for on-board autonomy). https://www.esa.int/Our_Activities/Space_Engineering_Technology/Proba_Missions. Accessed: 2018-07-19.

Evers, L., A. I. Barros, H. Monsuur, and A. Wagelmans
2014. Online stochastic uav mission planning with time windows and time-sensitive targets. *European Journal of Operational Research*, 238(1):348–362.

Fehse, W.
2003. *Automated rendezvous and docking of spacecraft*, volume 16. Cambridge university press.

Ferber, J. and G. Weiss
1999. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Fleetwood, K.
2004. An introduction to differential evolution. In *Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November, Brisbane, Australia*, Pp. 785–791.

Flint, M., M. Polycarpou, and E. Fernandez-Gaucherand
2002. Cooperative path-planning for autonomous vehicles using dynamic programming. *IFAC Proceedings Volumes*, 35(1):481–486.

Floudas, C. A. and P. M. Pardalos
1990. *A collection of test problems for constrained global optimization algorithms*, volume 455. Springer Science & Business Media.

Fudenberg, D. and D. K. Levine
1995. Consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19(5):1065–1089.

Gad, A. and O. Abdelkhalik
2011. Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization. *Journal of Spacecraft and Rockets*, 48(4):629–641.

Ganguly, S., N. Sahoo, and D. Das
2013. Multi-objective planning of electrical distribution systems using dynamic programming. *International Journal of Electrical Power & Energy Systems*, 46:65–78.

Gao, C., Z. Zhen, and H. Gong
2016. A self-organized search and attack algorithm for multiple unmanned aerial vehicles. *Aerospace Science and Technology*, 54:229–240.

Gao, K. B., G. H. Wu, and J. H. Zhu
2013. Multi-satellite observation scheduling based on a hybrid ant colony optimization. In *Advanced Materials Research*, volume 765, Pp. 532–536. Trans Tech Publ.

Gao, P.-A., Z.-X. Cai, and L.-L. Yu
2009. Evolutionary computation approach to decentralized multi-robot task allocation. In *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, volume 5, Pp. 415–419. IEEE.

Garcia, M. P., O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin
2009. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102–1110.

Gendreau, M., G. Laporte, C. Musaraganyi, and É. D. Taillard
1999. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26(12):1153–1173.

Gerkey, B. P., S. Thrun, and G. Gordon
2005. Parallel stochastic hill-climbing with small teams. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, Pp. 65–77. Springer.

Gill, E., O. Montenbruck, K. Arichandran, S. Tan, et al.
2004. High-precision onboard orbit determination for small satellites-the gps-based xnson x-sat. In *Small Satellites, Systems and Services*, volume 571.

Gill, E., O. Montenbruck, and K. Brieß
2000. Gps-based autonomous navigation for the bird satellite. In *15th International Symposium on Spaceflight Dynamics, Biarritz, Germany*.

Gill, E., O. Montenbruck, and S. D'Amico
2007. Autonomous formation flying for the prisma mission. *Journal of Spacecraft and Rockets*, 44(3):671–681.

Gill, E., O. Montenbruck, and H. Kayal
2001. The bird satellite mission as a milestone toward gps-based autonomous navigation. *Navigation*, 48(2):69–75.

Giovanini, L., J. Balderud, and R. Katebi
2007. Autonomous and decentralized mission planning for clusters of uuvs. *International Journal of Control*, 80(7):1169–1179.

Globus, A., J. Crawford, J. Lohn, and R. Morris
2002. Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach.

Globus, A., J. Crawford, J. Lohn, and A. Pryor
2003. Scheduling earth observing satellites with evolutionary algorithms.

Glover, F.
1989. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206.

Glover, F.
1990. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32.

Gong, D.-w., J.-h. Zhang, and Y. Zhang
2011. Multi-objective particle swarm optimization for robot path planning in environment with danger sources. *Journal of computers*, 6(8):1554–1561.

Gong, Y.-J., W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li
2015. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300.

Gounley, R., R. White, and E. Gai
1984. Autonomous satellite navigation by stellar refraction. *Journal of guidance, control, and dynamics*, 7(2):129–134.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. R. Kan
1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, Pp. 287–326. Elsevier.

Grecu, D. and P. Gonsalves
2000. Agent-based simulation environment for ucav mission planning and execution. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, P. 4481.

Guglieri, G., F. Maroglio, P. Pellegrino, and L. Torre
2014. Design and development of guidance navigation and control algorithms for spacecraft rendezvous and docking experimentation. *Acta Astronautica*, 94(1):395–408.

Gusev, G., B. Lomonosov, K. Pichkhadze, N. Polukhina, V. Ryabov, T. Saito, V. Sysoev, E. Feinberg, V. Tsarev, and V. Chechin
2006. Detection of ultrahigh-energy cosmic rays and neutrinos by radio method using artificial lunar satellites. *Cosmic Research*, 44(1):19–38.

Han, X., S. Mandal, K. R. Pattipati, D. L. Kleinman, and M. Mishra
2014. An optimization-based distributed planning algorithm: a blackboard-based collaborative framework. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(6):673–686.

Hart, P. E., N. J. Nilsson, and B. Raphael
1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hart, S. and A. Mas-Colell
2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150.

Hedar, A.-R.
2018a. Constrained test functions. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page422.htm. Accessed: 2018-08-08.

Hedar, A.-R.
2018b. Global optimization test problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm. Accessed: 2018-07-23.

Hogbom, J. and W. Brouw
1974. The synthesis radio telescope at westerbork. principles of operation, performance and data reduction. *Astronomy and Astrophysics*, 33:289.

Holmes, W.
1978. Nasa's tracking and data relay satellite system. *IEEE Communications Society Magazine*, 16(5):13–20.

Hong, T.-P. and H.-S. Wang
1996. A dynamic mutation genetic algorithm. In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, volume 3, Pp. 2000–2005. IEEE.

Hoos, H. H.
1999. On the run-time behaviour of stochastic local search algorithms for sat. In *AAAI/IAAI*, Pp. 661–666.

Howard, A., M. J. Mataric, and G. S. Sukhatme
2002. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed autonomous robotic systems*, 5:299–308.

IEEE
2018. The foundation for intelligent physical agents (fipa). http://www.fipa.org. Accessed: 2018-11-17.

Izzo, D. and L. Pettazzi
2007. Autonomous and distributed motion planning for satellite swarm. *Journal of Guidance, Control, and Dynamics*, 30(2):449–459.

Jiang, T., K. Khorasani, and S. Tafazoli
2008. Parameter estimation-based fault detection, isolation and recovery for nonlinear satellite models. *IEEE Transactions on control systems technology*, 16(4):799–808.

Jin, C., F. Li, M. Wilamowska-Korsak, L. Li, and L. Fu
2014. Bsp-ga: A new genetic algorithm for system optimization and excellent schema selection. *Systems Research and Behavioral Science*, 31(3):337–352.

Jin, L. and S. Li
2016. Distributed task allocation of multiple robots: A control perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48:693–701.

Johnson, D. S.
1990. Local optimization and the traveling salesman problem. In *International colloquium on automata, languages, and programming*, Pp. 446–461. Springer.

Juels, A. and M. Wattenberg
1996. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. In *Advances in Neural Information Processing Systems*, Pp. 430–436.

Kaiser, M. L., T. Kucera, J. Davila, O. S. Cyr, M. Guhathakurta, and E. Christian
2008. The stereo mission: An introduction. *Space Science Reviews*, 136(1-4):5–16.

Kang, W., A. Sparks, and S. Banda
2001. Coordinated control of multisatellite systems. *Journal of Guidance, Control, and Dynamics*, 24(2):360–368.

Kanury, S. and Y. Song
2006. Flight management of multiple aerial vehicles using genetic algorithms. In *2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory*, Pp. 33–37. IEEE.

Karr, C. L., L. M. Freeman, and D. L. Meredith
1990. Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm. In *Intelligent Control and Adaptive Systems*, volume 1196, Pp. 274–289. International Society for Optics and Photonics.

Karypis, G. and V. Kumar
2000. Multilevel k-way hypergraph partitioning. *VLSI design*, 11(3):285–300.

Kawano, I., M. Mokuno, T. Kasai, and T. Suzuki
1999. Result and evaluation of autonomous rendezvous docking experiment of ets-vii. In *Guidance, Navigation, and Control Conference and Exhibit*, P. 4073.

Keinänen, H.
2009. Simulated annealing for multi-agent coalition formation. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Pp. 30–39. Springer.

Kennedy, J.
2011. Particle swarm optimization. In *Encyclopedia of machine learning*, Pp. 760–766. Springer.

Koes, M., K. Sycara, and I. Nourbakhsh
2006. A constraint optimization framework for fractured robot teams. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, Pp. 491–493. ACM.

Korf, R. E.
1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109.

Korf, R. E.
1990. Depth-limited search for real-time problem solving. *Real-Time Systems*, 2(1-2):7–24.

Kose, H., U. Tatlidede, C. Meriçli, K. Kaplan, and H. L. Akin
2004. Q-learning based market-driven multi-agent collaboration in robot soccer. In *Proceedings of the Turkish symposium on artificial intelligence and neural networks*, Pp. 219–228.

Krohling, R. A. and L. dos Santos Coelho
2006. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1407–1416.

Kwek, S.
1997. On a simple depth-first search strategy for exploring unknown graphs. In *Workshop on Algorithms and Data Structures*, Pp. 345–353. Springer.

Lang, F. and A. Fink
2012. Collaborative single and parallel machine scheduling by autonomous agents. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, Pp. 76–83. IEEE.

Lang, F., A. Fink, and T. Brandt
2016. Design of automated negotiation mechanisms for decentralized heterogeneous machine scheduling. *European Journal of Operational Research*, 248(1):192–203.

Lengyel, J., M. Reichert, B. R. Donald, and D. P. Greenberg
1990. *Real-time robot motion planning using rasterizing computer graphics hardware*, volume 24. ACM.

Lerman, K., C. Jones, A. Galstyan, and M. J. Matarić
2006. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241.

Leung, C., T. Wong, K.-L. Mak, and R. Y. Fung
2010. Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers & Industrial Engineering*, 59(1):166–180.

Li, F. and T. Zhang
2009. Study on genetic algorithm based on schema mutation and its performance analysis. In *Electronic Commerce and Security, 2009. ISECS'09. Second International Symposium on*, volume 1, Pp. 548–551. IEEE.

Li, J. and X.-x. Sun
2008. A route planning's method for unmanned aerial vehicles based on improved a-star algorithm [j]. *Acta Armamentarii*, 7:788–792.

Li, S., R. Mehra, R. Smith, and R. Beard
2000. Multi-spacecraft trajectory optimization and control using genetic algorithm techniques. In *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*, volume 7, Pp. 99–108. IEEE.

Likhachev, M., D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun
2005. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, Pp. 262–271.

Lipowski, A. and D. Lipowska
2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196.

Liu, S., Y. Tian, and J. Liu
2004. Multi mobile robot path planning based on genetic algorithm. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 5, Pp. 4706–4709. IEEE.

LIU, S.-h., Y. Zhang, H.-y. WU, and J. Liu
2010. Multi-robot task allocation based on swarm intelligence. *Journal of Jilin University (Engineering and Technology Edition)*, 1:025.

Matossian, M. G.
1996. Earth observing system constellation design optimization through mixed integer programming. *Space Technology*, 5(16):233–243.

Mayfield, J., Y. Labrou, and T. Finin
1995. Evaluation of kqml as an agent communication language. In *International Workshop on Agent Theories, Architectures, and Languages*, Pp. 347–360. Springer.

McCurdy, H. E.
2001. *Faster, better, cheaper: Low-cost innovation in the US space program*. JHU Press.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller
1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mingyong, L. and C. Erbao
2010. An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2):188–195.

Mini, S., S. K. Udgata, and S. L. Sabat
2014. Sensor deployment and scheduling for target coverage problem in wireless sensor networks. *IEEE Sensors Journal*, 14(3):636–644.

Moehlman, T. A., V. R. Lesser, and B. L. Buteau
1992. Decentralized negotiation: An approach to the distributed planning problem. *Group decision and Negotiation*, 1(2):161–191.

Moore, E. F.
1959. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, Pp. 285–292.

Mukundan, R., R. Narayanan, and N. Philip
1994. A vision based attitude and position estimation algorithm for rendezvous and docking. *In its Journal of Spacecraft Technology, Volume 4, No. 2, July 1994 p 60-67 (SEE N95-13825 03-12)*, 4.

Murphey, R. A.
2000. Target-based weapon target assignment problems. In *Nonlinear assignment problems*, Pp. 39–53. Springer.

Muscettola, N., P. P. Nayak, B. Pell, and B. C. Williams
1998. Remote agent: To boldly go where no ai system has gone before. *Artificial intelligence*, 103(1-2):5–47.

NASA
2018a. Magnetospheric multiscale mission (mms). https://mms.gsfc.nasa.gov/. Accessed: 2018-07-18.

NASA
2018b. Multi-angle imager for aerosols. https://www.jpl.nasa.gov/missions/multi-angle-imager-for-aerosols-maia/. Accessed: 2018-07-17.

Ng, K. Y. and N. Sancho
2001. A hybrid 'dynamic programming/depth-first search'algorithm, with an application to redundancy allocation. *Iie Transactions*, 33(12):1047–1058.

Nikolos, I. K. and A. N. Brintaki
2005. Coordinated uav path planning using differential evolution. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, Pp. 549–556. IEEE.

Nilsson, N. J.
1971. Problem-solving methods in artificial intelligence. *Me Graw H*, 111.

Ning, X., L. Wang, X. Bai, and J. Fang
2013. Autonomous satellite navigation using starlight refraction angle measurements. *Advances in space research*, 51(9):1761–1772.

Nissim, R. and R. I. Brafman
2012. Multi-agent a* for parallel and distributed systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, Pp. 1265–1266. International Foundation for Autonomous Agents and Multiagent Systems.

Orcutt, M.
2018. Space over time. https://www.technologyreview.com/s/425120/space-over-time/. Accessed: 2018-07-17.

Patek, S. D., D. A. Logan, and D. A. Castanon
1999. Approximate dynamic programming for the solution of multiplatform path planning problems. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, Pp. 1061–1066. IEEE.

Pehlivanoglu, Y. V.
2012. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. *Aerospace Science and Technology*, 16(1):47–55.

Pei, Y., G. Jiang, X. Yang, and Z. Wang
2004. Auto-master-slave control technique of parallel inverters in distributed ac power systems and ups. In *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, volume 3, Pp. 2050–2053. IEEE.

Pohl, I.
1971. Bi-directional search. *Machine intelligence*, 6:127–140.

Ponda, S., J. Redding, H.-L. Choi, J. P. How, M. Vavrina, and J. Vian
2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC), 2010*, Pp. 3998–4003. IEEE.

Ponda, S. S., L. B. Johnson, A. Geramifard, and J. P. How
2015. Cooperative mission planning for multi-uav teams. *Handbook of Unmanned Aerial Vehicles*, Pp. 1447–1490.

Psiaki, M. L., L. Huang, and S. M. Fox
1993. Ground tests of magnetometer-based autonomous navigation (magnav) for low-earth-orbiting spacecraft. *Journal of Guidance, Control, and Dynamics*, 16(1):206–214.

Pugh, J. and A. Martinoli
2007. Inspiring and modeling multi-robot search with particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, Pp. 332–339. IEEE.

Quantalysus
2018. Different of system architectures. https://quantalysus.com/2018/04/30/the-byzantine-generals-problem/. Accessed: 2018-07-17.

Rabideau, G., S. Chien, J. Willis, and T. Mann
1999a. Using iterative repair to automate planning and scheduling of shuttle payload operations. In *AAAI/IAAI*, Pp. 813–820.

Rabideau, G., R. Knight, S. Chien, A. Fukunaga, and A. Govindjee
1999b. Iterative repair planning for spacecraft operations using the aspen system. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, P. 99.

Raffard, R. L., C. J. Tomlin, and S. P. Boyd
2004. Distributed optimization for cooperative agents: Application to formation flight. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, Pp. 2453–2459. IEEE.

Rakshit, P., A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar
2013. Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):814–831.

Reinhelt, G.
2014. {TSPLIB}: a library of sample instances for the tsp (and related problems) from various sources and of various types. http://comopt.ifi.uniheidelberg.de/software/TSPLIB95. Accessed: 2018-07-17.

Reinl, C. and O. Von Stryk
2007. Optimal control of multi-vehicle-systems under communication constraints using mixed-integer linear programming. In *Proceedings of the 1st international conference on Robot communication and coordination*, P. 3. IEEE Press.

Richards, A. and J. P. How
2002. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, Pp. 1936–1941. IEEE.

Richards, A., T. Schouwenaars, J. P. How, and E. Feron
2002. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764.

Rocha, R., F. Ferreira, and J. Dias
2008. Multi-robot complete exploration using hill climbing and topological recovery. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Pp. 1884–1889. IEEE.

Romano, M., D. A. Friedman, and T. J. Shay
2007. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. *Journal of Spacecraft and Rockets*, 44(1):164–173.

Ruel, S., T. Luu, and A. Berube
2012. Space shuttle testing of the tridar 3d rendezvous and docking sensor. *Journal of Field robotics*, 29(4):535–553.

Russell, S. J. and P. Norvig
2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Sabra, W., M. Khouzam, A. Chanu, and S. Martel
2006. Use of 3d potential field and an enhanced breadth-first search algorithms for the path planning of microdevices propelled in the cardiovascular system. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, Pp. 3916–3920. IEEE.

SalarKaleji, F. and A. Dayyani
2013. A survey on fault detection, isolation and recovery (fdir) module in satellite onboard software. In *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*, Pp. 545–548. IEEE.

Saska, M., M. Macas, L. Preucil, and L. Lhotska
2006. Robot path planning using particle swarm optimization of ferguson splines. In *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, Pp. 833–839. IEEE.

Schetter, T., M. Campbell, and D. Surka
2003. Multiple agent-based autonomy for satellite constellations. *Artificial Intelligence*, 145(1-2):147–180.

Schouwenaars, T., B. De Moor, E. Feron, and J. How
2001. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, Pp. 2603–2608. IEEE.

Searle, J. R.
1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Semsar, E. and K. Khorasani
2007. Optimal control and game theoretic approaches to cooperative control of a team of multi-vehicle unmanned systems. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, Pp. 628–633. IEEE.

Shamma, J.
2008. *Cooperative control of distributed multi-agent systems*. John Wiley & Sons.

Shao, G., F. Berman, and R. Wolski
2000. Master/slave computing on the grid. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, Pp. 3–16. IEEE.

Shehory, O. and S. Kraus
1998. Methods for task allocation via agent coalition formation. *Artificial intelligence*, 101(1):165–200.

Sheikh, S. I., D. J. Pines, P. S. Ray, K. S. Wood, M. N. Lovellette, and M. T. Wolff
2006. Spacecraft navigation using x-ray pulsars. *Journal of Guidance, Control, and Dynamics*, 29(1):49–63.

Shen, W. and D. H. Norrie
1999. Agent-based systems for intelligent manufacturing: a state-of-the-art survey. *Knowledge and information systems*, 1(2):129–156.

Shi, G., H. Iima, and N. Sannomiya
1996. A new encoding scheme for solving job shop problems by genetic algorithm. In *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, volume 4, Pp. 4395–4400. IEEE.

Shorshi, G. and I. Y. Bar-Itzhack
1995. Satellite autonomous navigation based on magnetic field measurements. *Journal of Guidance, Control, and Dynamics*, 18(4):843–850.

Shtub, A., L. J. LeBlanc, and Z. Cai
1996. Scheduling programs with repetitive projects: a comparison of a simulated annealing, a genetic and a pair-wise swap algorithm. *European Journal of Operational Research*, 88(1):124–138.

Shuai, P., S. Chen, Y. Wu, C.-q. ZHANG, and M. LI
2007. Navigation principles using x-ray pulsars. *Journal of Astronautics*, 6:021.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo
2010. *Robotics: modelling, planning and control*. Springer Science & Business Media.

Smyrnakis, M. and S. M. Veres
2016. Fictitious play for cooperative action selection in robot teams. *Engineering Applications of Artificial Intelligence*, 56:14–29.

Soma, P., S. Venkateswarlu, S. Santhalakshmi, T. Bagchi, and S. Kumar
2004. Multi-satellite scheduling using genetic algorithms. In *Space OPS 2004 Conference*, P. 515.

Srinivas, M. and L. M. Patnaik
1994a. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667.

Srinivas, M. and L. M. Patnaik
1994b. Genetic algorithms: A survey. *computer*, 27(6):17–26.

Stentz, A. et al.
1995. The focussed dˆ* algorithm for real-time replanning. In *IJCAI*, volume 95, Pp. 1652–1659.

Storn, R. and K. Price
1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Sun, B., W. Wang, X. Xie, and Q. Qin
2010. Satellite mission scheduling based on genetic algorithm. *Kybernetes*, 39(8):1255–1261.

Surjanovic, S. and D. Bingham
2018a. Virtual library of simulation experiments: Test functions and datasets. https://www.sfu.ca/~ssurjano/index.html. Accessed: 2018-08-08.

Surjanovic, S. and D. Bingham
2018b. Virtual library of simulation experiments: Test functions and datasets. http://www.sfu.ca/~ssurjano. Accessed: 2018-08-17.

Tanenbaum, A. S. and M. Van Steen
2007. *Distributed systems: principles and paradigms*. Prentice-Hall.

Tang, F. and L. E. Parker
2007. A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation. In *Robotics and Automation, 2007 IEEE International Conference on*, Pp. 3351–3358. IEEE.

Tang, H. and E. Miller-Hooks
2005. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407.

Tangpattanakul, P., N. Jozefowiez, and P. Lopez
2015a. Biased random key genetic algorithm for multi-user earth observation scheduling. In *Recent Advances in Computational Optimization*, Pp. 143–160. Springer.

Tangpattanakul, P., N. Jozefowiez, and P. Lopez
2015b. A multi-objective local search heuristic for scheduling earth observations taken by an agile satellite. *European Journal of Operational Research*, 245(2):542–554.

Tarjan, R.
1972. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160.

Thangiah, S. R., I. H. Osman, and T. Sun
1994. Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. *Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27*, 69.

Tompkins, P., A. Stentz, and D. Wettergreen
2006. Mission-level path planning and re-planning for rover exploration. *Robotics and Autonomous Systems*, 54(2):174–183.

Torreño, A., E. Onaindia, and Ó. Sapena
2012. An approach to multi-agent planning with incomplete information. In *20th European Conference of Artificial Intelligence (ECAI 2012)*, volume 242, Pp. 762–767.

Truszkowski, W., M. Hinchey, J. Rash, and C. Rouff
2004. Nasa's swarm missions: The challenge of building autonomous software. *IT professional*, 6(5):47–52.

Tsang, E.
2014. *Foundations of constraint satisfaction: the classic text*. BoD–Books on Demand.

University, H.
2018. {TSPLIB}: a library of sample instances for the tsp (and related problems) from various sources and of various types. https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Accessed: 2018-07-23.

Walsh, W. E. and M. P. Wellman
1998. A market protocol for decentralized task allocation. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, Pp. 325–332. IEEE.

Wang, H., M. Xu, R. Wang, and Y. Li
2009. Scheduling earth observing satellites with hybrid ant colony optimization algorithm. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 2, Pp. 245–249. IEEE.

Whitley, D.
1994. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

Wiegand, M.
1996. Autonomous satellite navigation via kalman filtering of magnetometer data. *Acta Astronautica*, 38(4-8):395–403.

Wikipedia
2018a. Astérix. https://en.wikipedia.org/wiki/Ast%C3%A9rix_(satellite). Accessed: 2018-08-17.

Wikipedia
2018b. Canadian advanced nanospace experiment program. https://en.wikipedia.org/wiki/Canadian_Advanced_Nanospace_eXperiment_Program. Accessed: 2018-07-19.

Wikipedia
2018c. Dong fang hong i. https://en.wikipedia.org/wiki/Dong_Fang_Hong_I. Accessed: 2018-08-17.

Wikipedia
2018d. Explorer 1. https://en.wikipedia.org/wiki/Explorer_1. Accessed: 2018-07-17.

Wikipedia
2018e. Gravity recovery and climate experiment (grace). https://en.wikipedia.org/wiki/Gravity_Recovery_and_Climate_Experiment. Accessed: 2018-07-18.

Wikipedia
2018f. Gravity recovery and interior laboratory (grail). https://en.wikipedia.org/wiki/GRAIL. Accessed: 2018-07-18.

Wikipedia
2018g. Job-shop scheudling. https://en.wikipedia.org/wiki/Job_shop_scheduling#Example. Accessed: 2018-07-23.

Wikipedia
2018h. Morning and afternoon constellation. https://satellitesafety.gsfc.nasa.gov/missions.html#morning. Accessed: 2018-07-18.

Wikipedia
2018i. Ohsumi. https://en.wikipedia.org/wiki/Ohsumi_(satellite). Accessed: 2018-08-17.

Wikipedia
2018j. Orbital express. https://en.wikipedia.org/wiki/Orbital_Express. Accessed: 2018-08-17.

Wikipedia
2018k. Smoke signal. https://en.wikipedia.org/wiki/Smoke_signal. Accessed: 2018-08-17.

Wikipedia
2018l. Sputnik 1. https://en.wikipedia.org/wiki/Sputnik_1. Accessed: 2018-07-17.

Wikipedia
2018m. Tandem-x. https://en.wikipedia.org/wiki/TanDEM-X. Accessed: 2018-07-18.

Williams, E. A. and W. A. Crossley
1998. Empirically-derived population size and mutation rate guidelines for a genetic algorithm with uniform crossover. In *Soft computing in engineering design and manufacturing*, Pp. 163–172. Springer.

Wu, W., X. Wang, and N. Cui
2018. Fast and coupled solution for cooperative mission planning of multiple heterogeneous unmanned aerial vehicles. *Aerospace Science and Technology*.

Xhafa, F., J. Sun, A. Barolli, A. Biberaj, and L. Barolli
2012. Genetic algorithms for satellite scheduling problems. *Mobile Information Systems*, 8(4):351–377.

Yilmaz, N. K., C. Evangelinos, P. F. Lermusiaux, and N. M. Patrikalakis
2008. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537.

Yingying, D., H. Yan, and J. Jingping
2003. Multi-robot cooperation method based on the ant algorithm. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, Pp. 14–18. IEEE.

Yuejin, L. Y. C. Y. T.
2005. The method of mission planning of the ground station of satellite based on dynamic programming [j]. *Chinese Space Science and Technology*, 1:007.

Yunfeng, D. and Z. Renwei
1995. Autonomous satellite navigation by stellar sensor [j]. *Journal of Astronautics*, 4.

Zak, A.
2018a. Mars exploration rover. http://www.russianspaceweb.com/spektr_origin.html. Accessed: 2018-11-17.

Zak, A.
2018b. Origin of the spektr series. http://www.russianspaceweb.com/spektr_origin.html. Accessed: 2018-11-17.

Zhang, X., P. Shuai, L. Huang, S. Chen, and L. Xu
2017. Mission overview and initial observation results of the x-ray pulsar navigation-i satellite. *International Journal of Aerospace Engineering*, 2017.

Zhang, Z., N. Zhang, and Z. Feng
2014. Multi-satellite control resource scheduling based on ant colony optimization. *Expert Systems with Applications*, 41(6):2816–2823.

Zhen, Z., D. Xing, and C. Gao
2018. Cooperative search-attack mission planning for multi-uav based on intelligent self-organized algorithm. *Aerospace Science and Technology*, 76:402–411.

Zheng, X. and S. Koenig
2009. K-swaps: Cooperative negotiation for solving task-allocation problems. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, volume 9, Pp. 373–378.

Zidi, I., K. Zidi, K. Mesghouni, and K. Ghedira
2011. A multi-agent system based on the multi-objective simulated annealing algorithm for the static dial a ride problem. In *18th World Congress of the International Federation of Automatic Control (IFAC), Milan (Italy)*.

Zlot, R. and A. Stentz
2005. Complex task allocation for multiple robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, Pp. 1515–1522. IEEE.

Zlot, R., A. Stentz, M. B. Dias, and S. Thayer
2002. Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA'02.*, volume 3, Pp. 3016–3023. IEEE.

Zlotkin, G. and J. S. Rosenschein
1996. Mechanisms for automated negotiation in state oriented domains. *Journal of Artificial Intelligence Research*, 5:163–238.

Zweben, M., E. Davis, B. Daun, and M. J. Deale
1993. Scheduling and rescheduling with iterative repair. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1588–1596.

ACKNOWLEDGEMENTS

When I start to write this acknowledgments, I suddenly realized that it's already been four years since I first got here. I spent my most valuable four years here in the Delft to pursue my doctoral degree and have enjoyed this harmony and friendly working environment. For all these, I would like to take this opportunity to express my appreciations to those who have supported and helped me.

Firstly, I would like to express my sincere gratitude to my promoter Prof. Eberhard Gill for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. His rigorous and careful attitude towards research is the most precious wealth that he taught me, and this will benefit me for lift. He has been very kind and patient and always willing to lend his service whenever I approached him and I acknowledge and appreciate him for all his efforts. I could not have imagined having a better promoter for my Ph.D study. For my daily supervisor Dr. Jian Guo, I want to express my special thanks for his patience, his support, his guidance, and his enlighten. I am really going to miss our discussions on both research and life, you are more than just my supervisor, but also my good friend.

Secondly, I want to say thank you to all my colleagues in the SSE group. Debby and Marielle are wonderful secretaries who are very patient to help me deal with a lot of trivial things. Angelo, Silvana, Nikitas, and Mehmet, I really enjoyed our PingPong time together and hope one day in the future we can play again. For Vidhya, I wish all the best for your coming baby and your "God knows where" driver's license. I am also grateful to the following staff: Hans, Samiksha, Chris, Trevor, Jasper, Stefano, Robert, Alessandra for their unfailing support and assistance. A very special gratitude goes out to Barry for his optimistic personality which really brings happiness to all the people around him, and also thanks for translating my summary and propositions in this thesis from grungy English to Dutch.

I would also like to express my appreciations to my fellow PhD colleagues. Jing(褚晶), thank you for your detailed introduction about how to start PhD, if it's not for you, I would waste time for following the wrong path. Minghe(明贺), thank you for your company on playing games, basketball, and getting drunk with me... Johan, Marsil, Dadui, Fiona, Dennis, Linyu, Mario, Victor, Lorenzo, Stefano, have all extended their support in a very special way, and I gained a lot from them, through their personal and scholarly interactions, their suggestions at various points of my research. Of course, the time we spent together for bowling, karting, walking and bunch of drinking would always be my most precious memory. The last four years were the best years of my life. I am sad to see them come to an end and incredibly excited for the future. A future in which I hope to cross paths with all of you again.

Furthermore, there are so many thanks go to my dear friends here in the Netherlands and in China. Lu(吕洋), Chuang(余总), Tian(天哥), Fan(冯大爷), Ding(丁儿),

Tian No.2(高天), Qiang(强哥), Ran Yi(毅然), Rui Yan(咸阳邓肯) and Jiawei Lai(广州球王), thank you all for playing basketball with me and letting me shoot 3-pointers and get rebounds so easily 😉. Yue(越越), Xiang(院士), Jian(方总), Yu(桌游玩儿哭的奇女子), Tiantian(甜甜), Zhijie(萌师兄), Jintao(老实人), the great moments have you all together, board games, dinners, trips, and of course, drinks. You guys have prevented me from sinking into a boring life. I would like also show my gratitude to my dear friends back in my hometown Xi'an, Hao(浩子一炮走), Xun(过马德里而不见的逊), Kai(靳大叔), Xianghao(小/老候), Zonglin(歌王钟霖), Jun Wan(军儿红脸), Guoqiang(国强吉拉个), Defu and Zhibang(小皮和小沙哥哥). In the past four years, every time I came back, you guys have welcomed me with open arms, which makes me very touched. However, it would be much better if these welcomes would not end with hangovers. For my friends in other cities, Zongxun(老二), Dan(嫂子), Yulu(骚玉禄), Ting(婷婷), Le(内蒙分酒器乐儿), Hao Dong, Jianhui Zhang, Zhong Zheng and Mingjiang Li, I would like to say thank you for hosting me whenever I visited your city.

Next, I would like to acknowledge Prof. Jianping Yuan(袁建平), who supervised my master degree. His valuable guidance, scholarly inputs and consistent encouragement all helped me recognize the path of my future, which affected my choice to continue my academic life (pursuing a doctoral degree). I also want to show my respect to people from National Key Laboratory of Aerospace Flight Dynamics in NWPU, who had supported me a lot in both my research and my life: Prof. Jianjun Luo, Prof. Qun Fang, Prof. Zhanxia Zhu, Prof. Xiaokui Yue, Prof. Xing Ning, Weihua Ma, Honghua Dai, Mingming Zhang, Chong Sun.

I owe a lot to my parents Caixia Jing(敬彩霞) and Yongan Zheng(郑永安), who encouraged and helped me at every stage of my personal and academic life, and longed to see this achievement come true. I consider myself nothing without them. They gave me enough moral support, encouragement and motivation to accomplish my personal goals. They have always supported me financially so that I only pay attention to the studies and achieving my objective without any obstacle on the way.

Last but not least, I want to show my deepest gratitude to my beloved wife Lingling(玲玲), I am very much indebted to you. You have been supported me every possible way even though you are also pursuing a doctoral degree. If it is not for you, I would probably die for starvation and loneliness 😇. It is you who makes my life colorful, joyful and meaningful. Just like a lyric in a song of U2: *You are the best thing about me, the best thing that ever happened a boy.* Although I am not a boy anymore (maybe I still am 😂), you still and will always be the best thing that ever happened to me.

Zixuan
April, 2019
Delft, the Netherlands

CURRICULUM VITÆ

Zixuan ZHENG

07-01-1990 Born in Xi'an, Shaanxi Province, China.

EDUCATION

2008–2012 Undergraduate in Detection Guidance and Control Technology
Harbin Institute of Technology
Harbin, China

2012–2014 Graduate in Spacecraft Design and Control
Northwestern Polytechnical University
Xi'an, China

2014–2018 PhD. Aerospace Engineering
Delft University of Technology

Thesis:	Onboard Autonomy for Multiple Satellite Systems
Promotor:	Prof. Dr. E. K. A. Gill
Copromotor:	Dr. J. Guo

LIST OF PUBLICATIONS

JOURNAL PAPERS

1. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Swarm Satellite Mission Scheduling & Planning using Hybrid Dynamic Muation Genetic Algorithm*, Acta Astronautica **137**, 243-253 (2017).

2. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Onboard Autonomous Mission Re-planning for Multi-Satellite System*, Acta Astronautica **145**, 28-43 (2018).

3. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Onboard Mission Allocation for Multi-Satellite System in Limited Communication Environment*, Aerospace Science and Technology **79**, 174-186 (2018).

4. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Distributed Onboard Mission Planning for Multi-Satellite Systems*, Aerospace Science and Technology (2019) (In Press).

5. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *A Survey on Artificial Intelligence Technologies Applied for Space Missions*, Progress in Aerospace Science (Under Preparation).

CONFERENCE PAPERS

1. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Multi-Satellite Onboard Behaviour Planning Using Adaptive Genetic Algorithm*, the 67th International Astronautical Congress, Guadalajara, Mexico. IAC-16-B4.3.9, (2016).

2. **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Distributed Agent-based Approaches for Multi-Satellite System Limited Coordination Problem*, 2018 SpaceOps Conference, Marseilles, France. 2391, (2018).

3. Salvatore Sarno, **Zixuan Zheng**, Jian Guo, Eberhard Gill. *Autonomous Reconfiguration of A Distributed SAR Driven By Remote Sensing*, 4S Symposium, Sorrento, Italy (2018).