A string-based representation and crossover operator for evolutionary design of dynamical mechanisms

Kuppens, P. Reinier; Wolfslag, Wouter J.

DOI
10.1109/LRA.2018.2800102

Publication date
2018

Document Version
Final published version

Published in
IEEE Robotics and Automation Letters

A String-Based Representation and Crossover Operator for Evolutionary Design of Dynamical Mechanisms

P. Reinier Kuppens and Wouter J. Wolfslag

***Abstract*—Robots would perform better when their mechanical structure is specifically designed for their designated task, for instance by adding spring mechanisms. However, designing such mechanisms, which match the dynamics of the robot with the task, is hard and time consuming. To assist designers, a platform that automatically designs dynamical mechanisms is needed. This letter introduces a novel string-based representation for mechanisms, including evolutionary operators, that allows an evolutionary algorithm to automatically design dynamical mechanisms for a designated task. The mechanism representation allows simultaneous optimization of topology and parameters. Simulation experiments investigate various algorithms to obtain best optimization performance. We show the efficacy of the representation, operators, and evolutionary algorithm by designing mechanisms that track straight lines and ellipses by virtue of both their kinematic and dynamic properties.**

***Index Terms*—Mechanism design, dynamics, optimization, optimal control.**

I. Introduction

ENGINEERS would benefit from a tool that automatically generates concepts for mechanisms in robotics. Evolutionary algorithms (EAs) could be the basis of such a tool, as they can search for optimal points in the space of part assemblies.

In particular we consider spring mechanisms used for their dynamical properties. By tuning the stiffness of a mechanism, the natural dynamics can be adapted to the desired operating frequency, thereby minimizing energy consumption [1]. Multiple uses for springs exist in legged robots, especially running robots [2] and robot arms [1], [3]. Springs can also be used to create energy free systems. These systems have constant potential energy over the complete work-envelope and can thus move quasi-statically without operating energy [4]. Examples are statically balanced parallel robots [5], gravity compensation [6] and vibration isolation systems [7]. Finally, rigid body spring mechanisms are important because of their use as a first approximation for compliant mechanisms [8], such as a fully compliant force balanced oscillator [9]. Note that most of these references cite papers discussing the design of a task specific spring mechanism. Such mechanisms would be more viable if concepts could be quickly and automatically found.

Searching for spring mechanisms is a mixed integer programming problem, because the topology of a mechanism is described by integers (components are either connected or they are not) while its behaviour also relies on real valued parameters, such as link length and spring stiffness. Furthermore, it is a non-standard problem , because the number of components is generally unequal for different mechanisms. In addition, there exists no explicit function to describe the solution space, since the value of a design can only be quantified via simulation.

The complexity and nonlinearity of this problem requires a method that is both flexible and robust. For this type of problem, literature shows that EAs are capable of finding good (not necessarily optimal) solutions. For example, the design of analog electronics [10], antennas [11], finding natural laws [12], compliant mechanisms [13], kinematics [14], [15] and robotics [16], [17]. More recently, first steps towards dynamics have been made in [18].

The exploratory qualities of an EA could be extended with more problem specific heuristics to get a hybrid EA [19]. For example, a gradient based optimization to fine-tune the real valued parameters of each mechanism.

The main challenge in using EAs is the genome (the representation of the mechanism) and the variational operators (crossover and mutation) that act upon it. Many approaches to representing mechanisms have been developed [20]–[24], in order to make conceptual design more systematic. Early attempts such as Reuleaux's symbolic notation [20] and Franke's condensed notation [22] require visual inspection. Other methods such as the Denavit-Hartenberg parameters [21] only describe a single spatial kinematic chain and are therefore not suited for generating general spring mechanisms with EAs.

In the 1960s graph theory was first used to investigate the kinematic structure of mechanisms [23], [25], [26]. Graph theory enables type synthesis and analysis of complex kinematic chains, because properties such as planarity, isomorphism and connectivity can be tested on the incidence matrix of a graph. Furthermore, concepts such as graph duality and graph contraction aid mechanism analysis, see [27]. This matrix representation enables effective computer implementation, as first demonstrated in the 1980s [28], [29]. However, it is not directly suited for use in an EA.

In this letter, we present a representation based on [30] and [18]. Specifically, we use a *simple* graph to describe the

Manuscript received August 25, 2017; accepted January 14, 2018. Date of publication January 31, 2018; date of current version February 27, 2018. This letter was recommended for publication by Associate Editor C. Gosselin and Editor P. Rocco upon evaluation of the reviewers' comments. This work was supported by the research program STW, which is (partly) funded by the Netherlands Organization for Scientific Research (NWO). (*Corresponding author: P. Reinier Kuppens.)*

The authors are with the Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, Delft 2628 CD, The Netherlands (e-mail: p.r.kuppens@tudelft.nl; w.j.wolfslag@tudelft.nl).

Digital Object Identifier 10.1109/LRA.2018.2800102

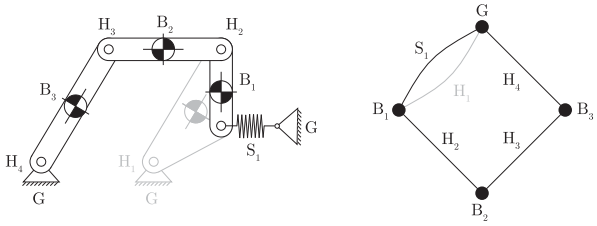

Fig. 1. A mechanism with its graph along with a specific schema. The grayed out components indicate a wildcard.

TABLE I
THE INCIDENCE MATRIX OF THE MECHANISM FROM FIG. 1 AND ITS SCHEMA

	S_1	H_1	H_2	H_3	H_4
G	1	1/□	0	0	1
B_1	1	1/□	1	0	0
B_2	0	0/□	1	1	0
B_3	0	0/□	0	1	1

kinematics and then arbitrarily assign directions to find the dynamics of the mechanism. The main contribution is to extend previous work by ensuring compatibility with variational operators. The focus is on single degree of freedom (DOF), 2-D rigid body spring mechanisms that are not actuated. Motion is caused by gravity and springs, such that a mechanisms inherent dynamics make it functional.

In Section II we encode dynamical mechanisms for efficient computer implementation. In Section III we explain how variational operators can manipulate these encodings to explore the space of part assemblies. Section IV gives details on the evolutionary algorithms. Section V explains our experiments, with their results in Section VI. Finally Sections VII and VIII provide discussion and conclusions.

II. REPRESENTING MECHANISMS

A simple graph $\mathcal{G}$ (from now on graph) is an ordered pair $(\mathcal{V}(\mathcal{G}), \mathcal{E}(\mathcal{G}))$ consisting of a set $\mathcal{V}(\mathcal{G})$ of vertices, a set $\mathcal{E}(\mathcal{G})$ of edges and an incidence function $\phi_{\mathcal{G}} : \mathcal{E} \rightarrow \mathcal{V} \times \mathcal{V}$ that associates with each edge of $\mathcal{G}$ an unordered pair of vertices of $\mathcal{G}$ and contains no loops or multiple edges [31].

Vertices are associated with bodies and edges with forces. Two types of bodies are included: rigid bodies (B) with a finite mass and a single ground (G) with infinite mass. Two types of connections between masses are included: spring forces (S) and hinges (H). An example is given in Fig. 1.

The incidence function of a graph $\phi_{\mathcal{G}}$ is defined by an $N \times M$ incidence matrix $I_{\mathcal{G}}$ with N being the number of vertices and M being the number of edges. The (i, j)-th element of $I_{\mathcal{G}}$ is equal to 1 if the i-th vertex is an end of the j-th edge and zero otherwise [31], e.g., see Table I.

Each edge carries associated parameters that describe properties such as hinge or spring locations. By carrying these properties within the edges, the ground can be represented by a single vertex such that a fixed vertex set $\mathcal{V}(\mathcal{G}) = \{G, B_1, B_2, \ldots, B_{N-1}\}$ can be used. Therefore, vertex labels can be omitted. In [18] grounds were labeled individually, making a fixed vertex-set impossible.

We indicate unlabeled topology data with an I, edgelabels with an E, and mass (B), hinge (H) and spring (S) data are of the form:

$$B_a = [m, x_c, ry_c],$$
$$S_b = [x_O, y_O, x_I, y_I, L_0, k], \quad H_c = [x_h, y_h] \qquad (1)$$

where a is the number of bodies, b the number of springs and c the number of hinges. All such data is given for the mechanism from Fig 6(c) in (7). The shape of each body is determined by the connections that are made to it: it is the convex hull of the location-data contained in S_b and H_c.

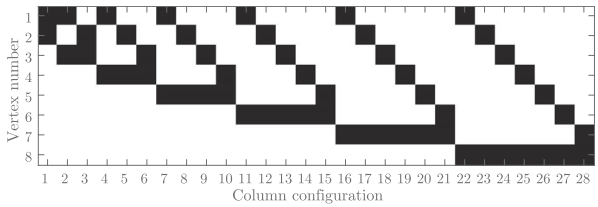

Fig. 2. The pattern generated by (4) up to $v = 8$. Each square is a 1 in a column.

A. Schemata as Building Blocks

The suggested encoding can be analysed by means of the schema theorem developed by Holland *et al.* [32]. It provides insight into how EAs manipulate individuals to accumulate wisdom within a population [33].

A schema H is a subspace of the original search space. For instance, consider minimizing the objective function f of a vector of decision variables $\mathbf{x}$. Suppose the optimal variables are $\mathbf{x}^* = [6, 3, 4]$ and that the □-symbol indicates a wildcard, i.e., it can be any value. Then examples of schemata that contain the solution are $H_1 = [\square, \square, 4]$, $H_2 = [\square, 3, \square]$, $H_3 = [6, \square, \square]$ and $H_4 = [\square, 3, 4]$, even the whole space $H_5 = [\square\square\square]$ and the optimum itself are schemata. A schema has an order, which equals the number of non-wildcard positions, and a defining length, which is the distance between the outermost non-wildcard positions. The optimal solution $\mathbf{x}^* = H = [6, 3, 4]$ can be constructed by intersection of the lower order schemata H_1, H_2 and H_3.

The schema theorem shows that short low-order schemata have a higher probability to propagate unharmed into subsequent generations than long high-order schemata, given the same chance on selection. This observation has led to the *building block hypothesis* [34]. This hypothesis states that EAs initially select short low-order schemata and progressively combine them into longer higher-order schemata. Based on the concept of building blocks the following six conditions for EA success were established [35]: 1) Ensure the EA is processing building blocks. 2) Ensure the mixing of good building blocks. 3) Ensure diversity of building blocks. 4) Ensure growth of good building blocks. 5) Decide well among competing building blocks. 6) Solve problems that can be build form building blocks, or encode them so they can. They will be used as a practical guide for the EA design.

A lower order schema can be found in a mechanism by replacing a column of its incidence matrix with wildcards. An example is shown in Fig. 1 and Table I. In this case the connection between B_1 and G is removed. At the level of the mechanism the shape of B_1 changes and the mechanism is reduced from a four-link to a three-link mechanism.

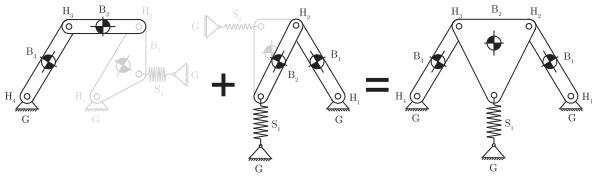

Fig. 3. Crossover on the level of mechanisms. The black parts indicate the selected tails from the genome.

This method can be used to create schemata for all submechanisms of a given mechanism or, vice versa, to combine schemata to form the mechanism. This means that our EA is processing building blocks.

B. Connectivity and Equivalent Topologies

The size of the solution space is reduced by using the properties of connectivity and isomorphism. Connectivity means all parts of a mechanism are connected and is directly tested on its graph. Isomorphism tests whether it is possible to relabel the parts of one mechanism (while also reordering the connections in the same way) such that the result is a second mechanism, which is therefore equivalent to the first.

Taking into account connectivity and isomorphism reduces the search space for our algorithm by orders of magnitude. For instance, there are $268 \cdot 10^6$ labeled simple graphs and $11 \cdot 10^3$ connected non-isomorphic graphs with 8 vertices [36].

Due to the different types of edges we cannot establish isomorphism for the whole graph at once. Four subgraphs of a different type can be identified, i.e., $\phi_{\mathcal{G}}(H) = (G, B)$, $\phi_{\mathcal{G}}(H) = (B, B)$, $\phi_{\mathcal{G}}(S) = (G, B)$, $\phi_{\mathcal{G}}(S) = (B, B)$. Only subgraphs of the same type will be compared. If all subgraphs are isomorphic to all subgraphs of another mechanism we consider the topology to be identical. This approach is however slightly conservative, because isomorphism between all subgraphs of two graphs does not necessarily mean that their represented mechanism topology is identical. We analyze isomorphism with the software package *nauty* [37].

C. Dynamical Model

To uncover the behavior of a mechanism we simulate its unactuated dynamics. Conservative forces (gravity G_i and linear spring forces S_i) cause any motion that occurs. Therefore, tuning the positions and stiffnesses of the springs is important for tracking a desired trajectory. This contrasts with a kinematic solution where a predefined input motion is specified. For example, the parent mechanisms in Fig. 3 move due to gravity and the springs. However, the child mechanism is in an equilibrium and would not move.

The equations for dynamic simulation of the mechanism are derived by way of virtual power, using Lagrange multipliers to add the constraints to implement the hinges [38]. This results in the following set of differential-algebraic equations:

$$\begin{bmatrix} M_{ij} & D_{k,i} \\ D_{k,j} & 0_{kk} \end{bmatrix} \begin{bmatrix} \ddot{x}_j \\ \lambda_k \end{bmatrix} = \begin{bmatrix} f_i \\ -D_{k,pq}\dot{x}_p\dot{x}_q \end{bmatrix} \tag{2}$$

where M_{ij} is a diagonal mass matrix, $D_{k,j}$ the Jacobian of the constraint vector D_k, $x_j = [x_1, y_1, \theta_1, \cdots, x_n, y_n, \theta_n]^T$ the global coordinates, λ_k the Lagrange multipliers, f_i the force vector and $D_{k,pq}\dot{x}_p\dot{x}_q$ the convective acceleration terms. This method does not require generalized coordinates, which can be hard to find automatically.

The scleronomic hinge constraints and the spring forces require the distance between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ located on two bodies:

$$\begin{aligned} \Delta P(\mathbf{x}_1, \mathbf{x}_2, \theta_1, \theta_2) = R(\theta_1)\left(\mathbf{p}_1 - \mathbf{x}_{c1}\right) + \mathbf{x}_1 \\ - R(\theta_2)\left(\mathbf{p}_2 - \mathbf{x}_{c2}\right) - \mathbf{x}_2 \end{aligned} \tag{3}$$

where $\mathbf{x}_1 = [x_1, y_1]^T$, $\mathbf{x}_{c1} = [x_{c_1}, y_{c_1}]^T$ and θ_1 are the current and initial center of mass position of the first body and its current angular position. $\mathbf{x}_2$; $\mathbf{x}_{c2}$ and θ_2 are defined analogously; and $R(\theta)$ is the 2D rotation matrix.

For each hinge two constraints of the form $\Delta P = 0$ are added to D_k with $\mathbf{p}_1 = \mathbf{p}_2 = \mathbf{x}_h$. If a body is connected to a ground $\mathbf{x}_2 = \mathbf{x}_h$ and $\mathbf{x}_{c2} = \mathbf{x}_h$ in addition.

The forces are given by $f_i = G_i + S_i$, where G_i is gravity and S_i are spring forces. Friction is neglected. Spring forces are computed as minus the partial derivative of the spring energy E_s with respect to the coordinates. The energy of a spring equals $\Delta L = |\Delta P - L0|$ with $\mathbf{p}_1 = [x_O, y_O]^T$ and $\mathbf{p}_2 = [x_I, y_I]^T$. When a spring is connected to a ground $\mathbf{x}_{c2} = [x_I, y_I]^T$ and $\mathbf{x}_2 = [x_I, y_I]^T$ hold as well. All constants are stored in M, H and S from (2).

Equation (2) (solved for $\ddot{x}_j$) is numerically integrated using the Runge-Kutta 4th order method with a fixed stepsize of $h = 0.05$. All simulations start at initial conditions $x_j(0) = [x_{c1}, y_{c1}, 0, \ldots, x_{ca}, y_{ca}, 0]$ and $\dot{x}_j(0) = \mathbf{0}$ and last for $T = 5$ seconds. To satisfy the constraints (i.e., $D_k = \mathbf{0}$) the Newton-Raphson method is used after each time step with a tolerance of 10^{-12} m. Numerical integration is stopped when singularities or extreme displacements (10 m per timestep) are detected. Two solutions along the optimized trajectory are given in Fig. 7.

D. Degrees of Freedom

One of the most fundamental properties of a mechanism is its number of DOFs. This describes the number of independent coordinates needed to fully describe motion, which plays a large role in a mechanisms functionality. The number of DOFs of a mechanism is found as the dimension of the nullspace of the Jacobian of its constraint vector $D_{k,j}$ at the initial configuration. This means singularities (causing changes in the number of DOFs) are not accounted for.

III. Variational Operators

Variational operators are responsible for creating new mechanisms in the population. Typically binary and unary operators are used. The binary operator (crossover) combines two genomes into one new mechanism. The unary operator (mutation) makes small random changes to one genome.

A. Crossover

A crossover operator, that combines two graphs, is non-trivial since incidence matrices are generally of a different size due to a different graph order. Also, this operator should ensure simple graphs and thus prevent loops and multiple edges. Furthermore, it should deal with topology and real valued parameters simultaneously. Many possible solutions arise from the different ways of storing the incidence matrix.

Incidence matrices contain many zeros, suggesting the use of a sparse matrix notation[39], [40]. However, the list describing a sparse matrix has length $2|\mathcal{E}|$, one entry for each non-zero element. Consequently, invalid edges can be created by splitting a sparse matrix in an inappropriate location. Another issue is how to unite the $|\mathcal{E}|$ data vectors belonging to each edge with $2|\mathcal{E}|$ list elements.

An edge-list may solve that, because it groups vertex pairs that form edges [31]. For example, the edge-list of the graph from Fig. 1 is given by $\{\{v_1, v_2\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_1, v_4\}\}$, which is written as a matrix of dimension $2 \times |\mathcal{E}|$. Even though an edge-list is easily united with $|\mathcal{E}|$ data vectors, it is not perfect for detecting loops and multiple edges. For example, when generating random graphs one must take good care not to generate loops of the form $\{v_1, v_1\}$ and multiple edges of the form $\{\{v_1, v_2\}, \{v_2, v_1\}\}$.

To explicitly exclude loops and multiple edges, we write the topology as a string of length $|\mathcal{E}|$. By properly defining a mapping $\mathcal{I} : \mathcal{E}(\mathcal{G}) \rightarrow \mathbb{N}$, we make sure that loops and multiple edges of the form $\{\{v_1, v_2\}, \{v_2, v_1\}\}$ cannot occur. Duplicate edges, such as $\{\{v_1, v_2\}, \{v_1, v_2\}\}$ are prevented by sampling without replacement. The mapping is given by

$$\mathcal{I}(\{v_1, v_2\}) = \frac{1}{2}(v_2 - 2)(v_2 - 1) + v_1 \quad \text{s.t.} \quad v_1 > v_2 \quad (4)$$

where $\{v_1, v_2\}$ is a pair of vertices that form an edge. The pattern this mapping creates is given in Fig. 2. It is created by noting that in a simple graph any column in $I_{\mathcal{G}}$ can have a finite number of configurations. This number is given by the binomial coefficient, e.g., six configuration for each column in Table I. Replacing each unique column in Table I with a number from the discrete set $\mathcal{I}_6 = \{1, 2, 3, 4, 5, 6\}$ given by (4) gives us the string. The pattern has the property that all possible configurations for a certain number of vertices are assigned before moving on to a higher number of vertices. Therefore, by using this pattern for translating the incidence matrix to a string of numbers, it is possible to combine mechanisms of different vertex-order.

Crossover is done with a slightly modified one point crossover [32], because it is simple and no obvious advantages exist over more complicated methods like n-point or uniform crossover [19]. The modification addresses that the length of both parents is in general not equal $L_1 \neq L_2$. We pick two random integers $r_1 \in [1, L_1 - 1]$ and $r_2 \in [1, L_2 - 1]$. Next, we split parent 1 at point r_1 and parent 2 at point r_2 and merge the tails to create a child. For example, the topology of the parents from Fig. 3 is given by $p_1 = [1\ 1\ 3\ 6\ 4]$ and $p_2 = [2\ 1\ 3\ 2]$ respectively. For $r_1 = 3$ and $r_2 = 1$ we get:

$$\left[\begin{array}{ccc|cc} 1 & 1 & 3 & 6 & 4 \\ \mathbf{d}_s & \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_h \end{array}\right] + \left[\begin{array}{c|ccc} 2 & 1 & 3 & 2 \\ \mathbf{d}_s & \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_s \end{array}\right] = \left[\begin{array}{cc|ccc} 6 & 4 & 1 & 3 & 2 \\ \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_h & \mathbf{d}_s \end{array}\right]$$

where $\mathbf{d}_h = [E_h, B_{a_1}, B_{a_2}, H_c]^T$ for hinges and $\mathbf{d}_s = [E_s, B_{a_1}, B_{a_2}, S_b]^T$ for springs. Data contained in d_h and d_s is associated with its respective edge and simply transported to the child mechanism. (7) gives an example of all data to build a genome for the mechanism from Fig. 6(c). To prevent multiple edges, duplicate incidence numbers with the same label are removed after crossover, along with their respective data vectors d_h or d_s. In our implementation, the mass data of each body is stored in the hinges and springs that are connected to them. This means that the mass data can be stored in multiple columns, which potentially causes conflicts after crossover. To resolve such conflicts, we take the average of all these instances for each mass after crossover.

B. Mutation

Mutation is done in two ways. The first is to randomly change a value in the representation. In this case we have to account for the meaning of each value, which can represent:

1. an incidence number, in which case we change the incidence number by one, in random direction.
2. an edge label, in which case we change it randomly.
3. a real valued datum about the hinges, springs or masses, in which case we change it by adding a value drawn from a normal distribution.

The second type of mutation causes changes on a more macroscopic scale: it acts upon complete parts, instead of single values. To emphasize mechanisms of a low complexity, we only *delete* parts. Again there are three options: to delete a mass, a hinge or a spring.

IV. Evolutionary Algorithm

This section discusses the details of the implemented EA. The goal of the EA is to provide an environment in which to test the proposed genome and operators.

A. Diversity

The power of EAs comes from two sources: exploration and exploitation [41]. They need to be well balanced, which can be done by maintaining diversity. Diversity measures the variety within a population and can be directly used in preventing premature convergence in local optima [42], [43].

Diversity D will be measured by the anti-log of the Shannon entropy, because it includes both the richness and evenness of species in a population. It is given by $D = e^H$ where $H = -\sum_{i=1}^{S} p_i \ln p_i$, with p_i the proportion of the population that belongs to species i and where S is the total number of species [44], [45]. A species is considered to be a unique topology as described in Section II-B, similar to genetic programming where diversity typically refers to structural differences [43].

Quantifying diversity allows us to monitor it. Maintaining diversity, without compromising convergence speed (i.e., lowering selection pressure), requires spatially distributing the population, specialized selection operators which need a similarity metric between individuals, or re-injecting the population with new genetic material [42]. Since no readily available similarity metric for mechanisms exists, we will maintain diversity by spatially distributing the population and by re-injecting genetic material.

The population will be spatially distributed with the island and the diffusion model. The island model splits the population into subgroups called islands. Gene flow between islands only takes place after an epoch, i.e., when the local populations stop improving for 10 generations. A circular topology of N islands is used, in line with results from [46].

The diffusion model lowers gene flow trough the population on each island by placing all individuals in its own cell on a toroidal grid. Crossover only occurs between neighboring individuals. This local flow has the same evolutionary power as global operators [47]. In addition, they also enable parallel computation [48], [49].

B. Selection

Parents, survivors, migrants and neighourhoods are selected with the linear ranking scheme with maximum selection pressure ($s = 2$) from [19]. The expected value is converted into sampling from the populations by the roulette wheel algorithm.

During neighbourhood selection an individual is selected by ranking the population in order of ascending fitness. A neighbourhood is then the locality of the selected individual. Next two parents are locally selected from the neighbourhood the same way. During survivor selection, one individual that will be deleted is locally selected by ranking the neighbourhood in descending order of fitness. During migrant selection one individual is selected on each island in the same way as neighbourhoods are selected. An individual μ_1 on island 1 will take the place of μ_2 on island 2, μ_2 will take the place of μ_3 on island 3 and so on.

C. Initialization

To ensure a maximally diverse initial population all members are non-isomorphic as defined in Section II-B. Members will have different graph orders, which raises the question how many samples of each graph order should be taken. According to the true distribution of non-isomorphic connected simple graphs, mostly samples of high order graphs [36]. However, low order graphs can be useful building blocks by reason of the building block hypothesis.

To compromise, we sample graph order v according an exponential distribution with scaling parameter $\lambda = 0.5$. By tuning λ more emphasis can be given to low or high order graphs. Next the number of edges e is uniformly chosen such that the mechanism can be connected and has at least one DOF. The graph is build by generating e uniformly random incidence numbers between 1 and $\mathcal{I}(v, v-1)$.

D. Fitness Function

Two design cases will be presented: 1) a one DOF mechanism that draws a straight line and 2) a one DOF mechanism that draws an ellipse. In both cases a feature based fitness function is used. The fitness function F is a bilinear form of the differences in desired (f_d) and measured (f_m) features, similar to [50]:

$$F = ||f_d - f_m||_A = \sqrt{(f_d - f_m)^T A (f_d - f_m)} \qquad (5)$$

In each case fitness is calculated for each center of mass and all hinges and spring-ends that do not connect to a ground (e.g., six trajectories in Fig. 1). The fitness of the mechanism is the minimum of these fitness values.

For the straight line mechanism four features are included in the feature vector. The first feature, $f_{m,1}$, measures the absolute total curvature of a trajectory. The local curvature κ_i is computed for each point i on the polygonal curve with the algorithm from [51]. The second feature $f_{m,2}$ is the trajectory length to prevent very small straight lines. (6) shows the first two features with $|p|$ is the number of points, x_i' and y_i' are the differences in the x and y directions between consecutive points. The errors are normalized by bounding and dividing the differences by 100.

$$f_{m,1} = \sum_{i=1}^{|p|} |\kappa_i|, \qquad f_{m,2} = \sum_{i=1}^{|p|-1} \sqrt{x_i'^2 + y_i'^2} \qquad (6)$$

TABLE II
PARAMETER SETTINGS FOR ALL ALGORITHMS

Parameter	Value	Parameter	Value
Max no. bodies	7	No. migrants	1
Neighbourhood size	3×3	Crossover rate	1
Population size	7×7	Parameter mutation rate	0.05
No. gen. per epoch	$\geq$10	Part mutation rate	0.03
No. children per gen.	10		

The third feature $f_{m,3}$ is the number of DOFs of the mechanism squared. For this feature we square the difference. Lastly, $f_{m,4}$, represents the complexity as the sum of all masses, hinges and springs of the mechanism. The differences for the DOFs and complexity features are normalized by the maximum value given the number of vertices in the largest allowed graph. The matrix A is chosen to be $A = diag(80, 70, 25, 25)$. The desired feature vector is $f_d = [0, 10, 1, 0]^T$.

For the ellipse mechanism, we use features describing the ellipticity, the length and the area of the closed polyon curve. Also, the L2 norm of the vertex-wise difference in position and curvature are computed to describe local variations between the desired and measured trajectories. These features are described in more detail in [50]. In addition we include the DOFs and complexity of the mechanisms as in the straight line case. A normalized ellipse with a major axis twice the minor axis is used as desired trajectory. We take matrix $A = diag(10, 10, 10, 20, 20, 5, 5)$.

V. EXPERIMENT DESCRIPTION

To investigate the performance of the EA, it is decomposed into five sub-algorithms. By comparing their output we are able to pick the best combination of methods.

Each algorithm is run with both 2 and 12 populations (islands) in parallel on an Intel i5-3320M and two Intel Xeon X5650 CPUs respectively with both 20 and 60 migrations. The results are the fitness values of the best mechanism. All algorithms are run 30 times to obtain a fair sample of the optimization performance. The samples are compared using three-way independent measures ANOVA.

The following algorithms are compared: a) Multistart algorithm; a simple EA with 2 and 12 non-interacting panmictic populations. b) Island model. c) Island with diffusion model. d) Island with diffusion model plus re-injection at a threshold of 80 effective species. e) Island with diffusion model plus re-injection at a threshold of 20 effective species.

Reasonable algorithm parameters have been determined by preliminary tests and values stated in literature, see Table II.

VI. RESULTS

The optimization results are summarized in Fig. 4. Nine out of 20 cases significantly deviated from normality, according to the Shapiro-Wilk test ($p < .05$). Furthermore, the assumption of homogeneity of variance for ANOVA was violated (Levene's test: $F(19, 580) = 25.493, p < .001$). Transforming the data did not rectify these problems, so the reported F-tests could be inaccurate.

The results show that the main effect of algorithm choice significantly affected the best fitness score ($F(4, 580) = 9.649$,

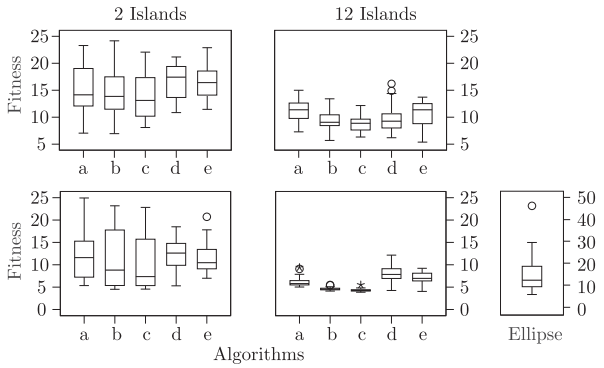

Fig. 4. Boxplots for all runs. The upper two plots show results for case I with 20 migrations for 2 and 12 islands. The bottom left two show results for case I with 60 migrations for 2 and 12 islands. The bottom right shows results for case II with 60 migrations and 12 islands. Small circles indicate moderate outliers and the stars indicate extreme outliers.

TABLE III
RUN TIME OVER 30 RUNS IN MINUTES

	20 Migrations		60 Migrations	
	2 Islands	12 Islands	2 Islands	12 Islands
Alg. a	9	16	21	40
Alg. b	16	14	20	25
Alg. c	7	13	18	25
Alg. d	17	29	48	75
Alg. e	13	24	39	72
Evals	≥4000	≥24000	≥12000	≥72000

$p < .001, \eta_p^2 = .062$). Post-hoc analysis using Tukey HSD revealed that algorithm c performed significantly better than the algorithms a, d and e (for all $p < .01$). Algorithm b performed significantly better than the algorithms d and e (both $p < .05$). No significant difference was found between the algorithms a, d and e (for all comparisons $p > 0.05$).

The main effect of the number of migrations significantly affects the best fitness ($F(1, 580) = 203.521, p < .001, \eta_p^2 = .26$). Furthermore, the main effect of the number of parallel populations affects the best fitness score significantly as well ($F(1, 580) = 379.509, p < .001, \eta_p^2 = .396$). The effect size ($\eta_p^2$) shows that the number of parallel populations and the number of migrations both have a large effect, whereas the effect of algorithm choice is small to medium.

No interaction effects among independent variables (algorithm choice, number of migrations and the number of populations) were found. Thus, it is likely that each variable can be adjusted without affecting the effect of another.

These findings suggest that algorithms b and c outperformed the other methods in terms of optimization performance. However, since only algorithm c was significantly better than algorithm a we pick algorithm c as the best.

Table III shows the run time for the experiments, which are considered reasonable. The run-time of algorithms d and e is significantly higher ($\approx$3 times), because complex mechanisms are repeatedly added to the population.

Two straight line solutions can be distinguished: single pendula and Roberts mechanisms. Pendula approach a straight line as they get longer and have very low complexity. Roberts

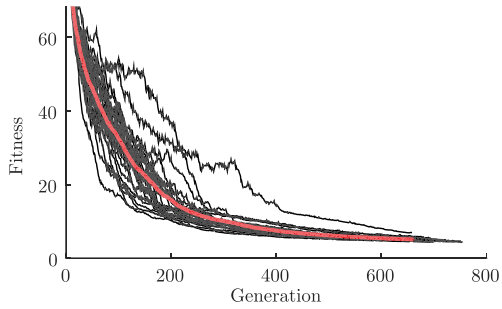

Fig. 5. Mean fitness of algorithm c. The black lines indicate individual runs and the red line indicates the mean.

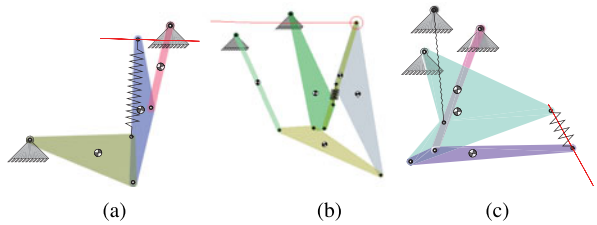

Fig. 6. Evolved straight line mechanisms.

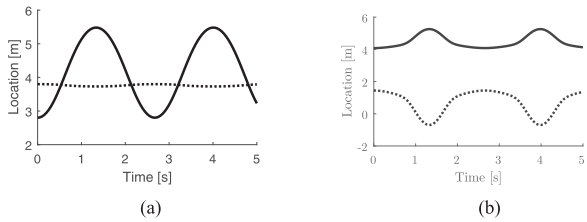

Fig. 7. The x- (solid line) and y- (dotted line) coordinates of resulting trajectory versus time.(a) Mechanism from Fig. 6(a). (b) Mechanism from Fig. 6(c).

mechanism is more accurate at smaller size, but has more parts. Three things are noted about the evolved Roberts mechanisms. First, they were never symmetric. Second, they often had redundant parts attached, meaning topology was different from case to case. Lastly, most Roberts mechanisms were produced by algorithms d and e, indicating those algorithms might be useful when the goal is to create a more diverse set of mechanisms. Examples are shown in Fig. 6

Fig. 5 shows the mean fitness of the population versus the time (in generations) of algorithm c. The black lines indicate individual runs and red the mean of all 30 runs. Diversity versus time looks very similar, albeit with a quicker convergence, suggesting parameters were further optimized after the topology was found.

Algorithm c with 12 islands and 60 migrations performs best on the straight line problem. That is, it produced the best fitness values with the least spread. The results from algorithm c suggest that the EA is able to successfully perform a search for mechanism topology and perform a parameter optimization at the same time. This is readily observed in the convergence in Fig. 5.

The best algorithm found for the straight line mechanism is also used for tracking an elliptic trajectory. The algorithm was successful in 29 out of 30 runs. The fitness values are summarized in Fig. 4. Two of the 29 mechanisms are shown in

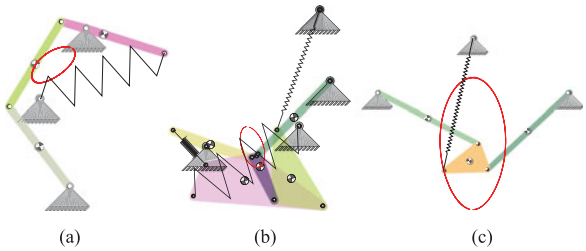

Fig. 8. (a) and (b) Evolved and (c) redesigned ellipse mechanisms.

Fig. 8(a) and 8(b). Both mechanism rely on the same topology, but due to their different parameters their operating principle is different. Finally, Fig 8(c) shows a human edited version of the 2nd mechanism, showing that the automatically evolved mechanisms can be used as an inspiration to design elegant and functional concepts.

$$
\begin{aligned}
I &= [1 \quad 5 \quad 2 \quad 6 \quad 6 \quad 2] \\
E &= [1 \quad 1 \quad 2 \quad 1 \quad 2 \quad 1] \\
B &= \begin{bmatrix} 0.29 & 0.49 & 0.71 \\ 0.70 & 0.31 & 0.43 \\ 0.46 & 0.40 & 0.61 \end{bmatrix} \\
S &= \begin{bmatrix} 1.04 & 3.94 & 1.05 & 0.92 & 1.05 & 2.27 \\ 3.88 & 0.83 & 4.08 & 1.44 & 1.13 & 4.51 \end{bmatrix} \\
H^T &= \begin{bmatrix} 2.26 & 0.58 & 0.03 & 0.80 \\ 3.44 & 0.47 & 0.02 & 2.84 \end{bmatrix}
\end{aligned} \tag{7}
$$

VII. Discussion

We have demonstrated evolutionary design of planar straight line mechanisms and mechanisms that draw an ellipse, an improvement over previous work [18]. To obtain these results, a novel representation for mechanisms was developed with emphasis on compatibility with EAs. In particular, we unified the representation of [18] with the schema theorem [32]. Compatibility with variational operators was achieved by writing topology as a string. This inherently avoids loops and multiple edges, preventing parts to be connected to themselves or multiple times to each other.

The underlying graph theory allows for extensive post processing. It allows for example analysis of kinematic chains by graph contraction, isomorphic topologies, planarity and connectivity. This means that after evolution the mechanisms can be further analyzed and developed by a design engineer. However, by representing the graph via an incidence string the description becomes more abstract. Therefore, analysis cannot directly be applied on the incidence string that we introduced. Our representation is not intended to be used directly by human engineers. Instead, it makes it easier for EAs to manipulate and accumulate data in a population enabling evolutionary design of mechanisms.

The evolved mechanisms move by virtue of their kinematic and dynamic properties. However, the fitness functions do not include any criteria involving speed or forces. Such criteria could be interesting for spring mechanisms for fast moving robots, or to optimize the natural frequency of oscillations. Implementation should be possible by using the computed Lagrange multipliers or displacement solutions such as shown in Fig. 7. However, dynamic criteria can be difficult to state or compute robustly. To that end, future research should study how to tightly couple more sophisticated simulation engines to our proposed algorithm.

No mechanisms that exactly track the desired trajectory (e.g., the linkage from [52]) were found. We believe that such mechanisms might be deceptive optima. That is, optima for which no continuous transformation along the gradient of fitness improvement towards a global optimum exist [53]. Deceptive problems are more likely to be solved by searching for novelty [54] rather than an explicit objective. To use novelty search, future research should aim to find an appropriate metric for the similarity between mechanisms.

In this research only springs, hinges, and rigid bodies were used as building blocks. The representation can be extended to include more parts, such as torsion springs, prismatic joints, linear and rotational motors and so on. For example, torsion springs can be added by using their potential energy $E_\theta = 1/2k\Delta\theta^2$, a prismatic joint can be added by constraining a rotational joint on a line instead of a plane, i.e., $D_p = [\sin(\alpha) \ -\cos(\alpha)]\Delta P = 0$, where α is the angle of the line, and motor functions can be added by including rheonomic constraints. As the underlying graph that encodes topology remains the same, this theoretically only requires more edge labels to denote different constraints and forces. The main challenge for future research is to find good solutions in the enlarged search space.

Similarly, we believe any system that can be encoded by a graph can potentially be designed by EA. For example, electric circuits that have been evolved in previous research [10] or neural networks that are parametrized graphs. Since the abstraction of their network structure is essentially the same, such systems could ultimately be unified into a single representation. Evolution of simple multi-physics systems has already been demonstrated [17]. An important, but challenging, next step is to evolve dynamic mechanisms with motors, in combination with evolving the motor controllers.

VIII. Conclusion

In this letter we presented a string based representation for dynamical mechanisms and variational operators that act upon it. The representation and operators were designed while carefully considering the schema theorem and the building block hypothesis such that both topology and parameters could be optimized simultaneously and efficiently. The performance of various algorithms was tested. For the test case of a straight line mechanism, a combination of the island and diffusion model worked best. Finally, we have demonstrated, for the first time, evolutionary design of simple single DOF mechanisms that track an approximate straight line and an approximate ellipse by virtue of their kinematic and dynamic properties.

References

[1] A. Jafari, N. G. Tsagarakis, and D. G. Caldwell, "Exploiting natural dynamics for energy minimization using an actuator with adjustable stiffness (awas)," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4632–4637.

[2] R. Alexander, "Three uses for springs in legged locomotion," *Int. J. Robot. Res.*, vol. 9, no. 2, pp. 53–61, 1990.

[3] M. Plooij and M. Wisse, "A novel spring mechanism to reduce energy consumption of robotic arms," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 2901–2908.

[4] J. L. Herder, *Energy-Free Systems. Theory, Conception and Design of Statically Balanced Spring Mechanisms*, Ph.D. dissertation, Delft Univ. Tech. The Netherlands, Nov. 2001.

[5] C. M. Gosselin and J. Wang, "Static balancing of spatial six-degree-of-freedom parallel mechanisms with revolute actuators," *J. Field Robot.*, vol. 17, no. 3, pp. 159–170, 2000.

[6] R. Barents, M. Schenk, W. D. van Dorsser, B. M. Wisse, and J. L. Herder, "Spring-to-spring balancing as energy-free adjustment method in gravity equilibrators," *J. Mech. Des.*, vol. 133, no. 6, 2011, Art. no. 061010.

[7] T. D. Le and K. K. Ahn, "A vibration isolation system in low frequency excitation region using negative stiffness structure for vehicle seat," *J. Sound Vib.*, vol. 330, no. 26, pp. 6311–6335, 2011.

[8] L. L. Howell, *Compliant Mechanisms*. Hoboken, NJ, USA: Wiley, 2001.

[9] S. L. Weeke, N. Tolou, G. Semon, and J. L. Herder, "A fully compliant force balanced oscillator," in *Proc. ASME 2016 Int. Des. Eng. Tech. Conf. Comput. Inf.*, 2016, Paper no. V05AT07A008.

[10] J. R. Koza, M. A. Keane, and M. J. Streeter, "Evolving inventions," *Sci. Amer.*, vol. 288, no. 2, pp. 52–59, 2003.

[11] G. Hornby, J. Lohn, and D. Linden, "Computer-automated evolution of an x-band antenna for NASA's space technology 5 mission," *Evol. Comput.*, vol. 19, no. 1, pp. 1–23, 2011.

[12] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Sci.*, vol. 324, no. 5923, pp. 81–85, 2009.

[13] R. Parsons and S. Canfield, "Developing genetic programming techniques for the design of compliant mechanisms," *Struct. Multidisciplinary Optim.*, vol. 24, no. 1, pp. 78–86, 2002.

[14] H. Lipson, "Evolutionary synthesis of kinematic mechanisms," *J. Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 22, no. 3, pp. 195–205, 2008.

[15] O. Chocron and P. Bidaud, "Evolutionary algorithms in kinematic design of robotic systems," in *Proc. 1997 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 1997, vol. 2, pp. 1111–1117.

[16] C. Leger, "Automated synthesis and optimization of robot configurations," Ph.D. dissertation, Pittsburgh, PA, USA: Carnegie Mellon University, Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 1999.

[17] J. B. Pollack and H. Lipson, "The golem project: Evolving hardware bodies and brains," in *Proc. 2nd NASA/DoD Workshop Evol. Hardware*. IEEE, Jul. 2000, pp. 37–42.

[18] I. C. Staal, "Evolutionary mechanisms," *TU Delft, Delft Univ. Tech., BioMech. Eng.*, 2014.

[19] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. New York, NY, USA: Springer, 2003.

[20] F. Reuleaux and E. S. Ferguson, *Kinematics of Machinery: Outlines of a Theory of Machines*. North Chelmsford, MA, USA: Courier Corporation, 2012.

[21] J. Denavit, "A kinematic notation for lower-pair mechanisms based on matrices." *J. Trans. Appl. Mech.*, vol. 22, pp. 215–221, 1955.

[22] R. Franke, *Vom Aufbau der Getriebe*. Berlin, Germany: VDI-Verlag, 1958.

[23] L. Dobrjanskyj and F. Freudenstein, "Some applications of graph theory to the structural analysis of mechanisms," *J. Manuf. Sci. Eng.*, vol. 89, no. 1, pp. 153–158, 1967.

[24] Y.-M. Moon and S. Kota, "Automated synthesis of mechanisms using dual-vector algebra," *Mech. Mach. Theory*, vol. 37, no. 2, pp. 143–166, 2002.

[25] F. Crossley, "The permutations of kinematic chains of eight members or less from the graph-theoretic viewpoint," *Develop. Theor. Appl. Mech.*, vol. 2, pp. 467–486, 1965.

[26] F. Freudenstein and L. Dobrjanskyj, "On a theory for the type synthesis of mechanisms," in *Proc. Appl. Mech*. Springer, 1966, pp. 420–428.

[27] H.-S. Yan and Y.-T. Chiu, "On the number synthesis of kinematic chains," *Mech. Mach. Theory*, vol. 89, pp. 128–144, 2015.

[28] T. Mruthyunjaya and M. Raghavan, "Computer-aided analysis of the structure of kinematic chains," *Mech. Mach. Theory*, vol. 19, no. 3, pp. 357–368, 1984.

[29] D. Olson, T. Thompson, D. Riley, and A. Erdman, "An algorithm for automatic sketching of planar kinematic chains," *ASME J. Mech. Transm. Autom. Des.*, vol. 107, no. 1, pp. 106–111, 1985.

[30] A. Van der Schaft and B. Maschke, "Port-hamiltonian systems on graphs," *SIAM J. Control Optim.*, vol. 51, no. 2, pp. 906–937, 2013.

[31] J. A. Bondy and U. Murty, "Graph theory, volume 244 of graduate texts in mathematics," London, U.K.: Springer, 2008.

[32] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, MI, USA: Michigan Press, 1975.

[33] D. E. Goldberg and K. Sastry, "A practical schema theorem for genetic algorithm design and tuning," in *Proc. GECCO*, 2001, pp. 328–335.

[34] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," Boston, MA, USA: Addion Wesley, 1989.

[35] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Syst.*, vol. 6, pp. 333–362, 1991.

[36] F. Harary and E. M. Palmer, *Graphical Enumeration*. Amsterdam, The Netherlands: Elsevier, 2014.

[37] B. D. McKay and A. Piperno, "Practical graph isomorphism, ii," *J. Symbolic Comput.*, vol. 60, pp. 94–112, 2014.

[38] R. Q. Van der Linde and A. L. Schwab, "Lecture notes multibody dynamics B, wb1413, course 1997/1998," Lab. for Eng. Mech., *Delft Univ. Technol.*, 2002.

[39] J. R. Gilbert, C. Moler, and R. Schreiber, "Sparse matrices in matlab: Design and implementation," *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 1, pp. 333–356, 1992.

[40] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, 2011, Art. no. 1.

[41] A. E. Eiben and C. Schippers, "On evolutionary exploration and exploitation," *Fundam. Inform.*, vol. 35, no. 1–4, pp. 35–50, 1998.

[42] R. K. Ursem, "Diversity-guided evolutionary algorithms," in *Proc. 7th Int. Conf. Parallel Probl. Solving Nature,* Springer, 2002, pp. 462–471.

[43] E. K. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: An analysis of measures and correlation with fitness," *Trans. Evol. Comput.*, vol. 8, no. 1, pp. 47–62, 2004.

[44] R. K. Peet, "The measurement of species diversity," *Annu. Rev. Ecol. Syst.*, vol. 5.1, pp. 285–307, 1974.

[45] J. P. Rosca, "Entropy-driven adaptive representation," in *Proc. Workshop Genetic Program.: From Theory to Real-World Appl.*, vol. 9. Citeseer, 1995, pp. 23–32.

[46] R. A. Lopes, R. C. Pedrosa Silva, A. R. Freitas, F. Campelo, and F. G. Guimarães, "A study on the configuration of migratory flows in island model differential evolution," in *Proc. 2014 Annu. Conf. Genetic Evol. Comput.*, ACM, 2014, pp. 1015–1022.

[47] C. C. Pettey, "Diffusion (cellular) models," in *The Handbook of Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, 1997, Chapter 6.4.

[48] D. Whitley, S. Rana, and R. B. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," *J. Comput. Inf. Technol.*, vol. 7, pp. 33–48, 1999.

[49] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, 2002.

[50] S. Coros *et al.*, "Computational design of mechanical characters," *J. ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 83.

[51] D.-J. Kroon, "2d line curvature and normals." [Online]. Available: https://nl.mathworks.com/matlabcentral/fileexchange/32696-2d-line-curvature-and-normals?s_tid=prof_contriblnk

[52] M. Gallet, C. Koutschan, Z. Li, G. Regensburger, J. Schicho, and N. Villamizar, "Planar linkages following a prescribed motion," *Math. Comput.*, vol. 86, no. 303, pp. 473–506, 2017.

[53] D. E. Goldberg, "Genetic algorithms and walsh functions-part ii: Deception and its analysis," *Complex Syst.*, vol. 3, pp. 153–171, 1989.

[54] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.