Joint Search of Optimal Topology and Trajectory for Planar Linkages

Zherong Pan[1, 4] and Min Liu[2] and Xifeng Gao[3, 4] and Dinesh Manocha[2]

Abstract

We present an algorithm to compute planar linkage topology and geometry, given a user-specified end-effector trajectory. Planar linkage structures convert rotational or prismatic motions of a single actuator into an arbitrarily complex periodic motion, which is an important component when building low-cost, modular robots, mechanical toys, and foldable structures in our daily lives (chairs, bikes, and shelves). The design of such structures require trial and error even for experienced engineers. Our research provides semi-automatic methods for exploring novel designs given high-level specifications and constraints. We formulate this problem as a non-smooth numerical optimization with quadratic objective functions and non-convex quadratic constraints involving mixed-integer decision variables (MIQCQP). We propose and compare three approximate algorithms to solve this problem: mixed-integer conic-programming (MICP), mixed-integer nonlinear programming (MINLP), and simulated annealing (SA). We evaluated these algorithms searching for planar linkages involving $10 - 14$ rigid links. Our results show that the best performance can be achieved by combining MICP and MINLP, leading to a hybrid algorithm capable of finding the planar linkages within a couple of hours on a desktop machine, which significantly outperforms the SA baseline in terms of optimality. We highlight the effectiveness of our optimized planar linkages by using them as legs of a walking robot.

Keywords
mixed integer, topology, geometry, optimization

Introduction

Over the past decades, robots have profoundly changed the industry, scaling up the productivity and simplifying the workflow of assembly lines. With their superior reliability and accuracy, however, comes a high fabrication cost and time-consuming maintenance. For robots to serve other aspects of our lives, they have to be versatile and adapt to rapidly changing tasks. To this end, we have witnessed an ongoing trend in both industry (HEBI 2019) and research (Murata et al. 2002; Hayakawa et al. 2020) communities towards low-cost, modular robot designs. This type of modular hardware opens the door to a huge space of infinitely many robot designs that can be exploited to accomplish a variety of tasks. A pivotal challenge faced by a robot designer is to determine the "optimal" design to accomplish a given task.

This paper addresses the problem of computational task-driven, design optimization for planar linkages. Planar linkages are mechanical structures built out of a set of rigid bodies connected by hinge joints. These structures are capable of converting simple rotational or linear motion into arbitrarily complex, curved motion. Being electronics-free and low-cost, planar linkages have been studied for centuries and used ubiquitously in mechanical tools, household accessories, and vehicles. As illustrated in Figure 1, they can also be integrated into low-cost robots to fulfill requirements of different types of locomotion for walking, swimming, and flying (Hernández et al. 2016; Thomaszewski et al. 2014). However, the design of planar structures requires significant trial and error, even for experienced engineers.
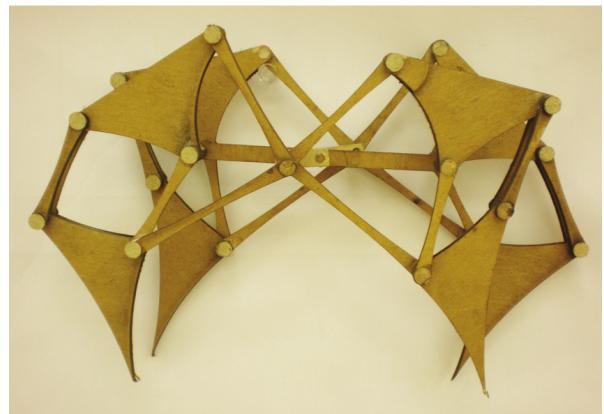

Figure 1. The Jansen's strandbeest can be manufactured by putting together 4 planar linkages to the left and right of the robot's body; see Nansai et al. (2013) for more details.

Computational robot design (Ha et al. 2018; Ha et al. 2018; Whitman et al. 2020) has recently drawn increasing attention, partly due to high-performance computers. The

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois IL 61801, USA. [2]Department of Computer Science and Electrical & Computer Engineering, University of Maryland at College Park, Maryland MD 20742, USA. [3]Department of Computer Science, Florida State University, Florida FL 32306, USA. [4]Lightspeed & Quantum Studio, Tencent America, USA.

Corresponding author:
Min Liu
Email: gfsliumin@gmail.com

decision space can grow exponentially with the complexity of the robot, so these methods are limited to designing sequential manipulators with no more than a few links. These links are standardized with fixed geometric shapes, so the design algorithm only needs to determine the connectivity (or topology) of those links, leaving only a discrete decision space of design variables to the designer. Prior works utilize algorithms such as A^* search (Ha et al. 2018) and Q-learning (Whitman et al. 2020) to solve the underlying optimization problem. Standardized links are sufficient for sequential manipulators because they can use multiple actuators to move the end-effector. With a single actuator, however, the end-effector of a planar linkage can only trace out a single curve, and a design algorithm must jointly search for both the geometry and topology of links to cover a large variety of curve shapes, which poses a much more challenging decision-making problem in a mixed continuous-discrete space.

Main Results: We propose a new algorithm to automatically search for topology and geometry for a large sub-class of planar linkages that are specified by prior works Kecskemethy et al. (1997); Thomaszewski et al. (2014); Bächer et al. (2015). Our main idea is to formulate the problem as a mixed-integer numerical optimization with quadratic objectives and non-convex, quadratic constraints, or MIQCQP. We then propose and compare the performance of three algorithms to approximate the optimal solutions: a mixed-integer conic programming (MICP) algorithm that uses a piecewise convex relaxation of non-convex constraints; a mixed-integer nonlinear programming (MINLP) algorithm that uses sequential quadratic programming (SQP) to find locally feasible solutions for non-convex constraints; a simulated annealing (SA) algorithm that randomizes both the geometry and topology, which is a variant of Zhu et al. (2012). We have evaluated our method in a row of optimization tasks with $5-7$ rigid bodies tracing out complex end-effector curves. These hybrid algorithms can find a solution within a couple of hours on a desktop machine, and the results exhibit an averaged $9.3\times$ higher optimality when compared with the SA-baseline.

This paper is an extended version of our prior work Pan et al. (2019), where we proposed the original MICP relaxation scheme to solve MIQCQP approximately. We extend over the prior work in three ways. First, we propose a new MIQCQP-approximation scheme based on MINLP. MINLP solver tries to satisfy non-convex constraints exactly and achieves a better balance between computational time and the optimality of the resulting solution.. Second, we initialize MINLP solver using a similar approach as MICP, which allows MINLP to exhaustively try more initial guesses and improves its success rate. We also introduce a local optimization move into the SA baseline algorithm to improve its efficacy. Finally, we conduct simulated experiments to illustrate the application of our optimized planar linkages on robot locomotion. Specifically, we use a linkage as legs of a walking robot and optimize its dynamics properties to maximize the walking performance via Bayesian exploration.

Related Work

In this section, we review related work in robot design optimization, mixed-integer programming, and planar-linkage design.

Robot Design Optimization is among the most challenging decision-making problems because the design algorithm must jointly reason about the robot design parameters and motion plans. This problem is a superset of conventional topology and truss optimization (Liu and Ma 2016), which does not involve movable components. Furthermore, the decision space of a robot design is oftentimes high-dimensional, and involves topology, geometry, and space-time variables. Existing approaches use one or more of those three variables to approximately search for optimal robot designs. In Umetani et al. (2014); Thomaszewski et al. (2014); Bächer et al. (2015), authors proposed human-in-the-loop design tools that either visualize the designed robot motion or locally optimize the robot's continuous geometric parameters. Our method is complementary to these works as it jointly optimizes the topology or geometry, although we can still find sub-optimal solutions or even fail to find a solution. In Zhu et al. (2012); Liao et al. (2019), the authors use stochastic optimization solvers such as SA and Bayesian optimization to search for robot topology. Our method provides an alternative, deterministic approach to solve the same problem. Most recent works (Ha et al. 2018; Spielberg et al. 2017; Saar et al. 2018; Hu et al. 2019) locally optimize robot's geometry given a user-provided topology and a geometric initial guess. However, figuring these topology and geometric initial guesses can still be labor-intensive.

Mixed-Integer Programming (MIP) is a standardized tool to formulate mathematical programming problems with non-convex constraints that can be expressed as a disjoint set. Although solving general MIP is NP-hard, practical branch-and-bound (BB) algorithms (Lawler and Wood 1966) can find global optima of MICP instances of small-to-medium sizes, where each member of the disjoint set is convex. BB algorithms rely on tight, convex relaxations to efficiently find lower bounds and cut off sub-optimal solutions at an early stage. BB serves as the computational engine of a large variety of problems, including inverse kinematics (Dai et al. 2017), network flows (Conforti et al. 2009), mesh generations (Bommes et al. 2009), motion planning with collision handling (Ding et al. 2011), and legged locomotion (Deits and Tedrake 2014). We adopt a similar technique as these methods to formulate our topology optimization problem, where constraints with integer variables ensure the correctness of link connectivity. However, if the members of the disjoint set are non-convex, as it is the case with our geometric optimization problem, finding the exact global optima is intractable and two approaches can be used to approximate them. First, big-M methods (Bertsimas and Tsitsiklis 1997), McCormick envelopes, and piecewise approximations (Liberti 2004) discretize a non-convex set as a union of convex sets, where the discretization error can be made arbitrarily small using higher resolutions of discretization and more integer decision variables. Second, MISQP algorithms (Exler and Schittkowski 2007) locally solve non-convex programs and use the solution in BB

algorithms, which is not guaranteed to be a lower bound. Prior works like Lobato et al. (2003); Kanno (2013) have formulated topology optimization problems as MIP. Nevertheless, our work is the first one to formulate the planar linkage problem as MIP, and to employ MIP to find the optimal topology, geometry, and trajectory of a planar linkage concurrently.

Planar Linkage is a set of 2D rigid objects connected together to convert motion and forces. We considered a subset of planar linkages connected by hinge joints. More generally, linkage structures can be coupled with pistons, gears, springs (Zhu et al. 2012), and compliant structures (Megaro et al. 2017) to exhibit more complex motion of the end-effector. But their designs are still relying heavily on human experiences. Parallel to their applications in robotics, the mathematical structure of planar linkages has been studied for centuries. In 1875, Kempe (1875) provided a constructive method to build a planar linkage that can trace out any algebraic curve, but the resulting linkage structure can be extremely complex. The recent work Gallet et al. (2017) proposed a construction leading to simpler structures, but their complexities are still too high for real-world applications. Therefore, practitioners rely on genetic algorithms (Zhu et al. 2012; Cabrera et al. 2002) or semi-automatic design tools (Thomaszewski et al. 2014; Bächer et al. 2015) to search for simple linkage structures that trace out a specified end-effector curve.

Planar Linkages Optimization Problem

In this section, we define a subset of the planar linkages considered in this paper. As illustrated in Figure 2 (a), we have a set of rod-like rigid bodies connected using hinge joints. The end points of these rigid bodies can take at most N distinct positions, denoted as a node set $\mathbf{n}_{1,\cdots,N}$, of which $\mathbf{n}_1$ is the rotational motor and $\mathbf{n}_N$ is the end-effector. Within one limit cycle, $\mathbf{n}_1$ traces out a circular curve centered at (X_C, Y_C) with a radius r:

$$\mathbf{n}_1(t) = \left(\sin(\pm t)r + X_C, \cos(\pm t)r + Y_C \right), \qquad (1)$$

which induces trajectories of other nodes $\mathbf{n}_i(t)$ via forward kinematics, where $t \in [0, 2\pi]$ is the time parameter. Throughout the paper, we use X, Y to denote the two axes of a 2D vector. The other $N - 2$ nodes can be one of two kinds: fixed or movable. In addition, a rigid body may exist between each pair of nodes $\mathbf{n}_{i,j}$, in which case $\|\mathbf{n}_i(t) - \mathbf{n}_j(t)\|$ must be a constant for all t. Given these definitions, we formulate the optimal planar linkage design problem as follows, where we take the following inputs:

- A user-provided target end-effector trajectory $\mathbf{n}_N^*(t)$.

- K: The maximal number of nodes in the planar linkage.

- T: The number of samples needed to discretize the end-effector trajectory $\mathbf{n}_N(t)$.

- S: The parameter controlling the accuracy of the MICP formulation. A larger S leads to greater accuracy and higher computational cost.

The output of our method is the tuple $\mathcal{L} = \langle N, C_{ji}, \mathbf{n}_i(t), X_C, Y_C, r \rangle$ defining both the topology and geometry of a planar linkage:

- An integer vector of size N (the number of nodes) containing the type of each node: fixed or movable.

- An $N \times N$ adjacent matrix $C^{N \times N}$ where $C_{ji} = 1$ means a rigid body connects $\mathbf{n}_i$ and $\mathbf{n}_j$.

- The position of $\mathbf{n}_{1,\cdots,N}(t)$ at a certain, arbitrary time instance t.

- X_C, Y_C, r are determined automatically by our MICP formulation.

The goal of our method is to find the above set of variables that minimizes the cost $\int_0^{2\pi} \|\mathbf{n}_N(t) - \mathbf{n}_N^*(t)\|^2 dt$. For two planar linkage structures, we claim that one is more accurate or optimal than the other if its end-effector trajectory incurs a smaller cost.

Constrained Linkage Kinematics

A valid planar linkage has only one degree of freedom, and the positions of all the nodes must be uniquely determined at a given time t and a fixed initial configuration. Therefore, planar linkages must use closed loops to eliminate all the redundant degrees of freedom. However, computing the forward kinematics for general, closed-loop articulated bodies involves solving constrained systems of equations (Featherstone 2014), which increases the complexity of the search of their topological structures. Therefore, we limit our research to a subset of linkage topology, which was originally proposed by Kecskemethy et al. (1997) and later adapted to human-assisted linked design in Bächer et al. (2015). The kinematics of this subset can be computed as easily as open-loop articulated bodies.

The key to our kinematic computation lies in the law-of-cosine. Specifically, if a node $\mathbf{n}_i$ is connected to two other nodes $\mathbf{n}_j, \mathbf{n}_k$ with known positions via rigid links with length l_{ji}, l_{ki}, then the position of $\mathbf{n}_i$ can be determined using the following function:

$$\mathbf{n}_i(\mathbf{n}_j, \mathbf{n}_k, l_{ji}, l_{ki}) \triangleq \frac{l_{ji}}{\|\mathbf{n}_j - \mathbf{n}_k\|} \mathbf{R}(\mathbf{n}_k - \mathbf{n}_j) + \mathbf{n}_j$$

$$\mathbf{R} \triangleq \begin{pmatrix} \cos & \pm\sqrt{1 - \cos^2} \\ \mp\sqrt{1 - \cos^2} & \cos \end{pmatrix} \qquad (2)$$

$$\cos \triangleq \frac{\|\mathbf{n}_j - \mathbf{n}_k\|^2 + l_{ji}^2 - l_{ki}^2}{2\|\mathbf{n}_j - \mathbf{n}_k\| l_{ji}},$$

where l_{ji} is the length of the link connecting $\mathbf{n}_i$ and $\mathbf{n}_j$ and $\mathbf{R}$ is a rotation matrix. Note that, if we have $\|\mathbf{n}_j - \mathbf{n}_k\| > l_{ji} + l_{ki}$ or $\|\mathbf{n}_j - \mathbf{n}_k\| < |l_{ji} - l_{ki}|$ at certain time instance, then the three nodes cannot form a triangle and such a configuration cannot be realized. The above law-of-cosine has two solutions (determined by the sign of off-diagonal entries of $\mathbf{R}$) corresponding to two mirrored triangles, but the linkage can only exhibit a unique, continuous motion without flipping any triangles. As proposed in Kecskemethy et al. (1997), we construct a planar linkage by recursively connecting a new node with two other nodes using rigid links. As a result, the position of each new node can be determined by the law-of-cosine. As illustrated in Figure 2 (b), such topological constraints can be formalized using a connectivity graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where the vertex set $\mathcal{V} =$
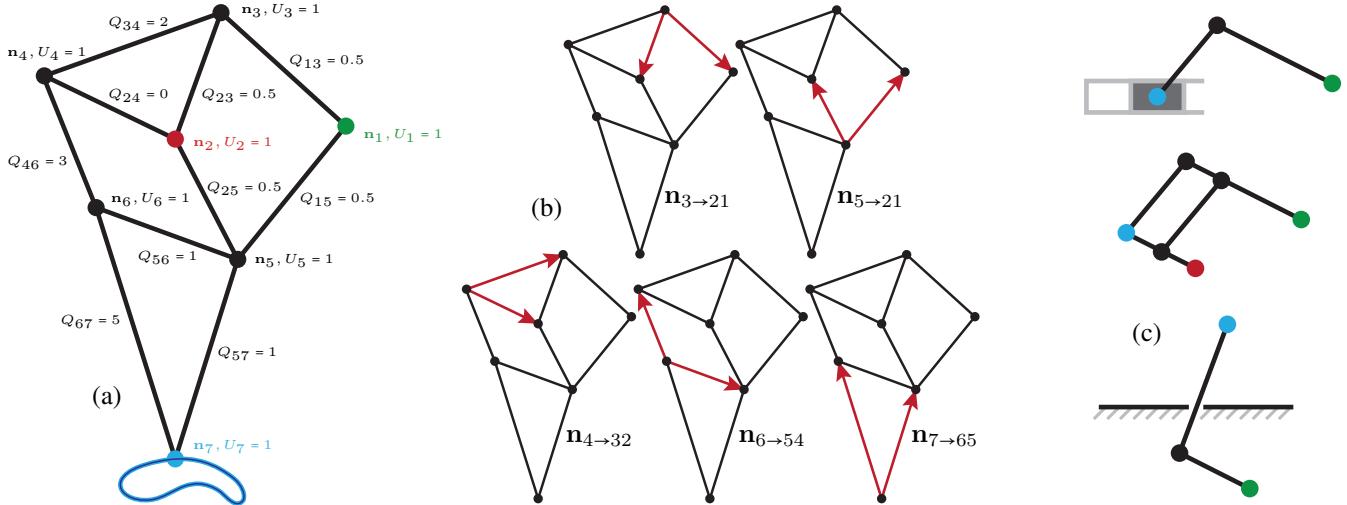
Figure 2. (a): Jansen's mechanics is a planar linkage involving 7 nodes. The motor node $\mathbf{n}_1$ is green, the fixed node $\mathbf{n}_2$ is red, the movable nodes $\mathbf{n}_{3,4,5,6}$ are black, and the end-effector node $\mathbf{n}_7$ is blue. (b): Our method is based on a prior symbolic representation (Kecskemethy et al. 1997; Bächer et al. 2015). This representation assumes that each node is connected to zero or two other nodes with lower indices (marked by arrows): $\mathbf{n}_{3\to21}$, $\mathbf{n}_{5\to21}$, $\mathbf{n}_{4\to32}$, $\mathbf{n}_{6\to54}$, $\mathbf{n}_{7\to65}$. (c): We show three important linkage structures beyond our topology subset, which involve pistons, complex loops, and pinhole constraints.

$\{\mathbf{n}_1, \cdots, \mathbf{n}_N\}$ consists of all the nodes and the edge set $\mathcal{E} = \{\mathbf{n}_{a\to b}\}$ consists of node pairs connected by a link, on which our topological constraint can be summarized below:

Assumption 1. *The vertex set $\mathcal{V}$ of the connectivity graph has a topological ordering, by which each node $\mathbf{n}_i$ is connected to either zero or two other nodes with lower indices (denoted as $\mathbf{n}_{i\to jk} \triangleq \{\mathbf{n}_{i\to j}, \mathbf{n}_{i\to k}\}$ with $i > j$ and $i > k$).*

With the topological ordering, we can determine the position of each node in ascending order. A node connected to zero lower-index nodes is either the actuator $\mathbf{n}_1$ or a fixed node, otherwise, we apply the law-of-cosine to $\mathbf{n}_{i\to jk}$ and determine the position of $\mathbf{n}_i$. Using this method, the forward kinematics of the entire linkage structure can be computed within $\mathcal{O}(N)$ and Jacobian matrix with respect to link lengths can also be computed within $\mathcal{O}(N)$ using the adjoint method as summarized in Algorithm 1. This Jacobian matrix can be used to locally optimize the geometry of a linkage structure using a gradient-based algorithm as done in Bächer et al. (2015). We emphasize that many important linkage structures, involving pistons, complex loops, or pinhole constraints, are beyond our subset, as illustrated in Figure 2 (c). However, our subset already encompasses a rich variety of end-effector trajectories as illustrated by our results.

Planar Linkage Optimization as MIQCQP

In this section, we show that the optimal planar linkage design problem can be reformulated as a MIQCQP. Such reformulation allows us to utilize mature algorithms and approximation techniques to find (nearly) optimal solutions. Unlike prior methods (Thomaszewski et al. 2014; Bächer et al. 2015) for planar linkage optimization that are based on minimal coordinates, we propose to use maximal coordinates to represent the configuration. Maximal coordinates treat all the node positions $\mathbf{n}_i$ as independent decision variables

---

Algorithm 1 Forward/Inverse Kinematics

1: Compute $\mathbf{n}_1$ using Equation 1
2: **for** $i = 2, \cdots, N$ **do**   ▷ Forward Kinematics
3:     **if** $\mathbf{n}_i$ is not fixed with $\mathbf{n}_{i\to jk} \subset \mathcal{E}$ **then**
4:         Compute $\mathbf{n}_i$ using Equation 2
5:         Output $\mathbf{n}_i$
6: **for** $i = 2, \cdots, N$ **do**   ▷ Inverse Kinematics
7:     **if** $\mathbf{n}_i$ is not fixed with $\mathbf{n}_{i\to jk} \subset \mathcal{E}$ **then**
8:         Compute $\frac{\partial \mathbf{n}_i}{\partial l_{ji}}, \frac{\partial \mathbf{n}_i}{\partial l_{ki}}, \frac{\partial \mathbf{n}_i}{\partial \mathbf{n}_j}, \frac{\partial \mathbf{n}_i}{\partial \mathbf{n}_k}$ using Equation 2
9:         $\frac{\partial \mathbf{n}_N}{\partial l_{ji}} \leftarrow \frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_i}\frac{\partial \mathbf{n}_i}{\partial l_{ji}}$ $\frac{\partial \mathbf{n}_N}{\partial l_{ki}} \leftarrow \frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_i}\frac{\partial \mathbf{n}_i}{\partial l_{ki}}$
10:        Output $\frac{\partial \mathbf{n}_N}{\partial l_{ji}}, \frac{\partial \mathbf{n}_N}{\partial l_{ki}}$
11:        $\frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_j} \leftarrow \frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_i}\frac{\partial \mathbf{n}_i}{\partial \mathbf{n}_j}$ $\frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_k} \leftarrow \frac{\partial \mathbf{n}_N}{\partial \mathbf{n}_i}\frac{\partial \mathbf{n}_i}{\partial \mathbf{n}_k}$ ▷ Adjoint

---

and introduce additional constraints to ensure the rigidity of each link. Although maximal coordinates use more variables and involve solving constrained systems of equations, the constraints take a simpler form to be handled by numerical optimization tools. A similar idea has been used by Dai et al. (2017) to compute globally optimal inverse kinematics for sequential manipulators. Conceptually, our goal is to solve the following infinite-dimensional program:

$$\underset{C_{ji}, \mathbf{n}_i(t), X_C, Y_C}{\operatorname{argmin}} \int_0^{2\pi} \|\mathbf{n}_N(t) - \mathbf{n}_N^*(t)\|^2 dt + \text{reg.}$$

$$\text{s.t. Topological Constraints}$$
$$\text{Geometric Constraints,}$$

(3)

where the main objective is to search for a linkage structure whose end-effector curve matches the user-provided target curve as much as possible. Meanwhile, we introduce a regularization term to reduce the fabrication cost, which penalizes the number of links and the total length of links. To ensure that the linkage is realizable and well-behaved, we introduce several sets of constraints as summarized in Table 1. The topological constraints ensure that the

linkage structure satisfies Assumption 1, and the geometric constraints ensure the kinematic feasibility. In the following sections, we use additional notations to mark the range of indices to which a constraint applies. If no notations are used, then the constraint applies to all the index combinations. Furthermore, we assume a variable is continuous with no bounds, unless otherwise specified (e.g., as a binary variable).

Constraint Set	Guarantees
NodeUsageConstraint	unambiguous node type definitions
NodeConnectivityConstraint	node connectivity satisfies Assumption 1
NoWasteConstraint	each node affects the end-effector trajectory
MovableNodeConstraint	movable nodes are connected to actuator
RealizabilityConstraint	linkage structure can be fabricated
AreaConstraint	end-effector trajectory is unique
MotorConstraint	motor is rotational

Table 1. A summary of topology and geometric constraint sets and the guarantees corresponding to each constraint.

Topological Constraints

We design four types of topological constraints. Our first set of constraints is denoted as **NodeUsageConstraint**, which allows a numerical optimizer to automatically determine the number of nodes and links to use. We further ensure that each node can either be movable or fixed, but not both. Since the number of nodes is unknown, we assume that the maximum number of nodes is $K \geq N$. We will have then all the joints defined, but only N of them should be present, which will be formalized by introducing a binary indicator variable U_i. For each node other than the first motor node $\mathbf{n}_1$, $U_i = 1$ indicates that $\mathbf{n}_i$ will be present as a part of the planar linkage structure. In addition, we need another binary variable F_i such that $F_i = 1$ indicates that $\mathbf{n}_i$ is fixed and $F_i = 0$ indicates that $\mathbf{n}_i$ is movable. These two sets of variables are under the constraint that only a used node can be movable. In addition, we assume that the last node $\mathbf{n}_K$ is the end-effector that must be used. In summary, we introduce the following sets of variables and node-state constraints:

$$U_i, F_i \in \{0, 1\}$$
$$1 - F_i \leq U_i \quad U_1 = U_K = 1 \quad F_1 = 0. \tag{4}$$

Our next set of constraints is denoted as **NodeConnectivityConstraint**, which ensures that each movable node is connected to exactly two other nodes with lower indices. As a result, the movable node and the two other ones will form a triangle and the position of the movable node can then be determined via the law-of-cosine. We introduce auxiliary variables C_{ji}^1 to indicate whether $\mathbf{n}_j$ is the first node to which $\mathbf{n}_i$ is connected. C_{ji}^2 indicates whether $\mathbf{n}_j$ is the second node to which $\mathbf{n}_i$ is connected. In addition, we introduce two verbose variables $C_{0i}^{1,2} = 1$ to indicate that $\mathbf{n}_i$ is connected to nothing, which is the case when $\mathbf{n}_i$ is fixed or unused. The resulting constraint set is:

$$C_{ji}^1, C_{ji}^2 \in \{0, 1\} \quad \forall 1 \leq j < i \leq K$$
$$C_{ji} = C_{ji}^1 + C_{ji}^2 \in [0, 1] \quad C_{ji}^1 \leq U_j \wedge C_{ji}^2 \leq U_j$$
$$\sum_{j=1}^{i-1} C_{ji} = 2 - 2F_i \quad \forall 2 \leq i \leq K \tag{5}$$
$$C_{0i}^d \in \{0, 1\} \quad \sum_{j=0}^{i-1} C_{ji}^d = 1.$$

When $\mathbf{n}_i$ is fixed in the above formulation, then $F_i = 1$ in Equation 5 and all C_{ji} are zero except for $C_{0i}^{1,2} = 1$ due to the sum-to-one constraints. If $\mathbf{n}_i$ is movable, then $F_i = 0$ and C_{ji} sums to two. As a result, there must be $j_1, j_2 < i$ such that $C_{j_1 i}^1 = 1$ and $C_{j_2 i}^2 = 1$. Note that j_1 and j_2 must be different because otherwise the constraint that $C_{ji} \in [0, 1]$ will be violated. In addition, since the first node $\mathbf{n}_1$ is the motor node, it is excluded from these connectivity constraints.

Our third set of constraints is denoted as **NoWasteConstraint**, which ensures that the linkage structure contains no wasted parts. In other words, each node must have some influence on the trajectory of the end-effector node and the first motor node must be connected to others. We model these constraints using the MICP formulation of network flows (Conforti et al. 2009). Specifically, each node $\mathbf{n}_i$ will generate a flux that equals to U_i, and we assume that there is a flow edge defined between each pair of nodes with capacity Q_{ji}. We require inward-outward flux balance for each node except the end-effector node:

$$Q_{ji} \geq 0 \quad \forall 1 \leq j < i \leq K$$
$$Q_{ji} \leq C_{ji} K$$
$$U_i + \sum_{j=1}^{i-1} Q_{ji} = \sum_{k=i+1}^{K} Q_{ik} \quad \forall 1 \leq i \leq K - 1, \tag{6}$$

where the lefthand (resp. righthand) side of the last equation equals to the inward (resp. outward) flux. The inward flux is a sum of the newly generated flux U_i and the flux passed on from nodes with lower-indices. Note that the flux-balance condition is imposed on every node except for the end-effector node, which means that only the end-effector node can deplete fluxes. As a result, every node must be connected to the end-effector node in order for its newly generated flux U_i to be depleted. We illustrate one solution of U_i, Q_{ji} in Figure 2 (a). Here we adopt the big-M method (Bertsimas and Tsitsiklis 1997) in the second constraint to ensure that only edges between connected nodes can have a capacity up to K. Big-M is a well-known method in mixed-integer modeling for choosing one element from a discrete set, or for choosing one case from several possible cases.

Finally, using a similar idea, we also formulate a constraint that restricts a movable node to be connected to at least one other movable node (otherwise the movable node never moves), which is denoted as **MovableNodeConstraint**. We assume that each node $\mathbf{n}_i$ generates a reversed outward flux that equals to $1 - F_i$ and there is a flow edge defined between each pair of nodes with capacity R_{ji}. We require an inward-outward flux balance for each node except for the motor

node:

$$R_{ji} \geq 0 \quad \forall 1 \leq j < i \leq K$$
$$R_{ji} \leq C_{ji}K \quad R_{ji} \leq (1 - F_j)K$$
$$\sum_{j=1}^{i-1} R_{ji} = 1 - F_i + \sum_{k=i+1}^{K} R_{ik} \quad \forall 2 \leq i \leq K. \tag{7}$$

These four constraints ensure that the planar linkage structure is symbolically correct, independent of the concrete geometric shape. In summary, our topological constraints involve:

Topological Constraints $\triangleq$

$$\begin{cases} \text{Equation 4 : NodeUsageConstraint} \\ \text{Equation 5 : NodeConnectivityConstraint} \\ \text{Equation 6 : NoWasteConstraint} \\ \text{Equation 7 : MovableNodeConstraint} \end{cases}$$

A planar linkage satisfies Assumption 1 if topological constraints are satisfied. All the equations in this section are mixed-integer linear constraints that can be satisfied using an off-the-shelf MICP solver such as Gurobi Optimization (2018) as long as a solution exists.

Reducing Binary Variables

MIP solvers build a search tree by branching on binary variables whose continuous relaxation is not exact, so the size of the search tree and the performance of MIP solvers is closely related to the number of binary variables. Altogether, our topological constraints use $4K + K(K + 1)$ binary variables, where the quadratic term comes from C_{ji}^1, C_{ji}^2. We can further reduce the number of variables to $\mathcal{O}(K\lceil \log K \rceil)$ by adopting the idea of a special ordered set of type 1 ($\mathcal{SOS}_1$) (Vielma and Nemhauser 2011). Intuitively, $\mathcal{SOS}_1$ is a constraint that only one out of a set of K variables can take a non-zero value. Their main idea is to order these variables from 1 to K and choose a number within this range. To this end, a binary variable is introduced to indicate whether each binary bit is 1. We can apply this idea to Equation 5 by observing that the two constraints $C_{ji}^d \in \{0, 1\}$ and $\sum_{j=0}^{i-1} C_{ji}^d = 1$ is equivalent to:

$$\{C_{ji}^d | j = 0, \cdots, i - 1\} \in \mathcal{SOS}_1$$
$$\sum_{j=0}^{i-1} C_{ji}^d = 1.$$

The $\mathcal{SOS}_1$ constraints can be converted to conventional linear constraints via Algorithm 2, where we introduce at most $\lceil \log K \rceil$ auxiliary binary variables denoted as: $\mathbb{I}_1, \cdots, \mathbb{I}_{\lceil \log K \rceil}$.

Geometric Constraints

We introduce three sets of geometric constraints to ensure that our linkage structure can be fabricated and generate unique end-effector trajectories. Our first constraint set is denoted as **RealizabilityConstraint**, which ensures that the node positions at all time instances t can be realized by the same set of links, so that the linkage structure can be fabricated. For a pair of nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ ($j < i$), there might

---

Algorithm 2 Constraint $\{C_{ji}^d | j = 0, \cdots, i - 1\} \in \mathcal{SOS}_1$

Input: $\mathbb{I}_1, \cdots, \mathbb{I}_{\lceil \log i \rceil} \in \{0, 1\}$
1: **for** $j = 0, \cdots, i - 1$ **do**
2:     **for** bit$=1, \cdots, \lceil \log i \rceil$ **do**
3:         **if** $j \wedge 2^{\text{bit}-1} = 0$ **then** $\triangleright$ Bitwise and
4:             Add constraint $C_{ji}^d \leq \mathbb{I}_{\text{bit}}$
5:         **else**
6:             Add constraint $C_{ji}^d \leq 1 - \mathbb{I}_{\text{bit}}$

---

be a link connecting them as indicated by the variable C_{ji}^d. Equation 5 dictates that there is a link between the two nodes if and only if $\sum_{d=1}^2 C_{ji}^d = 1$. Further, we ensure that the node never moves if $F_i = 1$. Put together, our realizability constraint takes the following form:

$$\|\mathbf{n}_i(t_1) - \mathbf{n}_j(t_1)\|^2 = \|\mathbf{n}_i(t_2) - \mathbf{n}_j(t_2)\|^2$$
$$\forall 0 \leq t_1 < t_2 \leq 2\pi \wedge \sum_{d=1}^2 C_{ji}^d = 1$$
$$\|\mathbf{n}_i(t_1) - \mathbf{n}_i(t_2)\| = 0 \quad \forall 0 \leq t_1 < t_2 \leq 2\pi \wedge F_i = 1.$$

We need to absorb the decision variable C_{ji}^d into the constraint to be handled by the optimizer, to which end the big-M method can be used to derive the following equivalent constraints:

$$\|\mathbf{n}_i(t) - \mathbf{n}_j(t) - \mathbf{d}_{di}(t)\| \leq (1 - C_{ji}^d)2\sqrt{2}B \quad \forall 2 \leq i \leq K$$
$$\|\mathbf{d}_{di}(t_1)\|^2 = \|\mathbf{d}_{di}(t_2)\|^2 \quad \forall 0 \leq t_1 < t_2 \leq 2\pi \tag{8}$$
$$\|\mathbf{n}_i(t_1) - \mathbf{n}_i(t_2)\| \leq (1 - F_i)2\sqrt{2}B \quad \forall 0 \leq t_1 < t_2 \leq 2\pi.$$

The big-M method requires an upper bound on the length difference between the two nodes at different time instances. If we assume the planar linkage is bounded inside a box with side length B, then a conservative upper bound is twice the diagonal length $2\sqrt{2}B$. Here we introduce the slack variables $\mathbf{d}_{di}(t)$ that are constrained to be equal to $\mathbf{n}_i(t) - \mathbf{n}_j(t)$ when $C_{ji}^d = 1$, in which case the equal-length constraint is specified for $\mathbf{d}_{di}(t)$ instead. Otherwise, when $\mathbf{n}_i$ is fixed, $\mathbf{d}_{di}(t)$ can take any value and the equal-length constraint can be trivially satisfied.

Our second constraint set ensures that the orientations of nodes are unambiguous during the entire limit cycle and a linkage structure generates a unique end-effector trajectory. This is denoted as **AreaConstraint**. According to Equation 2, there are two possible positions for a node $\mathbf{n}_i$ corresponding to the sign of off-diagonal terms in the rotation matrix $\mathbf{R}$. To ensure that the sign of the two off-diagonal terms never changes, it is enough to bound their values away from zero. Equivalently, we can bound the cosine value away from 1 by a small margin denoted as ϵ, i.e., $\cos \leq 1 - \epsilon$. After some rearrangement, we derive the following equivalence:

$$|\|\mathbf{n}_j(t) - \mathbf{n}_k(t)\| - l_{ji}|^2 \leq l_{ki}^2 - 2\epsilon \|\mathbf{n}_j(t) - \mathbf{n}_k(t)\| l_{ji}.$$

The above constraint is not a quadratic form, but an equivalent form exists by observing that $\cos = 1$ if and only if the area of the triangle formed by $\mathbf{n}_i, \mathbf{n}_j, \mathbf{n}_k$ has zero area. Therefore, we can bound the signed area away from zero as follows:

$$\langle [\mathbf{d}_{1i}(t)]^\perp, \mathbf{d}_{2i}(t) \rangle \geq \epsilon, \tag{9}$$

where the $\perp$ superscript denotes the vector rotated by 90 degrees clockwise. Note that the sign of the area does not matter because they correspond to two binary variable assignments $C_{ji}^1 = C_{ki}^2 = 1$ and $C_{ki}^1 = C_{ji}^2 = 1$, and our optimizer is free to choose one of the two cases. Additionally, the area constraint on the triangle between $\mathbf{n}_i, \mathbf{n}_j, \mathbf{n}_k$ should only be activated when one of the two binary variable assignments happens, but we do not need to consider this issue here as it has been done in the first constraint of Equation 8.

We introduce the last set of constraints, denoted as **MotorConstraint**, to specify the rotational motion of the first node $\mathbf{n}_1$. The motion specified by Equation 1 must be replaced with a quadratic form to be consumed by MIQCQP. Note that the motor can rotate clockwise or counter-clockwise. Therefore, we introduce a binary variable D to distinguish between these two cases. Our constraint set is then formulated as:

$$\begin{aligned}
\|\mathbf{R}(t)\mathbf{d}_{d1}(t) - \mathbf{d}_{d1}(0)\| &\leq D2\sqrt{2}B \\
\|\mathbf{R}(-t)\mathbf{d}_{d1}(t) - \mathbf{d}_{d1}(0)\| &\leq (1-D)2\sqrt{2}B \quad (10) \\
\mathbf{d}_{d1}(t) &= \mathbf{n}_1(t) - (X_C, Y_C),
\end{aligned}$$

where we have used the same upper bound for the big-M method. In summary, our geometric constraints involve:

Geometric Constraints $\triangleq$

$$\begin{cases}
\text{Equation } 8 : \text{RealizabilityConstraint} \\
\text{Equation } 9 : \text{AreaConstraint} \\
\text{Equation } 10 : \text{MotorConstraint}
\end{cases} .$$

We formalized the following result which shows that our linkage structure is bounded away from singularities under geometric constraints:

Lemma 1. *If Equation 8, Equation 9, and Equation 10 are satisfied, then for any t, the linkage structure has no singular configurations of any types (Gosselin and Angeles 1990), and the end-effector trajectory is unique.*

Proof. The forward kinematics of our linkage can be computed by solving the implicit equations:

$$\begin{aligned}
\mathbf{R}(\pm t)\mathbf{d}_{d1}(t) - \mathbf{d}_{d1}(0) &= 0 \\
\left. \begin{aligned} \|\mathbf{n}_i - \mathbf{n}_j\|^2 - l_{ji}^2 &= 0 \\ \|\mathbf{n}_i - \mathbf{n}_k\|^2 - l_{ki}^2 &= 0 \end{aligned} \right\} &\forall \mathbf{n}_{i \to jk},
\end{aligned}$$

all the equations of which hold by Equation 8 and Equation 10. We can summarize the above equations as a vector-valued implicit function $\mathbf{F}(t, \mathbf{n}) = 0$, where $\mathbf{n}$ is a concatentation of movable node positions. Only the first two rows of $\mathbf{F}$ correspond to the motor node $\mathbf{n}_1$ and each non-motor, movable node $\mathbf{n}_{i \to jk}$ occupies two additional rows. The sensitivity analysis leads to:

$$\frac{\partial \mathbf{F}}{\partial t} \dot{t} + \frac{\partial \mathbf{F}}{\partial \mathbf{n}} \dot{\mathbf{n}} = 0.$$

Since only the first two rows are functions of t, we immediately have $\partial \mathbf{F} / \partial t$ is a single non-zero column, which implies that the linkage does not have singular configuration of type (i) or (iii) (Gosselin and Angeles 1990). Next, we

focus on $\partial \mathbf{F} / \partial \mathbf{n}$, which is a square matrix. This is because each (motor or non-motor), movable node will introduce two rows and two columns. We first order the rows and columns of $\mathbf{F}$ in the topological ordering as implied by Assumption 1, then $\partial \mathbf{F} / \partial \mathbf{n}$ becomes a 2×2 block upper-triangular matrix of the following type:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{n}} = \begin{pmatrix} \ddots & & \mathbf{0} \\ \cdots & \begin{pmatrix} 2(X_i - X_j) & 2(Y_i - Y_j) \\ 2(X_i - X_k) & 2(Y_i - Y_k) \end{pmatrix} \end{pmatrix},$$

from which we immediately have $\det(\partial \mathbf{F} / \partial \mathbf{n}) \geq (4\epsilon)^{\bar{N}-1} > 0$ by Equation 9 and the linkage does not have singular configuration of type (ii) (Gosselin and Angeles 1990), where $\bar{N}$ denotes the number of movable nodes. Since $\det(\bullet)$ is a continuous function, the linkage structure is bounded away from singularity and the end-effector trajectory is unique. $\square$

The geometric constraint set consists of an infinite number of quadratic constraints. Unfortunately, the two crucial constraints (i.e.: the equal-length constraints in Equation 8 and Equation 9) are non-convex, so finding the feasible solution set for them is not tractable. In the next section, we propose methods to discretize and then approximate its optimal solution set.

Approximate MIQCQP Solver

The above-mentioned MIQCQP involves infinitely many variables, intractable integral in the objective function, and non-convex constraints. To derive a finite-dimensional problem, we discretize our trajectory, as well as the user-specified target trajectory, by sampling T nodes evenly at $t^q = 2\pi q / T$ with $q = 1, \cdots, T$. Under such discretization, our objective function can be approximated as:

$$\begin{aligned}
& \int_0^{2\pi} \|\mathbf{n}_N(t) - \mathbf{n}_N^*(t)\|^2 dt + \text{reg.} \\
& \approx \frac{2\pi}{T} \sum_{q=1}^T \|\mathbf{n}_N(t^q) - \mathbf{n}_N^*(t^q)\|^2 + \lambda \sum_{i=1}^N U_i, \quad (11)
\end{aligned}$$

where we have utilized the variable U_i as our regularization term, which encourages the optimizer to use as few links as possible, and λ is the weight coefficient that balances the exactness of the target curve matching and the simplicity of the linkage structure. Similarly, we can discretize the geometric constraint Equation 8 as:

$$\|\mathbf{n}_i(t^q) - \mathbf{n}_j(t^q) - \mathbf{d}_{di}(t^q)\| \leq (1 - C_{ji}^d)2\sqrt{2} \quad (12)$$

$$\|\mathbf{d}_{di}(t^q)\|^2 = \|\mathbf{d}_{di}(t^{q+1})\|^2 \quad (13)$$

$$\|\mathbf{n}_i(t^q) - \mathbf{n}_i(t^{q+1})\|^2 \leq (1 - F_i)2\sqrt{2}B, \quad (14)$$

Equation 9 as:

$$\left\langle [\mathbf{d}_{1i}(t^q)]^\perp, \mathbf{d}_{2i}(t^q) \right\rangle \geq \epsilon, \quad (15)$$

and finally Equation 10 as:

$$\begin{aligned}
\|\mathbf{R}(t^q)\mathbf{d}_{d1}(t^q) - \mathbf{d}_{d1}(0)\| &\leq D2\sqrt{2}B \\
\|\mathbf{R}(-t^q)\mathbf{d}_{d1}(t^q) - \mathbf{d}_{d1}(0)\| &\leq (1-D)2\sqrt{2}B \quad (16) \\
\mathbf{d}_{d1}(t^q) &= \mathbf{n}_1^d(t^q) - (X_C, Y_C).
\end{aligned}$$

In summary, our discrete MIQCQP takes the following form:

$$\text{argmin Equation } 11 \tag{17}$$
$$\text{s.t.} \quad \text{Equation } 4, 5, 6, 7, 12, 13, 14, 15, 16.$$

The constraint set of Equation 17 is a subset of Equation 3, which defines an outer approximation for the feasible set of Equation 3. Specifically, a feasible solution for Equation 17 might not be realizable or satisfy the area constraints in between two time samples, but such discretization error can be made arbitrarily close to zero as $T \to \infty$. Although Equation 17 is a finite-dimensional problem, finding its global optima is still intractable due to the non-convex constraints. In this section, we propose three algorithms to approximate its solution.

Mixed-Integer Conic Programming

Our first algorithm uses a convex outer approximation for each non-convex quadratic constraint and then uses off-the-shelf MICP solver such as Gurobi Optimization (2018) to find approximate solutions. To relax the non-convex constraint Equation 13, we borrow techniques from Liberti (2004). We notice that the non-convex constraint can be written as a linear constraint as the square of the node coordinates $[X_{di}]^2$, $[Y_{di}]^2$:

$$\|\mathbf{d}_{di}(t^q)\|^2 = \|\mathbf{d}_{di}(t^{q+1})\|^2 \Leftrightarrow$$
$$[X_{di}]^2 (t^q) + [Y_{di}]^2 (t^q) = [X_{di}]^2 (t^{q+1}) + [X_{di}]^2 (t^{q+1}),$$

where we assume $\mathbf{d}_{di} = (X_{di}, Y_{di})$. For any decision variable α, Liberti (2004) proposed a technique to derive a piecewise linear upper bound of α^2, denoted as $\alpha^2 \le \tilde{\alpha}$, and the approximation error $(\tilde{\alpha} - \alpha^2)$ can be made arbitrarily small by using more pieces. As illustrated in Figure 3, the upper bound is formed by evenly sampling $S + 1$ points on the curve α^2 and then connect the samples using straight line segments. If we know that $\alpha \in [-B/2, B/2]$, then the sample points are $\alpha_s = sB/S - B/2$, where $s = 0, \cdots, S$. To constraint $\tilde{\alpha}$ to lie on the set of line segments, we use the following set of constraints:

$$\binom{\alpha}{\tilde{\alpha}} = \sum_{s=0}^{S} \lambda_s \binom{\alpha_s}{\alpha_s^2} \tag{18}$$
$$\{\lambda_{0,\cdots,S}\} \in \mathcal{SOS}_2 \quad \sum_{s=0}^{S} \lambda_s = 1,$$

where we have used auxiliary variables λ_s that belong to the special ordered set of type 2 ($\mathcal{SOS}_2$) (Vielma and Nemhauser 2011). $\mathcal{SOS}_2$ requires that at most two of the variables in an ordered set with consecutive indices can take non-zero values. The $\mathcal{SOS}_2$ constraint can be converted to a set of linear constraints and a $\mathcal{SOS}_1$ constraint using Algorithm 3. With the upper bound, we can approximate the non-convex constraint with two linear constraints:

$$\|\mathbf{d}_{di}(t^q)\|^2 \le \tilde{X}_{di}(t^{q+1}) + \tilde{Y}_{di}(t^{q+1})$$
$$\|\mathbf{d}_{di}(t^{q+1})\|^2 \le \tilde{X}_{di}(t^q) + \tilde{Y}_{di}(t^q). \tag{19}$$

It can be shown that Equation 19 forms an outer approximation for the feasible region of the non-convex
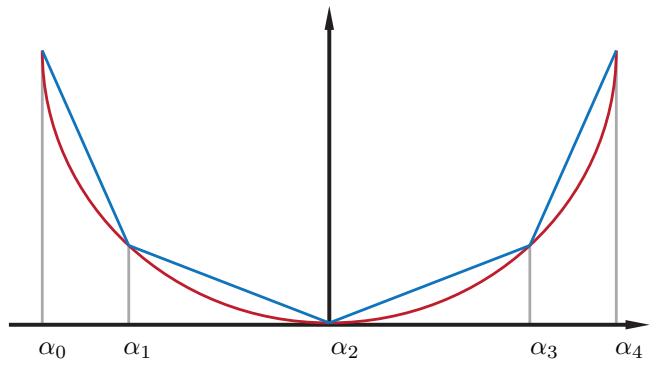

Figure 3. An illustration of the piecewise linear upper bound (blue) of the quadratic curve α^2 (red) with $S = 4$.

constraint and the approximation error diminishes as $S \to \infty$. A similar formulation has been used in Dai et al. (2017) to discretize the space of unit vectors. To formulate Equation 19, we need $4TK$ upper bounds, each one introducing $\lceil \log S \rceil$ binary decision variables, so we need $4TK\lceil \log S \rceil$ binary variables altogether.

Algorithm 3 Constraint $\{\lambda_0, \cdots, \lambda_S\} \in \mathcal{SOS}_2$

Input: Auxiliary variables $\bar{\lambda}_0, \cdots, \bar{\lambda}_{S-1} \in [0, 1]$
Input: $\bar{\lambda}_{-1} = \bar{\lambda}_S = 0$
1: Add constraint $\{\bar{\lambda}_0, \cdots, \bar{\lambda}_{S-1}\} \in \mathcal{SOS}_1$ ▷ Algorithm 2
2: **for** $i = 0, \cdots, S$ **do**
3:     Add constraint $\lambda_i \le \bar{\lambda}_{i-1} + \bar{\lambda}_i$

We adopt a similar technique to relax the area constraints Equation 15. These constraints involve two bilinear terms:

$$\langle [\mathbf{d}_{1i}(t^q)]^\perp, \mathbf{d}_{2i}(t^q) \rangle \ge \epsilon \Leftrightarrow$$
$$X_{1i}(t^q)Y_{2i}(t^q) - Y_{1i}(t^q)X_{2i}(t^q) \ge \epsilon,$$

which contributes to the non-convexity. A standardized technique to relax bilinear constraints is the McCormick envelop (see Liberti (2004) for more details), where the feasible domain is outer-approximated as a union of convex hulls. However, unlike for length constraints, we argue that inner approximations should be used for area constraints. This is because the parameter ϵ is a small constant and the McCormick envelope would also introduce relaxation errors with a larger magnitude than ϵ, so even if the area constraints after the McCormick relaxation are satisfied, the exact constraints are still violated, rendering the relaxation useless. Instead, we propose an inner-approximation scheme. We cut the 2D rotation group $\mathcal{SO}(2)$ into S sectors, as illustrated in Figure 4 (a), so that $\mathbf{d}_{1i}$ will only fall into one of the S sectors. If $\mathbf{d}_{1i}$ falls in a particular sector, then we restrict $\mathbf{d}_{2i}$ to its left half-space that is at least ϵ degrees apart, as shown in Figure 4 (b). If we use an $\mathcal{SOS}_1$ constraint to select the sector in which $\mathbf{d}_{1i}$ falls, then only $\mathcal{O}(TK\lceil \log S \rceil)$ binary decision variables are needed. A minor issue with this formulation is that the allowed region of $\mathbf{d}_{2i}$ jumps discontinuously as $\mathbf{d}_{1i}$ changes continuously. We can fix this problem by double-covering the region of $\mathcal{SO}(2)$ by using $2S$ sectors, as shown in Figure 4 (c). To formulate these constraints, we assume that each sector of $\mathcal{SO}(2)$ is flagged by a selector variable γ_l, which is bounded by its left/right
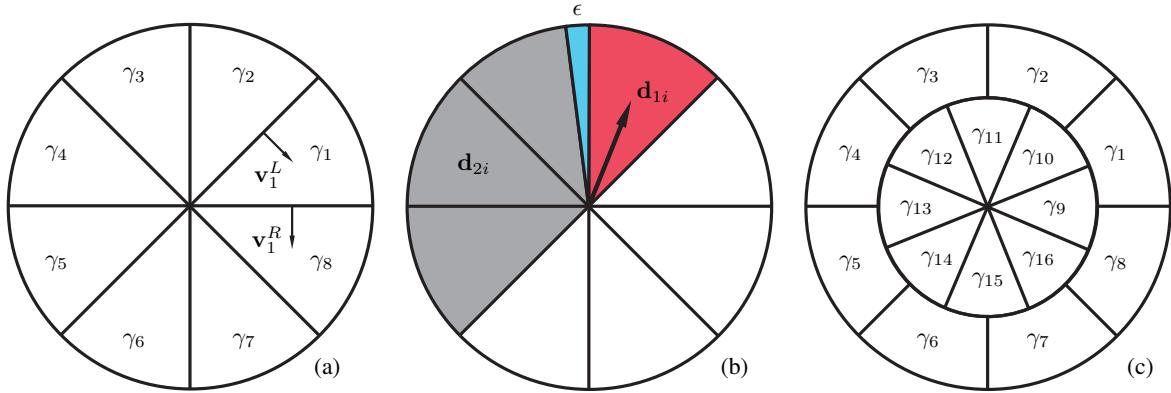
Figure 4. Linear relaxation of angle constraints. (a): We cut $\mathcal{SO}(2)$ into 8 sectors, each of which is selected by a γ-flag. The sector selected by γ_1 is bounded by its left/right normal vectors $\mathbf{v}_1^L/\mathbf{v}_1^R$. (b): If $\mathbf{d}_{1i}$ falls in the red area, then we restrict $\mathbf{d}_{2i}$ to the gray area, which is at least ϵ-apart (blue). However, when $\mathbf{d}_{1i}$ moves across sector boundaries, the gray area jumps discontinuously. (c): To avoid discontinuous changes for $\mathbf{d}_{2i}$ when $\mathbf{d}_{1i}$ undergoes continuous changes, we propose to double cover $\mathcal{SO}(2)$.

unit-length plane-normal vectors $\mathbf{v}_l^L/\mathbf{v}_l^R$. Combined with the fact that constraints should only be satisfied for one particular sector and for only movable nodes, we have the following formulation:

$$
\begin{aligned}
&< \mathbf{v}_l^L, \mathbf{d}_{1i}(t^q) >\geq \sqrt{2}B(\gamma_{l,i}(t^q) - 1) \\
&< \mathbf{v}_l^R, \mathbf{d}_{1i}(t^q) >\leq \sqrt{2}B(1 - \gamma_{l,i}(t^q)) \\
&< \mathbf{R}(\epsilon)\mathbf{v}_l^L, \mathbf{d}_{2i}(t^q) >\leq \sqrt{2}B(1 - \gamma_{l,i}(t^q)) \\
&< \mathbf{R}(\pi)\mathbf{v}_l^R, \mathbf{d}_{2i}(t^q) >\geq \sqrt{2}B(\gamma_{l,i}(t^q) - 1) \\
&\{\gamma_{1,1i}(t^q), \cdots, \gamma_{2S,i}(t^q)\} \in \mathcal{SOS}_1 \quad \sum_{l=1}^{2S} \gamma_{l,i}^d = 1,
\end{aligned}
\tag{20}
$$

where $TK\lceil \log(2S) \rceil$ binary variables are used.

We conclude that MICP can solve the relaxed and discretized MIQCQP problem by using $\mathcal{O}(K\lceil \log K \rceil + 4TK\lceil \log S \rceil + TK\lceil \log(2S) \rceil)$ binary variables (by replacing Equation 13 with Equation 19 and Equation 15 with Equation 20), where the first term is due to topological constraints, the second term due to relaxed length constraints, and the last term due to relaxed angle constraints. Furthermore, the error due to relaxation and discretization can be made arbitrarily small as $T, S \to \infty$. In practice, the solution of MICP does not satisfy the exact, non-convex constraints and we remedy the error by locally solving Equation 17 while fixing all the binary variables using an NLP solver.

Mixed-Integer Nonlinear Programming

Despite the theoretical advantage of MICP, its practical performance can be unacceptable due to an excessive number of binary variables. For example, if we only use a coarse discretization and relaxation with $K = 7, T = 10, S = 8$, the number of binary variables is already 1064, for which finding the exact global optima is impossible and we have to terminate the optimization early and return users the first few feasible solutions. To make things worse, the returned solution might be useless by not satisfying the exact non-convex constraints.

We observe that it is both impossible and unnecessary to solve the relaxed problem to get the global optima, because even the global optima might not satisfy the exact constraints under a coarse relaxation. Instead, it is worthwhile to return

a solution that is not globally optimal but satisfies the exact, non-convex constraints. Our second algorithm uses an off-the-shelf MINLP solver, such as Byrd et al. (2006), to find a locally optimal solution. Similar to MICP, an MINLP solver is also based on the BB algorithm and constructs a search tree by branching on binary decision variables. For each node, however, we do not assume that the problem is convex and use SQP to find a locally optimal, feasible solution as described in Exler and Schittkowski (2007). The node will be pruned for further expansion if no feasible solution can be found or the objective function is larger than the incumbent. Since SQP does not guarantee to find the global optima or even a feasible solution when one exists, MINLP might prune a node that contains useful solutions. On the other hand, all the returned solutions are guaranteed to satisfy the exact non-convex constraints.

Although MINLP can directly handle Equation 17 without any relaxation, we argue that a relaxation similar to MICP can also be used to force MINLP to try more initial points and increase the chance of finding better solutions. Take the equal length constraints for example, each $\mathbf{d}_{di} \in [-B/2, B/2]^2$ by assumption, so we can evenly divide the domain into S^2 blocks. This can be done by introducing two sets of sample points $\alpha_s = sB/S - B/2$ and $\beta_s = sB/S - B/2$ and the following constraints:

$$
q = 1 : \begin{cases} \mathbf{d}_{di}(t^q) = \left(\sum_{s=0}^{S} \lambda_s^X \alpha_s, \sum_{s=0}^{S} \lambda_s^Y \beta_s \right) \\ \{\lambda_{0,\cdots,S}^X\} \in \mathcal{SOS}_2 \quad \sum_{s=0}^{S} \lambda_s^X = 1 \\ \{\lambda_{0,\cdots,S}^Y\} \in \mathcal{SOS}_2 \quad \sum_{s=0}^{S} \lambda_s^Y = 1, \end{cases}
\tag{21}
$$

where $\lambda_s^{X,Y}$ are auxiliary variables. Adding these constraints will force the MINLP to insert nodes into the search tree corresponding to initializing $\mathbf{d}_{di}$ in different blocks. In practice, we find that it is enough to relax only the equal length constraints and apply the relaxation only to the first timestep $q = 1$. We conclude that MINLP can solve the discretized MIQCQP problem by using $\mathcal{O}(K\lceil \log K \rceil + 4K\lceil \log(S) \rceil)$ binary variables (by adding Equation 21 to Equation 17). This formulation uses much fewer binary variables (e.g., 98 variables when $K = 7, T = 10, S = 8$). Finally, we informally argue that if we apply relaxation to every $1 \leq q \leq T$ and let $T, S \to \infty$, then MINLP will also
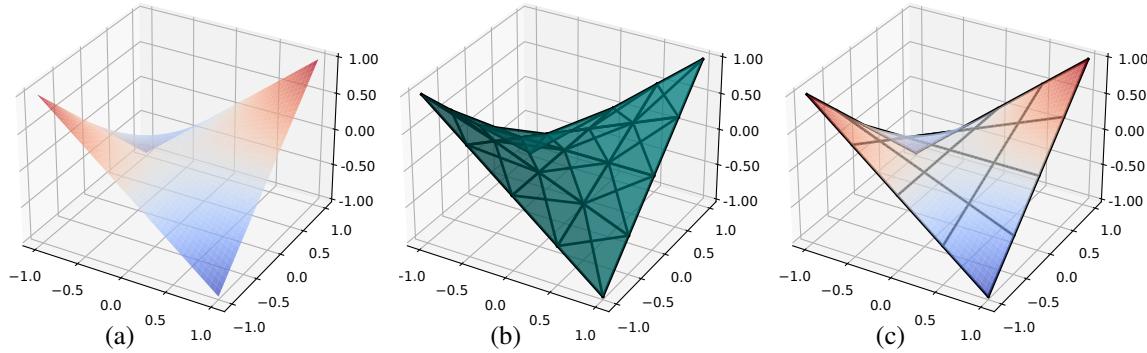
Figure 5. Taking the bilinear constraint $z = xy$ as an example, we compare the three different relaxation techniques. (a): The original constraint of MIQCQP is non-convex; (b): MICP approximates the feasible domain as the union of convex tetrahedra (teal); (c): MIQCQP keeps the non-convex constraint, but we introduce additional relaxations to force the solver to try multiple initial values that lie in each black block.

find the global optima of MIQCQP because this is essentially asking MINLP to initialize from all possible solutions.

Simulated Annealing

We introduce our third algorithm as a baseline for comparison, which is based on the SA framework, similar to Zhu et al. (2012); Cabrera et al. (2002) but adapted to our subclass of planar linkages satisfying Assumption 1. We use the SA algorithm described in Bertsimas et al. (1993) to minimize our discrete objective function (Equation 11). Starting from a trivial initial guess $\mathcal{L}_0$, SA generates a Markov chain by mutating $\mathcal{L}_i$ to $\mathcal{L}_{i+1}$ at the ith iteration and accept $\mathcal{L}_{i+1}$ with a probability proportional to the decrease in the objective function. Our initial guess $\mathcal{L}_0$ involves only one motor node $\mathbf{n}_1$ with $r = B/10, X_C = Y_C = 0$.

We propose a mutation scheme outlined in Algorithm 4 that chooses one of four moves with equal probability, where we use the function Valid($\bullet$) to check whether the linkage structure has valid kinematics (i.e., $\cos \in [0,1]$ for all $t \in [0, 2\pi]$). **TopologicalAdditionMove** adds a new node that can either be a fixed node or a movable node. A movable node $\mathbf{n}_{N+1 \to jk}$ is added by randomly selecting two existing nodes $\mathbf{n}_j$, $\mathbf{n}_k$ with $1 \le j < k \le N$ and then selecting two length parameters $l_{j(N+1)}, l_{k(N+1)} \sim \mathcal{U}(0,B)$ with uniform distribution. A fixed node is uniformly randomly selected within $[-B/2, B/2]^2$. We allow a TopologicalAdditionMove to happen if the total number of nodes is less than K, all the topological constraints are satisfied (if a movable node is added), and the planar linkage has valid kinematics. Our second **TopologicalSubtractionMove** simply removes the last node $\mathbf{n}_N$ from $\mathcal{L}_i$. We allow a TopologicalSubtractionMove to happen if there is at least one motor node remaining and all the topological constraints are satisfied. We check the topological constraints using our MICP solver after each topological move. Our third **GeometricPerturbationMove** would first randomly select a node $\mathbf{n}_{i \to jk}$ and then perturb its position at any time instance by adding a Gaussian noise $\mathcal{N}(B, B^2)$. We allow a GeometricPerturbationMove as long as the linkage structure has valid kinematics. Finally, we introduce a novel **LocalOptimizationMove** that locally minimizes Equation 11 with respect to all the geometric parameters (link lengths, X_C, Y_C, r) by using a gradient-based method.

Algorithm 4 SA-Mutation

Input: $\mathcal{L}_i$
1: Succ←False
2: **while** Not Succ **do**
3:     Choose move type, $\mathcal{L}_{i+1} \leftarrow \mathcal{L}_i$
4:     **if** type=TopologicalAdditionMove **then**
5:         IsFixed$\sim \mathcal{U}(0,1)$
6:         **if** IsFixed> 0.5 **then**
7:             Choose $\mathbf{n}_{N+1} \sim \mathcal{U}(-B/2, B/2)^2$
8:             $N \leftarrow N + 1$, Succ$\leftarrow N \le K$
9:         **else**
10:            Choose $1 \le j < k \le N$
11:            Choose $l_{j(N+1)}, l_{k(N+1)} \sim \mathcal{U}(0, B)$
12:            Add node $\mathbf{n}_{N+1 \to jk}, N \leftarrow N + 1$
13:            Succ$\leftarrow N \le K \wedge$ Satisfied(4,5,6,7) $\wedge$Valid($\mathcal{L}_{i+1}$)
14:     **else if** type=TopologicalSubtractionMove **then**
15:         Remove $\mathbf{n}_N, N \leftarrow N - 1$
16:         Succ$\leftarrow N \ge 1 \wedge$ Satisfied(4,5,6,7)
17:     **else if** type=GeometricPerturbationMove **then**
18:         Choose $1 \le i \le N, t \sim \mathcal{U}(0, 2\pi)$
19:         Choose $\Delta X, \Delta Y \sim \mathcal{N}(B, B^2)$
20:         $\mathbf{n}_i(t) \leftarrow \mathbf{n}_i(t) + (\Delta X, \Delta Y)$
21:         Succ←Valid($\mathcal{L}_{i+1}$)
22:     **else if** type=LocalOptimizationMove **then**
23:         ▷ The last term computed using Algorithm 1
24:         $d \leftarrow \frac{2\pi}{T} \Sigma_{q=1}^T \frac{\partial \|\mathbf{n}_N(t^q) - \mathbf{n}_N^\star(t^q)\|^2}{\partial \mathbf{n}_N(t^q)} \frac{\partial \mathbf{n}_N(t^q)}{\partial (l_{ji}, l_{ki}, X_C, Y_C, r)}$
25:         $\alpha \leftarrow$Armijo-Line-Search(d)
26:         $(l_{ji}, l_{ki}, X_C, Y_C, r) \leftarrow (l_{ji}, l_{ki}, X_C, Y_C, r) + \alpha d$
27: Return $\mathcal{L}_{i+1}$

This is a computationally costly move that involves gradient evaluation using Algorithm 1 and then we choose to only perform a single steepest descend step with an Armijo backtracing line-search to ensure the decrease of objective function. Some of these moves might be unsuccessful, in which case we keep selecting a new type of move until one is successful.

Results

We implement our three algorithms using Python, where we use the Supporting Hyperplane Optimization Toolkit

(SHOT) (Kronqvist et al. 2016) to solve both MICP and MINLP problems, with IPOPT (Wächter 2009) being the low-level NLP solver. Since the low-level NLP problems are non-convex, the solution of MINLP is sensitive to their initial guesses. Specifically, we use IPOPT in two phases, where phase-I ignores the objective function and only tries to satisfy the constraints and phase-II takes the objective functions into account. For each node of the BB search tree, we use the solution of phase-I of the parent node as an initialization. Our SA-baseline is implemented based on the algorithm described in Bertsimas et al. (1993). All the experiments are performed on a desktop computer with a 10-core Intel Xeon(R) W-2155 CPU. We use all the 10 cores to explore multiple nodes of the BB search tree in parallel. In this section, we discuss and compare our three algorithms in terms of computational cost, optimality, additional user constraints, and robot integration.

Computational Cost and Optimality

We designed a graphical user interface that allows a robot designer to sketch a curve. We then close the curve and evenly sample T points on the curve such that we have equal arc lengths between two consecutive points. In Table 2, we illustrate 10 testing target curves and the resulting planar linkages found by MICP, MINLP, and the SA-baseline. To get these results, we set $K = 7$ for all three algorithms. We use $T = 10, S = 8$ for MICP, $T = 20, S = 8$ for MINLP. The SA-baseline requires a cooling function for the temperature $\mathcal{T}$, for which we use:

$$\mathcal{T} = \mathcal{T}_{max} \exp(-\log(\frac{\mathcal{T}_{max}}{\mathcal{T}_{min}})\frac{i}{i_{max}}).$$

We set $\mathcal{T}_{max} = 2.5 \times 10^4$ and $\mathcal{T}_{min} = 2.5$, $i_{max} = 5 \times 10^4$ where i is the iteration number. Due to the limited computational resources and time, we allocate a maximum 24 hours running time for each test and return the best solution. We tune the SA-baseline such that its computational cost is comparable to MICP or MINLP, for which running 5×10^4 iterations takes approximately 1 hour and running 5×10^5 iterations takes roughly 10 hours (one iteration corresponds to one execution of an SA-Mutation Algorithm 4). We set up the parameters for our three algorithms, so that their computational times are roughly comparable. According to Table 2, the SA-baseline cannot generate satisfactory results in terms of matching the target curve. We notice that this is not because of the parameter settings of the SA-baseline. Indeed, the results are still unsatisfactory if we increase the iterations number 10 times, as shown in the rightmost column of Table 2. We further highlight that a larger S could lead to better solutions. As illustrated in Figure 6, we fix the parameters $K = 5, T = 20$, while comparing the objective function values under three choices: $S = 1, 4, 8$. Using a larger S reduce the objective function value for 10 out of 11 examples. In one of the example, the improvement can be as high as 78%.

We further analyze the computational cost and the convergence history of the MICP and MINLP algorithms as a function of K and S, as shown in Figure 7 and Figure 8, respectively. We use both algorithms to solve ten benchmark problems under different parameter settings. As K increases
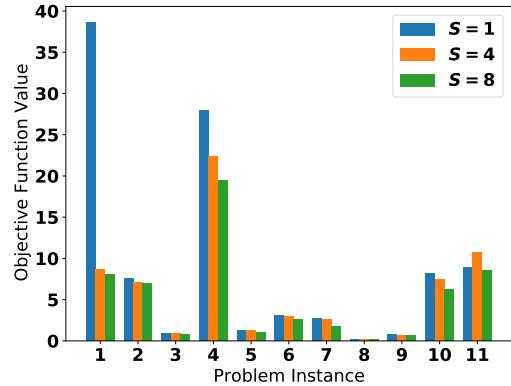

Figure 6. We compared the objective function values with three difference choices of $S = 1, 4, 8$, while using $K = 5$, $T = 20$. Using a larger S improves the solution for 10 out of 11 examples. In most examples, the improvement is minor, but the largest improvement can be as high as 78%.

from 5 to 7 or S increases from 4 to 16, the computational cost increases significantly for MICP. We found that S needs to be at least 8 because otherwise the relaxation is too coarse, rendering the solution of MICP nearly useless as it will not satisfy the non-convex constraints even after a local NLP solve as post-processing. The cost of a typical MICP solve is in the order of tens of hours and in many cases hits the 24 hours limit, especially when we use large S or T. MINLP has relatively faster performance, and typically accomplishes the computation within $5 - 10$ hours. The cost of MICP increases superlinearly with both S and T, while the cost of MINLP increases superlinearly with only S but not T. This is because our number of binary variables in MINLP does not increase with T. On the other hand, MINLP cannot match the target curve well in 2 out of 10 test cases from Table 2, while MICP always achieves an ideal match. We further notice from Figure 8 that MICP explores orders of magnitude more nodes than MINLP in the search tree of the BB algorithm, which is understandable due to a much larger number of binary variables. However, MINLP uses more time to explore each node due to a higher cost in solving a non-convex problem for each node. We observe that both optimizers update the solution less than 10 times before convergence and most of the computations are devoted to detecting and pruning impossible cases. We conclude that MINLP achieves an overall better computational efficacy than MICP at a minor sacrifice of optimality.

Alternative User Interfaces

Our default user interface is for a designer to provide a target end-effector curve. However, other types of objective functions and hard constraints are possible. Our MIQCQP can take any non-convex objective function and constraints, while the MICP solver can take only convex objectives and constraints. In this section, we evaluate several additional user editing operations. We allow users to draw a box and constrain all the nodes (except for the end-effector node) to reside only in the given box, which can be formulated as four additional convex constraints. This is useful for a linkage structure to be mounted on a legged robot, where the non-end-effector nodes should be a certain distance away from the ground to avoid collisions in case of uneven terrains.
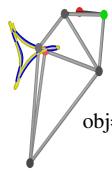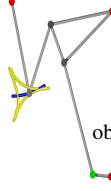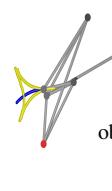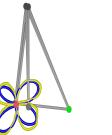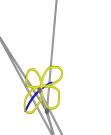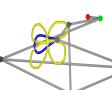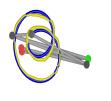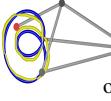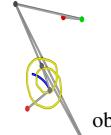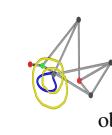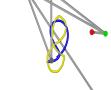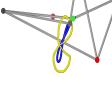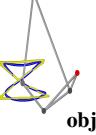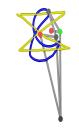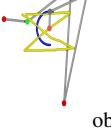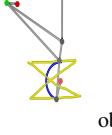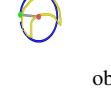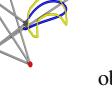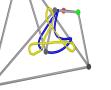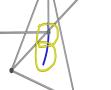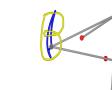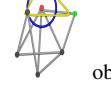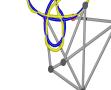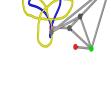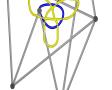
Input Curve	MINLP	MICP	SA-Baseline $i_{max} = 5 \times 10^4$	SA-Baseline $i_{max} = 5 \times 10^5$
	obj=0.62	**obj=0.31**	obj=3.94	obj=3.52
	obj=0.99	**obj=0.96**	obj=3.34	obj=1.78
	obj=0.7	obj=0.77	obj=6.2	obj=7.75
	obj=1.36	**obj=1.36**	obj=4.49	obj=7.85
	obj=0.71	obj=2.58	obj=4.96	obj=5.35
	obj=1.25	obj=1.37	obj=16.15	obj=4.57
	obj=1.39	obj=2.52	obj=2.66	obj=6.8
	obj=0.36	**obj=0.32**	obj=15.88	obj=15.8
	obj=0.52	**obj=0.51**	obj=10.03	obj=11.09
	obj=0.77	obj=1.28	obj=7.9	obj=9.67

Table 2. We show 10 testing curves and the results optimized by our three algorithms and we evaluated two variants of SA-baseline with different iteration numbers. On the left, we show the user's input curves and $T = 20$ sample points are drawn in yellow and green, respectively. For each optimized linkage structure, the fixed and movable nodes are shown in red and gray, respectively. The actual end-effector's curve is drawn in blue, and we mark the optimal objective function value on the lower-right corner.
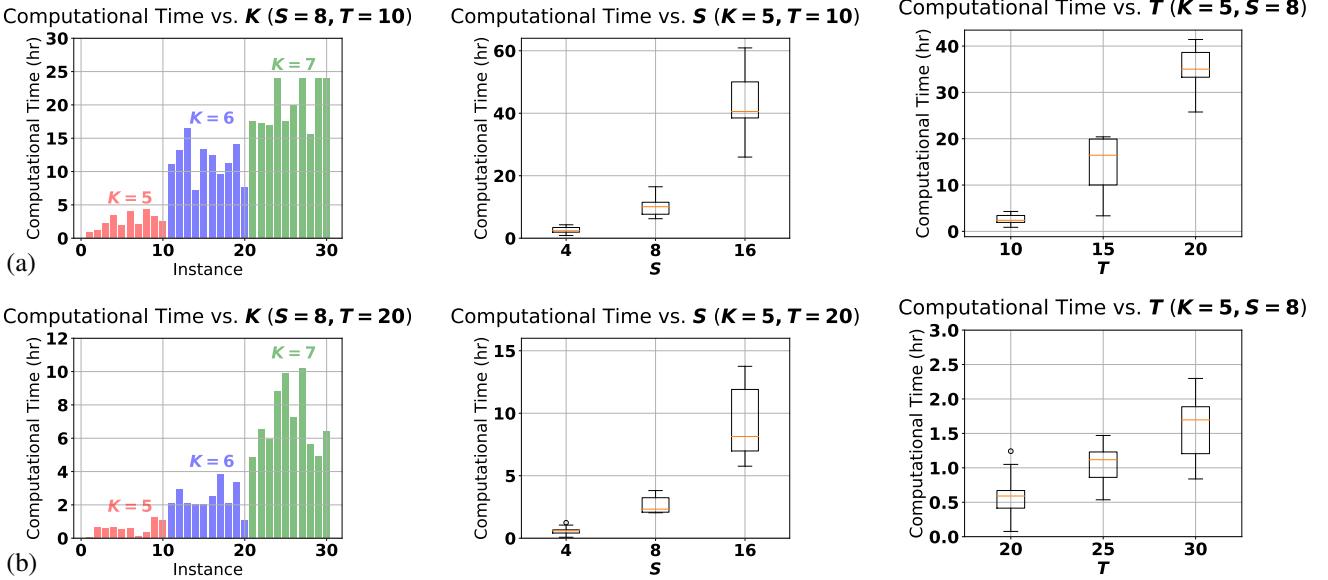
Figure 7. The cost of solving MICP (a) and MINLP (b) for 10 benchmark problems as a function of K, S, and T.
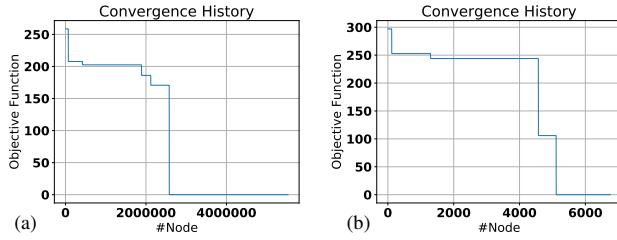

Figure 8. The convergence history of MICP (a) and MINLP (b) for the first problem in Table 2, plotted against the number of nodes explored on the BB search tree.

Some mechanical toys have limited space in the gearbox and such constraints can be employed to fit the structure inside. In Figure 9, we illustrate these two cases for the end-effector to trace out the same elliptical curve.

In addition, we allow users to draw a curve and optimize a linkage structure whose end-effector passes through the sampled points on the target curve with an arbitrary order, which is a typical case of coverage planning. For example, the order is unimportant for a planar linkage to hold a pen and fill out an area on a piece of paper. This requirement can be achieved by removing the first two equations in Equation 16, leaving only: $\mathbf{d}_{11}^d(t^q) = \mathbf{n}_1^d(t^q) - (X_C, Y_C)$. We highlight such an example in Figure 10, where the user provides an 8-shaped target curve. By default, the end-effector traces out a genus-2 curve, but it can also trace out a genus-1 curve to visit all the sampled points on the curve when the
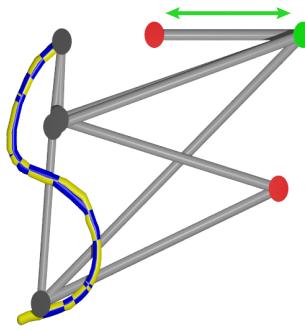

Figure 11. We illustrate a linkage structure that transforms linear motions into an S-shaped curve. The green node moves back and forth linearly along the green arrow.

order is arbitrary. Finally, our formulation is not limited to rotational motors. In Figure 11, we illustrate a case with a linear motor, where the motor is moving according to: $\mathbf{n}_1(t) = (X_C, Y_C) + t(X_C', Y_C')$. Such motion can be realized by replacing Equation 16 with: $\mathbf{n}_1(t^q) = (X_C, Y_C) + t^q(X_C', Y_C')$, where X_C, Y_C, X_C', Y_C' are additional decision variables for the starting position and the moving direction. We speculate that several other motor types can be also realized by using similar techniques. Incorporating various motor types allows our formulation to be used in the modular design of mechanical systems, where the motion of the motor node is realized by another module.

Robot Walking

Prior works (Liao et al. 2019; Spielberg et al. 2017) have demonstrated that it is possible to design walking robots where linkage structures are used to transform rotational motion into loops of footsteps. Their end-effectors trace out an oval-shaped target curve, of which a well-known design is shown in Figure 1, Figure 2, and analyzed in Nansai et al. (2013). Although the above-mentioned work uses a manually designed linkage topology and geometry, they rely on an additional fine-tuning optimization to adjust the linkage's mounting points on the robot and geometric parameters. They show that such fine-tuning is essential to maximize the robot's performance, such as its walking speed.

We explore the potential application of our optimized planar linkages in legged walking robotics, following a two-step semi-optimization approach similar to prior works (Soong and Yan 2007; Erkaya and Uzmay 2009; Coros et al. 2013; Thomaszewski et al. 2014). Specifically, we first optimize a variety of different linkage geometries and topologies. We then manually choose one of these linkages as robot legs. Finally, we optimize the robot performance in an end-to-end manner. To this end, we establish a testbed as illustrated in Figure 9 (d) where we mount a set of eight linkage structures shown in Figure 9 (c) onto a robot with a rectangular torso and two rotary motors, where four linkages are used as front legs and the other four as back legs. We
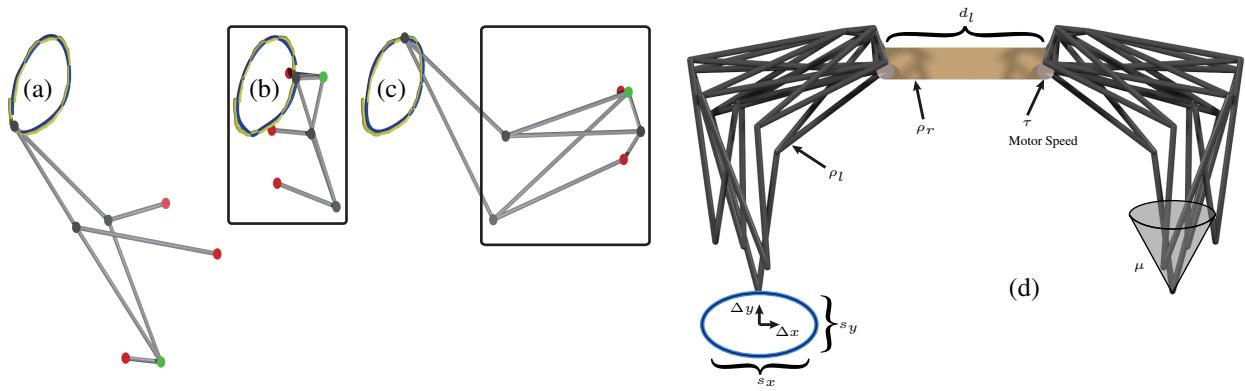
Figure 9. We illustrate the effect of additional constraints on node positions. (a): no constraint; (b): box constraint that limits the entire structure in a small vicinity around the target curve; (c): box constraint that ensures that all the nodes (except for the end-effector) are a certain distance away from the target curve. (d): We mount the linkage structure (c) on a walking robot and optimize its walking distance with respect to the parameters: target curve translation $\Delta x, \Delta y(m)$, target curve scale $s_x, s_y(m)$, frictional coefficient μ, robot mass density $\rho_r(kg/m^3)$, leg mass density $\rho_l(kg/m^3)$, motor torque $\tau(kgm^2/s^2)$, motor speed(m/s), and body length $d_l(m)$.
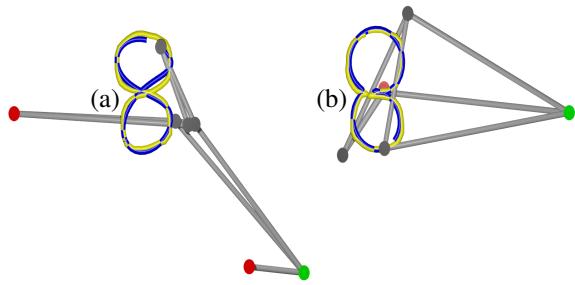

Figure 10. We illustrate the effect of allowing an arbitrary order to visit the set of points on the target 8-shaped curve. (a): the curve is genus-2 using fixed order; (b): the curve is genus-1 using arbitrary order.

simulate the robot walking on a flat terrain using the Bullet Physics Engine (Coumans et al. 2013). The robot motion is generated by creating a single rotary actuator on which a constant torque is applied, i.e. we assume the robot is not equipped with any sensor or controller. This is the case with many low-cost robots and mechanical toys. There are several additional parameters to set up the robot simulator: the separation distance between the front and back legs d_l, the robot-to-ground frictional coefficient μ, the motor torque τ, the motor speed, the robot's mass density ρ_r, and the leg's mass density ρ_l. Since our formulation only considers the end-effector's curve and does not care about the robot's performance, we speculate that some fine-tuning is needed. We perform the fine-tuning by using Bayesian optimization (Eggensperger et al. 2013), where our objective function is the distance traveled by the robot's center of mass over a simulated period of 10 seconds. We first investigate which parameters must be fine tuned, so we run six passes of fine-tuning for each parameter; the results are summarized in Figure 12 (a), where we observe the most significant performance increase by tuning τ, the motor speed, and ρ_l. Next, we run another pass of fine-tuning jointly in these three parameters and observe a $4.3\times$ overall performance boost as shown in Figure 12 (b) (with optimal values $\tau = 7247(kgm^2/s^2)$, motor speed$= 3.22(m/s)$, and $\rho_l = 4.8(kg/m^3)$). Finally, we plot the landscape of the objective

function that is approximated by using the Gaussian process in Figure 13, which is the output of Bayesian optimization after 200 iterations. We can see that high objective function values only occupy a small fraction of the domain, so we conclude that fine-tuning the robot-mounting parameters is a necessity for linkage structures to gain high performance on robots. For example, we found that the minimal torque to drive the robot is $4500(kgm^2/s^2)$ as shown in Figure 14.

The above fine-tuning modifies the mounting of linkage structure on the robot, while the geometry of the structure is fixed. We perform a separate fine-tuning that focuses on the geometry. There are too many parameters that specify the geometry and Bayesian optimization does not scale well to such high-dimensional decision spaces. To tackle this issue, we only modify the target curve and use local optimization to change the shape of the linkage structure. Specifically, we introduce four parameters: $(\Delta x, \Delta y) \in [-3,3]^2$ and $(s_x, s_y) \in [1,3]^2$ specify the translation and scaling of the target curve, respectively. Given $(\Delta x, \Delta y)$ and (s_x, s_y), we use the L-BFGS-B algorithm to minimize the following local objective function where gradients are calculated using Algorithm 1:

$$\int_0^{2\pi} \left\| \mathbf{n}_N(t) - \begin{pmatrix} s_x & \\ & s_y \end{pmatrix} \mathbf{n}_N^*(t) - \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right\|^2 dt,$$

where $\mathbf{n}_N^*(t)$ is the target curve of Figure 9. We fix the position of all fixed nodes, the center of motor (X_C, Y_C), and the radius of motor r, so that the linkage can be mounted on the robot in the same way. After local optimization, we mount the linkage structure on the robot and compute its walking distance over 10 seconds of simulation. We optimize the walking distance with respect to the four parameters $(\Delta x, \Delta y)$ and (s_x, s_y) by using Bayesian optimization. These four parameters are default to $\Delta x = \Delta y = 0$ and $s_x = s_y = 1$. We found that $\Delta x, s_x, s_y$ are the three most influential parameters. The walking distance as a function of these parameters and the optimized robot design are shown in Figure 15. We observe that there are many designs leading to high walking distances. Therefore, the robot performance is not sensitive to the linkage geometry.
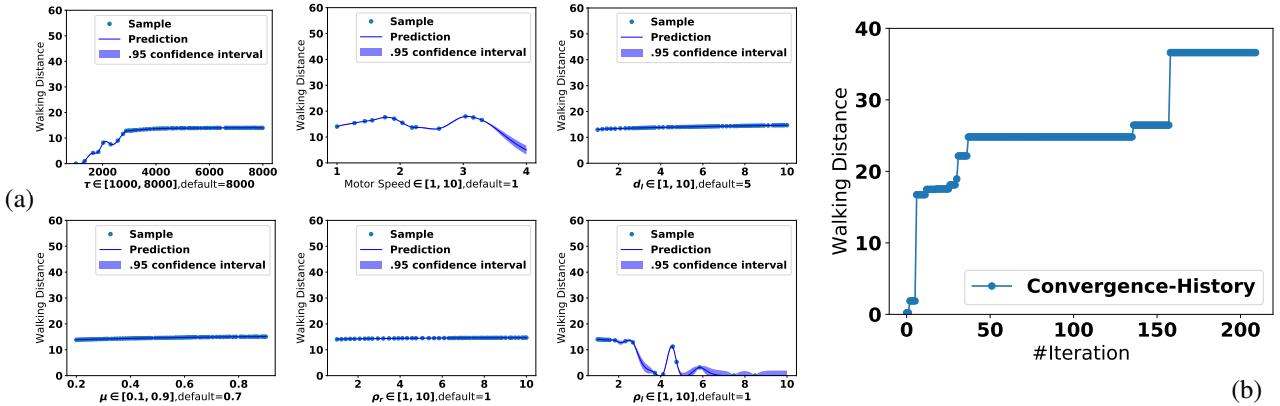
Figure 12. (a): We optimize the robot's walking distance (m) with respect to six parameters separately and we plot the objective landscape approximated by the Gaussian process. (b): We then optimize the three most important parameters: τ, motor speed, and ρ_l and plot the convergence history.

Conclusions and Discussions

We have proposed a deterministic algorithmic framework for optimizing a large subset of planar linkages, such that the end-effector traces out a curve that matches the user-specified target curve. We show that, by modeling the linkage structure by using maximal instead of minimal coordinates, the joint optimization of both the topology and geometry can be reformulated as an infinite-dimensional, non-convex MIQCQP. We further proposed a discretization scheme and three algorithms to solve MIQCQP approximately. Our first algorithm relaxes non-convex constraints as a disjoint convex set, allowing an MICP solver to find the global optima of the relaxed problem. Our second algorithm uses MINLP to find local feasible solutions via SQP. We highlight that, compared to the SA-baseline, our deterministic algorithm achieves $9.3\times$ higher optimality in a row of benchmarks (measured by the average ratio of objective function values in Table 2) and can take various additional constraints. These promising results can induce several avenues of future work.

Our work can find small linkage structures with up to 7 nodes. Even at such a small scale, solving MIQCQP is still computationally intensive, taking tens of hours on a desktop machine. There are, however, several ways to further accelerate the algorithm. First, with the availability of multi-core processors, the BB algorithm can be parallelized by exploring multiple nodes simultaneously. Second, the efficacy of the BB algorithm is closely related to the heuristic rules for expanding the search tree, generating cutting planes, and warm-starting the node solutions, for which dedicated heuristic rules can be designed for our problem. Finally, the relaxation scheme of non-convex constraints can be optimized to reduce the approximation error as described in Nagarajan et al. (2019). For example, the sample positions in Figure 3 can be adaptively selected. This method can also minimize the number of samples (S) and reduce the number of binary variables.

Our current implementation does not allow users to specify the timing for the end-effector to reach each sample point on the target curve. Currently, we support two default timing schemes: 1) even sampling the target curve and assuming equal travel time between consecutive samples; 2) arbitrary travel time and order for all the samples, as illustrated in Figure 10. This is due to two reasons. First,

it is difficult and non-intuitive for users to specify the exact timing via a GUI interface. Second, at such a small scale with up to 7 nodes, we have not observed significantly different designs using different timing schemes and cases such as Figure 10 are rare. We expect a larger solution space would lead to a variety of designs corresponding to more deliberate timing specification.

Our method can only approximate the solutions of MIQCQP and sometimes can fail at finding a feasible solution. Indeed, we only impose non-singular constraints (Equation 9) at discrete time instances to avoid infinite constraints and Lemma 1 is not guaranteed to hold as a result. Our MICP solver further approximates the non-convex constraints as piecewise convex ones. By comparison, the SA-baseline is guaranteed to return a solution, although it can drift far from the user's requirement. It is worthwhile to explore an approximation scheme for relaxing MIQCQP. One promising direction is to consider the semidefinite lower-bound that turns a quadratic constraint into a linear matrix inequality (Vandenberghe and Boyd 1996). More generally, the sum-of-squares programming allows any polynomial optimization to be converted into a semidefinite programming problem (Laurent 2009), and the conversion is exact under certain conditions. Such conversions can be used to derive the lower-bound in BB algorithms. Recent work (Pan et al. 2020) has applied this idea to the inverse kinematic problems of sequential manipulators. The main advantage of sum-of-squares programming is that the lower bound can be made arbitrarily tight.

Finally, a major limitation of our method is that we only optimize the kinematics and geometric features of the linkage structure, which is not sufficient for many robotic applications, especially when deployed onto a physical robot platform. In our robot walking results for example, optimizing the dynamics properties, e.g., joint torques, frictional coefficients, mass distributions, is key to the overall final performance. Our current experiments optimize these dynamics properties using Bayesian exploration as a separate post-process, assuming fixed geometry and topology. This is a standard approach used by several prior works to automatically tune the dynamics properties. For example, Bai et al. (2018) proposed an optimization method to reduce the vibration. Feng et al. (2002) optimized

(a): Walking Distance

Figure 14. We fix the two variables at their optimal values (motor speed= $3.22(m/s)$, and $\rho_l = 4.8(kg/m^3)$) and plot the robot's performance against the motor torque. As the torque changes between $[4500, 7500](kgm^2/s^2)$, the performance only changes by 20% of the optimal value.

Figure 13. (a): After 200 iterations of Bayesian optimization, we plot the approximate landscape of walking distance as a function of the three parameters: τ, motor speed, and ρ_l using volume rendering (red means higher objective function values). We also show the range and default value of each parameter. When a parameter is not optimized, we use its default value. (b): The best robot design in perspective and orthogonal view.

the mass distribution to reduce the needed joint force. Truss optimization (Sokół 2011) typically maximizes the strength of a linkage structure under external forces. Joint formulations such as Soong and Yan (2007) have also been proposed that simultaneously minimize the motor torques and maximize the structure strength. Erkaya and Uzmay (2009) used simulated annealing to adjust multiple dynamics parameters under the influence of joint clearance. However, we expect that higher performance can be achieved by considering kinematics and dynamics into a single, joint optimization formulation. In many applications, the dynamic properties, e.g., material densities and motor torques, are pre-determined by hardware specifications, and the optimized linkage structures should satisfy these specifications as hard constraints that may also include collision handling Govindaraju et al. (2005); Kim et al. (2002). Unfortunately, formulating these


Figure 15. (a): After 200 iterations of Bayesian optimization, we plot the approximate landscape of walking distance as a function of the three parameters: Δx, s_x, and s_y using volume rendering (red means higher objective function values). (b): The best robot design in perspective and orthogonal view.

considerations would significantly increase the complexity and computational time, so we leave them as future work.

Acknowledgement

References

Bächer M, Coros S and Thomaszewski B (2015) Linkedit: interactive linkage editing using symbolic kinematics. ACM Transactions on Graphics (TOG) 34(4): 99.

Bai ZF, Jiang X, Li F, Zhao JJ and Zhao Y (2018) Reducing undesirable vibrations of planar linkage mechanism with joint clearance. Journal of Mechanical Science and Technology 32(2): 559–565.

Bertsimas D, Tsitsiklis J et al. (1993) Simulated annealing. Statistical science 8(1): 10–15.

Bertsimas D and Tsitsiklis JN (1997) Introduction to linear optimization, volume 6. Athena Scientific Belmont, MA.

Bommes D, Zimmer H and Kobbelt L (2009) Mixed-integer quadrangulation. ACM Transactions On Graphics (TOG) 28(3): 77.

Byrd RH, Nocedal J and Waltz RA (2006) Knitro: An integrated package for nonlinear optimization. In: Large-scale nonlinear optimization. Springer, pp. 35–59.

Cabrera J, Simon A and Prado M (2002) Optimal synthesis of mechanisms with genetic algorithms. Mechanism and machine theory 37(10): 1165–1177.

Conforti M, Di Summa M, Eisenbrand F and Wolsey LA (2009) Network formulations of mixed-integer programs. Mathematics of Operations Research 34(1): 194–209.

Coros S, Thomaszewski B, Noris G, Sueda S, Forberg M, Sumner RW, Matusik W and Bickel B (2013) Computational design of mechanical characters. ACM Transactions on Graphics (TOG) 32(4): 1–12.

Coumans E et al. (2013) Bullet physics library. Open source: bulletphysics. org 15(49): 5.

Dai H, Izatt G and Tedrake R (2017) Global inverse kinematics via mixed-integer convex optimization. In: International Symposium on Robotics Research, Puerto Varas, Chile. pp. 1–16.

Deits R and Tedrake R (2014) Footstep planning on uneven terrain with mixed-integer convex optimization. In: 2014 IEEE-RAS international conference on humanoid robots. IEEE, pp. 279–286.

Ding H, Reißig G, Groß D and Stursberg O (2011) Mixed-integer programming for optimal path planning of robotic manipulators. In: 2011 IEEE International Conference on Automation Science and Engineering. IEEE, pp. 133–138.

Eggensperger K, Feurer M, Hutter F, Bergstra J, Snoek J, Hoos H and Leyton-Brown K (2013) Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS workshop on Bayesian Optimization in Theory and Practice, volume 10. p. 3.

Erkaya S and Uzmay I (2009) Determining link parameters using genetic algorithm in mechanisms with joint clearance. Mechanism and Machine Theory 44(1): 222–234.

Exler O and Schittkowski K (2007) A trust region sqp algorithm for mixed-integer nonlinear programming. Optimization Letters 1(3): 269–280.

Featherstone R (2014) Rigid body dynamics algorithms. Springer.

Feng B, Morita N and Torii T (2002) A new optimization method for dynamic design of planar linkage with clearances at joints—optimizing the mass distribution of links to reduce the change of joint forces. J. Mech. Des. 124(1): 68–73.

Gallet M, Koutschan C, Li Z, Regensburger G, Schicho J and Villamizar N (2017) Planar linkages following a prescribed motion. Mathematics of Computation 86(303): 473–506.

Gosselin C and Angeles J (1990) Singularity analysis of closed-loop kinematic chains. IEEE Transactions on Robotics and Automation 6(3): 281–290. DOI:10.1109/70.56660.

Govindaraju NK, Lin MC and Manocha D (2005) Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware. In: IEEE Proceedings. VR 2005. Virtual Reality, 2005. IEEE, pp. 59–66.

Gurobi Optimization L (2018) Gurobi optimizer reference manual.

Ha S, Coros S, Alspach A, Bern JM, Kim J and Yamane K (2018) Computational design of robotic devices from high-level motion specifications. IEEE Transactions on Robotics 34(5): 1240–1251. DOI:10.1109/TRO.2018.2830419.

Ha S, Coros S, Alspach A, Kim J and Yamane K (2018) Computational co-optimization of design parameters and motion trajectories for robotic systems. The International Journal of Robotics Research 37(13-14): 1521–1536.

Hayakawa T, Kamimura T, Kaji S and Matsuno F (2020) Autonomous distributed system for gait generation for single-legged modular robots connected in various configurations. IEEE Transactions on Robotics 36(5): 1491–1510. DOI:10.1109/TRO.2020.2992983.

HEBI (2019) Hebi robotics. URL http://www.hebirobotics.com/.

Hernández A, Gómez C, Crespo J and Barber R (2016) A home made robotic platform based on theo jansen mechanism for teaching robotics. In: INTED2016 Proceedings, 10th International Technology, Education and Development Conference. IATED. ISBN 978-84-608-5617-7, pp. 6689–6698.

Hu Y, Liu J, Spielberg A, Tenenbaum JB, Freeman WT, Wu J, Rus D and Matusik W (2019) Chainqueen: A real-time differentiable physical simulator for soft robotics. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 6265–6271. DOI:10.1109/ICRA.2019.8794333.

Kanno Y (2013) Topology optimization of tensegrity structures under compliance constraint: a mixed integer linear programming approach. Optimization and Engineering 14(1): 61–96.

Kecskemethy A, Krupp T and Hiller M (1997) Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. Multibody System Dynamics 1(1): 23–45.

Kempe AB (1875) On a general method of describing plane curves of the nth degree by linkwork. Proceedings of the London Mathematical Society 1(1): 213–216.

Kim YJ, Otaduy MA, Lin MC and Manocha D (2002) Fast penetration depth computation for physically-based animation. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation. pp. 23–31.

Kronqvist J, Lundell A and Westerlund T (2016) The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization 64(2): 249–272.

Laurent M (2009) Sums of squares, moment matrices and optimization over polynomials. In: Emerging applications of algebraic geometry. Springer, pp. 157–270.

Lawler EL and Wood DE (1966) Branch-and-bound methods: A survey. Operations research 14(4): 699–719.

Liao T, Wang G, Yang B, Lee R, Pister K, Levine S and Calandra R (2019) Data-efficient learning of morphology and controller for a microrobot. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 2488–2494. DOI:10.1109/ICRA.2019.8793802.

Liberti L (2004) Reformulation and convex relaxation techniques for global optimization. PhD Thesis, Springer.

Liu J and Ma Y (2016) A survey of manufacturing oriented topology optimization methods. Advances in Engineering Software 100: 161 – 175.

Lobato E, Echavarren F, Rouco L, Navarrete M, Casanova R and Lopez G (2003) A mixed-integer lp based network topology optimization algorithm for overload alleviation. In: 2003 IEEE Bologna Power Tech Conference Proceedings,, volume 2. IEEE, pp. 5–pp.

Megaro V, Zehnder J, Bächer M, Coros S, Gross MH and Thomaszewski B (2017) A computational design tool for compliant mechanisms. ACM Trans. Graph. 36(4): 82–1.

Murata S, Yoshida E, Kamimura A, Kurokawa H, Tomita K and Kokaji S (2002) M-tran: self-reconfigurable modular robotic system. IEEE/ASME Transactions on Mechatronics 7(4): 431–441. DOI:10.1109/TMECH.2002.806220.

Nagarajan H, Lu M, Wang S, Bent R and Sundar K (2019) An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. Journal of Global Optimization 74(4): 639–675.

Nansai S, Elara MR and Iwase M (2013) Dynamic analysis and modeling of jansen mechanism. Procedia Engineering 64: 1562–1571.

Pan Z, He L and Gao X (2020) Inner-approximation of manipulable and reachable regions using bilinear matrix inequalities. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6585–6592. DOI:10.1109/IROS45743.2020.9341017.

Pan Z, Liu M, Gao X, Xu K and Manocha D (2019) Globally optimal joint search of topology and trajectory for planar linkages. The International Symposium on Robotics Research (ISRR) .

Saar KA, Giardina F and Iida F (2018) Model-free design optimization of a hopping robot and its comparison with a human designer. IEEE Robotics and Automation Letters 3(2): 1245–1251.

Sokół T (2011) A 99 line code for discretized michell truss optimization written in mathematica. Structural and Multidisciplinary Optimization 43(2): 181–190.

Soong RC and Yan HS (2007) Simultaneous minimization of shaking moment, driving torque, and bearing reactions of complete force balanced linkages. J. Chin. Soc. Mech. Eng 28(3): 243–254.

Spielberg A, Araki B, Sung C, Tedrake R and Rus D (2017) Functional co-optimization of articulated robots. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 5035–5042.

Thomaszewski B, Coros S, Gauge D, Megaro V, Grinspun E and Gross M (2014) Computational design of linkage-based characters. ACM Trans. Graph. 33(4): 64:1–64:9.

Umetani N, Koyama Y, Schmidt R and Igarashi T (2014) Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. ACM Transactions on Graphics (TOG) 33(4): 1–10.

Vandenberghe L and Boyd S (1996) Semidefinite programming. SIAM review 38(1): 49–95.

Vielma JP and Nemhauser GL (2011) Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming 128(1-2): 49–72.

Wächter A (2009) Short tutorial: Getting started with ipopt in 90 minutes. In: Naumann U, Schenk O, Simon HD and Toledo S (eds.) Combinatorial Scientific Computing, number 09061 in Dagstuhl Seminar Proceedings. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. URL http://drops.dagstuhl.de/opus/volltexte/2009/2089.

Whitman J, Bhirangi R, Travers M and Choset H (2020) Modular robot design synthesis with deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34. pp. 10418–10425.

Zhu L, Xu W, Snyder J, Liu Y, Wang G and Guo B (2012) Motion-guided mechanical toy modeling. ACM Trans. Graph. 31(6): 127:1–127:10.