Task-Based Design of Ad-hoc Modular Manipulators

Thais Campos[1*], Jeevana Priya Inala[2*], Armando Solar-Lezama[2], and Hadas Kress-Gazit[1]

***Abstract*— The great promise of modular robots is the ability to create on demand robots; however, choosing the "right" design based on a task is still a challenging problem. In this paper, we present an approach to automatically synthesize both the design and control for modular robots from a task description. In particular, we focus on manipulators composed of one degree-of-freedom (DoF) modules. Our approach is able to handle partially infeasible tasks by either identifying the infeasible part and finding a design that satisfies the feasible part or searching for multiple designs that together satisfy the entire task. We compare our approach to a baseline genetic algorithm in a series of increasingly complex environments.**

I. INTRODUCTION

Modular robots are collections of modules, each capable of sensing and actuation, that are arranged and connected in different configurations, depending on the task they are performing [1], [2]. These tasks can vary from locomotion [2], [3] to manipulation [2], to complex high-level tasks as in [4], [5]. While such robots can be tailored to the task at hand, manually coming up with a configuration to do a specific task is a tedious process due to the large design space that contains both discrete and continuous parameters, and it involves reasoning about complex kinematic equations. This gives rise to the challenge of automatically synthesizing modular robot configurations and controls based on specific tasks. In this paper, we focus on solving the design problem for a particular kind of modular robots – modular manipulators which are composed of one degree of freedom (DoF) modules, links and mounting brackets (see Fig. 1).

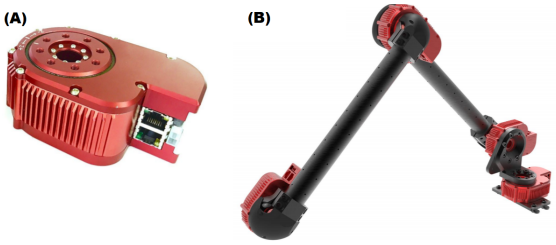


Fig. 1: Modular manipulator. (A) One DoF X-Series Actuator (Hebi Robotics); (B) Four DoFs robotic arm built using X-Series Actuators, links and mounting brackets [6].

*Thais Campos and Jeevana Priya Inala contributed equally to this work.

[1]Thais Campos and Hadas Kress-Gazit are with the Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca NY 14850, USA {tcd58, hadaskg}@cornell.edu

[2]Jeevana Priya Inala and Armando Solar-Lezama are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge MA 02139, USA {jinala, asolar}@csail.mit.edu

This work was supported by ONR N00014-17-1-2699 and NSF CNS-1837506

The synthesis problem is: given a task, here a set of points in 3D and obstacles in the environment, find a design such that the manipulator can trace a trajectory that includes all the task points without colliding with the obstacles. Our approach divides the problem into two parts—*reachability synthesis* and *trajectory verification*. In the reachability synthesis step, we solve a set of constraints on the design so that the manipulator can reach the task points individually. Then, after a design is found, the trajectory verification step searches for a trajectory connecting the task points to ensure that the task is feasible by the design found (i.e. the manipulator can move between the individual points). This is done using RRT* [7], a sampling-based motion planner.

For solving the constraints on the design, recent approaches include a variety of optimization techniques, such as Covariance Matrix Adaptation [8], DIRECT Algorithm [9] and Simulated Annealing and its variations [10], [11]. Genetic Algorithms (GA) and its variations are the most used technique [12]–[17]. GA is a heuristic optimization technique that can handle the mixed discrete and continuous search space. However, approaches based on GA have difficulties finding solutions when the search space is highly constrained (e.g. a large number of obstacles).

In this paper, we use the Sketch synthesis system [18], [19] to solve the constraints on the design. Sketch allows users to specify a synthesis problem as a partial program with unknown parameters and a set of assertions to specify the constraints. For problems involving discrete and continuous variables, Sketch uses a combination of numerical optimization and Boolean reasoning, a variant of the algorithm in [20]. A key feature of Sketch is that it works with a symbolic representation of the search space and the constraints, which allows it to search in a very directed manner compared to GA that only sees the fitness of the individual candidates it tries. The Sketch algorithm is sound i.e. if a solution is found, it is guaranteed to be correct—and is efficient in practice. However, the algorithm is not complete, so there might be scenarios for which a solution exists but the algorithm cannot find one in the allotted time.

The main challenge in using Sketch for design synthesis is to formulate the problem in such a way that the synthesis problem becomes tractable. The synthesis problem involves both searching over the design space and solving the inverse kinematics (IK) problem to assert that the task can be done by the manipulator. We found that a naïve formulation of the IK problem, based on using the forward kinematics equations, introduces numerous local minima for numerical optimization. In this paper, we present a new formulation to solve the IK problem that includes constraints to avoid

collisions incorporated directly into the optimization.

Unique to our approach is the ability to synthesize design(s) for partially infeasible tasks. If the entire task cannot be solved by a single design configuration, our approach can either separate the task into feasible and infeasible portions for one best configuration or find multiple configurations that together are capable of achieving the whole task, if they exist.

In summary, the paper's contributions are:

- A framework for synthesizing provably-correct design and control for modular manipulators.
- An approach that handles partially infeasible tasks.
- Comparison to an approach based on genetic algorithms.

II. Definitions

We introduce definitions regarding modular manipulators based on the Denavit-Hartenberg (DH) convention [21]. For notation, we use the . operator to refer to named parameters of an item; and, [.] with an index to refer to a specific item of an enumerated variable. We use $\{a..b\}$ to denote discrete intervals and $[a, b]$ for continuous intervals.

Definition 1 (Module): A module m is a single atomic unit of a manipulator which is composed of one rotational DoF (actuator) and one link (as shown in Fig. 2). It is defined as the tuple $m_j = \langle r_j, \alpha_j, d_j \rangle$ and has a local coordinate frame $(\overrightarrow{x_j}, \overrightarrow{y_j}, \overrightarrow{z_j})$ associated to it. The origin of the local coordinate frame is at the base of the module. The z-axis is the axis of rotation of the actuator, the x-axis is chosen parallel to the previous link according to the DH convention, and the y-axis follows the right hand rule. The symbol r_j is the link length, α_j is the angle between $\overrightarrow{z_{j-1}}$ and $\overrightarrow{z_j}$, and d_j is the offset of the actuator along $\overrightarrow{z_j}$. We assume $\alpha_1 = 0$ i.e. the axis of the rotation of the first actuator coincides with the z-axis of the global coordinate frame. Here, r_j is a continuous value in $[0, r_{\max}]$ for a chosen maximum length $r_{\max}$. Similarly, α_j is a continuous value in $[0, \pi]$. The offsets d_j are fixed and depend on the actuator's thickness, for example, for the X-series actuators they are 5.5 cm.

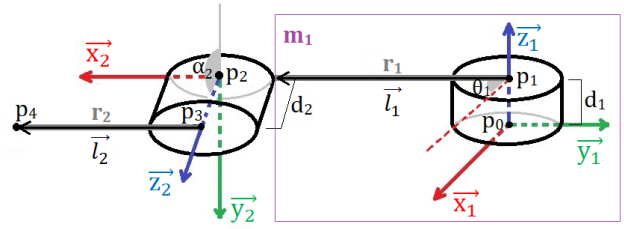


Fig. 2: The structure of a 2 DoF manipulator. The purple box represents a single module which contains an actuator (cylinder) and a link $\overrightarrow{l_1}$. Each actuator has a coordinate frame attached to it $(\overrightarrow{x_j}, \overrightarrow{y_j}, \overrightarrow{z_j})$.

Definition 2 (Configuration): A configuration C is defined as a list of n_{mod} modules i.e. $C = \{m_1, m_2, \cdots, m_{n_{\text{mod}}}\}$ such that the end of module m_j is connected to the base of module m_{j+1}. Moreover, the end of $m_{n_{\text{mod}}}$ is the end-effector and m_1 is fixed at the origin of the global frame.

Definition 3 (Joint angle (control)): A joint angle θ_j of a module is the angle between the module's link and the x-axis $\overrightarrow{x_j}$ about $\overrightarrow{z_j}$ (see Fig. 2). We use Θ to denote the joint angles of all the modules of the manipulator, $\Theta = \{\theta_1, \theta_2, \cdots, \theta_{n_{\text{mod}}}\}$.

Definition 4 (State): For a configuration C and a control Θ, the state of the manipulator, denoted as $P(C, \Theta) = \{p_0, p_1, \cdots, p_{2n_{\text{mod}}}\} \in \mathbb{R}^{(2n_{\text{mod}}+1)\times 3}$, is the global 3D Cartesian coordinates of the $2n_{\text{mod}} + 1$ points that define the end points of all the links in C, in addition to the origin, as shown in Fig. 2.

We overload the symbol P to also refer to a variable for a state. We define $P.\overrightarrow{v_h} = \overrightarrow{p_{h-1}p_h}$ for $h \in \{1..2n_{\text{mod}}\}$ to denote the line segments between every pair of neighboring points in P. The link segment of module j is defined as $P.\overrightarrow{l_j} = P.\overrightarrow{v_{2j}} = \overrightarrow{p_{2j-1}p_{2j}}$. The position of the end-effector is given by $P.\text{end} = p_{2n_{\text{mod}}}$. The z-axis of the module-fixed frame is $P.\overrightarrow{z_j} = P.\overrightarrow{v_{2j-1}} = \overrightarrow{p_{2j-2}p_{2j-1}}$.

A state P' is "valid" for a manipulator with configuration C iff there exists a control Θ such that $P(C, \Theta) = P'$.

Definition 5 (Task): A task T is defined as a set of n_{task} points in the global reference frame.

Definition 6 (Obstacles): The environment for the manipulator is characterized by a set of n_{obs} obstacles, O. An obstacle $o \in O$ is modeled as a sphere described by its center's position in the global reference frame $\langle x_c, y_c, z_c \rangle$ and its radius r_c.

Definition 7 (Collision-Free Workspace): Given a set of obstacles O and a configuration C, the collision-free workspace of the manipulator, denoted as W(C, O), includes all states that the manipulator can achieve without colliding with the obstacles or the manipulator itself. $W(C, O) = \{P \mid P \text{ is a valid state of the manipulator } \wedge \text{NoCollision}(P)\}$. The constraints for NoCollision are formally defined in Equations 2 and 3 in the next section.

Definition 8 (State Space Trajectory): A trajectory, Γ, in the state space is a connected set of states of the manipulator.

We use $\text{End}(S)$ to refer to the global 3D Cartesian coordinates of the end-effector for all states in the set S.

III. Approach

A. Problem Formulation and Overview

In this paper, we address the challenge of finding a single design that satisfies a task (Problem 1), but if no solution is found, we provide partial solutions (Problems 2 and 3):

Problem 1 (Single-Design Synthesis): Given a task T and obstacles O, the goal is to find a configuration C such that there is a trajectory Γ in the collision-free workspace that can trace all points in T. Formally,

$$\text{Find } C, \Gamma \text{ s.t. } \Gamma \subseteq W(C, O) \ \wedge \ T \subseteq \text{End}(\Gamma)$$

We break the synthesis problem of finding both the configuration and the trajectory into two parts. First, we synthesize a configuration C that can reach all the task points individually i.e.

$$\text{Find } C \text{ s.t. } T \subseteq \text{End}(W(C, O)) \qquad \text{(Reachability)}$$

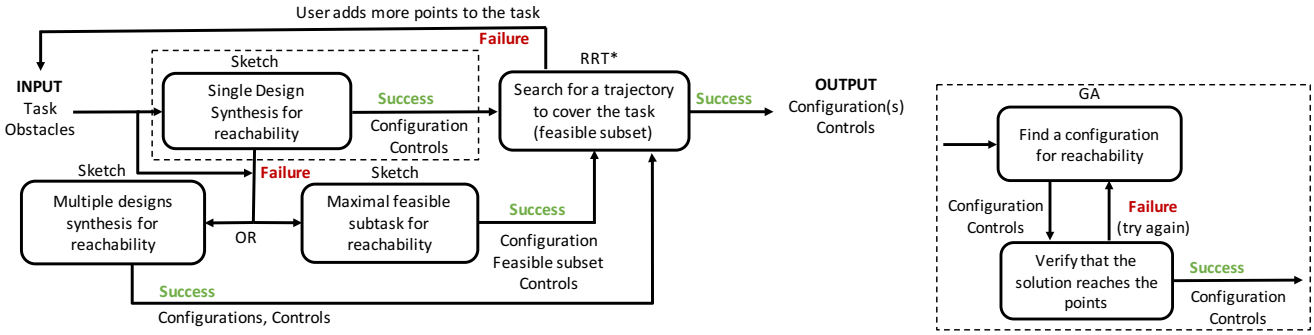


Fig. 3: Overview of the approach (left). For comparison, we show the equivalent of the dotted box for GA (right).

Second, we use the configuration found by the above problem (C_0) to search for a trajectory i.e.

$$\text{Find } \Gamma \text{ s.t. } \Gamma \subseteq W(C_0, O) \wedge T \subseteq \text{End}(\Gamma) \qquad \text{(Trajectory)}$$

The reachability problem is solved by encoding the necessary constraints in Sketch. To find a trajectory, we use a motion planning algorithm, specifically RRT* [7]. The RRT* algorithm starts from the joint angles (which are derived from the states) found by Sketch for $T[i]$ and incrementally constructs the path by sampling joint angles that satisfy the NoCollision constraints until it reaches the joint angles for $T[i+1]$. If a trajectory is found, the configuration and the controls to execute the trajectory are the returned solution. The motion planner will not find a trajectory if the task is spread across many disconnected components in the workspace of C_0. In this case, we ask the user to intervene and add more points from the anticipated trajectory to the task and then, the design process is repeated. Adding more points increases the likelihood that the task is in a single connected component of the workspace. Fig. 3 depicts our approach.

Problem 2 (Maximal Feasible Subtask): If no solution is found for problem 1, we separate the task into the feasible subset T_F and infeasible subset T_I and find a configuration C such that the feasible subset can be achieved without any collisions. Similar to problem 1, this problem is split into two parts.

$$\begin{aligned}
&\underset{C,T_F}{\text{maximize}} \ |T_F| \text{ s.t. } T_F \subseteq T \ \wedge \\
&T_F \subseteq \text{End}(W(C,O)) \qquad \text{(Reachability)} \\
&\exists \Gamma.\ \Gamma \subseteq W(C,O) \ \wedge \ T_F \subseteq \text{End}(\Gamma) \qquad \text{(Trajectory)}
\end{aligned}$$

Problem 3 (Multiple-Designs Synthesis): If no solution is found for problem 1, we find n_{conf} configurations, $C_1, C_2, \cdots, C_{n_{\text{conf}}}$, such that T can be achieved, without collisions, by the combination of configurations.

$$\begin{aligned}
&\underset{C_1,\cdots,C_{n_{\text{conf}}}}{\text{minimize}} \ n_{\text{conf}} \text{ s.t.} \\
&T \subseteq \text{End}(W_1) \cup \cdots \cup \text{End}(W_{n_{\text{conf}}}) \qquad \text{(Reachability)} \\
&\forall u \in [1, n_{\text{conf}}].\ \exists \Gamma_u.\ \Gamma_u \subseteq W_u \ \wedge \\
&\qquad T \subseteq \text{End}(\Gamma_1) \cup \cdots \cup \text{End}(\Gamma_{n_{\text{conf}}}) \qquad \text{(Trajectory)}
\end{aligned}$$

where $W_u = W(C_u, O)$.

We will now describe the encodings for the three reachability synthesis problems in the Sketch language.

B. Encoding Constraints for Single Design Synthesis

Finding a single configuration C to reach the task points requires searching over the discrete parameter n_{mod}, the number of modules in the configuration, and the continuous parameters r_j and α_j of the modules. Since the number of discrete choices for n_{mod} is small (we consider 2 to 4 DoF manipulators) and a smaller value for n_{mod} is preferable, we explicitly enumerate over the possible values for n_{mod}. Thus, we solve one synthesis problem for each different value of n_{mod} in a bounded synthesis process [22] that terminates when a solution is found. The following is an encoding of the synthesis problem for a fixed n_{mod}.

A straightforward approach is to search for the configuration parameters along with the joint angles Θ_i for the i-th point in the task such that the forward kinematics would lead the manipulator to the desired point without collisions. The main disadvantage of this approach is that most of the constraints such as the end-effector constraint and the collision avoidance constraints are in terms of the state of the manipulator; however, the function that determines the state from the free variables, Θ_i, is composed of highly non-linear functions (arising from the forward kinematics equations). This gives rise to numerous local minima making the synthesis problem hard.

We propose an alternate approach that encodes the constraints directly in terms of the state of the manipulator. In this encoding, the synthesis problem is to find the parameters of the different modules $r_1, \cdots, r_{n_{\text{mod}}}, c\_\alpha_1, \cdots, c\_\alpha_{n_{\text{mod}}}$ along with the states of the manipulator $P_1, \cdots, P_{n_{\text{task}}}$ to reach each of the n_{task} points in the task where c_α_j represents $\cos(\alpha_j)$. We use the subscript i to index a point in the task and j to index a module of the manipulator.

The first constraint establishes that the end-effector of the manipulator reaches the task points.

Constraint 1 (End-effector): $\forall i \in \{1..n_{\text{task}}\}$,

$$P_i.\text{end} = T[i] \qquad (1)$$

The collision avoidance constraints are defined as follows: First, to avoid collision with obstacles, the distance between all line segments that constitute the manipulator and the center of all the spheres has to be greater than their respective radii + thickness of the modules (Δ).

Constraint 2 (Obstacle avoidance): $\forall i \in \{1..n_{\text{task}}\}$, $\forall h \in \{1..2n_{\text{mod}}\}$, $\forall o \in O$,

$$\text{Dist}\left(P_i.\overrightarrow{v_h}, \langle o.x_c, o.y_c, o.z_c\rangle\right) > o.r_c + \Delta \tag{2}$$

where Dist is the minimal distance between a line segment and the center of a sphere.

Second, to avoid self-collision, the distance between all non-consecutive line segments has to be greater than the thickness of the modules (2Δ).

Constraint 3 (Self-collision avoidance): $\forall i \in \{1..n_{\text{task}}\}$, $\forall h \in \{1..2n_{\text{mod}}\}$, $\forall h' \in \{h+2..2n_{\text{mod}}\}$,

$$\text{Dist}\left(P_i.\overrightarrow{v_h}, P_i.\overrightarrow{v_{h'}}\right) > 2\Delta \tag{3}$$

The link lengths must be fixed, i.e. the lengths of the link segments of each module in the different states are the same and equal to the length of the module in the configuration parameters.

Constraint 4 (Consistent link length): $\forall i \in \{1..n_{\text{task}}\}$, $\forall j \in \{1..n_{\text{mod}}\}$,

$$||P_i.\overrightarrow{l_j}|| = r_j \tag{4}$$

Similarly, the next constraint is to ensure that the angle between the axes of two consecutive modules is consistent and equal to α in the configuration parameters.

Constraint 5 (Consistent alpha): $\forall i \in \{1..n_{\text{task}}\}$, $\forall j \in \{2..n_{\text{mod}}\}$,

$$P_i.\overrightarrow{z_j} \cdot P_i.\overrightarrow{z_{j-1}} = d * d * c\_\alpha_j \tag{5}$$

$$P_i.\overrightarrow{z_j} \cdot (P_i.\overrightarrow{z_{j-1}} \times P_i.\overrightarrow{l_{j-1}}) \geq 0 \tag{6}$$

Here, (6) is required to restrict α_j to $[0, \pi]$. d is the fixed offset distance.

Geometrical constraints enforced by the fixed actuator offsets d are encoded as follows:

Constraint 6 (Offset): $\forall i \in \{1..n_{\text{task}}\}$, $\forall j \in \{1..n_{\text{mod}}\}$,

$$||P_i.\overrightarrow{z_j}|| = d \tag{7}$$

Finally, to facilitate manipulator assembly, we enforce that the axis of the rotation of any actuator must be orthogonal to the previous and next link segments.

Constraint 7 (Orthogonality): $\forall i \in \{1..n_{\text{task}}\}$,

$$\forall j \in \{1..n_{\text{mod}}\}.\ P_i.\overrightarrow{l_j} \cdot P_i.\overrightarrow{z_j} = 0 \tag{8}$$

$$\forall j \in \{2..n_{\text{mod}}\}.\ P_i.\overrightarrow{l_{j-1}} \cdot P_i.\overrightarrow{z_j} = 0 \tag{9}$$

With these constraints, the synthesis problem is:

$$\begin{aligned}&\text{Find } r_j, c\_\alpha_j, P_i \quad i \in \{1..n_{\text{task}}\}, j \in \{1..n_{\text{mod}}\}\\ &\text{s.t. constraints (1 - 9)}\end{aligned}$$

C. Maximal Feasible Subtask

If Sketch does not find a single configuration to solve the task, it must solve either problem 2 or problem 3. In problem 2, the goal is to find a configuration that can reach as many points as possible. In addition to the number of modules (n_{mod}), the new discrete choices in this problem include the number of feasible task points (denoted as n_{f}, $n_{\text{f}} \leq n_{\text{task}}$) and whether an individual point in the task belongs to the feasible set. To solve this, the system iterates over a series of synthesis problems for different choices of n_{f} and n_{mod}, and chooses a solution that first maximizes n_{f} and then minimizes n_{mod}.

Given n_{mod} and n_{f}, we make the following changes to the encoding used to solve problem 1. First, to indicate which points in the task are feasible, we introduce n_{task} boolean variables $\{b_1, b_2, \cdots, b_{n_{\text{task}}}\}$ for each point in the task where $b_i = 1$ implies that the point i is feasible. Second, we define conditional versions of some of the previous constraints so that the constraints need to be satisfied only if $b_i = 1$.

Constraint 8 (Conditional): $\forall i \in \{1..n_{\text{task}}\}$, $\forall j \in \{1..n_{\text{mod}}\}$, $\forall h \in \{1..2n_{\text{mod}}\}$, $\forall h' \in \{h+2..2n_{\text{mod}}\}$, $\forall o \in O$

$$\begin{aligned} b_i \implies & P_i.\text{end} = T[i] && \wedge\\ & \text{Dist}\left(P_i.\overrightarrow{v_h}, \langle o.x_c, o.y_c, o.z_c\rangle\right) > o.r_c + \Delta && \wedge\\ & \text{Dist}\left(P_i.\overrightarrow{v_h}, P_i.\overrightarrow{v_{h'}}\right) > 2\Delta && \wedge\\ & ||P_i.\overrightarrow{l_j}|| = r_j && \wedge\\ & P_i.\overrightarrow{z_j} \cdot P_i.\overrightarrow{z_{j-1}} = d * d * c\_\alpha_j \end{aligned} \tag{10}$$

The synthesis problem is encoded as:

$$\begin{aligned}&\text{Find } r_j, c\_\alpha_j, P_i, b_i \quad i \in \{1..n_{\text{task}}\}, j \in \{1..n_{\text{mod}}\}\\ &\text{s.t. } \sum_{i=1}^{n_{\text{task}}} b_i >= n_{\text{f}} \ \wedge \ \text{constraints (6-10)}\end{aligned}$$

The constraint $\sum_{i=1}^{n_{\text{task}}} b_i >= n_{\text{f}}$ ensures that the cardinality of the feasible set is at least n_{f}. We do not condition the constraints (6) - (9) since they are geometric constraints and do not depend on either T or the parameters (r_j, α_j).

D. Multiple Designs Synthesis

As an alternative to problem 2, one could also choose to solve problem 3 that attempts to find multiple configurations that together reach all the points in the task. Here, we solve one synthesis problem for the different choices of n_{mod} and the number of configurations (n_{conf}) and choose the solution that first minimizes n_{conf} and then minimizes n_{mod}.

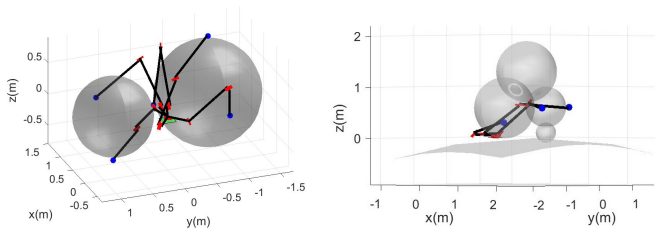
For the encoding, we introduce $n_{\text{task}} * n_{\text{conf}}$ boolean variables b_i^u for each point i and configuration u and the synthesis problem is: Find

$$\begin{aligned}&r_j^u, c\_\alpha_j^u, P_i^u, b_i^u \ \ i \in \{1..n_{\text{task}}\}, j \in \{1..n_{\text{mod}}\}, u \in \{1..n_{\text{conf}}\}\\ &\text{s.t. } \forall i.\ \bigvee_u (b_i^u = 1) \wedge \ \ (\forall u.\ \text{ constraints (6-10)})\end{aligned}$$

where $\bigvee_u (b_i^u = 1)$ ensures that at least one configuration can reach the point i.

IV. Results

In this section, we evaluate our approach on various tasks and compare the results with a genetic algorithm based approach (baseline). We divide the tasks into three categories – (a) feasible tasks without obstacles, (b) feasible tasks with obstacles, and (c) partially infeasible tasks. In this evaluation, we answer the following questions: How does our approach and the baseline scale with the number of points and the number of DoFs? Can the approaches find a design configuration when the environment is highly constrained due to obstacles? Can our approach find a single configuration or



(a) (b)

Fig. 4: (a) The solution returned by our approach for $n_{\text{obs}} = 2$, $n_{\text{task}} = 5$ and $n_{\text{mod}} = 3$. The actuators are in red, the links in black, the task points are the blue dots, and the origin is at the center of the green square. (b) The solution returned by our approach for $n_{\text{obs}} = 5$, $n_{\text{task}} = 3$ and $n_{\text{mod}} = 3$.

a set of configurations for partially infeasible tasks? Finally, we present examples that highlight the interaction between design synthesis and verification through motion planning.

In the examples, we treat the approach as successful if it was able to find a configuration and a set of joint angles that drives the end-effector to a distance smaller than $1mm$ from each of the task points while avoiding collisions. For all cases, we set $r_{max} = 1m$. All the simulations were run on a standard desktop machine 64 bit running 14.04 Ubuntu with 8 GB RAM and 3.6 GHz processor.

A. Baseline - GA

The population of configurations is initially generated at random and evaluated according to a score given by the fitness function; for each configuration, we solve the IK problem for every point in the task to get Θ_i and P_i. The fitness function is:

$$F = \frac{1}{\sum_{i=1}^{n_{\text{task}}} F_i} \qquad \text{where}$$

$$F_i = \; || \, P_i.\text{end} - T[i] \, || + \begin{cases} 10^7, \text{ if there is collision} \\ 10^{-7}, \text{ otherwise} \end{cases}$$

The 20 configurations with the highest fitness scores compose the second generation. The next generations are altered by the mechanisms of mutation, selection and crossover.

The algorithm terminates if a configuration in the population has $F > 1/(n_{\text{task}} * 0.001)$. The algorithm also terminates and returns the best configuration so far when the maximum number of iterations (here 15) is reached or the runtime exceeds a maximum value. For the GA approach, we check whether the solution is correct by calculating the forward kinematics; if the solution does not satisfy the task, we reseed the population and re-run the algorithm.

B. Feasible task without obstacles

We randomly generated 5 feasible tasks for each $n_{\text{task}} = \{3, 6, 9\}$ and $n_{\text{mod}} = \{2, 3, 4\}$. To create a feasible task, we randomly generate C and a set of controls Θ then, using FK, we calculate T, which is the input for both approaches.

The success rates of the two approaches are shown in Table I. It can be seen that Sketch can solve all the tasks irrespective of the number of points or number of DoFs.

		Genetic Algorithm		Sketch	
Task	DoFs	% Success	Runtime (s)	% Success	Runtime (s)
3 Points	2	60	42.8 ± 3.1	100	0.8 ± 0.01
	3	100	46.6 ± 23.0	100	1.3 ± 0.01
	4	100	74.5 ± 7.5	100	2.9 ± 0.1
6 Points	2	80	105.6 ± 45.8	100	3.7 ± 3.4
	3	100	152.0 ± 54.1	100	5.3 ± 2.5
	4	100	145.9 ± 10.9	100	11.5 ± 3.6
9 Points	2	60	125.1 ± 6.4	100	9.9 ± 5.3
	3	100	311.9 ± 115.7	100	6.1 ± 5.2
	4	100	211.5 ± 5.1	100	32.7 ± 13.7

TABLE I: Rate of success for a feasible task in an environment without obstacles for GA and Sketch.

Obstacle radius (m)	GA solved?	Runtime	Sketch solved?	Runtime
0.2	Yes	2.3s	Yes	3.1s
0.4	Yes	48.47min	Yes	2.8s
0.6	No	-	Yes	9.4s
0.8	No	-	Yes	13.6s
1.0	No	-	Yes	10.6s

TABLE II: GA and Sketch results for $n_{\text{obs}} = 1$, $n_{\text{mod}} = 3$ and $n_{\text{task}} = 5$.

Genetic algorithm, on the other hand, fails some tasks with 2 DoFs given a maximum runtime of 10min. This is because with fewer DoFs, there are less configurations that can satisfy the task which decreases the likelihood of GA finding a solution. This evaluation shows that our approach performs well even when the set of valid solutions is highly restricted.

C. Feasible task with obstacles

In this evaluation, we introduce an obstacle in the environment. We fix $n_{\text{mod}} = 3$ and $n_{\text{task}} = 5$, but increase the size of the obstacle; the larger the obstacle, the more constrained the environment is. The results are shown in Table II. Again, our approach is able to solve all the tasks but GA can only solve for small obstacles given a maximum runtime of 60min.

Next, we consider environments with multiple obstacles (more obstacles imply more constrained environments). Fig. 4(a) shows an example with two obstacles and Fig. 4(b) shows an example with five obstacles. Our approach can find solutions for both the tasks; in contrast, the solutions found by GA either collide with the obstacles or cannot reach the points. The results again show that our approach works better than GA in highly constrained environments.

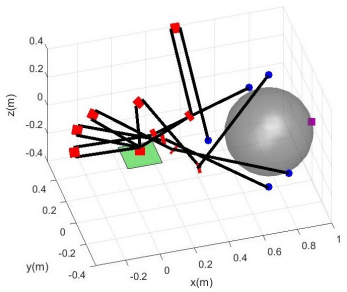
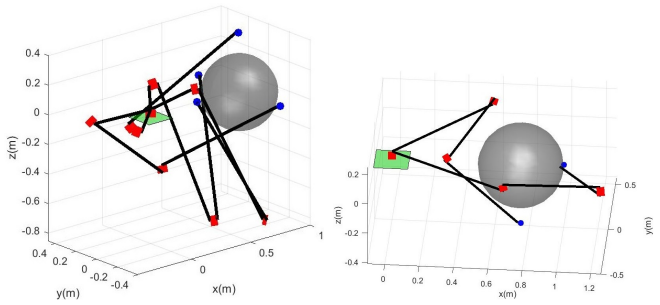
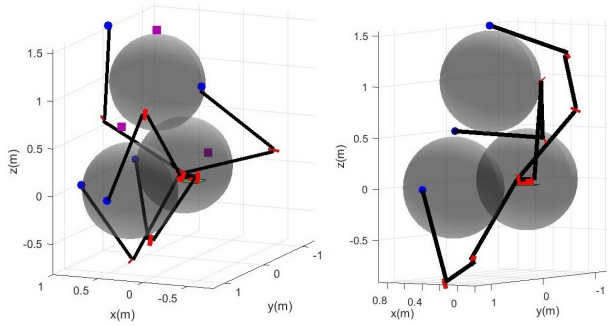


Fig. 5: Solution for Problem 2 that satisfies $n_{\text{f}} = 5$, $n_{\text{task}} = 6$, $n_{\text{obs}} = 1$, and $n_{\text{mod}} = 3$. The infeasible point is the purple square.



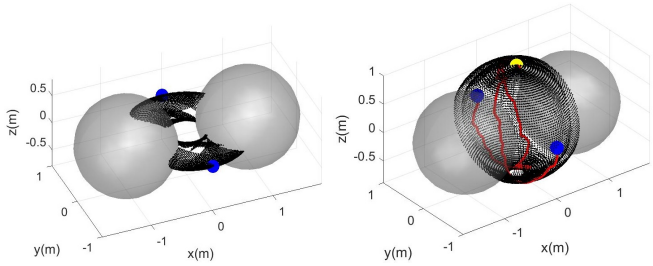
(a) (b)

Fig. 6: (a) First, and (b) Second configurations returned for Problem 3 with $n_{\text{conf}} = 2$.



(a) (b)

Fig. 7: (a) Solution for Problem 2 that satisfies $n_{\text{f}} = 5$, $n_{\text{obs}} = 3$ and $n_{\text{mod}} = 3$. The infeasible points are the purple squares. (b) Solution for Problem 1 with $n_{\text{task}} = 3$, $n_{\text{conf}} = 1$, $n_{\text{obs}} = 3$ and $n_{\text{mod}} = 4$.



(a) (b)

Fig. 8: (a) Collision-free workspace of solution returned for Problem 1 with $n_{\text{obs}} = 2$, $n_{\text{task}} = 2$, and, $n_{\text{mod}} = 2$. RRT* was not able to find a trajectory due to disconnected workspace. (b) Collision-free workspace after additional user input (yellow dot), and trajectory connecting T (red curve).

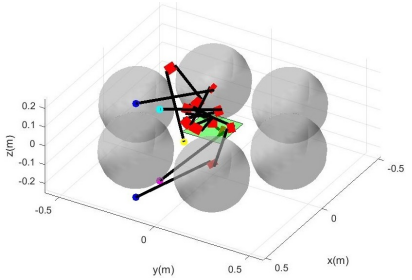


Fig. 9: Solution returned for Problem 1 with $n_{\text{obs}} = 6$, $n_{\text{task}} = 5$, and, $n_{\text{mod}} = 3$. The blue dots are the original task, the others are additional user inputs.

D. Partially infeasible task

We considered two partially infeasible tasks (Fig. 5 and 7). The first task has 6 points and 1 obstacle. Sketch was not able to find a solution for a single configuration (Problem 1) in the allotted time. Solving Problem 2 resulted in classifying one point as infeasible. Fig. 5 shows the solution found by Sketch, in which the purple square indicates the infeasible point. In addition, our approach was able to find a solution for Problem 3 with two configurations that can satisfy the entire task, as shown in Fig. 6.

The second task has 8 points and 3 obstacles. Again Sketch could not find a solution for Problem 1, however it solved Problem 2, classifying three of the eight task points as infeasible (Fig. 7a). Sketch could not find a solution for Problem 3 with $n_{\text{mod}} = 3$; instead we took the infeasible set T_I from Problem 2 and solved Problem 1 where $T = T_I$. Our approach was able to find a solution (Fig 7b) for those 3 points using a 4 DoF manipulator, a more complex design than the one found for the original feasible set T_F which was a 3 DoF design.

E. Iterative task refinement

For the tasks discussed so far, we were able to find a trajectory for the synthesized manipulator that connects the task points, thus ensuring the task can be achieved. Here, we consider a task that requires additional user input. Fig. 8a shows the collision-free workspace (in black) as well as the task points (blue dots) for the configuration returned as solution for Problem 1 with $n_{\text{obs}} = 2$, $n_{\text{task}} = 2$ and $n_{\text{mod}} = 2$. As it can be seen, the workspace is disconnected, thus it is impossible for RRT* to find a feasible trajectory. Fig. 8b shows the new collision-free workspace for the configuration returned after a new point (yellow dot) is added by the user to the task, and the trajectory found (red curve).

Fig. 9 shows a more complex environment that needs multiple user interactions. The two blue dots are the original task, the yellow was the first point added by the user, the magenta the second and the cyan the third.

V. Conclusions and Future Work

In this paper, we present a framework for synthesizing provably-correct task-based designs for manipulators. Our approach is able to handle partially infeasible tasks and can generate more than one configuration if the task cannot be solved by a single one. In addition, our approach outperforms GA in highly constrained environments.

The limitations of the approach are that it is not complete, i.e. it may not find a solution within the time constraints even if a solution exists, and it requires user input when a design is synthesized but a trajectory is not found.

In the future, we will explore techniques to automatically infer the new points required for trajectory generation. We will also encode more complex tasks such as following trajectories, as well as additional physical constraints such as torque limits. Moreover, we will extend our approach to handle more complex modular robots that combine locomotion and manipulation.

REFERENCES

[1] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[2] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: a modular reconfigurable robot," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 514–520.

[3] B. Salemi, W.-M. Shen, and P. Will, "Hormone-controlled metamorphic robots," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4194–4199.

[4] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, "An end-to-end system for accomplishing tasks with modular robots." in *Robotics: Science and Systems*, 2016.

[5] T. Tosun*, J. Daudelin*, G. Jing, H. Kress-Gazit, M. Campbell, and M. Yim, "Perception-informed autonomous environment augmentation with modular robots," in *IEEE International Conference on Robotics and Automation*.

[6] HebiRobotics. [Online]. Available: https://www.hebirobotics.com/x-series-smart-actuators

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1177/0278364911406761

[8] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Task-based limb optimization for legged robots," in *IEEE/RSJ Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2062–2068.

[9] E. Van Henten, D. Vant Slot, C. Hol, and L. Van Willigenburg, "Optimal manipulator design for a cucumber harvesting robot," *Computers and electronics in agriculture*, vol. 65, no. 2, pp. 247–257, 2009.

[10] C. Baykal and R. Alterovitz, "Asymptotically optimal design of piecewise cylindrical robots using motion planning," in *Robotics: Science and Systems*, 2017.

[11] S. Patel and T. Sobh, "Task based synthesis of serial manipulators," *Journal of advanced research*, vol. 6, no. 3, pp. 479–492, 2015.

[12] W. K. Chung, J. Han, Y. Youm, and S. Kim, "Task based design of modular robot manipulator using efficient genetic algorithm," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 1. IEEE, 1997, pp. 507–512.

[13] J.-O. Kim and P. K. Khosla, "A formulation for task based design of robot manipulators," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1993, pp. 2310–2317.

[14] I.-M. Chen and J. W. Burdick, "Determining task optimal modular robot assembly configurations," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 1. IEEE, 1995, pp. 132–137.

[15] S. Farritor, S. Dubowsky, N. Rutman, and J. Cole, "A systems-level modular design approach to field robotics," in *IEEE International Conference on Robotics and Automation*, 1996, pp. 2890–2895.

[16] G. S. Hornby, H. Lipson, and J. B. Pollack, "Evolution of generative design systems for modular physical robots," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4146–4151.

[17] S. Tabandeh, W. Melek, M. Biglarbegian, S.-h. P. Won, and C. Clark, "A memetic algorithm approach for solving the task-based configuration optimization problem in serial modular and reconfigurable robots," *Robotica*, vol. 34, no. 9, pp. 1979–2008, 2016.

[18] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, "Combinatorial sketching for finite programs," *ACM Sigplan Notices*, vol. 41, no. 11, pp. 404–415, 2006.

[19] A. Solar-Lezama, "Open source sketch synthesizer," 2012. [Online]. Available: https://bitbucket.org/gatoatigrado/sketch-frontend/

[20] J. P. Inala, S. Gao, S. Kong, and A. Solar-Lezama, "REAS: combining numerical optimization with SAT solving," *CoRR*, vol. abs/1802.04408, 2018. [Online]. Available: http://arxiv.org/abs/1802.04408

[21] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.

[22] B. Finkbeiner and S. Schewe, "Bounded synthesis," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5-6, pp. 519–539, 2013.