CAPO: Control and Actuator Placement Optimization for Large-Scale Problems with Nonlinear Dynamics

Mitchell B. Fogelson[1], Giusy Falcone[2], and Zachary Manchester[1]

[1] Carnegie Mellon University, Pittsburgh, PA, USA
[2] University of Michigan, Ann Arbor, MI, USA
Corresponding Author Email: mfogelson@cmu.edu

Abstract. Actuator selection and placement are critical aspects of robot design that are tightly coupled to task performance. Jointly optimizing actuator placement and control policies or trajectories can enhance efficiency and robustness. However, current optimization methods struggle to scale to high-dimensional systems with many degrees of freedom, including soft robots, large flexible spacecraft, or cloth manipulation. We propose CAPO, a scalable Control and Actuator Placement Optimization method that jointly optimizes the number, type, and placement of actuators, along with control trajectories. CAPO concurrently optimizes over all the control and design parameters in a single computationally tractable nonlinear program that scales favorably with system size and complexity. CAPO is evaluated against a state-of-the-art genetic algorithm and mixed-integer programming solvers on six problems, including an acrobatic multirotor aircraft, a spinning space structure, cloth manipulation, and a soft robotic swimmer. On small-scale problems, CAPO finds comparable solutions with similar objective values in 2.5-218x less time than existing methods. On large-scale problems, CAPO is the only method capable of finding to a feasible solution, and it achieves actuator configurations that reduce the total number of actuators by 12%-27.5% compared to baselines.

Keywords: Control Theory and Optimization · Optimal Control · Design Optimization

1 INTRODUCTION

Nature provides many examples of physical structures adapted to perform a behavior efficiently. For instance, humans can swim long distances but much slower than dolphins. Similarly, the design of a robotic system significantly influences its ability to complete tasks, as demonstrated by Boston Dynamics' Atlas robot being one of the few humanoids to perform a backflip [1]. From a designer's perspective, creating a backflipping robot requires significant effort and many iterations to select and place actuators. From the control engineer's perspective, significant effort is applied to enable the robot to achieve the backflip while operating within the constraints of the selected actuators. Within the robotics community, the type and placement of actuators are essential factors in robot design across various applications and tasks, from soft robot locomotion [5], maneuvering continuum robots [2, 3, 7, 17, 25, 31, 36], to resource-constrained robot control [33].

Design decisions like actuator selection are discrete choices that expand the design space combinatorially as the number of actuators considered grows. Current design and control optimization approaches are performed in separate sequential loops, resulting in inefficient evaluation of intermediate designs. Direct collocation (DIRCOL) is an optimal control method used to determine open-loop control trajectories that align with desired reference trajectories while adhering to dynamics and thrust constraints. DIRCOL optimizes state and control variables simultaneously, making it well-suited for handling complex nonlinear dynamics [14]. A unique feature of DIRCOL is that intermediate iterations in the optimization loop do not need to be dynamically feasible; only the final result is required to satisfy all constraints. Including parameters like actuator type and position as decision variables in DIRCOL can mitigate the costly evaluation of intermediate designs. A simplified diagram showing the joint actuator and control optimization problem can be seen in Figure 1. Applying these parameters in a DIRCOL formulation and solving the joint problem of actuator selection and control optimization problem leads to a mixed integer nonlinear program (MINLP), which is NP-hard [28]. This makes it computationally impractical to find globally optimal solutions in most cases. Thus, designing efficient algorithms that can quickly find feasible sub-optimal solutions while scaling to large problems is a crucial area of research in robotics.

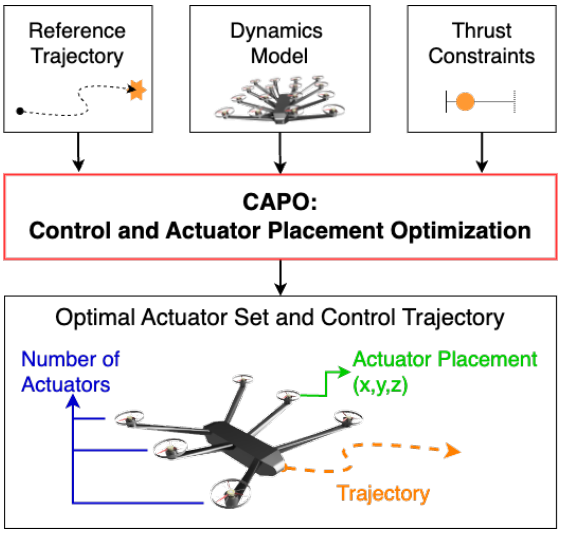

Fig. 1. Example of joint robot control and design optimization, which takes user-defined reference trajectory, dynamics models, and thrust constraints leading to an output of a feasible design configuration and control trajectory. The diagram shows finding the optimal set and placement of propellers on a multirotor to achieve a flip and the open-loop control trajectory.

To tackle the challenges described above of joint actuator and control optimization for robot design, we propose CAPO, a single-level optimization formulation for actuator placement and control that jointly minimizes the number of actuators, control effort, and tracking error to perform a desired task. CAPO achieves this by relaxing the actuator-placement problem into a computationally tractable nonlinear program (NLP),

providing a framework that scales well for complex, high-dimensional robots and trajectories. To evaluate its performance, CAPO is compared to three state-of-the-art algorithms: a genetic algorithm (GA) [38], a mixed-integer quadratic program (MIQP) solver [12], and a mixed-integer nonlinear program (MINLP) solver [18]. These methods are compared in Table 1, which highlights CAPO's advantages. The methods are evaluated on four low-dimensional actuator-placement design problems in simulation, including an over-actuated double integrator, an acrobatic multirotor aircraft, a flexible space structure, and a cloth manipulation task. CAPO is further demonstrated on three additional high-dimensional problems that the other methods failed to solve, including a 120-degree-of-freedom (DoF) flexible space structure, a 150-DoF cloth-manipulation task, and a 234-DoF soft-robotic swimmer. Our contributions include:

1. A scalable algorithm, CAPO, for the simultaneous control and actuator-selection-and-placement problem that concurrently considers actuator type, placement, and control parameters.
2. A domain randomization approach for improved design robustness that maintains problem sparsity structure and scalability.
3. Experimental validation of CAPO on various high-dimensional robotics problems demonstrating the computational advantages over the state-of-the-art.

The paper proceeds as follows: Section 2 reviews background information on nonlinear programming, integer-programming relaxations, and L1 regularization. Section 3 surveys related works that address the actuator-placement problem. Section 4 formalizes the control-and-actuator-placement problem that the method addresses. Section 5 derives the CAPO algorithm. Section 6 provides implementation details and presents the experimental results. Finally, Section 7 summarizes the findings and proposes directions for future work.

Table 1. Comparison table of the proposed method, CAPO, with three state-of-the-art approaches: Genetic Algorithm, Mixed Integer Quadratic Program, and Mixed Integer Nonlinear Program. CAPO's primary advantage is its ability to optimize all parameters in a single optimization, handle nonlinear constraints, and scale effectively with an increasing number of binary variables.

	CAPO (Ours)	Genetic Algorithm (GA)	Mixed Integer Quadratic Program (MIQP)	Mixed Integer Nonlinear Program (MINLP)
Single Level	✓	×	✓	✓
Nonlinear Constraints	✓	✓	×	✓
Scalable to High Dim.	✓	×	×	×

2 Background

We now review background information on Linear and Nonlinear Programming, Mixed-Integer Programming, the branch and bound method for solving Mixed-Integer Programs, and the L1 regularization method for optimization.

2.1 Linear, Nonlinear, and Mixed-Integer Programming

Smooth nonlinear programs (NLPs) can be expressed in the following standard form,

$$\underset{\mathbf{x},\,\mathbf{y}}{\text{minimize}} \quad \ell(\mathbf{x},\,\mathbf{y}) \tag{1a}$$

$$\text{subject to} \quad f(\mathbf{x},\,\mathbf{y}) = \mathbf{0}, \tag{1b}$$

$$g(\mathbf{x},\,\mathbf{y}) \geq \mathbf{0}, \tag{1c}$$

where, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ are decision variables, $\ell(\cdot)$ is a scalar-valued objective function, and $f(\cdot)$ and $g(\cdot)$ are equality and inequality constraints, respectively. Functions are assumed to be at least twice continuously differentiable. If $\ell(\cdot)$, $f(\cdot)$, and $g(\cdot)$ are linear functions, the problem is known as a linear program (LP). If $\ell(\cdot)$ is quadratic and both $f(\cdot)$ and $g(\cdot)$ are linear, the problem is called a quadratic program (QP). In all of these cases, locally optimal solutions satisfy the Karush-Kuhn-Tucker (KKT) conditions (first-order necessary conditions) [20].

A mixed-integer nonlinear program (MINLP) adds the constraint,

$$\mathbf{y} \in \{0,1\}^m \tag{2}$$

where the $\mathbf{y}$ variables are constrained to take on integer values. Note that, without loss of generality, binary constraints $\mathbf{y} \in \{0,1\}^m$ are considered. Similar to the continuous case, if $\ell(\cdot)$, $f(\cdot)$ and $g(\cdot)$, are all linear functions, the problem is known as a mixed-integer linear program (MILP), with mixed-integer quadratic programs (MIQP) defined similarly. Mixed-integer programs are NP-hard in general, and their computational complexity grows combinatorially in the problem size [40].

The branch-and-bound method is a prevalent technique for solving mixed-integer programs [21]. In its basic form, branch and bound first relaxes the binary constraints (2), known as a continuous relaxation, then solves the relaxed problem to optimality. This relaxed problem is strictly easier than the original binary-constrained version, and will have an objective value that lower bounds the original problem. The solution to this subproblem returns $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})$, where j is the subproblem index. If all values of $\mathbf{y}^{(j)}$ happen to take on values of 0 or 1, then the solution is also an optimal solution to the original MIP. If this is not the case, the problem is branched into two new problems for each element $y_i^{(j)} \notin \{0,1\}$; one with the additional constraint $y_i = 0$ and the other with the additional constraint $y_i = 1$. The new subproblems are solved, and the branching continues until all values in the solution satisfy the original problem constraints. If a feasible solution is found, its objective value is used to prune branches that have relaxed solutions with higher objective values, as these branches cannot yield a better solution than the one already found.

2.2 L1 Regularization Methods

One-norm or L1 regularization (also called "LASSO regularization") is commonly used in statistics and machine learning to drive many model coefficients to zero, producing sparse solutions [8, 11, 32, 37, 42]. L1 regularization presents itself as an additional penalty term in the objective function:

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad \ell(\mathbf{x},\mathbf{y}) + \sum_{i=1}^{m} \alpha |y_i| \tag{3a}$$

where α is a scalar penalty weight. While this method is well-known and widely used in the statistics and machine-learning communities, it is less frequently applied in the design community. Notably, Ha et al. [13] and Skouras et al. [34] have utilized L1-regularization to promote sparsity in design optimization applications. The hyperparameter α controls the sparsity of the solution, with larger values encouraging solutions with more zero elements. One may need to evaluate multiple α values to find a desirable solution.

3 RELATED WORKS

Several common approaches exist for addressing the mixed-integer nature of design optimization. A brief survey of sampling-based methods, mixed-integer programming, and continuous-optimization methods that solve relaxations of mixed-integer programs is presented.

3.1 Sampling Methods

Sampling-based methods are a common technique for robot configuration design. These methods systematically generate or select robot configurations from the design space. Genetic algorithm (GA), Simulated Annealing (SA), and reinforcement learning (RL) approaches are included in this category. In general, while sampling-based methods are very flexible and straightforward to implement, they can be computationally expensive due to the combinatorial nature of design problems and, subsequently, large design space.

Sampling-based approaches can leverage modern multi-core computing capabilities to parallelize the evaluation of many configurations simultaneously. Rao et al. [30] used a GA approach to find the optimal placement of actuators on a two-bay truss to dissipate the structure's energy. Borairi et al. [23] and Molter et al. [24] both use GA to place actuators along flexible structures. Baykal et al. [3] and Kuntz et al. [2] use an adaptive SA approach for robot kinematic design. Using the observability and controllability matrices from the linearized dynamics, Hu et al. sample many configurations to find the optimal placement of reaction wheels for hydroelastic bodies [15]. Bergeles et al. [7] and Anor et al. [36] both use direct search methods, which iteratively samples points around a current guess for concentric tube robots.

Zhao et al. [41] introduced Robogrammar, an RL approach to iteratively apply discrete grammar rules to construct robot configurations that maximize locomotion speed over specific terrains. This approach uses learning to direct the search in the large design space to find the best configurations. However, each design must be evaluated through

an expensive control optimization and simulation loop. In this work, concurrent optimization of design, controls, and simulation state enables greater efficiency, addressing a limitation of sampling-based methods, which struggle to optimize these variables simultaneously due to the growth in dimensionality.

3.2 Mixed-Integer Programming

Mixed-integer programs (MIPs) can natively handle discrete actuator-selection variables. Chanekar et al. [9] used an MIQP approach for the optimal placement of actuators in linear systems. Their formulation allowed for globally optimal solutions to be found but suffered from the computational burden of the branch-and-bound algorithm as the number of parameters increased. Deenen et al. [10] used Gurobi, a solver for MILPs and MIQPs, to optimize actuator placement for a hypothermia treatment for cancer patients. MIQP formulations have two fundamental limitations: the evaluation time scales poorly with the number of binary variables considered, and they cannot handle nonlinear dynamics constraints easily.

An increasing number of solvers have been developed for mixed-integer nonlinear programs [18, 19, 26]. However, these solvers must still deal with solving a large number of relaxed NLP subproblems during branching and often fail to find constraint-satisfying solutions efficiently.

3.3 Nonlinear Programming

Studies have developed strategies to relax or eliminate discrete parameters within the realm of robotic co-design. For example, Skouras et al. [34] presented a method for the design of tether-actuated deformable characters that could satisfy a set of poses. In a multi-step process, the approach identified a minimal set of tethers, their locations, and the optimal material parameters to best match the desired poses. A regularization technique akin to LASSO was employed to zero the force vector from redundant tethers, thus being able to handle the discrete problem using continuous optimization. However, their method is limited to static pose configurations.

Spielberg et al. [35] present a formulation for the design and control of task-driven robots, including continuous design parameters such as link lengths and actuator locations. They also include discrete parameters such as actuator type. They relax actuator-type parameters through a linear interpolation between actuator efforts. However, the number of actuators in the design was pre-specified. Lin et al. [17] used a scalable NLP solver to design concentric tube robots, but excluded any integer variables from the problem formulation.

4 The Control-and-Actuator-Placement Problem

This work seeks to determine the optimal type, location, and number of actuators a robot needs to track a specified trajectory while minimizing control input. This work

extends the standard DIRCOL formulation:

$$\begin{aligned} \underset{\mathbf{x}_{2:N},\,\mathbf{u}_{2:N-1}}{\text{minimize}} \quad & \ell(\mathbf{x}_k,\mathbf{u}_k) && \text{(4a)}\\ \text{subject to} \quad & f(\mathbf{x}_{k+1},\mathbf{x}_k,\mathbf{u}_k)=\mathbf{0} \quad \forall k, && \text{(4b)}\\ & g(\mathbf{x}_k,\mathbf{u}_k)\geq\mathbf{0} \quad \forall k && \text{(4c)} \end{aligned}$$

where $\mathbf{x}_k$ is the discrete-time robot state (including the robot's position, orientation, and velocity) and $\mathbf{u}_k$ is the discrete-time control input (describing the external force or moment being applied to the robot), to include additional actuator design parameters. To this end, a time-invariant actuator-selection vector $\boldsymbol{\beta}$ and a time-invariant continuous actuator position variable $\mathbf{r}$ are introduced:

$$\boldsymbol{\beta}=\begin{bmatrix}\beta_1\\ \vdots\\ \beta_M\end{bmatrix}, \quad \mathbf{r}=\begin{bmatrix}x_1,y_1,z_1\\ \vdots\\ x_M,y_M,z_M\end{bmatrix} \tag{5}$$

Here, $\boldsymbol{\beta}$ is a binary vector where the index value describes the existence of an actuator, and $\mathbf{r}$ specifies the position of the actuators on the robot. For example, $\boldsymbol{\beta}$ could consist of a set of thrusters and reaction wheels of various specifications to be considered for a spacecraft design where $\boldsymbol{\beta}[1]$ is an ion thruster and $\mathbf{r}[1]=(x_1,y_1,z_1)_B$ is the position of the thruster in the body frame. Therefore, if $\boldsymbol{\beta}[1]=1$, there is an ion thruster at $\mathbf{r}[1]=(x,y,z)_B$, and if $\boldsymbol{\beta}[1]=0$, the ion thruster is not added to the spacecraft. A dense set of actuator types (Eg. thrusters, reaction wheels, propellors, etc.) within specified control volumes (to prevent overlapping actuators) is enumerated for a particular design problem *a-priori* by the designer. – While not strictly necessary, the authors found it easiest to start with an actuator set that made the system fully controllable.. The new problem is to find a robot actuator configuration that enables the robot to best track a user-specified reference motion in the presence of disturbances. This problem can be formulated as the following MINLP:

$$\begin{aligned} \underset{\mathbf{x}_{2:N},\,\mathbf{u}_{2:N-1},\,\boldsymbol{\beta},\,\mathbf{r}}{\text{minimize}} \quad & \sum_k \ell(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\beta},\mathbf{r}), \quad \forall\,\mathbf{x}_1\in\mathbb{X}_1 && \text{(6a)}\\ \text{subject to} \quad & f(\mathbf{x}_{k+1},\mathbf{x}_k,\mathbf{u}_k\odot\boldsymbol{\beta},\mathbf{r})=\mathbf{0} \quad \forall k, && \text{(6b)}\\ & g(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\beta},\mathbf{r})\geq\mathbf{0} \quad \forall k, && \text{(6c)}\\ & \boldsymbol{\beta}\in\{0,1\}^m, && \text{(6d)} \end{aligned}$$

where $\mathbf{x}_k$ is the discrete-time system state, $\mathbf{u}_k$ is the control input vector, N is the total number of knot points (which is problem dependent). $\boldsymbol{\beta}$ is the binary vector of variables that specifies active actuator configuration, and $\mathbf{r}$ is the vector of actuator positions. $\ell(\cdot)$ is the objective function related to trajectory tracking, control minimization, and actuator reduction. $f(\cdot)$ is the nonlinear discrete-time dynamics, and $g(\cdot)$ are additional task-specific inequality constraints, such as thrust limits and actuator position limitations. The operator $\odot$ seen in Equation (6b) denotes an element-wise multiplication coupling the actuator selector vector, $\boldsymbol{\beta}$, and the control input, $\mathbf{u}_k$.

To avoid overfitting the robot actuator configuration, $\boldsymbol{\beta}$, to a specific initial condition $\mathbf{x}_1$, we include that the solution for $\boldsymbol{\beta}$ must satisfy all initial conditions, $\mathbb{X}_1$, in Equation

(6). In the next section, we transform this complete problem, which is computationally intractable for high-dimensional problems, into a formulation that scales well for large-scale problems.

5 The CAPO Algorithm

The CAPO algorithm provides a single-level optimization formulation that finds feasible solutions to the MINLP shown in Equation (6). CAPO, particularly, is designed to support high-dimensional robot-design problems when the size of $\boldsymbol{\beta}$ increases, thus considering a large number of potential actuators. This is achieved by relaxing the binary actuator-selection vector, $\boldsymbol{\beta}$, including an L1 regularization on $\boldsymbol{\beta}$, and sampling and solving over a finite number of trajectories from the set of initial conditions $\mathbb{X}_1$ to ensure solutions are valid over a wide range of the state-space and encourage robustness to disturbances.

5.1 Discrete Relaxation and L1 Regularization

Following standard practice for MINLP relaxation, a continuous relaxation transforms the constraint Equation (6d) into an inequality,

$$\mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1}, \tag{7}$$

which transforms Equation (6) into a continuous NLP, which can be solved relatively efficiently for very high dimensional problems. However, having a $\boldsymbol{\beta}$ vector with many non-zero values does not support a parsimonious actuator configuration. To achieve a sparse set of actuators, an L1 regularization is applied to $\boldsymbol{\beta}$. Consequently, the objective function in Equation (6a) is modified as follows:

$$\ell(\mathbf{x}, \mathbf{u}, \boldsymbol{\beta}, \mathbf{r}) + \alpha|\boldsymbol{\beta}|_1 \tag{8}$$

Because $\boldsymbol{\beta}$ is time-invariant, this regularization nullifies an actuator at all times steps, as opposed to applying regularization directly to the control inputs, which would only nullify the control at specific time instances. This is an important design decision of this paper, as the inclusion of additional decision variables $\boldsymbol{\beta}$ effectively decouples the actuator selection from the control optimization problem. The parameter α may require tuning between multiple trials for each problem to find a balanced regularization term.

5.2 Configuration Robustness

It is infeasible to account for all possible initial conditions in pursuing a robust configuration. Hence, CAPO optimizes over a set of sampled initial conditions and resulting state trajectories. Samples are drawn from a normal distribution:

$$\mathbf{x}_1^{(i)} \sim \mathcal{N}(\bar{\mathbb{X}}_1, \Sigma). \tag{9}$$

This approach ensures that the solution obtained is robust against disturbances. The covariance Σ should be selected based on the expected or desired amount of disturbance

the robot should recover from. While sampling introduces more decision variables, the problem structure remains sparse, thereby facilitating efficient solutions. As an aside, given that multiple control trajectory samples are being solved for, a feedback control policy $u = \pi_\theta(x)$ could be recovered from these samples using supervised learning with a neural network function approximator.

5.3 Relaxed Nonlinear Program

From the changes described above, CAPO, our sparse and computationally tractable relaxed single-level NLP is summarized as:

$$\underset{\mathbf{x}_{2:N}^{(1:I)}, \mathbf{u}_{2:N-1}^{(1:I)}, \boldsymbol{\beta}, \mathbf{r}}{\text{minimize}} \quad \sum_i \sum_k \ell(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \boldsymbol{\beta}, \mathbf{r}) + \alpha |\boldsymbol{\beta}|_1 \tag{10a}$$

$$\text{subject to} \quad f(\mathbf{x}_{k+1}^{(i)}, \mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)} \odot \boldsymbol{\beta}, \mathbf{r}) = \mathbf{0} \;\; \forall (k,i), \tag{10b}$$

$$g(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \boldsymbol{\beta}, \mathbf{r}) \geq \mathbf{0} \;\; \forall (k,i), \tag{10c}$$

$$\mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1}, \tag{10d}$$

where $\mathbf{x}_{2:N}^{(1:I)}$ is I instances of the robot state trajectory consisting of N knot points and $\mathbf{u}_{2:N-1}^{(1:I)}$ is I instances of the robot control trajectory consisting of N knot points. Note that there is only one instance of $\boldsymbol{\beta}$ and $\mathbf{r}$ as they are invariant across all trajectories and time instances. In practicality, these decision variables are stacked in a single large matrix formulation. In this work, CAPO only considers five instances of this problem, $I = 5$; however, modern sparsity-exploiting NLP solvers can solve very large problem instances.

5.4 Projection

After finding a solution to Equation (10), it is straightforward to project this solution onto the feasible set of the original MINLP from Equation (6). The solution to $\boldsymbol{\beta}$, due to the L1 regularization, will consist of values that are either zero or non-zero. Since the $\boldsymbol{\beta}$ and $\mathbf{u}$ terms are coupled in the constraints, the control trajectories $\mathbf{u}$ can be updated as follows:

$$\tilde{\mathbf{u}}_k^{(i)} = \mathbf{u}_k^{(i)} \odot \boldsymbol{\beta} \quad \forall (k,i) \tag{11}$$

where $\tilde{\mathbf{u}}$ denotes the final, projected control values. For the terms where $\boldsymbol{\beta}$ is zero, the $\mathbf{u}$ terms will be zero for all time steps and instances. The projection always makes $\tilde{\mathbf{u}} \leq \mathbf{u}$; however, zero must be included in the feasible control range. After projecting the control values, the $\boldsymbol{\beta}$ terms can be projected onto the binary constraint through the following operation:

$$\tilde{\beta}[m] = \begin{cases} 0, & \beta[m] = 0 \\ 1, & \beta[m] > 0 \end{cases} \qquad \forall\, 1:M \tag{12}$$

where $\tilde{\boldsymbol{\beta}}$ denotes the final binary actuator-selector vector. Due to floating point numerics, a small threshold (Eg. 1e-8) should be used to apply the final $\boldsymbol{\beta}$ projection.

6 Experiments

6.1 Optimization and Implementation

Solvers CAPO is implemented in Julia using the IPOPT solver [39]. IPOPT is an interior-point method that can efficiently solve nonlinear programs with a large number of decision variables and constraints.

To provide a basis for comparison, a bi-level genetic algorithm inspired by [6] was implemented using the Evolutionary.jl package [38]. The genetic algorithm uses an inverse roulette selection method, a uniform crossover with a crossover rate of 0.8, and a flip mutation with a mutation rate of 0.1. The population size is 10, and the minimum elitism is 10%. To evaluate each sample in the population, an NLP is employed using IPOPT to optimize only the continuous variables. For efficiency, the samples are evaluated in parallel using the Distributed.jl package in Julia [4].

The MIQP formulation, similar to the approach described in [9], is implemented and solved using Gurobi [12]. Linearized dynamics replace the nonlinear dynamics constraint described in Equation (10b) if possible. A feasibility tolerance of 1e-3 and a MIPGap of 90% is used as termination criteria for the solver; these values were found to achieve the best performance after trial and error.

The MINLP formulation is implemented as described in Equation 6a, and solved using Juniper [18], where IPOPT is used as NLP solver and HiGHS as MIP solver [16, 39].

Hyperparameters All experiments were run on an AMD EPYC 7502 32-Core processor. Five initial conditions were sampled from a normal distribution $\mathcal{N}(\bar{x}_0, 0.001)$. Each method was run five times on each problem with perturbed initial guesses. The following objective function was used,

$$\sum_{i=1}^{I}\sum_{k=2}^{N-1}\left((\mathbf{x}_k^{(i)}-\bar{\mathbf{x}}_k)^T Q(\mathbf{x}_k^{(i)}-\bar{\mathbf{x}}_k)+(\mathbf{u}_k^{(i)}-\bar{\mathbf{u}}_k)^T R(\mathbf{u}_k^{(i)}-\bar{\mathbf{u}}_k)\right)+\alpha\,|\boldsymbol{\beta}|_1 \tag{13}$$

where Q=10 I, R=1e3 I and α=9e6 for the acrobatic multirotor problem and Q=10 I, R=I and α=10 for all other problems. $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ are reference state and control trajectories, respectively.

Dynamics A simplified maximal-coordinate dynamics formulation was used for each problem representing the robot as a series of rigid bodies connected by springs and dampers whose stiffness can be tuned to represent various soft joints. The dynamics are described by:

$$\mathbf{x}=\begin{bmatrix}\mathbf{x}_1\\ \vdots\\ \mathbf{x}_n\end{bmatrix},\ \mathbf{x}_i=\begin{bmatrix}\mathbf{p}_i\\ \mathbf{q}_i\\ \mathbf{v}_i\\ \boldsymbol{\omega}_i\end{bmatrix},\ \dot{\mathbf{x}}=\begin{bmatrix}\dot{\mathbf{x}}_1\\ \vdots\\ \dot{\mathbf{x}}_n\end{bmatrix},\ \dot{\mathbf{x}}_i=\begin{bmatrix}\mathbf{v}_i\\ \frac{1}{2}\mathbf{q}_i\otimes\boldsymbol{\omega}_i\\ M_i^{-1}\cdot(\mathbf{f}_{\text{ext},i}+\mathbf{f}_{\text{int},i})\\ J_i^{-1}\cdot(\boldsymbol{\tau}_{\text{ext},i}+\boldsymbol{\tau}_{\text{int},i}-\boldsymbol{\omega}_i\times J_i\cdot\boldsymbol{\omega}_i)\end{bmatrix} \tag{14}$$

$$\mathbf{f}_{\text{ext},i}=\mathbf{u}_{f,i}+M_i\cdot\mathbf{g},\quad \mathbf{f}_{\text{int},i}=-K_x\Delta\mathbf{p}_{ij}-C_x\Delta\mathbf{v}_{ij} \tag{15}$$

$$\boldsymbol{\tau}_{\text{ext},i}=\mathbf{u}_{\tau,i}+\mathbf{p}_i\times\mathbf{q}_i\otimes(\mathbf{f}_{\text{ext},i}+\mathbf{f}_{\text{int},i}),\quad \boldsymbol{\tau}_{\text{int},i}=-K_t\Delta\boldsymbol{\phi}_{ij}-C_t\Delta\boldsymbol{\omega}_{ij} \tag{16}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ represents the position vector, $\mathbf{q}_i \in \mathbb{R}^4$ is the quaternion representing orientation, $\mathbf{v}_i \in \mathbb{R}^3$ is the linear velocity, and $\boldsymbol{\omega}_i \in \mathbb{R}^3$ is the angular velocity. $\mathbf{u}_{f,i}$ and $\mathbf{u}_{\tau,i}$ represent the control inputs that exert forces and torques on the body, respectively. $\mathbf{K}_x, \mathbf{C}_x, \mathbf{K}_t,$ and$\mathbf{C}_t$ are the spring and damper matrices and $\Delta\mathbf{p}_{ij}, \Delta\boldsymbol{\phi}_{ij}, \Delta\mathbf{v}_{ij},$ and$\Delta\boldsymbol{\omega}_{ij}$ are the position, orientation and velocity differences between two connected bodies. The mass matrix M_i and the inertia matrix J_i define the physical properties of the body, respectively. This formulation assumes the body's mass is much greater than the mass of the actuators and ignores the effects of adding or removing actuators to the mass and inertial matrices. Additionally, no complex aerodynamic or fluid effects, such as drag, are considered.

CAPO Formulation The choice of actuator type and location in design problems varies by application: 1-DoF vertical propellers for multirotor, 1-DoF thrusters and reaction wheels at the center of mass (COM) for space structures, and 1-DoF forces and torques at COM for cloth and soft robots. The full CAPO formulation for the experiments are as follows:

$$\begin{aligned} \underset{\mathbf{x}_{2:N}^{(1:5)}, \mathbf{u}_{2:N-1}^{(1:5)}, \boldsymbol{\beta}, \mathbf{r}}{\text{minimize}} \quad & (13) \\ \text{subject to} \quad & \mathbf{x}_k^{(i)} + dt * f(\frac{\mathbf{x}_k^{(i)} + \mathbf{x}_{k+1}^{(i)}}{2}, \boldsymbol{\beta} \odot \mathbf{u}_k^{(i)}, \mathbf{r}) = \mathbf{x}_{k+1}^{(i)}, \\ & \mathbf{x}_N^{(i)} = \bar{\mathbf{x}}_N, \\ & \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}, \\ & \mathbf{r}_{min} \leq \mathbf{r} \leq \mathbf{r}_{max}, \\ & \mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1} \end{aligned} \tag{17}$$

where the dynamics constraint is specified using an implicit midpoint integration. In the latter three cases, position variables are not considered due to the assumption of full actuation at the COM. However, these assumptions can be modified for future research or different problems.

6.2 Scalability

The scalability of the four methods is compared empirically using an over-actuated double integrator problem. The number of control inputs is swept from $m = 2:200$ for GA and MINLP and $m = 2:300$ control inputs for CAPO and MIQP to increase the difference resolution between these two methods. The CAPO formulation for this problem can be written as follows:

$$\begin{aligned} \min_{\mathbf{x}_{2:10}, \mathbf{u}_{2:9}, \boldsymbol{\beta}} \quad & (13) \\ \text{s.t.} \quad & \mathbf{x}_k + dt(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}_{2x2} \frac{\mathbf{x}_k + \mathbf{x}_{k+1}}{2} + \begin{bmatrix} 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{bmatrix}_{2xm} (\beta \odot \mathbf{u}_k)) = \mathbf{x}_{k+1}, \\ & \mathbf{x}_{10} = \bar{\mathbf{x}}_{10}, \\ & \mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1}, \\ & Q = 10.0 * I_{2x2},\ R = 1.0 * I_{2xm},\ \alpha = 10.0 \end{aligned} \tag{18}$$

where $\mathbf{x}_k$ is a vector with position and velocity, $\mathbf{u}_k$ is a vector with m acceleration values, $\boldsymbol{\beta}$ is a binary vector with m values. An implicit midpoint integration scheme is used for the dynamics constraint. For this toy example, the optimal solution is known to be a single control input, and all models found this solution for all instances of m. The runtime for each solver can be seen in Fig. 2.

CAPO achieves a 257% improvement over MIQP at 300 control inputs. This is due to the many binary variables in the MIQP formulation and the corresponding large branching factor. On the other hand, the binary relaxation enables CAPO to maintain favorable performance and optimal solutions as the number of actuators increases. While CAPO only shows moderate improvement on the high dimensional problems for this example with linear dynamics, larger improvements are expected on nonlinear problems since Gurobi cannot handle these systems easily.

The GA performs poorly in the proposed toy example due to the sparsity of the solution, which can cause the GA to search for a long time before satisfying the exit criteria. Additionally, the overhead for parallelization on this problem, which consists of few parameters, impedes its performance. The plateau seen in the results from the GA is due to a time-limit exit condition and not due to improved efficiency on larger problems.

The MINLP algorithm performs poorly because it must solve many consecutive NLP problems. Given knowledge about the problem, tuning the exit conditions could also improve the MINLP's performance.

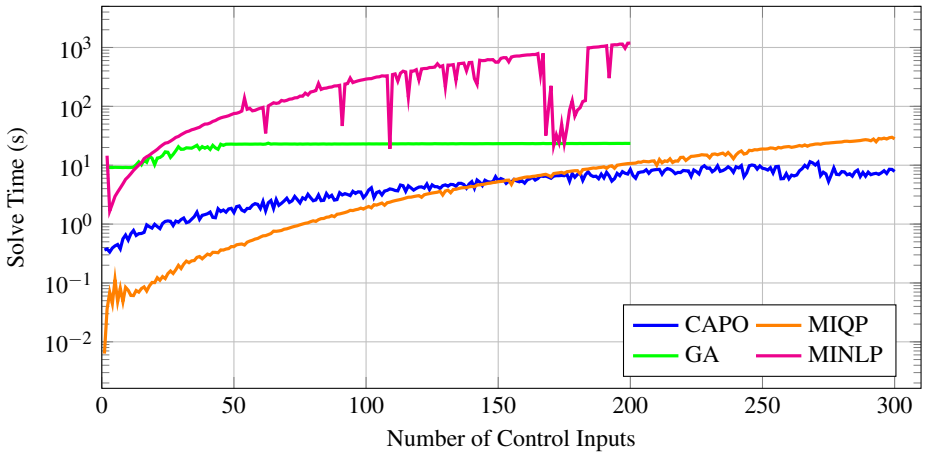

Fig. 2. Log plot of runtime scaling comparison for CAPO, GA, MIQP, and MINLP on an over-actuated double-integrator problem with 2-200 control inputs and up to 300 control inputs for CAPO and MIQP, highlighting CAPO's superior scaling with large binary parameters reaching 257% improvement at 300. CAPO is significantly faster than MINLP and GA methods at all scales and outperforms the MIQP solver as binary variables exceed 150. However, CAPO can also handle nonlinear constraints and objectives natively, which the MIQP does not always support.

6.3 Benchmark Problems

Multirotor Flip The first task aims to design a multirotor to perform an acrobatic flip maneuver, seen in Fig. 3a. The multirotor is modeled as a single rigid body with the

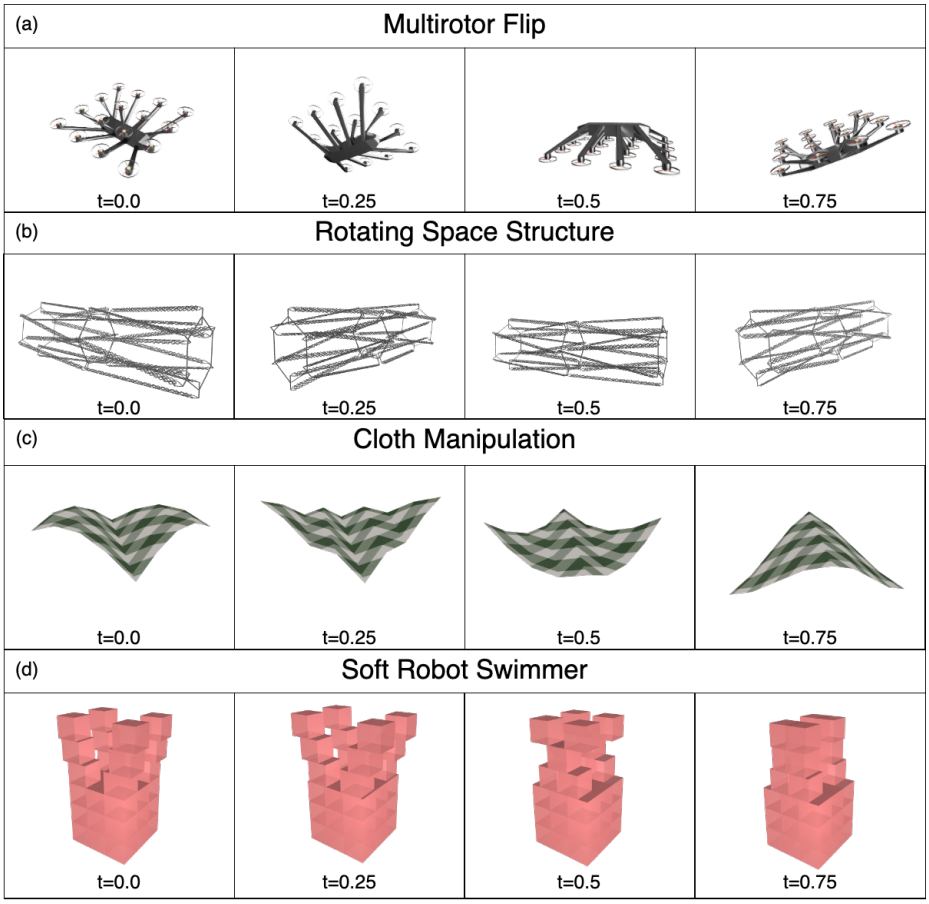

Fig. 3. Reference trajectories for (a) multirotor flip (6 DoF) with a total of 16 thrusters to be selected and placed on the body, (b) rotating space structure (12 DoF) with a total of 12 actuators (6 1-D thrusters, 6 1-D reaction wheels) to be selected and placed on the structure, (c) cloth manipulation with 25 nodes in a 5x5 grid (150 DoF) with a potential for 150 control points at each of the nodes, consisting of potential 1-D forces or moments applied to the cloth nodes, and (d) soft robotic swimmer with 39 discrete actuatable body units (234 DoF) with a total of 234 actuators to be placed on the discrete bodies to apply forces or moments at that body unit.

quadrotor dynamics are described by (14). The multirotor is initialized with 16 propellers placed in a uniform grid around the multirotor's body. Each actuator is confined to a predefined control volume so that actuators cannot overlap, and the controls are limited to positive thrust, $\mathbf{u}_{min} = \mathbf{0}$. The reference trajectory was designed by hand, consisting of 50-knot points, leading to 7218 decision variables. The results are reported in Table 2 for CAPO, MINLP, and GA. The MIQP was excluded from this experiment due to the highly nonlinear dynamics, which do not linearize well.

CAPO found solutions to this problem with a 2.55X speed increase. CAPO finds better solutions than GA and performs similarly to MINLP's. Due to poor initialization and the nonconvexity of this problem, CAPO failed to satisfy the task twice. The

MINLP approach is less susceptible to these issues as it exhaustively searches through branching and finds successful solutions in all cases. This problem sees the greatest decrease in actuators due to the robot's small number of DoF and a high number of initial actuators.

Space Structure The second task is stabilizing a multi-body flexible structure rotating at 1 Hz in space. The dynamics of this problem are described by the following (14). This task is challenging due to lightly damped structural vibration modes. The space structure, seen in Fig. 3b, is described by 2 bodies connected in series by springs and dampers, resulting in a 12 DoF system. This problem is initialized with 3 thrusters and 3 reaction wheels placed at the COM of each body, making the system fully controllable. The CAPO formulation is similar to that stated in (17), excluding the thrust and position limits. This small-scale problem still has around 1852 decision variables; the results are reported in Table 2.

In this problem, CAPO is 19x faster than the MIQP, 43x faster than the GA, and 65x faster than the MINLP. The MINLP and GA find the smallest objective values; however, CAPO finds a solution that is very close to the other methods and outperforms the MIQP. The objective function balances minimizing the reference trajectory, actuator effort, and the number of actuators, and the problem formulation is non-convex due to the coupled dynamics constraint, causing local minima in the solution landscape. Therefore, although CAPO's solution had an average actuator reduction of 10% versus the GA, MIQP, and MINLP solutions, which reduced the number of actuators by 25%, this only accounts for a difference of 1-2 actuators.

Cloth Manipulation The cloth manipulation task is to track a reference flapping motion seen in Fig. 3c. This problem can be related to coordinating robotic arms in manipulating deformable objects like shirts or towels. The cloth dynamics are approximated using the same dynamics described in (14), consisting of a series of rigid bodies connected by springs and dampers representing a soft spherical joint. In the small-scale problem, the cloth is modeled as 4 rigid bodies in a 2x2 grid. The CAPO formulation is similar to that stated in (17), excluding the thrust and position limits. This problem consists of 3704 decision variables, and the results are reported in Table 2.

CAPO is 7x faster on average than the MIQP formulation, 30x faster on average than the GA, and 218x faster on average than the MINLP. This time, CAPO finds the same objective value of the GA, and MINLP formulations and is better than the MIQP. CAPO and GA find solutions that reduce the number of actuators by 16.7%, whereas the MIQP finds a solution that reduces the number of actuators by 25%. MINLP finds a solution that reduces the number of actuators by 37.5%. One possible area of discrepancy with the MIQP formulation is due to the linearization of the dynamics constraints, while the branch and bound approach enables the MINLP to get out of local minima given sufficient time. However, it is apparent that, compared to GA, MIQP, and MINLP, CAPO provides a comparable solution in significantly less time.

6.4 Large-Scale Problems

CAPO is demonstrated on a 120 DoF flexible space structure, a 150 DoF cloth manipulation, and a 234 DoF soft robotic swimmer which the other methods failed to find a constraint-satisfying solution within a predefined time limit of 5 hrs. These problems

use the same dynamics described in (14) and leverage the same CAPO formulation shown in (17) excluding the thrust and actuator position limits. Each problem is initialized with 6 actuators placed at the COM of each body approximating the structure, 3 1-DOF force-generating actuators, and 3 1-DOF torque-generating actuators. These large-scale problems have 18520, 21900, and 36114 decision variables, respectively. The space structure and cloth manipulation problems were revisited using 20 bodies and 25 bodies in a 5x5 grid, respectively. The results for CAPO on these two large-scale tasks are reported in Table 2. The MIQP and MINLP failed to find a feasible solution for these larger problems, and the GA failed due to memory limits. In the large space structure task, CAPO reduces the total number of actuators by 14% and find the solution in an average time of 423 seconds (7.05 min). For the cloth manipulation, CAPO reduces the total number of actuators by an average of 27.5% and solves in an average time of 878 seconds (14.6 min). The actuator configurations are illustrated in Fig. 4, where force actuators are represented by cones and torque actuators are represented by toruses.

A soft-robotic swimmer task, which aims to achieve a reference swimming motion seen in Fig. 3 was also solved. Soft swimmers have been an active area of research for design optimization [22, 27, 29]. The soft-swimmer model consists of 39 bodies, 27 bodies describing the body, and 12 bodies defining the 4 flippers, resulting in a 234 DoF model. The results seen in Table 2 show that, for the soft swimmer problem, CAPO can reduce the total number of actuators by an average of 12% and finds a solution in an average time of 8536 seconds (2.37 hrs). The final actuator configuration is shown in Fig. 4. For these large-scale problems, CAPO can find reduced actuator configurations that satisfy the problem constraints. This demonstration shows that CAPO scales well even as the number of parameters increases significantly. The other methods either fail to find a solution or require large computational resources that exceed the testing setup.

7 Conclusions

Tools from trajectory optimization are powerful and capable of solving high-dimensional nonlinear problems. Sparse actuator configurations can be found when considering the design problem in these frameworks. Leveraging this, CAPO provides an efficient algorithm to address task-driven robot design by solving high-dimensional control-and-actuator-placement problems for complex, nonlinear systems across many applications. This is achieved by reformulating NP-hard computationally expensive MINLP for control-and-actuator-placement problems into a single-level NLP that can be solved 7-218 times faster than GA, MIQP, and MINLP approaches. There is still much room for future work: One limitation of CAPO's current formulation is its assumption that the system properties, such as mass and inertia, stay constant. This assumption holds for the space structure and cloth manipulation examples, but likely wouldn't hold for the multirotor problem and other important robotic platforms. Another important limitation of CAPO is its inherent non-convexity. This means that the feasible solutions found by CAPO are only local minima at best, and it is generally impossible to know whether a global minimum has been found. Future research will investigate more rigorous guarantees for CAPO, further validating the algorithm's reliability and robustness for real-world applications. Finally, efficiency gains can be harnessed by further leveraging the sparsity structure of the CAPO problem in customized solver algorithms.

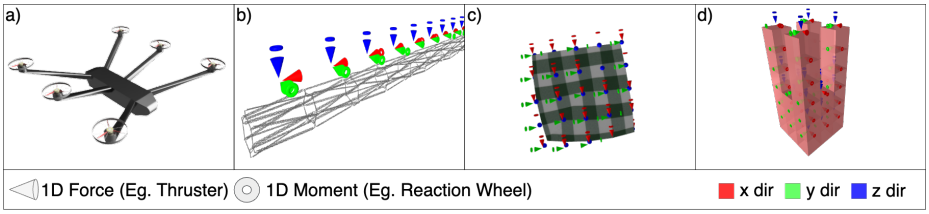

Fig. 4. Final actuator configurations for various optimization tasks a) the multirotor flip with a final configuration of 6 actuators placed at the extents of the body, b) 120 DoF flexible space structure with around 100 actuators placed along the body with most of the roll reaction wheels pruned, c) 150 DoF cloth manipulation with 105 force and moments applied to the various cloth nodes to achieve the desired trajectory, and d) 234 DoF soft robotic swimmer with 205 actuators along the discrete body units to enable the swimming motion. Cones represent 1-D force actuators, e.g. thruster, and toruses represent moment actuators, e.g. reaction wheel. Red represents the force/moment applied to the X-axis, Green represents the Y-axis, and Blue represents the Z-axis.

Table 2. This table presents the results from CAPO, GA, MIQP, and MINLP on 3 low dimensional optimization tasks and 3 high dimensional optimization tasks. The table presents each method's average objective value, constraint violation, runtime, percent actuator reduction, and trial success across 5 trials. The standard deviation is presented below in parentheses. The algorithm with the best performance in a particular category are highlighted in **Bold**. CAPO was the only method that successfully solved the high-dimensional tasks.

Low Dim. Tasks	AVG (STD)	CAPO (ours)	GA	MIQP	MINLP	High Dim. Tasks	AVG (STD)	CAPO (ours)	GA	MIQP	MINLP
Multi-rotor (6 DoF)	Obj. Value	1.07e8 (7.3e5)	1.54e8 (-)	-	**1.06e8 (2.5e5)**	Space Struct. (120 DoF)	Obj. Value	**1.47e7 (31.3)**	-	-	-
	Const. Viol.	1.96e-8 (1.8e-8)	2.60e-9 (-)	-	6.37e-9 (9.8e-9)		Const. Viol.	**1.33e-14 (0.0)**	-	-	-
	Runtime [sec]	**417.14 (116.27)**	2996.14 (-)	-	1065.59 (514.56)		Runtime [sec]	**423.0 (66.7)**	-	-	-
	Actuator Reduct.	65% (3.6%)	12% (-)	-	**66.25% (3.4%)**		Actuator Reduct.	**14% (0.3%)**	-	-	-
	Trial Success	3/5	1/5	0/5	**5/5**		Trial Success	**5/5**	0/5	0/5	0/5
Space Struct. (12 DoF)	Obj. Value	2.710e4 (0.056)	2.708e4 (0.0	1.620e5 (0.0)	**2.708e4 (0.0)**	Cloth Manip. (150 DoF)	Obj. Value	**9.61e5 (13.2)**	-	-	-
	Const. Viol.	3.11e-14 (0.0)	1.03e-13 (0.0)	0.001 (0.0)	8.88e-16 (0.0)		Const. Viol.	**9.7e-13 (0.0)**	-	-	-
	Runtime [sec]	**4.059 (1.2)**	174.61 (31.23)	76.27 (6.55)	266.5 (133.10)		Runtime [sec]	**878.0 (111.7)**	-	-	-
	Actuator Reduct.	10% (3.7%)	**25% (0.0%)**	**25% (0.0%)**	**25% (0.0%)**		Actuator Reduct.	**27.5% (0.6%)**	-	-	-
	Trial Success	5/5	5/5	5/5	5/5		Trial Success	**4/5**	0/5	0/5	0/5
Cloth Manip. (24 DoF)	Obj. Value	**1.234e5 (0.008)**	**1.234e5 (0.0)**	1.249e5 (0.0)	**1.234e5 (0.0)**	Soft Robot Swim. (234 DoF)	Obj. Value	**9.0e3 (46.6)**	-	-	-
	Const. Viol.	4.88e-13 (0.0)	2.54e-14 (0.0)	0.0026 (0.0)	1.2e-9 (1.6e-09)		Const. Viol.	**8.54e-13 (0.0)**	-	-	-
	Runtime [sec]	**15.4 (2.32)**	473.1 (1.8)	108.12 (1.81)	3370.0 (940.0)		Runtime [sec]	**8536.1 (5764.55)**	-	-	-
	Actuator Reduct.	16.7% (0.0%)	16.7% (0.0%)	25% (0.0%)	**37.5% (0.0%)**		Actuator Reduct.	**12% (2%)**	-	-	-
	Trial Success	5/5	5/5	5/5	5/5		Trial Success	**5/5**	0/5	0/5	0/5

Acknowledgments. This work was supported by a NIAC award from NASA's Space Technology Mission Directorate (Grant Number 80NSSC21K0446). Tools like ChatGPT and Grammarly were used for general editing and grammar enhancement.

Disclosure of Interests. The authors have filed for a provisional patent (docket no. 2024-220) on the method described in this paper.

References

1. Leaps, Bounds, and Backflips, https://bostondynamics.com/blog/leaps-bounds-and-backflips/
2. A. Kuntz, C. Bowen, C. Baykal, A. W. Mahoney, P. L. Anderson, F. Maldonado, R. J. Webster, R. Alterovitz: Kinematic Design Optimization of a Parallel Surgical Robot to Maximize Anatomical Visibility via Motion Planning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 926–933 (May 2018). https://doi.org/10.1109/ICRA.2018.8461135, journal Abbreviation: 2018 IEEE International Conference on Robotics and Automation (ICRA)
3. Baykal, C., Bowen, C., Alterovitz, R.: Asymptotically optimal kinematic design of robots using motion planning. Autonomous Robots **43**(2), 345–357 (Feb 2019). https://doi.org/10.1007/s10514-018-9766-x, https://doi.org/10.1007/s10514-018-9766-x
4. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM review **59**(1), 65–98 (2017), https://doi.org/10.1137/141000671
5. Bhatia, J.S., Jackson, H., Tian, Y., Xu, J., Matusik, W.: Evolution gym: A large-scale benchmark for evolving soft robots. Advances in Neural Information Processing Systems **34**, 2201–2214 (12 2021), http://evogym.csail.mit.edu.
6. Bruant, I., Gallimard, L., Nikoukar, S.: Optimal piezoelectric actuator and sensor location for active vibration control, using genetic algorithm. Journal of Sound and Vibration **329**(10), 1615–1635 (2010). https://doi.org/https://doi.org/10.1016/j.jsv.2009.12.001, https://www.sciencedirect.com/science/article/pii/S0022460X09009869
7. C. Bergeles, A. H. Gosline, N. V. Vasilyev, P. J. Codd, P. J. del Nido, P. E. Dupont: Concentric Tube Robot Design and Optimization Based on Task and Anatomical Constraints. IEEE Transactions on Robotics **31**(1), 67–84 (Feb 2015). https://doi.org/10.1109/TRO.2014.2378431
8. Candès, E.: Mathematics of sparsity (and a few other things). pp. 235–258 (2014)
9. Chanekar, P.V., Chopra, N., Azarm, S.: Optimal actuator placement for linear systems with limited number of actuators. Proceedings of the American Control Conference pp. 334–339 (6 2017). https://doi.org/10.23919/ACC.2017.7962975
10. Deenen, D.A., Sebeke, L.C., de Jager, B., Heijman, E., Grüll, H., Heemels, W.P.M.H.: Target-conformal optimization-based actuator placement for ultrasound-mediated hyperthermia in cancer treatments. IEEE Transactions on Control Systems Technology **31**(4), 1926–1933 (2023). https://doi.org/10.1109/TCST.2023.3245336
11. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. The Annals of Statistics **32**(2), 407 – 499 (2004). https://doi.org/10.1214/009053604000000067
12. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), https://www.gurobi.com
13. Ha, S., Coros, S., Alspach, A., Bern, J.M., Kim, J., Yamane, K.: Computational design of robotic devices from high-level motion specifications. IEEE Transactions on Robotics **34**(5), 1240–1251 (2018). https://doi.org/10.1109/TRO.2018.2830419

14. Hargraves, C., Paris, S.: Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics **10**(4), 338–342 (1987). https://doi.org/10.2514/3.20223
15. Hu, Q., Zhang, J.: Placement optimization of actuators and sensors for gyroelastic body. Advances in Mechanical Engineering **7**(3), 1687814015573765 (Mar 2015). https://doi.org/10.1177/1687814015573765, https://doi.org/10.1177/1687814015573765, publisher: SAGE Publications
16. Huangfu, Q., Hall, J.A.J.: Parallelizing the dual revised simplex method. Mathematical Programming Computation **10**(1), 119–142 (Mar 2018). https://doi.org/10.1007/s12532-017-0130-5
17. J. -T. Lin, C. Girerd, J. Yan, J. T. Hwang, T. K. Morimoto: A Generalized Framework for Concentric Tube Robot Design Using Gradient-Based Optimization. IEEE Transactions on Robotics **38**(6), 3774–3791 (Dec 2022). https://doi.org/10.1109/TRO.2022.3180627
18. Kröger, O., Coffrin, C., Hijazi, H., Nagarajan, H.: Juniper: An open-source nonlinear branch-and-bound solver in julia. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 377–386. Springer International Publishing, Cham (2018), https://doi.org/10.1007/978-3-319-93031-2_27
19. Kronqvist, J., Bernal, D.E., Lundell, A., Grossmann, I.E.: A review and comparison of solvers for convex MINLP. Optimization and Engineering **20**(2), 397–455 (Jun 2019). https://doi.org/10.1007/s11081-018-9411-8
20. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability pp. 481–492 (1950)
21. Land, A.H., Doig, A.G.: An Automatic Method of Solving Discrete Programming Problems. Econometrica **28**(3), 497 (Jul 1960). https://doi.org/10.2307/1910129, https://www.jstor.org/stable/1910129?origin=crossref
22. Lee, J.H., Michelis, M.Y., Katzschmann, R., Manchester, Z.: Aquarium: A fully differentiable fluid-structure interaction solver for robotics applications (2023), https://doi.org/10.48550/arXiv.2301.07028
23. M. Borairi, M. Soufian: Optimal actuator\sensor placement and controller design for large flexible space structures and robotics. In: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE). pp. 1398–1403 (Jun 2017). https://doi.org/10.1109/ISIE.2017.8001450, journal Abbreviation: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)
24. Molter, A., da Silveira, O.A.A., Fonseca, J.S.O., Bottega, V.: Simultaneous Piezoelectric Actuator and Sensor Placement Optimization and Control Design of Manipulators with Flexible Links Using SDRE Method. Mathematical Problems in Engineering **2010**(1), 362437 (Jan 2010). https://doi.org/10.1155/2010/362437, https://doi.org/10.1155/2010/362437, publisher: John Wiley & Sons, Ltd
25. Morimoto, T.K., Greer, J.D., Hawkes, E.W., Hsieh, M.H., Okamura, A.M.: Toward the Design of Personalized Continuum Surgical Robots. Annals of Biomedical Engineering **46**(10), 1522–1533 (Oct 2018). https://doi.org/10.1007/s10439-018-2062-2, https://doi.org/10.1007/s10439-018-2062-2
26. Nagarajan, H., Lu, M., Wang, S., Bent, R., Sundar, K.: An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. Journal of Global Optimization (2019). https://doi.org/10.1007/s10898-018-00734-1
27. Nava, E., Zhang, J.Z., Michelis, M.Y., Du, T., Ma, P., Grewe, B.F., Matusik, W., Katzschmann, R.K.: Fast aquatic swimmer optimization with differentiable projective dynamics and neural network hydrodynamic models. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 16413–16427. PMLR (17–23 Jul 2022), https://proceedings.mlr.press/v162/nava22a.html

28. Olshevsky, A.: Minimal controllability problems. IEEE Transactions on Control of Network Systems **1**(3), 249–258 (2014). https://doi.org/10.1109/TCNS.2014.2337974
29. Patel, D.K., Huang, X., Luo, Y., Mungekar, M., Jawed, M.K., Yao, L., Majidi, C.: Highly dynamic bistable soft actuator for reconfigurable multimodal soft robots. Advanced Materials Technologies **8**(2), 2201259 (2023). https://doi.org/https://doi.org/10.1002/admt.202201259, https://onlinelibrary.wiley.com/doi/abs/10.1002/admt.202201259
30. Rao, S.S., Pan, T.S., Venkayya, V.B.: Optimal placement of actuators in actively controlled structures using genetic algorithms. AIAA journal **29**, 942–943 (5 1991). https://doi.org/10.2514/3.10683, https://arc.aiaa.org/doi/10.2514/3.10683
31. S. Niyaz, A. Kuntz, O. Salzman, R. Alterovitz, S. S. Srinivasa: optimizing Motion-Planning Problem Setup via Bounded Evaluation with Application to Following Surgical Trajectories. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1355–1362 (Nov 2019). https://doi.org/10.1109/IROS40897.2019.8968575, journal Abbreviation: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
32. Santosa, F., Symes, W.W.: Linear inversion of band-limited reflection seismograms. SIAM Journal on Scientific and Statistical Computing **7**(4), 1307–1330 (1986). https://doi.org/10.1137/0907087
33. Seifried, R.: Two approaches for feedforward control and optimal design of underactuated multibody systems. Multibody System Dynamics **27**, 75–93 (1 2012). https://doi.org/10.1007/S11044-011-9261-Z/METRICS, https://link.springer.com/article/10.1007/s11044-011-9261-z
34. Skouras, M., Thomaszewski, B., Coros, S., Bickel, B., Gross, M.: Computational design of actuated deformable characters. ACM Transactions on Graphics (TOG) **32** (7 2013). https://doi.org/10.1145/2461912.2461979, https://dl.acm.org/doi/10.1145/2461912.2461979
35. Spielberg, A., Araki, B., Sung, C., Tedrake, R., Rus, D.: Functional co-optimization of articulated robots. Proceedings - IEEE International Conference on Robotics and Automation pp. 5035–5042 (7 2017). https://doi.org/10.1109/ICRA.2017.7989587
36. T. Anor, J. R. Madsen, P. Dupont: Algorithms for design of continuum robots using the concentric tubes approach: A neurosurgical example. In: 2011 IEEE International Conference on Robotics and Automation. pp. 667–673 (May 2011). https://doi.org/10.1109/ICRA.2011.5980311, journal Abbreviation: 2011 IEEE International Conference on Robotics and Automation
37. Tibshirani, R.: Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (12 2018). https://doi.org/10.1111/j.2517-6161.1996.tb02080.x
38. wildart: Evolutionary.jl. https://github.com/wildart/Evolutionary.jl (2022)
39. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**, 25–57 (5 2006). https://doi.org/10.1007/S10107-004-0559-Y/METRICS, https://link.springer.com/article/10.1007/s10107-004-0559-y
40. Yuan, G., Ghanem, B.: Binary optimization via mathematical programming with equilibrium constraints. arXiv: Optimization and Control (2016), https://api.semanticscholar.org/CorpusID:54069864
41. Zhao, A., Xu, J., Konaković-Luković, M., Hughes, J., Spielberg, A., Rus, D., Matusik, W.: Robogrammar: Graph grammar for terrain-optimized robot design. ACM Trans. Graph **39**, 16 (2020). https://doi.org/10.1145/3414685.3417831, https://doi.org/10.1145/3414685.3417831
42. Zou, H., Hastie, T.: Regularization and Variable Selection Via the Elastic Net. Journal of the Royal Statistical Society Series B: Statistical Methodology **67**(2), 301–320 (03 2005). https://doi.org/10.1111/j.1467-9868.2005.00503.x