2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Learning Mixed-Integer Convex Optimization Strategies for Robot Planning and Control

Abhishek Cauligi*, Preston Culbertson*, Bartolomeo Stellato,
Dimitris Bertsimas, Mac Schwager, and Marco Pavone

***Abstract*—Mixed-integer convex programming (MICP) has seen significant algorithmic and hardware improvements with several orders of magnitude solve time speedups compared to 25 years ago. Despite these advances, MICP has been rarely applied to real-world robotic control because the solution times are still too slow for online applications. In this work, we present the CoCo (Combinatorial Offline, Convex Online) framework to solve MICPs arising in robotics at very high speed. CoCo encodes the combinatorial part of the optimal solution into a *strategy*. Using data collected from offline problem solutions, we train a multiclass classifier to predict the optimal *strategy* given problem-specific parameters such as states or obstacles. Compared to [1], we use task-specific strategies and prune redundant ones to significantly reduce the number of classes the predictor has to select from, thereby greatly improving scalability. Given the predicted strategy, the control task becomes a small convex optimization problem that we can solve in milliseconds. Numerical experiments on a cart-pole system with walls, a free-flying space robot, and task-oriented grasps show that our method provides not only 1 to 2 orders of magnitude speedups compared to state-of-the-art solvers but also performance close to the globally optimal MICP solution.**

I. Introduction

Mixed-integer convex programming (MICP) is a powerful technique to model robotic tasks such as task and motion planning [2, 3], planning for systems with contact [4, 5] and dexterous manipulation [6, 7]. By modeling discrete control decisions with integer variables, we can, theoretically, solve MICPs to global optimality with branch-and-bound or outer approximation algorithms [8]. However, computing the optimal solutions is, in practice, computationally challenging.

Despite recent advances [9], MICPs have seen limited application in real-world robotic tasks for the following two reasons:

- *Slow computation times:* it is still challenging to solve MICPs in online settings where real-time computations are crucial. For example, many planning and control tasks require solutions in few milliseconds, while typical solution times range from seconds to minutes [5].
- *Non-embeddable algorithms:* the best off-the-shelf solvers such as Gurobi [10] or Mosek [11] rely on complex multithreaded implementations that are not suitable for embedded robot platforms.

This is why there is still a large gap between modeling and real-world implementation of MICP-based controllers in robotics systems. To fill this gap, in this paper we exploit the idea that by solving a large number of MICP instances one can generate a large amount of offline data that can be purposefully used to significantly accelerate online solutions.

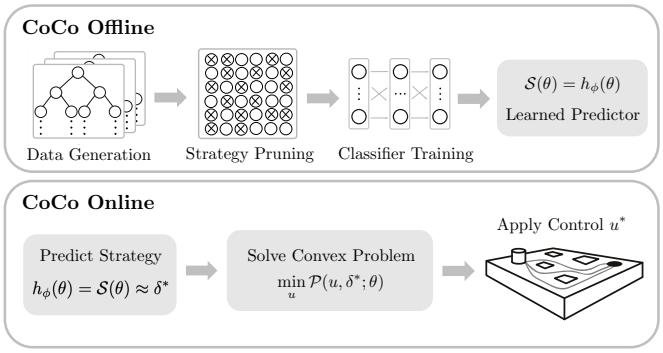


Fig. 1: CoCo framework for a motion planning task with free-flyin g space robots. A strategy classifier is trained offline on a dataset of problems. We added a block to simplify the number of strategies to select. Online, we use the strategy classifier to predict optimal trajectories given a new task specification for the MICP.

A. Statement of Contributions

To leverage offline data to accelerate online solutions, in this paper we introduce Combinatorial Offline, Convex Online (CoCo), an approach that builds upon the machine learning optimizer (MLOPT), recently introduced in [1, 12]. In detail, both MLOPT and CoCo consist of offline and online phases that are notionally represented in Figure 1. In the offline phase we collect data by solving a given control problem for several values of the key parameters (e.g., initial state, obstacles, object locations), obtaining the optimal solutions. Given each solution, MLOPT saves the optimal integer assignment, which we term an *integer* strategy, while CoCo saves the optimal *logical* strategy, both of which we define later. After the dataset generation we train a neural network which classifies the best strategy to apply given new problem parameters. Online, the approach consists of a forward neural network strategy prediction from a set of problem parameters, followed by a convex optimization which can be carried out very efficiently [13]. However, MLOPT naïvely applied to robotics problems cannot in general provide reliable time solve times, since the number of unique integer strategies can be too large to obtain high-accuracy multiclass classification.

Here, we demonstrate the efficacy of CoCo, which leverages problem-specific information and structure to solve MICPs

A. Cauligi, M. Schwager, and M. Pavone are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305. {acauligi, schwager, pavone} @stanford.edu.
P. Culbertson is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305. pculbertson@stanford.edu.
B. Stellato is with the Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544. bstellato@princeton.edu.
D. Bertsimas is with the Operations Research Center, MIT, Cambridge, MA 02139. dbertsim@mit.edu.
This work was supported in part by NASA under the NASA Space Technology Research Fellowship Grants NNX16AM78H and 80NSSC18K1180.
*These authors contributed equally to this work.

arising in robotics with improved performance and reliability compared to MLOPT. Our detailed contributions are as follows. First, we propose a methodology to identify unique logical strategies which encompass a set of redundant integer solutions corresponding to the same globally optimal continuous solution. This happens often in robotic settings, for example, when we obtain multiple integer assignments for the same obstacle avoidance path or multi-contact trajectory. Second, we introduce the notion of *task-specific* strategies that allows us to exploit the separable structure of robotics problems, thereby greatly improving scalability. Examples of structured problems include path planning where we can decouple combinatorial decisions obstacle-wise. For these settings, we learn a predictor for a single combinatorial decision, e.g., on which side of the obstacle the robot should pass. Online, we can apply the same predictor for every combinatorial decision we have to make. This idea greatly simplifies the number of unique strategies encountered by focusing on every independent decision. Finally, numerical experiments on a cart-pole system with walls, a free-flying space robot, and task-oriented grasps show that our method achieves strategy prediction accuracy which is at least 90% feasible with more than 90% of solutions being globally optimal. In addition, we obtain computational speedups from 1 to 2 orders of magnitude compared to MICP solvers Gurobi [10] and Mosek [11]. Therefore, the proposed algorithm is suitable to compute MICP solutions in real-time reliably and at very high-speed.

B. Related Work

Recently, there has been a surge of interest in applying data-driven methods in accelerating solution times for optimization-based controllers. In [14, 15], neural networks are used to warm start a solver for a QP-based controller used in a receding horizon fashion. Tang, et al. also consider non-convex optimization-based controllers by learning warm starts for an SQP-based trajectory optimization library [16]. Using tools from differentiable convex optimization, a learning-based approach to tune optimization-based controllers is proposed in [17]. While these methods have shown considerable promise, they only consider the setting of continuous optimization.

Relatively less attention has been paid to using data-driven techniques for accelerating integer program solutions in control. A popular class of methods has been to pose branch-and-bound as a sequential decision making process and apply imitation learning to learn effective variable and node selection strategies [18, 19]. However, these approaches can still be too computationally expensive for applications in control, as they require solving branch-and-bound and multiple forward passes of a neural network online.

Masti and Bemporad [20] use a regression based approach to train a neural network to learn binary variable assignments. However, the authors only demonstrate this approach on an MIQP with 14 binary variables. An approximate dynamic programming approach is presented in [4], where the cost-to-go function for an MIQP controller is learned. However, the approach still requires solving an expensive MIQP using branch-and-bound online. Hogan, et al. [7] propose an approach similar to ours in that they train a classifier to predict mode sequences for a manipulation task, but our formulation is more general in being able to handle a larger class of mixed logical dynamical systems.

II. Technical Background

A. Mixed-Integer Convex Programs

In this work, we focus our attention on a specific class of discrete optimization problems known as parametrized mixed-integer convex programs. The general form of this problem is:

$$\begin{array}{ll} \text{minimize} & f_0(x, \delta; \theta) \\ \text{subject to} & f_i(x, \delta; \theta) \leq 0, \quad i = 1, \ldots, m_f \\ & \delta \in \{0,1\}^{n_\delta}, \end{array} \tag{1}$$

where $x \in \mathbf{R}^{n_x}$ are continuous decision variables, $\delta \in \{0,1\}^{n_\delta}$ are binary decision variables, and $\theta \in \mathbf{R}^{n_p}$ are the problem parameters. The objective function f_0 and inequality constraints f_i are convex. While problem (1) is $\mathcal{NP}$-complete due to the inclusion of discrete decision variables δ [21], it can be solved to global optimality using algorithms such as branch-and-bound, which operates by sequentially building an enumeration tree of relaxed problems to the original problem.

B. Strategies for MICPs

Here, we briefly review the notion of an integer strategy as defined in [1, 12]. For an MICP defined by parameters $\theta \in \mathbf{R}^{n_p}$, an *integer* strategy $\mathcal{I}(\theta)$ consists of a tuple $\left(\delta^*, \mathcal{T}(\theta)\right)$ where (x^*, δ^*) is an optimizer of the problem (1), and $\mathcal{T}(\theta) = \{i \in \{1, \ldots, m_f\} \mid f_i(x^*, \delta^*; \theta) = 0\}$ is the corresponding set of active inequality constraints.

Given an optimal integer strategy $\mathcal{I}(\theta)$, we can obtain an optimal solution for (1) by solving a convex optimization problem,

$$\begin{array}{ll} \text{minimize} & f_0(x, \delta^*; \theta) \\ \text{subject to} & f_i(x, \delta^*; \theta) \leq 0, \;\; i \in \mathcal{T}(\theta), \end{array}$$

which is much easier than the original MICP (1). In fact, if the original problem (1) is a mixed-integer quadratic program (MIQP) or mixed-integer linear program (MILP), solving the reduced problem corresponds to simply solving a set of linear equations defined by the KKT conditions.

This insight motivates the supervised learning problem considered in [1], wherein the authors aim to learn an approximate mapping h_ϕ from problem parameters θ to a corresponding integer strategy $\mathcal{I}(\theta)$. The authors pose this as a multiclass classification problem over a dataset $\mathcal{D} = \{(\theta_i, \mathcal{I}_i)\}_{i=1}^T$ of problem parameters θ_i and their corresponding strategies $\mathcal{I}_i$. Here, the θ_i are sampled from $p(\Theta)$, which is a (known) parameter distribution representative of the problems encountered in practice.

C. Big-M Formulations of Mixed Logical Dynamical Systems

A common modeling choice in MICPs is to use the binary variables δ to capture high-level discrete or logical behavior of the system (e.g., contact, task assignment, hybrid control logic), and to enforce the resulting high-level behavior on the continuous variables x using what is known as big-M formulation [22].

For example, suppose we have a logical variable δ_i, which determines if the constraint $g_i(x; \theta) \leq 0$ on the continuous variables x is active, i.e., we seek to impose the relation

$$[\delta_i = 1] \implies [g_i(x; \theta) \leq 0]\,. \tag{2}$$

To this end, let us define

$$M_i(\theta) = \sup_x g_i(x;\theta),$$

which is simply an upper bound on the constraint function, and may be precomputed. Then, (2) can be enforced by imposing the constraint

$$g_i(x;\theta) \leq M_i(\theta)(1-\delta_i), \tag{3}$$

which we note is linear in δ_i. By inspection, when $\delta_i = 1$, we must have $g_i(x;\theta) \leq 0$, and when $\delta_i = 0$, we have the trivial constraint $g_i(x;\theta) \leq M_i(\theta)$. More complex logical behavior (such as disjunctive constraints) can be achieved using additional logical variables and constraints; for a more thorough treatment, we refer the reader to [22].

More generally, let us denote a "big-M" constraint as

$$g_i(x;\theta) \leq a_i(\theta)\delta_i,$$

where $a_i(\theta)$ are constants which, in general, bounds on the constraint functions to impose the desired logical behavior.

D. Uniqueness of Global Optima

We further note that since MICPs are non-convex, they may admit multiple global optima. Yet, for many physical systems, while the program itself is non-convex, the continuous optimizers x^*, (e.g., the shortest path through an obstacle field) are generally unique. Thus, following [22], we say a program is *well-posed* if it admits a unique continuous minimizer x^*, and we say a program is *completely well-posed* if it admits a unique global minimizer (x^*, δ^*). In this work, we assume the problems considered are well-posed, meaning they may admit multiple discrete optimizers δ^*.

To this end, for a particular solution (x^*, δ^*), we say a big-M constraint is *tight* if

$$g_i(x^*;\theta) \leq a_i(\theta)\delta_i \iff \delta_i = \delta_i^*,$$

meaning that this constraint may only be satisfied for the continuous global minimizer x^* by the values from the particular discrete solution δ^*. For example, let us again consider the implication constraint (2). Suppose for the continuous solution x^*, we have $g_i(x^*;\theta) \leq 0$. Then, since the inequality (3) may be satisfied by either $\delta_i = 0$ or $\delta_i = 1$, the particular solution δ_i^* is not unique; thus, this big-M constraint is not tight.

In this work, we use big-M constraints exclusively to relate the continuous variables x and logical variables δ. If m_M is the number of big-M constraints used, we can write more specifically the class of MICPs studied:

$$\begin{array}{ll} \text{minimize} & f_0(x,\delta;\theta) \\ \text{subject to} & f_i(x;\theta) \leq 0, \quad i = 1,\dots,m_f \\ & g_i(x;\theta) \leq a_i(\theta)\delta_i, \quad i = 1,\dots,m_M \\ & \delta \in \{0,1\}^{n_\delta}, \end{array} \tag{$\mathcal{P}(\theta)$}$$

III. Supervised Learning Strategies for MICPs

In this section we show how to apply CoCo to solve online MICP control tasks through the use of logical strategies and by designing task-specific strategies exploiting the structure of the problem. We finally introduce the complete algorithm and discuss online and offline details.

A. Pruning Redundant Strategies

Although the strategy classification problem presented in [1] provided promising results, it is unable to handle the general class of MICP problems commonly appearing in robotics, and specifically those using integer variables to model mixed logical dynamical systems [22]. A specific pitfall with naïvely using the integer strategies $\mathcal{I}(\theta)$ from [1] occurs when considering a problem which is well-posed (i.e., it has a unique optimal value and continuous optimizer x^*), but admits multiple discrete optimizers $\{\delta^*\}$. In this case, there exist multiple optimal strategies $\{(\delta^*, \mathcal{T}(\theta))\}$, and thus multiple correct labels for the same problem data θ, which makes the supervised learning problem ill-posed. This can occur, for example, when implication constraints (2) are included in the model, or when using logical "OR" constraints between variables.

To ameliorate this issue, we leverage the insight that for our problem $\mathcal{P}(\theta)$, a strategy $\mathcal{S}(\theta)$ can be equivalently, and uniquely, defined by considering only the set of big-M constraints which are tight.

Thus, let us instead define a logical strategy $\mathcal{S}(\theta)$ as a tuple $(\delta^*(\theta), \mathcal{T}_M(\theta))$, where $\delta^*(\theta)$ is a particular integer solution, and $\mathcal{T}_M(\theta)$ is the set of tight big-M constraints,

$$\mathcal{T}_M(\theta) = \{i \mid g_i(x^*;\theta) \leq a_i(\theta)\delta_i \iff \delta_i = \delta_i^*\}. \tag{4}$$

Since each $\delta_i \in \{0,1\}$, this set may be easily computed. Thus, by definition, our classification problem is once again well-posed.

While we could additionally label which continuous constraints are active, as in [1], we note that many common problems in robotics are more general than MIQPs or MILPs, and thus cannot be solved via the KKT conditions. Instead, at runtime, we choose to solve the corresponding convex optimization which results from simply setting $\delta = \delta^*$.

Although this revised notion of strategies for MICPs alleviates the concerns of multiple integer assignments δ^*, in the case of an ill-posed problem, there may also exist multiple optimal continuous solutions x^* which achieve the globally optimal objective. In this work, we neglect such cases, since they are somewhat rare for physical problems, and usually correspond to such pathological cases.

B. Task-Specific Strategies

Extending the notion of the logical strategy $\mathcal{S}(\theta)$ defined in (4), we can refine the definition of a strategy to exploit problem structure commonly found in many robotics problems — separability. In physical planning problems, it frequently occurs that sets of logical variables are used to model repeated, but spatially or temporally distinct, phenomena. For example, in the obstacle avoidance problem outlined in [2], a set of 4 logical variables is used to encode each obstacle; except through the optimal continuous variables, these logical variables are completely decoupled and do not appear together in common constraints. Similar structure can be found in multi-surface contact planning (variable sets per surface), piecewise-affine dynamics (variable sets per dynamics region), and so on.

We formalize this notion by considering problems in which the underlying mixed logical constraints can be represented as a conjunction of Boolean formulas,

$$F_1 \wedge F_2 \wedge \ldots \wedge F_\ell,$$

Algorithm 1 CoCo Offline

Require: Batch of training data $\{\theta_i\}_{i=1,\dots,T}$, problem $\mathcal{P}(\theta)$
Initialize strategy dictionary $\mathcal{S} \leftarrow \{\}$, train set $\mathcal{D} \leftarrow \{\}$
$k \leftarrow 0$
for each θ_i **do**
  Solve $\mathcal{P}(\theta)$
  if $\mathcal{P}(\theta)$ is feasible **then**
    Construct optimal strategy $\mathcal{S}^*$
    if $\mathcal{S}^*$ **not in** $\mathcal{S}$ **then**
      Add $\mathcal{S}^*$ to $\mathcal{S}$, assign class k
      $k \leftarrow k + 1$
    end if
    Assign training label y_i of strategy class S^*
    Add (θ_i, y_i) to $\mathcal{D}$
  end if
end for
Choose network weights ϕ which minimize cross-entropy loss $\mathcal{L}(h_\phi(\theta_i), y_i)_{i=1,\dots,T}$ via stochastic gradient descent
return h_ϕ, $\mathcal{S}$

where F_i is a distinct sub-formula of literals involving continuous and logical variables, and each integer variable δ_j is associated with only one, sub-formula F_i. Again, for illustration, the constraints in [2] can be represented in this form, where each sub-formula F_i encodes that the robot position must lie outside of a single obstacle at a single timestep.

Suppose the mixed logical constraints of MICP $\mathcal{P}(\theta)$ may be expressed in this manner. Then, the strategy $\mathcal{S}(\theta)$ can be decomposed further into sub-formula strategies $\mathcal{S}_1(\theta), \dots, \mathcal{S}_\ell(\theta)$, without loss of generality, since trivially the value of variable δ_j can be determined solely from the logical value of its parent sub-formula F_i and the continuous solution x^*.

The practical advantages of this view are twofold: first, since the number of possible strategies can grow exponentially with the number of integer variables $|\delta|$, decomposing the strategies in this way results in smaller sub-formulas which take on far fewer values individually than the total problem strategy $\mathcal{S}$. Second, many of the individual formulas are themselves repeated (e.g. multiple obstacles, but of various sizes and locations); decomposing the problem strategy into sub-formulas exposes this structure as well. Taken together, these advantages of strategy decomposition greatly help the strategy classifier proposed below, since now the number of possible classes becomes much lower, and a single classifier can be trained for all sub-formulas of identical structure, which in effect augments the training dataset.

We further note that this approach has diminishing returns; since, at runtime, a set of sub-formula strategies $\mathcal{S}_i$ must be recombined to create the total problem strategy $\mathcal{S}$, any machine learning-based algorithm must now produce ℓ correct predictions to reconstruct the globally optimal strategy $\mathcal{S}$. Thus, when choosing how to decompose the strategy, there is a practical trade-off between the number of possible assignments per sub-formula, and the number of total sub-formula assignments which must be predicted overall.

C. Algorithm Overview

We now detail our proposed procedure for training and deploying a strategy classifier. The offline portion of CoCo is described in Algorithm 1. In this offline phase, the optimization problem $\mathcal{P}(\theta)$ is solved for a batch of sample points θ_i sampled from a parameter distribution $p(\Theta)$. We maintain a dictionary $\mathcal{S}$ of logical strategies encountered in the dataset, and label each point θ_i with the index y_i of its strategy class $\mathcal{S}^*$. Finally, we use stochastic gradient descent to choose neural network weights ϕ which approximately minimize a cross-entropy loss over the training set.

Algorithm 2 CoCo Online

Require: Problem parameters θ, strategy dictionary $\mathcal{S}$, trained neural network h_ϕ, n_{evals}
Compute class scores $h_\phi(\theta)$
Identify top n_{evals}-scoring strategies in $\mathcal{S}$
for $j = 1, \dots, n_{\text{evals}}$ **do**
  if $\mathcal{P}(\theta)$ is feasible for strategy $\mathcal{S}^{(j)}$ **then**
    return Feasible solution (x^*, δ^*)
  end if
end for
return failure

As detailed in Algorithm 2, at runtime, we use the classifier h_ϕ and dictionary $\mathcal{S}$ online to solve the MICP given new task parameters. Provided a new task specification θ, we then compute a forward pass $h_\phi(\theta)$ and identify the strategies with the highest n_{evals} scores. For each candidate strategy, we fix the integer variables δ to an optimal assignment δ^* for the strategy class. We then solve the resulting convex problem to find optimal continuous values x^*. If a feasible solution is found, the algorithm terminates.

D. Feasible Solutions

We note that CoCo forgoes the optimality guarantees provided by branch-and-bound for finding the globally optimal solution for an MICP. However, we emphasize that in the context of robotics, controllers are often deployed in a receding horizon fashion wherein the controller is executed multiple times through a task or trajectory. Thus, some suboptimality in relatively rare instances can be tolerated if fast execution enables safe behavior and, in some cases, a well-tuned terminal cost can also account for long horizon behavior. Moreover, as we show next, in practice CoCo provides better quality solutions within a handful of strategy evaluations than the equivalent of branch-and-bound terminated after a fixed number of iterations.

IV. Numerical Experiments

In this section, we numerically validate our approach on three benchmark problems in robotics: the control of an underactuated cart-pole with multiple contacts, the robot motion planning problem, and dexterous manipulation for task-specific grasping. These problems demonstrate the rich set of combinatorial constraints that arise in robotics and demonstrate the broad applicability of CoCo in handling these constraints. Table I provides an overview of the three systems.

System	n_x	n_δ	n_p
Free-Flyer	34	160	44
Cart-Pole	74	40	13
Manipulator	1092	30	12

TABLE I: Overview of system dimensions in numerical experiments.

A. Implementation Details

For each system, we first generate a dataset by sampling θ from $p(\Theta)$, until a sufficient number of problems $\mathcal{P}(\theta)$ are solved. For each system, we separate 90% of the problems for training and the remaining 10% for evaluation. For our

neural network architecture, we implemented a standard ReLU feedforward network with three layers and 128 neurons per layer for the free-flyer and 32 neurons per layer for the cart-pole and manipulator.

We implemented each example in this section in a library written in Julia 1.2 and the `PyTorch` machine learning library to implement our neural network models with the ADAM optimizer for training. The mixed-integer convex problems were written using the JuMP modeling framework and solved using Gurobi [10] and Mosek [11]. We further benchmark CoCo against Gurobi, Mosek, and the regression framework from [20]. For all problems except the grasp optimization, we disable presolve and multithreading to better approximate the computational resources of an embedded processor. The network architecture chosen for the regressor was identical to the strategy classifier, updated with the appropriate number of integer outputs. The code for our algorithm is available at `https://github.com/StanfordASL/CoCo`.

B. Cart-Pole with Soft Walls

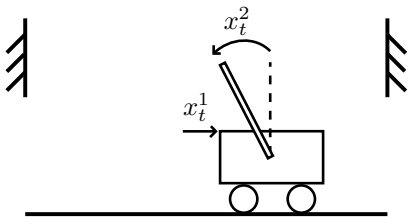


Fig. 2: 4D cart-pole with wall system.

For our first system, we consider problems in robotics that deal with planning multi-contact behaviors. While a myriad of approaches have been proposed for generating open-loop plans for systems with contact or discontinuities, developing controllers that can operate online to quickly react to disturbances has proven challenging. Here, we consider the cart-pole with contact system, a well-known underactuated, multi-contact problem in robot control [4, 23]. As depicted in Figure 2, the system consists of a cart and pole and the optimal control problem entails regulating the system to a goal x_g:

$$\begin{aligned}
\text{minimize} \quad & \|x_N - x_g\|_2 + \sum_{\tau=0}^{N-1} \|x_\tau - x_g\|_2 + \|u_\tau\|_2 \\
\text{subject to} \quad & x_{t+1} = Ax_t + Bu_t + Gs_t, \quad t = 0, ..., N-1 \\
& u_{\min} \le u_t \le u_{\max}, \quad t = 0, ..., N-1 \\
& s_t = \begin{cases} \kappa\lambda_t + \nu\gamma_t & \text{if } \lambda_t \ge 0 \text{ and } \kappa\lambda_t + \nu\gamma_t \ge 0 \\ 0, & \text{otherwise} \end{cases} \\
& \qquad t = 0, ..., N-1 \\
& x_{\min} \le x_t \le x_{\max}, \quad t = 0, ..., N \\
& x_0 = x_{\text{init}},
\end{aligned} \tag{5}$$

where the state $x_t \in \mathbf{R}^4$ consists of the position of the cart x_t^1, angle of the pole x_t^2, and their derivatives x_t^3 and x_t^4, respectively. The force applied to the cart is $u_t \in \mathbf{R}$ and $s_t \in \mathbf{R}^2$ are the contact forces imparted by the two walls. The relative distance of the tip of the pole with respect to the left and right walls is $\lambda_t \in \mathbf{R}^2$ and the time derivative of this relative distance $\gamma_t \in \mathbf{R}^2$. Finally, κ and ν are parameters associated with the soft contact model used.

As the contact force s_t becomes active only when the tip of the pole makes contact with either wall, we must introduce binary variables to enforce the logical constraints given in (5). We denote the relative distance of the tip of the pole with respect to the left and right walls as λ_t^1 and λ_t^2, respectively, and their time derivatives as γ_t^1 and γ_t^2 (we refer the reader to [5] for a thorough derivation of the system constraints).

To constrain contact forces s_t to become active only when the pole tip strikes a wall, we introduce four binary variables $\delta_t^i, \quad i = 1, ..., 4$. Using the formulation from [22], we enforce the following constraints for $k = 1, 2$:

$$\begin{aligned}
\lambda_{\min}^k(1 - \delta_t^{(2k-1)}) \le \lambda_t^k \le \lambda_{\max}^k \delta_t^{(2k-1)} \\
s_{\min}^k(1 - \delta_t^{(2k)}) \le \kappa\lambda_t^k + \nu\gamma_t^k \le s_{\max}^k \delta_t^{(2k)}
\end{aligned}$$

Finally, we impose constraints on s_t, for $k = 1, 2$:

$$\nu\gamma_{\max}^k(\delta_t^{(2k-1)} - 1) \le s_t^k - \kappa\lambda_t^k - \nu\gamma_t^k \le s_{\min}^k(\delta_t^{(2k)} - 1)$$

There are then a total of $4N$ integer variables in this MIQP.

1) Results: In this work, the value for N was set to 10, yielding 40 total integer variables. The parameter space $\theta \in \mathbf{R}^{13}$ consists of the initial state $x_0 \in \mathbf{R}^4$, goal state $x_g \in \mathbf{R}^4$, the relative distance of the tip to the left and right walls $(\lambda_0^1, \lambda_0^2) \in \mathbf{R}^2$ for the initial state and $(\lambda_g^1, \lambda_g^2) \in \mathbf{R}^2$ for the goal state, and the distance $||x_0 - x_g||_2$.

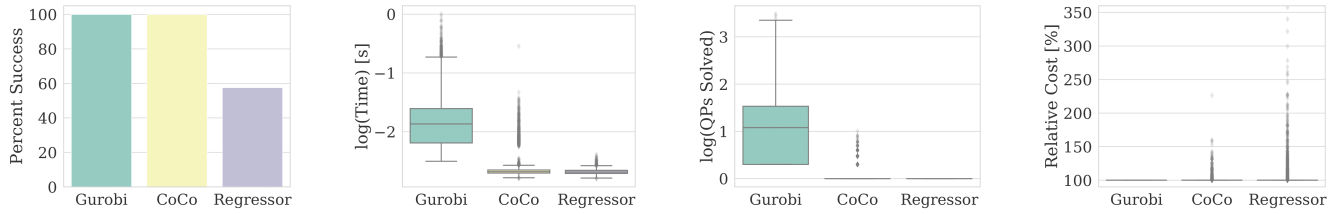
Figure 3 shows the results for the cart-pole system over a test set of ten thousand problems. Figure 3a reports the percent of feasible solutions found over the test set (note that all problems are feasible, so Gurobi reports 100% here). Among solutions for each method that were feasible, Figure 3b reports the solution time for the forward pass of the network plus computation time for QPs solved before a feasible solution was found. Figures 3c and 3d report the number of convex relaxations solved per problem and the cost of the feasible solution relative to the globally optimal solution.

We see that cart-pole system outperforms Gurobi and the regressor benchmarks. For this system, CoCo finds feasible solutions for 99% of the problems compared to only 58% for the regressor. Moreover, we see in Figure 3c that CoCo is able to find a feasible solution after one QP solve for 95% of its feasible problems compared to the average value of 50 QP solves required for Gurobi's branch-and-bound implementation. Note that the regressor always requires one QP solve as only one candidate solution is considered for every forward pass. However, this solution speed-up for CoCo does not come at the cost of optimality. As shown in Figure 3d, 99% of the feasible solutions found by CoCo are also the globally optimal solution for that problem.

C. Free-Flying Space Robots

Motion planning in the presence of obstacles is a fundamental problem in robotics. Here, we consider a free-flying spacecraft robot in the plane that must navigate through a workspace with multiple obstacles. In this section, we show how using the task-specific strategies approach discussed in III-B make this problem tractable to solve with CoCo.

We denote the position as $p_t \in \mathbf{R}^2$ and velocity as $v_t \in \mathbf{R}^2$ in the 2-dimensional plane. The state is therefore $x_t = (p_t, v_t)$. The input $u_t \in \mathbf{R}^2$ consists of the forces produced by the thruster. Letting $\mathcal{X}_{\text{safe}}$ be the free space which the robot must navigate through, the optimal control problem is to plan a



(a) Success percentage (b) Solution times (c) Num. QPs solved (d) Normalized cost [%]

Fig. 3: Simulation results for the cart-pole system. CoCo can quickly compute solutions with near-global feasibility and optimality.

collision free trajectory towards a goal state x_g:

$$\begin{aligned} \text{minimize} \quad & \|x_N - x_g\|_2 + \sum_{\tau=0}^{N-1} \|x_\tau - x_g\|_2 + \|u_\tau\|_2 \\ \text{subject to} \quad & x_{t+1} = Ax_t + Bu_t, \quad t = 0, ..., N-1 \\ & ||u_t||_2 \leq u_{\max}, \quad t = 0, ..., N-1 \\ & x_{\min} \leq x_t \leq x_{\max}, \quad t = 0, ..., N \\ & x_0 = x_{\text{init}} \\ & x_t \in \mathcal{X}_{\text{safe}}, \quad t = 0, ..., N, \end{aligned} \tag{6}$$

Here, the crucial constraint distinguishing (6) from a typical optimal control problem is the safety constraint $x_t \in \mathcal{X}_{\text{safe}}$. In the presence of obstacles, this constraint is typically highly non-convex and requires a global combinatorial search to solve this problem. One class of methods used to solve (6) entails posing it as an MICP and using binary variables to enforce collision-avoidance constraints [24]. Given a set of N_{obs} obstacles, the workspace is decomposed into keep-in and keep-out polytope regions, where binary variables are then used to enforce collision avoidance with the keep-out regions.

To simplify the presentation, we consider axis-aligned rectangular obstacles, but the framework can be generalized to any obstacles represented as convex polygons [25]. We represent an obstacle m with the coordinates of the rectangle for the lower-left hand corner $(x^m_{\min}, y^m_{\min})$ and upper right-hand corner $(x^m_{\max}, y^m_{\max})$. Given the state x_t, the collision avoidance constraints with respect to obstacle m are:

$$x^m_{\max} + M\delta^{m,1}_t \leq x^1_t \leq x^m_{\min} + M\delta^{m,2}_t \tag{7}$$

$$y^m_{\max} + M\delta^{m,3}_t \leq x^2_t \leq y^m_{\min} + M\delta^{m,4}_t \tag{8}$$

For an obstacle set of cardinality N_{obs}, each obstacle thus requires four integers variables $\delta^{m,i}_t$ at each time step. Because the robot must be in violation of at least one of the keep-out constraints, $\delta^{m,i}_t = 1$ corresponds to being on one side of face i of the rectangle. A final constraint is enforced to ensure that the robot does not collide with the obstacle:

$$\sum_{i=1}^{4} \delta^{m,i}_t \leq 3, \quad m = 1, ..., N_{\text{obs}}, \quad t = 1, ..., N. \tag{9}$$

Due to the ℓ_2-norm constraints imposed on the thruster forces, this problem is an MIQCQP with $4N_{\text{obs}}N$ variables.

1) Task-Specific Strategy Decomposition: We show here how the the underlying structure of the obstacle avoidance constraints can be leveraged for effectively training a strategy classifier, as discussed in III-B. We note that each binary variable depends only on the three other variables associated with the same obstacle at the same timestep. Thus, we can decompose the strategy on a per-obstacle basis. Thus, each strategy $\mathcal{S}(\theta; m)$ is comprised of a tuple $(\delta^{m,*}, \mathcal{T}^m_M)$, where $\delta^{m,*}$ and $\mathcal{T}^m_M$ are defined as in (4) for only the integer assignments and big-M constraints for obstacle m.

Rather than training N_{obs} separate strategy classifiers for each $\mathcal{S}(\theta; m)$, we train a single classifier using θ and a one-hot vector to encode which obstacle strategy is being queried.

2) Results: In this work, the horizon N was set to 5 and the number of obstacles N_{obs} to 8, yielding 160 total integer variables. The parameter space $\theta \in \mathbf{R}^{44}$ for this problem included the initial condition $x_0 \in \mathbf{R}^4$ and coordinates $(x^m_{\min}, x^m_{\max}, y^m_{\min}, y^m_{\max}) \in \mathbf{R}^4$ of each of the eight obstacles $m = 1, \ldots, 8$, and a one-hot encoding of the obstacle index.

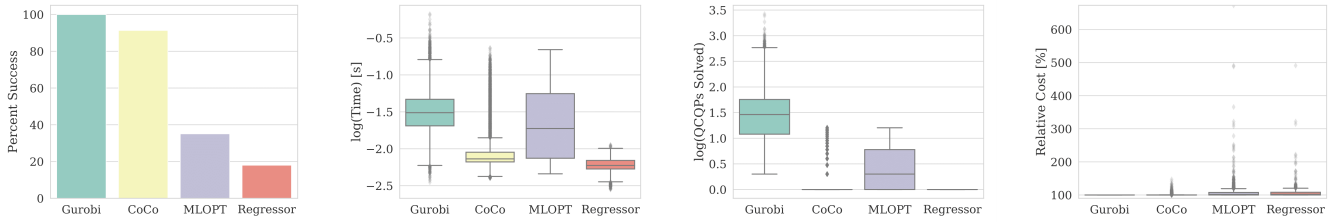
Figure 4 shows the results over the ten thousand test problems. The trained CoCo classifier is compared against Gurobi and the regressor. We also benchmark against an implementation of MLOPT that does not decompose the strategies over each obstacle separately. The number of integer strategies observed in the training set for MLOPT was approximately 81 thousand, whereas the task-specific strategy decomposition for CoCo resulted in 516 strategies. As shown in Figure 4a, CoCo finds feasible solutions for 92% of the training set compared to 18% for the regressor. Crucially, we see that the MLOPT finds feasible solutions for only 35% of the test set. CoCo is on average also able to find solutions faster than Gurobi and MLOPT. Figure 4c shows that this faster computation time is achieved by CoCo having to solve only one QCQP for 85% of the test problems. Once again, CoCo is still able to find high quality feasible solutions and finds the optimal solution for 92% of the time as illustrated in Figure 4d.

For this system, we see that the task-specific strategy approach was required in enabling CoCo to reliably find solutions for the MIQCQPs. MLOPT finds feasible solutions for 56% fewer problems than the task-specific CoCo. This gap is likely attributable to the fact that the number of integer strategies MLOPT has to consider is approximately 81 thousand compared to 516 for the task-specific CoCo. Thus, the task-specific strategies also allow for better supervision of the classifier as there are more labels available per class.

D. Task-Oriented Optimization of Dexterous Grasps

As a final system, we consider the problem of grasp optimization for task-specific dexterous grasps. Practically, during dexterous manipulation (especially with environmental contact) the actual trajectory of the object can diverge significantly from the planned trajectory; further, high-level tasks (such as placing a peg into a hole) may require multiple grasps for various subtasks. Thus, we are interested in enabling online computation of optimal dexterous grasps for both fast replanning and regrasping.

While task-agnostic grasp optimization has been well-studied [26], the problem choosing and even evaluating dex-



(a) Success percentage (b) Solution times (c) Num. QCQPs solved (d) Normalized cost [%]

Fig. 4: Simulation results for the free-flyer show how task-specific strategies are critical for enabling the use of CoCo for this system.

terous grasps for specific tasks (such as pushing or rotating objects, tool use) is still of considerable interest. While works such as [27] have proposed metrics which can be used to evaluate grasps for various tasks, they leave the problem choosing grasps which optimize these metrics unstudied.

Here we consider the problem of choosing n contact points for a multifingered robot hand from a set of points $p_1, \cdots, p_M \in \mathbf{R}^3$, sampled from the object surface in order to optimize the task-oriented grasp metric proposed in [27]. We model the contacts between the fingers and the object as point contacts with friction. Thus, at each candidate point, if selected, a finger could apply a local contact force $f_i = (f_i^x, f_i^y, f_i^z) \in \mathbf{R}^3$, where the local coordinate frame has the x- and y-axes tangent to the surface, and the z-axis along the inward surface normal. Intuitively, f_i^z is the component of the contact force which is normal to the object surface, and f_i^x, f_i^y are its tangential components.

Under this contact model, we constrain the contact force f_i to lie within the *friction cone* $\mathcal{K}^{(i)}$. Let us further define the contact force vector $f = (f_1, \ldots, f_M) \in \mathbf{R}^{3M}$, which is the vector of all contact forces.

Using the definition of a grasp matrix from [26], we can express the wrench applied to the object (from all contact forces) as Gf, where $G = [G_1, \ldots, G_M]$.

However, contact forces may not be applied at all candidate points. To this end, we introduce the logical variables $\delta_i \in \{0, 1\}$, with $\delta_i = 1$ iff point p_i is selected for the grasp. Thus, we enforce the constraint

$$f_i^z \leq \delta_i,$$

which constrains the normal forces of all unused grasps to be zero, and to be bounded by unity otherwise. Thus, for a choice of grasps $\delta = (\delta_1, ..., \delta_M)$, the set of possible object wrenches is defined as $\mathcal{W}(\delta) = \{Gf \mid f \in \mathcal{K}_i, f_i^z \leq \delta_i\}$.

In [27], the authors propose a task-based grasp metric using *task wrenches* $\hat{F}_t$, which are specific directions in wrench space that characterize the applied wrenches necessary to complete the task. For instance, if the desired task is to push the object along the $+x$-axis, then this task could be described using $\hat{F} = (1, 0, 0, 0, 0, 0)$, and so on. For a task described by a single wrench, the grasp quality can be defined as

$$\mu_1(\delta, \hat{F}_t) = \sup \left\{ \alpha \geq 0 \mid \alpha \hat{F}_t \in \partial \mathcal{W}(\delta) \right\},$$

where $\partial \mathcal{W}$ denotes the boundary of $\mathcal{W}$. However, most tasks are best described by a set of wrenches which must be generated, rather than a single direction in wrench space. Thus, the authors propose describing this set as the positive span of T task vectors; in turn, the grasp metric is defined as

$$\mu(\delta, \hat{F}_1, \ldots, \hat{F}_T) = \sum_{t=1}^{T} w_t \alpha_t,$$

where $w_i \geq 0$ are the relative weightings of the task vectors, and $\alpha_t = \mu_1(\delta, \hat{F}_t)$. This can, in turn, be computed by solving T SOCPs.

We seek δ^* which maximizes this grasp metric, which yields a mixed-integer second-order cone program (MISOCP):

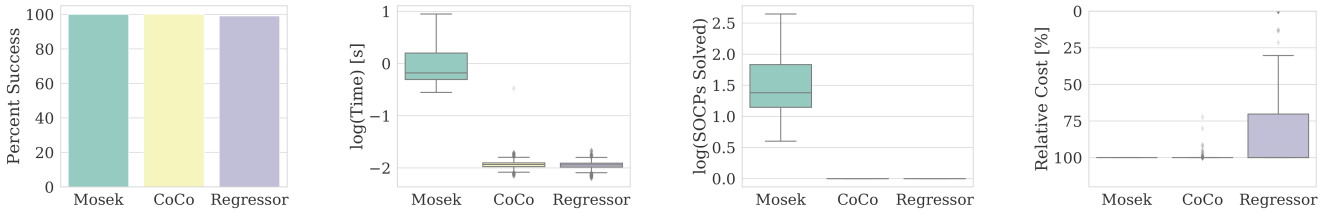
$$\begin{aligned} \text{maximize} \quad & \sum_{t=1}^{T} w_t \alpha_t \\ \text{subject to} \quad & G f^t = \alpha_t \hat{F}_t, \quad t = 1, \ldots, T \\ & f_i^t \in \mathcal{K}^{(i)}, \quad i = 1, \ldots M, \; t = 1, \ldots, T \\ & f_i^{z,t} \leq \delta_i, \quad i = 1, \ldots, M, \; t = 1, \ldots, T \\ & \sum_{i=1}^{M} \delta_i \leq n \\ & \delta \in \{0, 1\}^M \end{aligned} \tag{10}$$

1) Results: For our numerical experiments, we aimed to learn strategies for a single rigid body and set of candidate points which generalized across task weightings w. We set the number of candidate grasp point M equal to 30 and plan grasps for a four finger manipulator $n = 4$.

For training, we collected a set of 4,500 problems, using task vectors $\hat{F}_t$ which corresponded to the basis vectors $\pm e_i \in \mathbf{R}^6$ for $i = 1, \ldots, 6$. The parameter vector $\theta \in \mathbf{R}^{12}$ consists of the desired task weightings. Since these task vectors span $\mathbf{R}^6$, task weightings w with all $w_i > 0$ correspond to generating force closure grasps. We generate w_i by calculating the softmax of a vector sampled from a multivariate normal distribution with covariance matrix $\Sigma = 10\mathbf{I}$.

Figures 5 and 6 show the results for this system. Note that because α_t can be set to 0 i.e., resist a zero wrench, all grasp mode sequences with four contacts are feasible. However, we see in Figure 5a that the regressor can report infeasible solutions if it guesses more than four active binary assignments. Although all CoCo candidate strategies are feasible, we see in Figure 6 that maximizing the grasp metric is nonetheless challenging for the regressor and simple heuristics. Thus, Figure 5d illustrates that the feasible solution found by CoCo is also the globally optimal grasp for 99% of the problems (note that the scale is inverted in this figure as the MISOCP is a maximization problem).

Crucially, this ability to find the globally optimal solution for the majority of problems after solving only one SOCP relaxation leads to solutions in tens of milliseconds rather than the seconds needed by Mosek. On average, CoCo is able to



(a) Success percentage (b) Solution times (c) Num. SOCPs solved (d) Normalized cost [%]

Fig. 5: Simulation results for manipulation example. CoCo reduces solution times for (10) between 2–3 orders of magnitude.

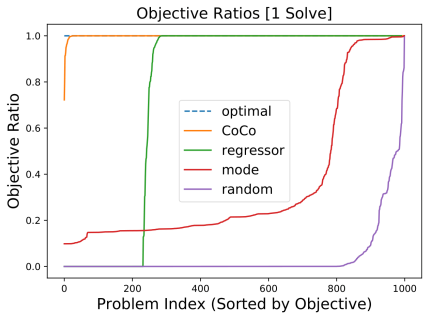


Fig. 6: The objective ratio (higher is better) plotted across the training dataset for grasp modes chosen by CoCo, regressor, the most common mode in the training data, and at random.

accelerate finding solutions for this problem by two to three orders of magnitude, making solving MISOCPs in real-time with high-quality solutions tractable.

V. Conclusion

In this paper, we presented CoCo, a machine learning technique to accelerate the solution of online MICPs by exploiting the structure of problems arising in practical applications. We discarded redundant strategies corresponding to equivalent globally optimal solutions. In addition, we exploited the separable structure of robotics problems to design task-specific strategies. Numerical examples show that our approach results in greater than 90% feasibility, with more than 90% of solutions being globally optimal in common robotics setups such as a cart-pole system with walls, a free-flying space robot, and task-oriented grasps. We also obtained 1-2 orders of magnitude speedups compared to commercial solvers. The proposed algorithm is, therefore, suitable to compute MICP solutions in real-time with high reliability and speed.

Future contributions will focus on providing theoretical guarantees on how to characterize the strategy space by effectively sampling the parameter space of the problems and to bound the optimality gap of feasible solutions found online. Additionally, we would like to explore the application of this proposed framework towards mixed-integer non-convex problems appearing in robotics such as task planning.

References

[1] D. Bertsimas and B. Stellato, "Online mixed-integer optimization in milliseconds," *INFORMS Journal on Computing*, 2020.

[2] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *AIAA JGCD*, vol. 25, no. 4, pp. 755–765, 2002.

[3] P. Culbertson, S. Bandyopadhyay, and M. Schwager, "Multi-robot assembly sequencing via discrete optimization," in *IROS*, 2019.

[4] R. Deits, T. Koolen, and R. Tedrake, "LVIS: Learning from value function intervals for contact-aware robot controllers," in *ICRA*, 2019.

[5] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *ICRA*, 2017.

[6] K. Hauser, S. Wang, and M. R. Cutkosky, "Efficient equilibrium testing under adhesion and anisotropy using empirical contact force models," *IEEE TRO*, vol. 34, no. 5, pp. 1157–1169, 2018.

[7] F. R. Hogan, E. R. Grau, and A. Rodriguez, "Reactive planar manipulation with convex hybrid MPC," in *ICRA*, 2018.

[8] J. Lee and S. Leyffer, Eds., *Mixed Integer Nonlinear Programming*. Springer-Verlag, 2012.

[9] R. E. Bixby, "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica*, 2012.

[10] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2020, available at http://www.gurobi.com.

[11] Mosek APS. The MOSEK optimization software. Available at http://www.mosek.com.

[12] D. Bertsimas and B. Stellato, "The voice of optimization," *Machine Learning*, 2020.

[13] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ. Press, 2004.

[14] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari. (2019) Large scale model predictive control with neural networks and primal active sets. Available at https://arxiv.org/pdf/1910.10835.pdf.

[15] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Safe and near-optimal policy learning for model predictive control using primal-dual neural networks," in *ACC*, 2019.

[16] G. Tang, W. Sun, and K. Hauser, "Learning trajectories for real-time optimal control of quadrotors," in *IROS*, 2018.

[17] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato, "Learning convex optimization control policies," in *L4DC*, 2019.

[18] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *NeurIPS*, 2017.

[19] H. He, H. Daumé III, and J. Eisner, "Learning to search in branch-and-bound algorithms," in *NeurIPS*, 2014.

[20] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *ECC*, 2019.

[21] R. M. Karp, "On the computational complexity of combinatorial problems," *Networks*, vol. 5, no. 1, pp. 45–68, 1975.

[22] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, 1999.

[23] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE TAC*, 2020.

[24] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *ECC*, 2001.

[25] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *ICRA*, 2016.

[26] C. Ferrari and J. Canny, "Planning optimal grasps," in *ICRA*, 1992.

[27] R. Haschke, J. J. Steil, I. Steuwer, and H. Ritter, "Task-oriented quality measures for dextrous grasping," in *CIRA*, 2005.