CoCo: Learning Strategies for Online Mixed-Integer Control

Abhishek Cauligi[1], **Preston Culbertson**[1], **Mac Schwager**[1], **Bartolomeo Stellato**[2], **Marco Pavone**[1]
[1]Stanford University, [2]Princeton University
{ acauligi, pculbertson, schwager, pavone}@stanford.edu, bstellato@princeton.edu

1 Introduction

Discrete and combinatorial optimization problems arise in a wide variety of applications and are used as a tool to formulate and solve challenging problems in robotics [4, 8], autonomous mobility-on-demand [15], and power network design [9, 14], among others. One approach to solving such problems is to model them through the lens of mixed integer convex programming (MICP), which are optimization problems with convex objective functions and constraints, but which are made non-convex through the inclusion of discrete decision variables. While MICPs have emerged as a popular tool for solving a variety of problems, the integrality constraints make them $\mathcal{NP}$-complete and often exceptionally difficult to solve in the worst case [7].

Despite great strides made in the past few decades to improve solution times by several orders of magnitude through algorithmic and hardware improvements, MICPs are rarely deployed in real-world robotics and control applications that require finding solutions in real time (i.e., 10-100Hz). Further, state-of-the-art commercial solvers such as Gurobi [6] and Mosek [11] are not suitable for use on embedded robotic systems as they rely on complex multithreaded implementations. Thus, the ability to solve MICPs in real-time on limited computational resources available on robotics remains an open and pressing challenge.

In this work, we build on results from [3], which presented the Combinatorial Offline, Convex Online (CoCo) algorithm. CoCo is a two-stage, data-driven algorithm that seeks to leverage tools from parametric convex optimization and supervised machine learning to quickly find feasible solutions for MICPs. Using the idea of task-specific strategy decompositions, we present new results on how CoCo can be extended to yield improved (1) *generalizability* and (2) *scalability*. We use a novel convolutional neural network architecture to allow CoCo to generalize to solve problems online with a different number of discrete decision variables than what it was trained for offline. We demonstrate how this architecture can scale to problems with different horizons and discrete variables through numerical experiments on a spacecraft motion planning problem. Finally, we propose future directions of research to allow for online adaptation of CoCo for improved performance.

2 Approach

We consider parametrized mixed-integer convex programs with general form,

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

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. A popular use for MICPs is to model logical constraints in systems using what is known as big-M constraints [1]. In the big-M formulation, binary variables δ capture high-level discrete or logical behavior of the system and enforce the resulting high-level behavior on the continuous variables x. An example of such a use case is, given a binary variable δ_i, the imposition of the logical constraint:

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

1st Workshop on Learning Meets Combinatorial Algorithms @ NeurIPS 2020, Vancouver, Canada.

In big-M formulation, such a logical constraint can be enforced through the constraint:

$$g_i(x;\theta) \le M_i(\theta)(1-\delta_i), \tag{2}$$

where $M_i(\theta) = \sup_x g_i(x;\theta)$. Note that the desired logical behavior is achieved through (2): the constraint is active when $\delta_i = 1$ and, when $\delta_i = 0$, we have the trivial constraint $g_i(x;\theta) \le M_i(\theta)$.

The linear structure of the constraint in (2) allows us to define the notion of a logical strategy. Given parameters $\theta \in \mathbf{R}^{n_p}$, a logical strategy $\mathcal{S}(\theta)$ consists of 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,

$$\mathcal{T}_M(\theta) = \{i \mid g_i(x^*;\theta) \le a_i(\theta)\delta_i \iff \delta_i = \delta_i^*\}. \tag{3}$$

Unlike the integer strategy definition presented in [2], logical strategies account for the existence of multiple integer solutions satisfying some logical proposition. As shown in the results in [3], logical strategies allow for the supervised learning problem used for CoCo to be well-posed, prune redundant integer strategies, and allow for classifier performance.

The utility of defining a logical strategy $\mathcal{S}(\theta)$ can then be made apparent by using $\delta^*(\theta)$ and $\mathcal{T}_M(\theta)$ to solve the MICP as a convex optimization problem. Specifically, given an optimal strategy $\mathcal{S}(\theta)$, the optimal solution for (1) can be found by solving,

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

which is a convex optimization problem that be solved efficiently. The CoCo algorithm leverages this insight to learn a mapping from problem parameters θ to a corresponding logical strategy $\mathcal{S}(\theta)$. Specifically, CoCo trains a neural network classifier h_ϕ offline over a dataset $\mathcal{D} = \{(\theta_i, \mathcal{S}_i)\}_{i=1}^N$ consisting of problem parameters θ_i and the corresponding optimal strategies $\mathcal{S}_i$. Online, presented a new problem parameter θ, a single forward pass of the network ranks candidate logical strategies and a finite number of user-specified convex optimization problems are solved until either a feasible solution is found or the algorithm terminates with failure.

2.1 Task-Specific Strategy Decomposition

In our work, we leverage the notion of task-specific strategies introduced in [3]. Task-specific strategies exploit the underlying structure and separability that commonly arise in many robotics problems such as piecewise-affine dynamics and multi-surface contact planning. For example, for the motion planning problem in [13], the logical constraints that enforce collision avoidance between the robot and an obstacle m can be individually treated by only regarding the subset of binary variables used in those constraints. This separability leads to a decoupling between the logical variables in the problem, as they do not appear together in common constraints.

Our key insight is that logical constraints under the CoCo framework can be decoupled by training the strategy predictor to predict the logical strategy for each constraint separately. Further, by separating the prediction of each constraint, we can compose strategies to solve control problems with different numbers of binary variables. Indeed, this ability to solve multiple tasks with a varying number of discrete decision variables is a distinguishing feature of our work as compared to [2, 3, 10, 16].

3 Experimental Results

3.1 Free-Flying Space Robots

We focus our attention on a fundamental problem in robotics: motion planning in the presence of obstacles. In this scenario, we consider a free-flying space robot in the plane that must navigate through a workspace with multiple obstacles. The optimal control problem for this system entails planning a collision free trajectory in free space $\mathcal{X}_{\text{safe}}$ towards a goal state x_g:

$$\begin{array}{ll} \text{minimize} & \|x_N - x_g\|_2 + \sum_{\tau=0}^{N-1} \|x_\tau - x_g\|_2 + \|u_\tau\|_2 \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \quad t = 0, \dots, N-1 \\ & ||u_t||_2 \le u_{\max}, \quad t = 0, \dots, N-1 \\ & x_{\min} \le x_t \le x_{\max}, \quad t = 0, \dots, N \\ & x_0 = x_{\text{init}} \\ & x_t \in \mathcal{X}_{\text{safe}}, \quad t = 0, \dots, N, \end{array} \tag{4}$$

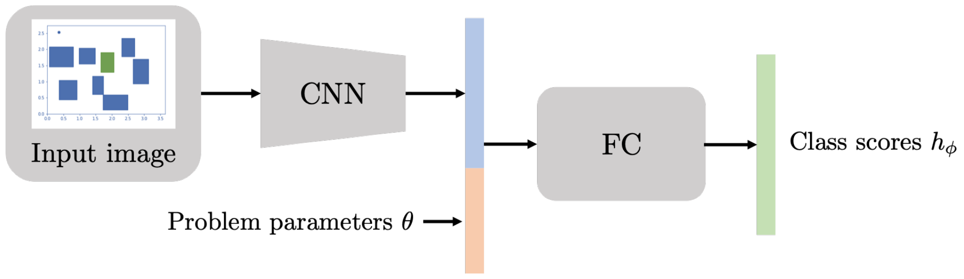


Figure 1: The neural network architecture for the proposed CoCo extension consists of concatenating the output of a convolutional pass of a synthetic image of the scene (in blue) with the remaining problem parameters (in red). The network then scores the candidate logical strategies stored from offline training.

where $p_t \in \mathbf{R}^2$ and $v_t \in \mathbf{R}^2$ are the position and velocity, respectively, 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.

The constraint in (4) that makes it highly non-convex and necessitates the inclusion of binary decision variables is the safety constraint $x_t \in \mathcal{X}_{\text{safe}}$. Indeed, the global combinatorial search required to solve this problem is an extensive area of study and one class of methods used to solve (4) entails posing it as an MICP [13]. The procedure consists of decomposing the workspace into keep-in and keep-out polytope regions given a set of N_{obs} obstacles. For simplicity, we consider axis-aligned rectangular obstacles. For an obstacle m, we denote the coordinates of the lower-left hand corner $(x^m_{\text{min}}, y^m_{\text{min}})$ and upper right-hand corner $(x^m_{\text{max}}, y^m_{\text{max}})$. The collision avoidance constraint between state x_t and obstacle m can then be written as:

$$x^m_{\text{max}} + M\delta_t^{m,1} \leq x_t^1 \leq x^m_{\text{min}} + M\delta_t^{m,2} \tag{5}$$

$$y^m_{\text{max}} + M\delta_t^{m,3} \leq x_t^2 \leq y^m_{\text{min}} + M\delta_t^{m,4} \tag{6}$$

where each obstacle requires four integer variables $\delta_t^{m,i}$ at each time step. An additional constraint is introduced to enforce that the the robot lies on at least one side of the obstacle without being in collision:

$$\sum_{i=1}^{4} \delta_t^{m,i} \leq 3, \quad m = 1, \ldots, N_{\text{obs}}, \quad t = 1, \ldots, N. \tag{7}$$

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

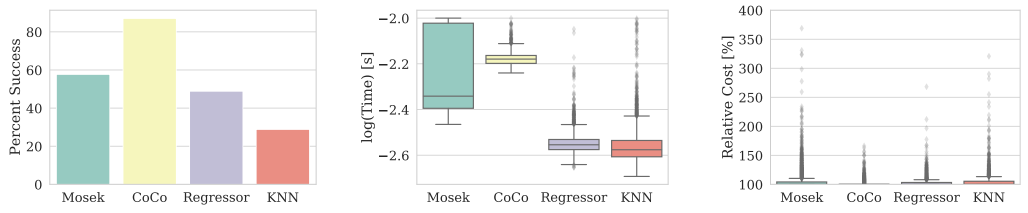
3.2 Architecture

Here, we show how the notion of task-specific strategies can be leveraged to (1) more effectively training the strategy classifier h_ϕ and (2) allow for the trained network to generalize to problems with a different number of integer strategies. In [3], this was accomplished by training a single classifier with parameters θ and a one-hot vector encoding as the input. Thus, the network was fixed to a particular number of obstacles m, with the one-hot vector encoding which obstacle strategy is being queried. However, this network architecture limits the use of CoCo to a fixed number of obstacles. For example, if a robot traverses to a new environment with a different number of obstacles, then the trained classifier h_ϕ is no longer useful for online control.

Instead, we propose using the convolutional neural network architecture (CNN) shown in Figure 1 for accomplishing this task. A CNN architecture confers several advantages. First, it allows for querying strategies for an arbitrary number of obstacles, thereby allowing it to be used for problems with different number of binary variables. Second, it allows for solving problems with various obstacle geometries depicted in the scene, as the strategy classifier is not fixed to a particular geometry parameterization. Finally, the ability to solve problems with a different number of binary variables also paves the way for online adaptation approaches, wherein the network parameters are tuned online for improved performance in new environments.

3.3 Results

The machine learning models were implemented in `PyTorch` [12] and the convex optimization problems solved using Mosek [11]. The network architecture used for the free-flyer motion planning



(a) Success percentage (b) Solution times (c) Normalized cost [%]

Figure 2: Simulation results for the free-flyer show how task-specific strategies are critical for enabling the use of CoCo for this system.

problem is depicted in Figure 1 and consists of three convolutional layers on a synthetic image of the obstacle. The output of this convolutional block is then concatenated with the remaining problem parameters and input to a standard ReLU feedforward network with three layers and 128 neurons per layer. We further benchmark our proposed architecture against Mosek [11], the regression framework from [10], and the k-nearest neighbor (KNN) approach from [16]. The CoCo, regressor, and KNN methods are trained using ninety thousand MICPs, with parameters θ drawn from a random distribution of initial conditions x_0 and obstacle coordinates, and evaluated on a test set of ten thousand feasible problems. The MICP used $N = 5$ and $N_{\text{obs}} = 8$, leading to a total of 160 binary variables. The code will be made publicly available.

Figure 2 shows the results of our numerical simulations for Mosek and each data-driven framework. For all methods, we cap the solver at 10ms (i.e., 100Hz) and compare the number of feasible solutions found, solution time, and the quality of these feasible solutions compared to the globally optimal solution. We see that task-specific CoCo finds feasible solutions for 87% of the test set, compared to 58%, 49%, and 29% for Mosek, the regressor, and the KNN, respectively. Figure 2c depicts the relative cost of first feasible solution found by Mosek and the three data-driven frameworks compared to the globally optimal solution for the problem. Crucially, we see that 80% of the feasible solutions solved by CoCo are indeed the globally optimal solution, compared to only 69%, 57%, and 53% for Mosek, regression, and KNN, respectively. Thus, we see that CoCo outperforms the commerical solver and the benchmarks from [10, 16] for this problem.

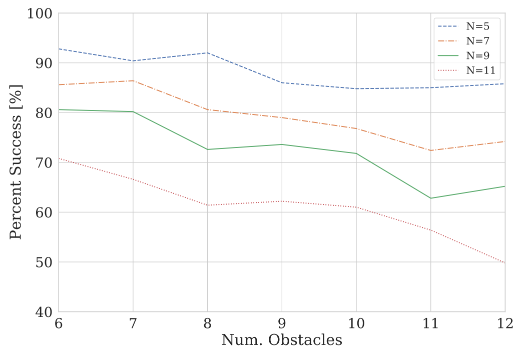


Figure 3: Generalization results for a CoCo strategy classifier network trained on environment with $N_{\text{obs}} = 8$ for various horizons N.

Figure 3 demonstrates the ability of CoCo to generalize to a distribution of problems with a varying of number binary variables. We trained multiple strategy classifiers corresponding to horizons of $N = \{5, 7, 9, 11\}$ with environments of eight obstacles. We evaluated the ability of these networks to environments with an increasing number of obstacles $N_{\text{obs}} = \{6, \dots, 12\}$. As shown, we see that the efficacy of the strategy classifier diminishes with an increasing horizon length N due to a corresponding increase in the number of strategies. Intuitively, we also see that performance decreases with an increase obstacles simply due to the increased difficulty of the planning problem. However, we note that the dropoff in performance remains roughly linear rather than an exponential decrease of performance stemming from including additional binary variables. Indeed, we believe that these results motivate improving the performance of the classifier for environments with a different number of obstacles as a future area of research. For example, one future avenue could be to explore meta-learning inspired approaches such as MAML [5] to allow CoCo to adapt network parameters online for improved performance using information gained from solving problems online.

References

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

[2] D. Bertsimas and B. Stellato. Online mixed-integer optimization in milliseconds, 2019. Available at https://arxiv.org/abs/1907.02206.

[3] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone. Learning mixed-integer convex optimization strategies for robot planning and control. In *Proc. IEEE Conf. on Decision and Control*, 2020. In Press.

[4] R. Deits, T. Koolen, and R. Tedrake. LVIS: Learning from value function intervals for contact-aware robot controllers. In *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[5] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Int. Conf. on Machine Learning*, 2017.

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

[7] R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.

[8] B. Landry, R. Deits, P. R. Florence, and R. Tedrake. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *Proc. IEEE Conf. on Robotics and Automation*, 2016.

[9] S. Mashayekh, M. Stadler, G. Cardoso, and M. Heleno. A mixed integer linear programming approach for optimal DER portfolio, sizing, and placement in multi-energy microgrids. *Applied Energies*, 2017.

[10] D. Masti and A. Bemporad. Learning binary warm starts for multiparametric mixed-integer quadratic programming. In *European Control Conference*, 2019.

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

[12] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017.

[13] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, 2001.

[14] D. Tenfen and E. C. Finardi. A mixed integer linear programming model for the energy management problem of microgrids. *Electric Power Systems Research*, 2015.

[15] R. Zhang, F. Rossi, and M. Pavone. Model predictive control of Autonomous Mobility-on-Demand systems. In *Proc. IEEE Conf. on Robotics and Automation*, 2016.

[16] J.-J. Zhu and G. Martius. Fast non-parameteric learning to accelerate mixed-integer programming for online hybrid model predictive control, 2019. Available at https://arxiv.org/pdf/1911.09214.pdf.