Tailored Presolve Techniques in Branch-and-Bound Method for Fast Mixed-Integer Optimal Control Applications

Quirynen, Rien; Di Cairano, Stefano

Abstract

Mixed-integer model predictive control (MI-MPC) can be a powerful tool for con- trolling hybrid systems. In case of a linear-quadratic objective in combination with linear or piecewise-linear system dynamics and inequality constraints, MI-MPC needs to solve a mixed-integer quadratic program (MIQP) at each sampling time step. This paper presents a collection of exact block-sparse presolve techniques to effi- ciently remove decision variables, and to remove or tighten inequality constraints, tailored to mixed-integer optimal control problems. In addition, we describe a novel approach based on a heuristic presolve algorithm to compute a feasible but possibly suboptimal MIQP solution. We present benchmarking results for a C code imple- mentation of the proposed BB-ASIPM solver, including a branch-and-bound (B&B) method with the proposed tailored presolve techniques and an active-set based inte-rior point method (ASIPM), compared against multiple state-of-the-art MIQP solvers on a case study of motion planning with obstacle avoidance constraints. Finally, we demonstrate the feasibility and computational performance of the BB-ASIPM solver in embedded system on a dSPACE Scalexio real-time rapid prototyping unit for a second case study of stabilization for an underactuated cart-pole with soft contacts

Tailored Presolve Techniques in Branch-and-Bound Method for Fast Mixed-Integer Optimal Control Applications [†]

Rien Quirynen* | Stefano Di Cairano

[1]Control for Autonomy, Mitsubishi Electric Research Laboratories, Massachusetts, USA

Correspondence
*Email: quirynen@merl.com

Present Address
201 Broadway, 8th Floor, Cambridge, MA 02139-1955

Abstract

Mixed-integer model predictive control (MI-MPC) can be a powerful tool for controlling hybrid systems. In case of a linear-quadratic objective in combination with linear or piecewise-linear system dynamics and inequality constraints, MI-MPC needs to solve a mixed-integer quadratic program (MIQP) at each sampling time step. This paper presents a collection of exact block-sparse presolve techniques to efficiently remove decision variables, and to remove or tighten inequality constraints, tailored to mixed-integer optimal control problems. In addition, we describe a novel approach based on a heuristic presolve algorithm to compute a feasible but possibly suboptimal MIQP solution. We present benchmarking results for a C code implementation of the proposed `BB-ASIPM` solver, including a branch-and-bound (B&B) method with the proposed tailored presolve techniques and an active-set based interior point method (ASIPM), compared against multiple state-of-the-art MIQP solvers on a case study of motion planning with obstacle avoidance constraints. Finally, we demonstrate the feasibility and computational performance of the `BB-ASIPM` solver in embedded system on a dSPACE Scalexio real-time rapid prototyping unit for a second case study of stabilization for an underactuated cart-pole with soft contacts.

KEYWORDS:
Mixed-integer programming, Numerical optimization algorithms, Hybrid model predictive control

1 | INTRODUCTION

Optimization-based motion planning and control techniques, such as model predictive control (MPC), allow a model-based design framework in which the dynamics, constraints and objectives are directly taken into account[1]. This framework has been extended to hybrid systems[2], providing a powerful technique to model a large range of hybrid control problems, e.g., including switched dynamical systems[3,4], discrete/quantized actuation[5], motion planning with obstacle avoidance[6], logic rules and temporal logic specifications[7]. However, the resulting optimal control problems (OCPs) are non-convex combinatorial optimization problems, because they contain variables that only take integer or binary values, and they are $\mathcal{NP}$-hard to solve in general[8]. When using a linear-quadratic objective in combination with linear or piecewise-linear system dynamics and inequality constraints, the resulting OCP can be formulated as a mixed-integer quadratic program (MIQP).

In the present work, we aim to solve MIQP problems of the following form:

$$\min_{X,U} \quad \sum_{i=0}^{N} \frac{1}{2} \begin{bmatrix} x_i \\ u_i \end{bmatrix}^{\top} H_i \begin{bmatrix} x_i \\ u_i \end{bmatrix} + \begin{bmatrix} x_i \\ u_i \end{bmatrix}^{\top} \begin{bmatrix} q_i \\ r_i \end{bmatrix} \tag{1a}$$

$$\text{s.t.} \quad x_{i+1} = \begin{bmatrix} A_i & B_i \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} + a_i, \quad \forall i \in \{0, \ldots, N-1\}, \tag{1b}$$

$$\begin{bmatrix} \underline{x}_i \\ \underline{u}_i \end{bmatrix} \leq \begin{bmatrix} x_i \\ u_i \end{bmatrix} \leq \begin{bmatrix} \bar{x}_i \\ \bar{u}_i \end{bmatrix}, \quad \forall i \in \{0, \ldots, N\}, \tag{1c}$$

$$\underline{c}_i \leq \begin{bmatrix} C_i & D_i \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} \leq \bar{c}_i, \quad \forall i \in \{0, \ldots, N\}, \tag{1d}$$

$$u_{i,j} \in \mathbb{Z}, \quad \forall j \in \mathcal{I}_i, \quad \forall i \in \{0, \ldots, N\}, \tag{1e}$$

where the state variables are $x_i \in \mathbb{R}^{n_x^i}$, the control variables are $u_i \in \mathbb{R}^{n_u^i}$ and $\mathcal{I}_i$ denotes the index set of integer decision variables, i.e., the cardinality $|\mathcal{I}_i| \leq n_u^i$ denotes the number of integer variables at each time step $i \in \{0, 1, \ldots, N\}$. The objective in (1a) defines a linear-quadratic function with positive semi-definite Hessian matrix $H_i \succeq 0$ and gradient vectors $q_i \in \mathbb{R}^{n_x^i}$ and $r_i \in \mathbb{R}^{n_u^i}$. The constraints include state dynamic equality constraints in (1b), simple bounds in (1c), affine inequality constraints in (1d) and integer feasibility constraints in (1e). The optimization problem in (1) is an optimal control structured MIQP or a block-sparse MIQP due to the block-structured sparsity of the Hessian and constraint matrix, i.e., the objective in (1a) and inequality constraints in (1c)-(1d) are separable per time step $i \in \{0, 1, \ldots, N\}$ but the variables are coupled via the dynamics in (1b). Note that an initial state constraint $x_0 = \hat{x}_0$, where $\hat{x}_0$ is a current state estimate, can be enforced using the simple bounds in (1c). A compact notation is used to denote the optimization variables of the MIQP in (1) as the state $X = [x_0^{\top}, \ldots, x_N^{\top}]^{\top}$ and control trajectory $U = [u_0^{\top}, \ldots, u_N^{\top}]^{\top}$. In the present paper, we occasionally refer to all optimization variables in (1) as $Z = [z_0^{\top}, \ldots, z_N^{\top}]^{\top}$, where $z_i = [x_i^{\top}, u_i^{\top}]^{\top}$ for $i \in \{0, 1, \ldots, N\}$. Unlike standard OCP formulations, the MIQP in (1) defines a mixed-integer OCP (MIOCP) that includes control variables on the terminal stage, $u_N \in \mathbb{R}^{n_u^N}$, which may include auxiliary variables to formulate the mixed-integer inequality constraints of the hybrid control system. A binary optimization variable $u_{i,j} \in \{0, 1\}$ can be defined as an integer variable $u_{i,j} \in \mathbb{Z}$ in (1e), including the simple bounds $0 \leq u_{i,j} \leq 1$ in (1c). Without loss of generality and for simplicity of notation, the integer optimization variables in (1) are restricted to control variables, even though the methods in this paper can be trivially extended to MIOCPs with integer state variables. MPC for any hybrid system can be formulated as in (1), for example, by leveraging a mixed logical dynamical (MLD) model[2]. There are typically different ways to formulate a hybrid control problem in the form of the MIQP in (1), providing a tradeoff between the number of decision variables and strength of the formulation[9]. The strength of an MIQP corresponds to how close the convex relaxations are to the exact MIQP solution, e.g., see the formulations in[3] for piecewise-affine systems.

Mixed-integer MPC (MI-MPC) implementations for motion planning and control aim to solve the MIOCP in (1) at each sampling time step. This is challenging due to the $\mathcal{NP}$-hard complexity of solving MIQPs in general[8], and given the relatively small computational resources and available memory on embedded microprocessors for real-time control applications[10]. Therefore, several tailored solution strategies have been proposed for MI-MPC. These approaches can generally be divided into *heuristic* techniques, which seek to efficiently find feasible but suboptimal solutions to the problem, and *exact* algorithms that solve the MIQPs to optimality. Examples of the former include rounding schemes[11,12], the feasibility pump[13], approximate optimization[14,15], approximate dynamic programming[16], and approximate explicit hybrid MPC[17]. Machine learning can be used to approximately solve combinatorial optimization problems[18], e.g., using supervised learning to train a network architecture to quickly compute feasible but suboptimal solutions online as in[19,20,21,22,23,24]. The downside of fast heuristic approaches is often the lack of guarantees of finding an optimal, or even a feasible, solution.

Due to the complexity of solving MIQPs, explicit methods that compute *offline* the optimal control as a function of the system parameters have been developed, e.g., see[25,26]. The application of these explicit methods is generally limited to small-dimensional systems with few discrete variables. We therefore focus on the use of *online* solution methods in the present paper. The MIQP in (1) is a mixed-integer convex program (MICP), i.e., it becomes a convex problem after relaxing the integer feasibility constraints in (1e). Most of the exact optimization algorithms for MIQPs are based on the classical branch-and-bound (B&B) technique[27]. Specifically for the structured MIQP in (1), the B&B strategy has been combined with tailored algorithms for solving the relaxed convex QPs. For example, a B&B algorithm for MI-MPC has been proposed in combination with a dual active-set solver in[28], a primal active-set solver in[29,30], an interior point algorithm in[31], dual projected gradient methods in[32,15], a nonnegative least squares solver in[33], and with the alternating direction method of multipliers (ADMM) in[34].

Machine learning could also be used to speed up the exact solution of combinatorial optimization problems, e.g., by improving the tree search in B&B methods [18,35]. B&B methods for solving mixed-integer nonlinear OCPs have also been studied, e.g., in [36].

The computational efficiency of B&B methods is affected by the node selection and branching rules [37,38], and by the convex relaxation solutions [39,40], but there are many other important factors that make state-of-the-art MIQP solvers like GUROBI [41] and MOSEK [42] successful. A crucial algorithmic ingredient is the *presolve* routine [43], which typically is called in each node of the B&B method (see Figure 1) before solving the convex relaxation, and it performs a collection of operations to remove decision variables, and to remove or tighten constraints. In the present paper, we explicitly refer to these presolve techniques as *exact* operations to emphasize that they preserve feasibility and optimality, i.e., a feasible and optimal solution to the reduced problem exists as long as a feasible and optimal solution exists to the original MIQP. Exact presolve techniques are vital for the good performance of current state-of-the-art MICP solvers, such that B&B methods can often solve seemingly intractable problems in practice [44]. Especially for MI-MPC applications, warm starting strategies exist that aim to reuse the explored B&B tree at one time step to reduce the computational cost of the B&B method at the next control time step. In recent years, different variants of B&B warm starting have been proposed for MI-MPC, e.g., in [33,29,45].

Unlike state-of-the-art mixed-integer solvers, e.g., GUROBI [41] and MOSEK [42], our aim is to propose a tailored algorithm and its solver implementation for fast embedded MI-MPC applications, i.e., running on microprocessors with considerably less computational resources and available memory [10], while leveraging the special structure of the MIQP in (1). The optimization algorithm should be relatively simple to code with a moderate use of resources, while the software implementation is preferably compact and library independent. In the present paper, we will use a tailored active-set based interior point method (ASIPM) that was presented in [46] to solve the block-sparse convex QP relaxations in the B&B method, resulting in an MIQP solver that will further be referred to as BB-ASIPM. The recent work in [47] showed how infeasibility detection and early termination based on duality can be implemented efficiently for an infeasible primal-dual interior point method (IPM) based on a computationally efficient projection strategy that will be used within our BB-ASIPM solver.

Our Contributions: A first contribution of the present paper is an exact block-sparse presolve routine that is tailored to MIQPs of the form in (1). Our previous work in [29,47] showed how domain propagation [43,48] can be applied to the condensed form of (1). In the present paper, we propose an algorithm for domain propagation that can be applied directly to the MIQP in (1), based on a forward-backward propagation of variable bounds, in order to speed up the computation times of a B&B method. We additionally present tailored algorithms for other presolve techniques, including the removal of trivial constraints, dual fixings, constraint coefficient strengthening and binary variable probing. A second contribution involves the use of the proposed presolve routine in a heuristic procedure to compute a feasible but possibly suboptimal MIQP solution. This heuristic is an extension of the idea in [24] for improving supervised learning of MICP solutions. A third contribution is the benchmarking results for a C code implementation of the BB-ASIPM algorithm against state-of-the-art MIQP solvers, including GUROBI [41], MOSEK [42], GLPK [49], Cbc [50], and Matlab's `intlinprog`, based on a case study of mobile robot motion planning with obstacle avoidance constraints. A fourth and final contribution is the demonstration of the computational performance of the BB-ASIPM solver on a dSPACE Scalexio rapid prototyping unit, using a second case study of stabilization for an underactuated cart-pole with soft contacts.

The paper is organized based on each of the following algorithmic ingredients that are typically important for a good computational performance of a B&B method for mixed-integer optimal control applications:

- variable branching decisions and node selection strategies (Section 2),
- structure-exploiting convex solver for efficient QP solutions (Section 3),
- exact presolve techniques for variable fixings and tight relaxations (Section 4),
- fast primal heuristic algorithm to find integer-feasible solutions (Section 5),
- warm starting and embedded software for mixed-integer MPC (Section 6).

The performance of our proposed MIQP solver is illustrated based on two MI-MPC case studies in Section 7, including hardware-in-the-loop simulations on a dSPACE Scalexio rapid prototyping unit. Finally, Section 8 concludes the paper.

2 | PRELIMINARIES ON MIXED-INTEGER QUADRATIC PROGRAMMING

We first introduce some of the basic concepts in mixed-integer quadratic programming (MIQP) solvers based on branch-and-bound (B&B) methods, such as convex QP relaxations, node selection and branching strategies.

LB: UB:
$-\infty$ ∞
10   15
12   13
13

$J_0 = 10$

P_0

$u_1 \leftarrow 0$ $u_1 \leftarrow 1$

$J_1 = 13$ P_1 P_2 $J_2 = 12$

$u_2 \leftarrow 0$ $u_2 \leftarrow 1$ $u_2 \leftarrow 0$ $u_2 \leftarrow 1$

P_3 P_4 P_5 P_6

$J_3 = 15$ $J_4 = \infty$ $J_5 = 13$ $J_6 = 16$
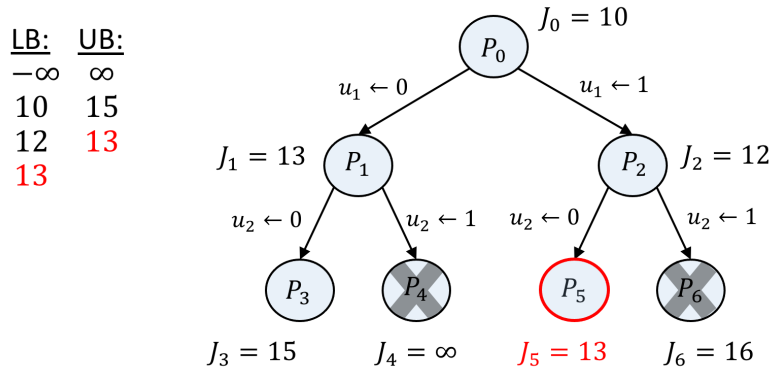
FIGURE 1 Illustration of the branch-and-bound (B&B) method as a binary search tree. A selected node can be either *branched*, e.g., resulting in two partitions for each binary variable $u_{i,j} \in \{0, 1\}$, or *pruned* based on feasibility or the current upper bound.

2.1 | Branch-and-Bound Algorithm

The main idea of the B&B optimization algorithm is to sequentially create partitions of the original MIQP problem and attempt to solve each of these partitions. While solving each partition may still be challenging, it is fairly efficient to obtain local lower bounds on the optimal objective value, e.g., by solving convex relaxations of the MIQP subproblem. If we happen to obtain an integer-feasible solution while solving a relaxation, we can then use it to obtain a global upper bound for the solution to the original problem. This may help to avoid solving or branching certain partitions that were already created, i.e., such partitions or nodes can be *pruned*. The general algorithmic idea of partitioning is better illustrated as a binary search tree, see Figure 1. A key step in this approach is how to create the partitions, i.e., which node to choose and which discrete variable to select for branching. Since we solve a convex relaxation at every node of the tree, it is natural to branch on one of the discrete variables with fractional values in the optimal solution of the relaxation. Therefore, if a binary variable, e.g., $u_{i,j} \in \{0, 1\}$ has a fractional value in a convex relaxation, we create two partitions where we add the equality constraints $u_{i,j} = 0$ and $u_{i,j} = 1$, respectively. Another key choice is the order in which the created subproblems are solved. These steps have been extensively explored in the literature and various heuristics are implemented in state-of-the-art tools[37].

2.2 | Convex Quadratic Program Relaxations

We use a B&B method to solve the MIQP (1) by solving convex quadratic programming (QP) relaxations that are constructed by dropping the integer feasibility constraints in (1e), e.g., $u_{i,j} \in \{0, 1\}$ is relaxed to a bounded continuous variable $0 \le u_{i,j} \le 1$. Other convex relaxations for MIQPs have been studied in the literature such as moment or SDP relaxations that may be tighter than QP relaxations[39,40], however they are often relatively expensive to solve, even if they may drastically reduce the nodes explored in a B&B method. In this paper, we restrict to standard QP relaxations and we present a tailored active-set based interior point method (ASIPM) in Section 3. The ASIPM solver has been shown to be competitive with state-of-the-art QP solvers for embedded MPC[46], it benefits from warm-starting and it allows for an efficient implementation of infeasibility detection and early termination based on duality[47]. For using the ASIPM solver, the relaxations need to be convex, i.e., the Hessian matrices H_i need to be positive semi-definite in (1a) such that each solution to a QP relaxation is globally optimal.

2.3 | Tree Search: Node Selection Strategies

A common implementation of the B&B method is based on a *depth-first* node selection strategy, which can be implemented by a last-in-first-out (LIFO) buffer. The next node to be solved is selected as one of the children of the current node and this process is repeated until integer feasibility or until a node is pruned because the node is either infeasible or dominated by the upper bound, which is followed by a backtracking procedure. Instead, a *best-first* strategy selects the node with the lowest local lower bound so far. In this paper, we will employ a combination of the depth-first and best-first node selection approach. This combination aims to find an integer-feasible solution quickly at the start of the B&B procedure (depth-first) to allow for early pruning, followed by a more greedy search for better feasible solutions (best-first).

2.4 | Reliability Branching for Variable Selection

Many branching rules exist such as "most infeasible" branching which selects the integer variable with fractional part in the QP relaxation that is closest to 0.5. Even though this rule is used quite often, e.g., in [33,34], it generally does not perform very well in practice [37]. We instead use *reliability branching* which is based on a combination of two concepts for variable selection: strong branching and pseudo-costs [37]. Strong branching relies on temporarily branching, both up (to higher integer) and down (to lower integer), for every integer variable that has a fractional value in the solution of a convex QP relaxation in a given node, before committing to the variable that provides the highest value for a particular score function. The increase in objective values $\Delta_{i,j}^+$, $\Delta_{i,j}^-$ are computed when branching the integer variable $u_{i,j}$, respectively, up and down. Given these quantities, a simple scoring function score$(\cdot, \cdot)$ is computed for each integer variable, such as the product score function [38]

$$S_{i,j} = \text{score}(\Delta_{i,j}^-, \Delta_{i,j}^+) = \max(\Delta_{i,j}^+, \epsilon) \cdot \max(\Delta_{i,j}^-, \epsilon), \tag{2}$$

given a small positive value $\epsilon > 0$. Full strong branching has been empirically shown to provide smaller search trees in practice [37], but it is relatively expensive since several QP relaxations are solved in order to select one variable to branch on.

The idea of pseudo-costs aims at approximating the increase of the objective function to decide which variable to branch on, without solving additional QP relaxations. This can be done by keeping statistic information for each integer variable, i.e., the *pseudo-costs* that represent the average increase in the objective value per unit change in that particular integer variable when branching. The current pseudo-cost values are computed based on branching decisions that occurred in different parts of the B&B tree [37]. Each variable has two pseudo-costs, $\phi_{i,j}^-$ when the variable is branched "down" and $\phi_{i,j}^+$ when it is branched "up". However, at the beginning of the B&B algorithm, the pseudo-costs are not yet initialized, which is when branching decisions typically impact the tree size the most. *Reliability branching* uses strong branching to initialize the pseudo-costs until a certain condition of reliability is satisfied, e.g., it moves to using pseudo-costs for a particular variable once it has been branched on a specified number η_{rel} of times [37]. Thus, reliability branching coincides with pseudo-cost branching if $\eta_{rel} = 0$, with strong branching if $\eta_{rel} = \infty$, but typically a value $1 \leq \eta_{rel} \leq 3$ is chosen. This rule is further augmented by implementing a look ahead limit in the number of candidates, as well as a limit on the number of QP iterations in the strong branching step.

3 | CONVEX RELAXATION SOLVER: ACTIVE-SET INTERIOR POINT METHOD (ASIPM)

We reformulate the MIQP in (1) by the following compact notation

$$\min_{z} \quad \frac{1}{2} z^\top H z + h^\top z \tag{3a}$$

$$\text{s.t.} \quad G z \leq g, \quad F z = f, \tag{3b}$$

$$z_j \in \mathbb{Z}, \qquad j \in \mathcal{I}, \tag{3c}$$

where z includes all primal optimization variables and the index set $\mathcal{I}$ denotes the integer variables. For many practical MPC formulations, the objective can be written as $\frac{1}{2} z^\top H z + h^\top z = \frac{1}{2} v^\top Q v + h_v^\top v + h_y^\top y$, where $H \succeq 0$ and $Q \succ 0$, by partitioning z into v and y, entering in the linear-quadratic and linear-only terms, respectively. The objective of any MIQP (3) can be reformulated in the latter form by a change of variables, e.g., based on the eigenvalue decomposition for the Hessian $H \succeq 0$.

We focus on the efficient solution of convex QP relaxations in a B&B optimization method. For a particular node of the B&B tree, a convex QP is obtained by relaxing the integer feasibility constraint in (3c) as $\underline{z}_j \leq z_j \leq \bar{z}_j$, $\forall j \in \tilde{\mathcal{I}}$, where the index set $\tilde{\mathcal{I}} \subseteq \mathcal{I}$ denotes each integer variable that has not been fixed due to branching, or due to the presolve routine, in the current node of the B&B tree. The values $\underline{z}_j$ and $\bar{z}_j$ denote the lower and upper bound values for each integer variable $j \in \tilde{\mathcal{I}}$, respectively. This section describes an overview of recent work on the efficient implementation of infeasibility detection and early termination based on duality [47], applied to the active-set based interior point method (ASIPM) that was proposed in [46].

3.1 | Dual QP Problem Formulation

We consider the primal convex QP of the form

$$\min_{\boldsymbol{v},\boldsymbol{y}} \quad \phi(\boldsymbol{v},\boldsymbol{y}) := \frac{1}{2}\boldsymbol{v}^\top Q \boldsymbol{v} + h_{\boldsymbol{v}}^\top \boldsymbol{v} + h_{\boldsymbol{y}}^\top \boldsymbol{y} \tag{4a}$$

$$\text{s.t.} \quad G_{\boldsymbol{v}}\boldsymbol{v} + G_{\boldsymbol{y}}\boldsymbol{y} \leq g, \tag{4b}$$

$$F_{\boldsymbol{v}}\boldsymbol{v} + F_{\boldsymbol{y}}\boldsymbol{y} = f, \tag{4c}$$

where $Q \succ 0$ in the primal objective $\phi(\boldsymbol{v},\boldsymbol{y})$, and the inequality constraints (4b) include both the original inequalities from (3b) and the convex relaxations of the integer feasibility constraints. We additionally define the compact notation $\boldsymbol{z} := [\boldsymbol{v}^\top \boldsymbol{y}^\top]^\top$, $G := [G_{\boldsymbol{v}}|G_{\boldsymbol{y}}]$, $F := [F_{\boldsymbol{v}}|F_{\boldsymbol{y}}]$, $h := [h_{\boldsymbol{v}}^\top h_{\boldsymbol{y}}^\top]^\top$ and $H := \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$. The dual QP of (4) reads as

$$\max_{\mu,\lambda} \quad \psi(\mu,\lambda) := -\frac{1}{2}\|\hat{h}(\mu,\lambda)\|_{Q^{-1}}^2 - \begin{bmatrix} g \\ f \end{bmatrix}^\top \begin{bmatrix} \mu \\ \lambda \end{bmatrix} \tag{5a}$$

$$\text{s.t.} \quad G_{\boldsymbol{y}}^\top \mu + F_{\boldsymbol{y}}^\top \lambda = -h_{\boldsymbol{y}}, \tag{5b}$$

$$\mu \geq 0, \tag{5c}$$

where λ and μ denote the Lagrange multipliers for the equality and inequality constraints, respectively, and $\hat{h}(\mu,\lambda) := h_{\boldsymbol{v}} + G_{\boldsymbol{v}}^\top \mu + F_{\boldsymbol{v}}^\top \lambda$ is defined for simplifying the dual objective function $\psi(\mu,\lambda)$ in (5a).

3.2 | Primal-dual Interior Point Method

A primal-dual IPM uses a Newton-type method to solve a sequence of relaxed Karush-Kuhn-Tucker (KKT) conditions for the convex QP in (4). An iteration of the IPM typically solves the reduced linear system[51]

$$\begin{bmatrix} H & F^\top & G^\top \\ F & 0 & 0 \\ G & 0 & -W^k \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{z}^k \\ \Delta \lambda^k \\ \Delta \mu^k \end{bmatrix} = - \begin{bmatrix} r_z^k \\ r_\lambda^k \\ \bar{r}_\mu^k \end{bmatrix}, \tag{6}$$

where $W^k = \text{diag}\left(w^k\right) > 0$, $w^k \in \mathbb{R}^{n_{\text{ieq}}}$ and $w_i^k = s_i^k / \mu_i^k > 0$, given current values of the slack variables $s^k \in \mathbb{R}^{n_{\text{ieq}}}$ and Lagrange multipliers $\mu^k \in \mathbb{R}^{n_{\text{ieq}}}$ for the inequality constraints in the kth iteration of the Newton-type method. The right-hand side in (6) denotes the residual value for the optimality conditions and reads as

$$\begin{aligned} r_z^k &= H\boldsymbol{z}^k + F^\top \lambda^k + G^\top \mu^k + h, \quad r_\lambda^k = F\boldsymbol{z}^k - f, \\ r_\mu^k &= G\boldsymbol{z}^k - g + s^k, \quad r_s^k = M^k S^k 1 - \tau^k 1, \quad \bar{r}_\mu^k = r_\mu^k - M^{k^{-1}} r_s^k, \end{aligned} \tag{7}$$

based on the barrier parameter $\tau^k \to 0$ for $k \to \infty$, where $M^k = \text{diag}\left(\mu^k\right)$, $S^k = \text{diag}\left(s^k\right)$ and the slack variables are updated as $\Delta s^k = -(W^k \Delta \mu^k + M^{k^{-1}} r_s^k)$. We consider an infeasible primal-dual IPM for which the starting point $\left\{\left(\boldsymbol{z}^0, \mu^0, \lambda^0, s^0\right)\right\}$ may not be primal and/or dual feasible, but the slack variables and Lagrange multipliers are positive at each iteration, i.e., $s^k \geq 0$ and $\mu^k \geq 0$. See[51] for details on properties and implementation of primal-dual IPMs.

Remark 1

Fixed variables, i.e., $\underline{z}_j = \bar{z}_j$, and redundant constraints should be removed from each convex QP for computational efficiency. In some cases, $n_x^0 = 0$ if an initial state value $x_0 = \hat{x}_0$ is imposed in (1), and many other variables are fixed due to exact presolve operations. Therefore, we require the QP solver to allow defining a varying number of state $x_i \in \mathbb{R}^{n_x^i}$ and control variables $u_i \in \mathbb{R}^{n_u^i}$, and a varying number of inequality constraints n_c^i at each time step $i = 0, 1, \ldots, N$ in the prediction horizon.

3.3 | Active-set based Inexact Newton Method

In this paper, we use the active-set based inexact Newton implementation of `ASIPM` from[46], which allows for block-sparse structure exploitation, reduced computations, warm starting and improved numerical conditioning. For inequality constraints that are strictly active at the optimum, $s_i^k \to 0$ and $\mu_i^k > 0$ such that $w_i^k \to 0$ for $k \to \infty$. On the other hand, for inactive inequality constraints, $\mu_i^k \to 0$ and $s_i^k > 0$ such that $w_i^k \to \infty$ for $k \to \infty$. Thus, the w-values become increasingly small and large for active and inactive inequality constraints, respectively, which highlights the well-known numerical ill-conditioning that

must be tackled when implementing IPMs[52]. Based on lower and upper bound values $0 < w_{\min} \ll w_{\max}$, at each IPM iteration, we classify the inequality constraints into the following three categories:

- *inactive*: constraints that are likely to be inactive at the solution, with index set $\mathcal{I}_{\text{in}} := \{ i \mid w_i \geq w_{\max} \}$,
- *active*: constraints that are likely to be active at the solution, with index set $\mathcal{I}_{\text{act}} := \{ i \mid w_i \leq w_{\min} \}$,
- *guessing*: constraints that are uncertain, i.e., not in previous categories, with index set $\mathcal{I}_{\text{g}} := \{ i \mid w_{\min} < w_i < w_{\max} \}$.

In the inexact Newton-type algorithm of `ASIPM`[46], we solve the linearized KKT system in Eq. (6) approximately by solving the following reduced block-tridiagonal linear system

$$\left[H + \frac{1}{\epsilon} F^\top F + \frac{1}{w_{\min}} G_{\text{act}}^\top G_{\text{act}} + G_{\text{g}}^\top W_{\text{g}}^{k-1} G_{\text{g}} + \frac{1}{w_{\max}} G_{\text{in}}^\top G_{\text{in}} \right] \Delta z^k = -\bar{r}_z^k, \tag{8}$$

where the right-hand side reads $\bar{r}_z^k = r_z^k + \frac{1}{\epsilon} F^\top r_\lambda^k + \frac{1}{w_{\min}} G_{\text{act}}^\top \bar{r}_{\mu,\text{act}}^k + G_{\text{g}}^\top W_{\text{g}}^{k-1} \bar{r}_{\mu,\text{g}}^k + G_{\text{in}}^\top W_{\text{in}}^{k-1} \bar{r}_{\mu,\text{in}}^k$, by using a block-tridiagonal Cholesky factorization for the augmented Hessian matrix in (8). The inequality constraints and w-values have been reordered and grouped together according to their category, i.e., $W_{\text{in}}^k := W_{i \in \mathcal{I}_{\text{in}}}^k$, $W_{\text{act}}^k := W_{i \in \mathcal{I}_{\text{act}}}^k$ and $W_{\text{g}}^k := W_{i \in \mathcal{I}_{\text{g}}}^k$. Similarly, we split G and $\bar{r}_\mu$ into the corresponding blocks. The search directions for the Lagrange multipliers are computed as

$$\Delta \lambda^k = \frac{1}{\epsilon} \left(r_\lambda^k + F \Delta z^k \right), \qquad \Delta \mu_{\text{act}}^k = \frac{1}{w_{\min}} \left(\bar{r}_{\mu,\text{act}}^k + G_{\text{act}} \Delta z^k \right), \tag{9a}$$

$$\Delta \mu_{\text{g}}^k = W_{\text{g}}^{k-1} \left(\bar{r}_{\mu,\text{g}}^k + G_{\text{g}} \Delta z^k \right), \quad \Delta \mu_{\text{in}}^k = W_{\text{in}}^{k-1} \left(\bar{r}_{\mu,\text{in}}^k + G_{\text{in}} \Delta z^k \right), \tag{9b}$$

and the update to the slack variables as $\Delta s^k = -M^{k-1}(S^k \Delta \mu^k + r_s^k)$.

3.4 | Early Termination based on Duality and Infeasibility Detection

We do not need to solve a convex QP relaxation in the B&B method if

- the convex QP relaxation is infeasible,
- the optimal solution has an objective value that exceeds the current global upper bound.

In both cases, the node, and hence the corresponding subtree, can be pruned from the B&B tree. A considerable computational effort can be avoided if the above scenarios are detected early, i.e., more quickly than the time for solving the convex QP relaxations. Based on our work in[47], we describe a tailored early termination strategy for infeasible primal-dual IPMs to handle both cases and to reduce the computational effort of the B&B method without affecting the quality of the optimal solution.

Due to duality properties, see, e.g.,[53], for a dual feasible point (μ, λ) that satisfies (5b)-(5c) and a primal feasible point (v, y) that satisfies (4b)-(4c)

$$\psi(\mu, \lambda) \leq \psi^\star \leq \phi^\star \leq \phi(v, y), \tag{10}$$

where $\phi^\star$ and $\psi^\star$ are the primal and dual optima, respectively. Based on (10), we propose an approach to find a dual feasible point that allows for early termination when $\psi(\mu, \lambda) > \text{UB}$ for the current upper bound (UB) to the optimum of the MIQP.

3.4.1 | Projection Strategy for Dual Feasibility

A dual feasible solution, i.e., (μ^k, λ^k) satisfying (5b)-(5c) is required in order to perform early termination based on the duality result in (10). Since an infeasible IPM generally does not provide a solution that satisfies the equality constraint in (5b) until convergence, we proposed[47] a projection step to compute new values $(\mu^+, \lambda^+) = (\mu^k + \Delta\mu, \lambda^k + \Delta\lambda)$ satisfying (5b) and (5c) by solving the optimization problem

$$\begin{aligned} \min_{\Delta v, \Delta \lambda, \Delta \mu} \quad & \frac{1}{2} \|\Delta v\|_Q^2 + \frac{1}{2} \|\Delta \lambda\|_{\epsilon_{\text{dual}}}^2 + \frac{1}{2} \|\Delta \mu\|_{W^k}^2 \\ \text{s.t.} \quad & \begin{bmatrix} Q \\ 0 \end{bmatrix} \Delta v + F^\top \Delta \lambda + G^\top \Delta \mu = - \begin{bmatrix} 0 \\ r_y^k \end{bmatrix}, \end{aligned} \tag{11}$$

where $r_y^k := F_y^\top \lambda^k + G_y^\top \mu^k + h_y$, $W^k = \text{diag}\left(w^k \right)$ and $w_i^k = \frac{s_i^k}{\mu_i^k} > 0$.

There are three advantages for the projection (11) over a standard minimum-norm projection. Neither of these projections directly enforces the positivity constraints $\mu^k + \Delta \mu > 0$, which would require solving an inequality constrained QP. A first advantage of (11) is that, since $w_i^k = \frac{s_i^k}{\mu_i^k} > 0$, the term $\|\Delta \mu\|_{W^k}^2 = \sum_i \left(\frac{s_i^k}{\mu_i^k} \Delta \mu_i^2 \right)$ in the objective of (11) penalizes the step $\Delta \mu_i$ to remain small when $\frac{s_i^k}{\mu_i^k}$ is relatively large, i.e., when $\mu_i^k > 0$ is close to zero. This makes the step smaller when approaching the

positivity constraint, such that it is more likely to satisfy $\mu_i^k + \Delta\mu_i > 0$ without making the update step always small. Second, the solution $(\Delta\mu, \Delta\lambda)$ to the optimization problem (11) is equivalent[47] to solving the symmetric system

$$
\begin{bmatrix} H & F^\top & G^\top \\ F & -\epsilon_{\text{dual}}I & 0 \\ G & 0 & -W^k \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\lambda \\ \Delta\mu \end{bmatrix} = - \begin{bmatrix} \begin{bmatrix} 0 \\ r_y^k \end{bmatrix} \\ 0 \\ 0 \end{bmatrix},
\tag{12}
$$

which is an IPM iteration similar to (6) with the only differences being the right-hand side and the augmented Lagrangian type regularization $\epsilon_{\text{dual}} > 0$. The equivalence means that the same block-tridiagonal matrix factorization for the solution of the reduced linear system (8) in each iteration of ASIPM[46] can be reused to compute the projection step in (12). Third, the projection (12) aims at retaining the IPM progress towards the optimum, since the projection step does not increase the residual value for the remaining optimality conditions in (7), due to the zero elements in the right-hand side of (12).

3.4.2 | Early Termination Strategy

We use the projection step in (12) for our early termination strategy, see Algorithm 1. As discussed later, one dual objective evaluation is computationally cheaper than a projection on a dual feasible point. Therefore, Algorithm 1 performs the projection in (12) if and only if the dual objective $\psi(\mu^k, \lambda^k)$ is larger than the current UB (line 5). Multiple evaluations of the projection step in (12), reusing the same matrix factorization, may be needed to ensure dual feasibility $\|F_y^\top\lambda + G_y^\top\mu + h_y\| < tol$ when using the inexact Newton implementation of[46] in the ASIPM solver.

Algorithm 1 Early termination for IPM in B&B method.

Input: Warm start $\{(z^0, \mu^0, \lambda^0, s^0)\}$, tol, and UB.

1: **while** $\max\{\tau^k, \|r^k\|\} > tol$ **do**
2:      **if** $\psi(\mu^k, \lambda^k) > $ UB $\wedge$ dual_feasible **then**
3:          **break** while loop.   ▷ Early termination
4:      **else if** $\psi(\mu^k, \lambda^k) > $ UB **then**
5:          Compute projection step $(\Delta\mu, \Delta\lambda)$ in (12).
6:          $\mu \leftarrow \mu^k + \Delta\mu$, $\lambda \leftarrow \lambda^k + \Delta\lambda$, and $r_y \leftarrow F_y^\top\lambda + G_y^\top\mu + h_y$.
7:          **if** $\mu > 0$ $\wedge$ $\|r_y\| < tol$ **then**
8:              $\mu^k \leftarrow \mu$, $\lambda^k \leftarrow \lambda$, $r_y^k \leftarrow r_y$, and dual_feasible $\leftarrow 1$.
9:              **if** $\psi(\mu^k, \lambda^k) > $ UB **then break** while loop.   ▷ Early termination
10:          **end if**
11:      **end if**
12:      Perform an IPM iteration (6), e.g., see[46].
13: **end while**

The standard iterates of an IPM can also be used to generate certificates of infeasibility[54]. Proposition 1, proved in[47], states conditions such that the dual objective $\psi(\mu^k, \lambda^k)$ is unbounded in the limit of the primal-dual IPM iterations. Therefore, our proposed early termination strategy is effective for infeasibility detection and, given a tight UB value from the B&B optimization method, it may lead to termination even before a certificate of infeasibility can be found.

Proposition 1. If the sequence of primal-dual iterates $\{(z^k, \mu^k, \lambda^k, s^k)\}$ of the IPM satisfy $\mu^{k^\top} s^k \le \mu^{0^\top} s^0$ and $\|\mu^k\| \to \infty$, then the dual objective $\psi(\mu^k, \lambda^k) \to \infty$.

3.4.3 | Computational Complexity

The proposed early termination strategy requires two computational steps, i.e., the projection and the dual objective evaluation, which are typically not needed in a standard IPM. Considering the optimal control structured program (1), the evaluation of the dual objective value (5a) requires $N(n^2 + 2nm + 2np)$ operations to compute $L^{-1}(G_v^\top\mu + F_v^\top\lambda)$ based on a block-diagonal Cholesky factorization $Q = LL^\top$, in which n, m and p are the number of variables in v (see (4)), the number of equality and inequality constraints, respectively, per control interval, and N is the number of control intervals.

Based on (12), we can perform one projection step at the computational cost of one IPM iteration. This allows for reusing the corresponding matrix factorization in the subsequent IPM iteration if the projection is not successful. Based on the particular ASIPM implementation from Section 3.3, as originally proposed in[46], one iteration requires a block-tridiagonal Cholesky factorization for the matrix in (8), for which the dominant terms in the computational cost are

$$N \left(\frac{7}{3} n_x^3 + 4 n_x^2 n_u + 2 n_x n_u^2 + \frac{1}{3} n_u^3 \right), \tag{13}$$

where n_x and n_u denote the number of state and control variables per interval in (1), respectively. Then, reusing this matrix factorization, the linear system for the projection step in (12) can be solved by

$$N \left(6 n_x^2 + 8 n_x n_u + 2 n_u^2 + 2(n_x + n_u)(m + p) \right), \tag{14}$$

operations for the resulting block-structured forward and backward substitution. Since $n \le (n_x + n_u)$, and often $n \ll (n_x + n_u)$ due to many auxiliary variables in hybrid systems[2] for which the Hessian contribution is zero, the cost for a dual objective evaluation is considerably smaller than the projection cost (14), as anticipated.

4 | EXACT PRESOLVE TECHNIQUES FOR MIXED-INTEGER OPTIMAL CONTROL

A presolve routine is a collection of computationally efficient operations that should be used in each node of the B&B method (see Figure 1) before solving the convex relaxation, in order to remove decision variables, and to remove or tighten constraints[44]. Presolve techniques are often crucial in strengthening convex relaxations such that typically fewer nodes need to be explored in a B&B optimization method, sometimes to such an extent that seemingly intractable problems become computationally tractable. We present a collection of tailored variants of presolve techniques with block-sparse structure exploitation for mixed-integer optimal control, based on presolve routines in state-of-the-art solvers for general-purpose MIQPs, e.g., see[43]. We explicitly refer to these presolve techniques as *exact* operations, differentiating from the heuristic approach in Section 5, and to emphasize that all presolve methods in the present Section preserve feasibility and optimality, i.e., the reduced problem is infeasible or unbounded only if the original problem is infeasible or unbounded, and any feasible or optimal solution of the reduced problem can be mapped to a feasible or optimal solution of the original problem. For example, an exact presolve method will fix a binary variable to 0 or 1 only if the method can guarantee that this variable is fixed to that same value in an optimal solution.

4.1 | Block-Structured Domain Propagation

Several strengthening techniques are implemented as part of "presolve" routines in state-of-the-art commercial solvers[43]. One technique that is particularly suitable to mixed-integer optimal control is based on *domain propagation*, in which the goal is to strengthen bound values based on the constraints of the MIQP in (1). In previous work[29], we suggested to apply domain propagation to the inequality constraints of a condensed MIOCP formulation, i.e., to the smaller but dense problem formulation after numerically eliminating each of the state variables based on the state dynamic constraints in (1b). However, based on a simple illustrative example, we show that it can often be advantageous to apply each of the presolve operations, and domain propagation in particular, directly to the block-sparse MIQP formulation in (1), including the state variables and state dynamic equality constraints (1b). The key insight is that tight lower and upper bounds for state variables can be used to tighten lower and upper bounds on control variables and vice versa.

Example 1

$$\min_{X, U, \delta} \quad \sum_{i=0}^{2} \frac{1}{2} \begin{bmatrix} x_i \\ u_i \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} \tag{15a}$$

$$\text{s.t.} \quad x_0 = 0, \tag{15b}$$

$$x_{i+1} = x_i + u_i, \qquad \forall i \in \{0, 1\}, \tag{15c}$$

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \le \begin{bmatrix} x_i \\ u_i \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \forall i \in \{0, 1, 2\}, \tag{15d}$$

$$x_2 \ge 2 - 10\delta, \qquad \delta \in \{0, 1\}, \tag{15e}$$

$$\min_{U, \delta} \quad \frac{1}{2} \left(\begin{bmatrix} u_0 \\ u_1 \end{bmatrix}^\top \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} + u_2^2 \right) \tag{16a}$$

$$\text{s.t.} \quad \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \le \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{16b}$$

$$-1 \le u_0 + u_1 \le 1, \tag{16c}$$

$$u_0 + u_1 \ge 2 - 10\delta, \qquad \delta \in \{0, 1\}. \tag{16d}$$

A standard bound strengthening procedure, as explained in [43,29], can be applied to each of the inequality constraints in either the block-sparse MIQP formulation in (15) or the equivalent but condensed MIQP in (16). For example, the inequality in (15e) can be used to compute a bound on the binary variable $\delta \in \{0, 1\}$ as follows:

$$10\delta \geq 2 - x_2 \underset{x_2 \leq 1}{\Rightarrow} 10\delta \geq 2 - 1 \Rightarrow \delta \geq \frac{1}{10} \underset{\delta \in \{0,1\}}{\Rightarrow} \delta = 1,$$

resulting in a fixing of the binary optimization variable $\delta = 1$. Alternatively, using the condensed inequality in (16d), the same procedure leads to

$$10\delta \geq 2 - u_0 - u_1 \underset{u_0 \leq 1, \, u_1 \leq 1}{\Rightarrow} 10\delta \geq 2 - 1 - 1 \Rightarrow \delta \geq 0,$$

which means that the binary optimization variable $\delta \in \{0, 1\}$ cannot be eliminated in this case, even though the condensed MIQP in (16) is equivalent to the block-sparse MIQP formulation in (15). □

Remark 2

In Example 1, for the condensed inequality in (16d), fixing of the binary variable $\delta = 1$ could be detected by exploiting a bound for the expression $u_0 + u_1 \leq 1$ from (16c). However, as discussed in [43, Section 5.4], a bound strengthening procedure considering multiple constraints at once is typically too expensive for MIQP solvers, but more general optimization-based bound tightening (OBBT) techniques are common for mixed-integer nonlinear programming (MINLP).

Motivated by the above illustrative example, we propose a novel block-sparse variant of domain propagation as described in Algorithm 2 that is tailored to the MIQP formulation in (1). The method consists of a forward iterative procedure for $i = 0, 1, \ldots, N$ (see Line 1-14), followed by a backward iterative procedure for $i = N - 1, \ldots, 0$ (see Line 15-21). The general intuition behind the forward-backward propagation is to have a quick propagation of variable bounds from near the beginning of the prediction horizon towards the end of the prediction horizon, as well as a quick propagation of variable bounds from near the end of the prediction horizon towards the beginning. The proposed forward-backward implementation can reduce the amount of times that the domain propagation in Alg. 2 needs to be called in order to achieve a particular amount of bound strengthening in practice, even though such a performance improvement cannot be guaranteed in general. However, by design, it can be guaranteed that the updated bound values $[\underline{z}_i^+, \bar{z}_i^+]$ that are computed by Alg. 2 are tighter than the original bound values $[\underline{z}_i, \bar{z}_i]$, i.e., $\underline{z}_i \leq \underline{z}_i^+ \leq z_i \leq \bar{z}_i^+ \leq \bar{z}_i$ for $i \in \{0, 1, \ldots, N\}$, without eliminating any feasible solution of the MIQP in (1) when replacing the bound values by the updated values $[\underline{z}_i^+, \bar{z}_i^+]$ in (1c). Algorithm 2 can result in strengthening of bound values for both continuous and integer/binary optimization variables. In addition, the tailored block-sparse MIQP structure exploitation in Alg. 2 considerably reduces the computational cost for each operation in the domain propagation.

Each iteration of the forward procedure for $i = 0, 1, \ldots, N$ (Line 1-14) performs domain propagation for each variable $z_{i,j}$ for $j = 1, \ldots, n_x + n_u$ based on the affine inequality constraints $\underline{c}_i \leq E_i z_i \leq \bar{c}_i$ (see Line 2-6), followed by domain propagation for each state variable $x_{i+1,j}$ for $j = 1, \ldots, n_x$ based on the state dynamic equality constraints $x_{i+1} = a_i + F_i z_i$ (see Line 7-13). The computation of $\underline{z}_{i,j}$ and $\bar{z}_{i,j}$ on Line 3 and 4, respectively, is defined using a compact OBBT notation. However, as described in the next section, we instead perform a single-row approximation for each inequality constraint individually, in order to reduce the computational cost for each iteration of the domain propagation. An MIQP subproblem is detected to be infeasible whenever the gap between a lower and upper bound value is below a particular threshold value $-\epsilon$, where ϵ is a small positive value, for example, see Lines 5, 11 and 19. The computation of $\underline{a}_i = \min\{a_i + F_i z_i\} = a_i + \left(F_i^+ \underline{z}_i + F_i^- \bar{z}_i\right)$ and $\bar{a}_i = \max\{a_i + F_i z_i\} = a_i + \left(F_i^+ \bar{z}_i + F_i^- \underline{z}_i\right)$ on Line 8 is used for domain propagation based on the state dynamics, where F_i^+, F_i^- contain all positive and negative elements of the matrix $F_i = F_i^+ + F_i^-$, respectively. Each iteration of the backward procedure $i = N - 1, \ldots, 0$ (see Line 15-21) performs domain propagation for each variable $z_{i,j}$ for $j = 1, \ldots, n_x + n_u$ based on the state dynamic equality constraints $x_{i+1} = a_i + F_i z_i$. Similarly, OBBT is used to update the bound values $\underline{z}_{i,j}$ and $\bar{z}_{i,j}$ on Line 17 and 18, respectively, but a computationally efficient single-row approximation can be used.

4.2 | Approximation of Optimization-based Bound Tightening

The computation of bound values $\underline{z}_{i,j}$, $\bar{z}_{i,j}$ on Lines 3-4 and Lines 17-18 of Alg. 2 require the solution of a linear programming (LP) problem. Since this operation needs to be performed for each variable on each time step in the horizon and for each domain propagation call in each iteration of the presolve routine, it is necessary to perform a computationally cheap single-row approximation instead as implemented in Alg. 3 based on the following lemma.

Algorithm 2 Block-sparse forward-backward operations for domain propagation and bound strengthening

---

Input: Bound values $[\underline{z}_i, \bar{z}_i]$, $E_i = \begin{bmatrix} C_i & D_i \end{bmatrix}$, $F_i = \begin{bmatrix} A_i & B_i \end{bmatrix}$, $i \in \{0, \dots, N\}$ and MIQP of the form (1).

1: **for** $i = 0, \dots, N$ **do**   ▷ Forward domain propagation
2:    **for** $j = 1, \dots, (n_x + n_u)$ **do**   ▷ Affine inequality constraints: $\underline{c}_i \leq E_i z_i \leq \bar{c}_i$
3:      $\underline{z}_{i,j} \leftarrow \min\{z_{i,j} : \underline{c}_i \leq E_i z_i \leq \bar{c}_i, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\}$, see Alg. 3
4:      $\bar{z}_{i,j} \leftarrow \max\{z_{i,j} : \underline{c}_i \leq E_i z_i \leq \bar{c}_i, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\}$, see Alg. 3
5:      **if** $\bar{z}_{i,j} - \underline{z}_{i,j} < -\epsilon$ **then return** `infeasible_flag`   ▷ MIQP subproblem infeasible
6:    **end for**
7:    **if** $i < N$ **then**
8:      $\underline{a}_i \leftarrow a_i + \left(F_i^+ \underline{z}_i + F_i^- \bar{z}_i\right)$ and $\bar{a}_i \leftarrow a_i + \left(F_i^+ \bar{z}_i + F_i^- \underline{z}_i\right)$.
9:      **for** $j = 1, \dots, n_x$ **do**   ▷ State dynamic constraints: $x_{i+1} = a_i + F_i z_i$
10:        $\underline{x}_{i+1,j} \leftarrow \max(\underline{x}_{i+1,j}, \underline{a}_{i,j})$ and $\bar{x}_{i+1,j} \leftarrow \min(\bar{x}_{i+1,j}, \bar{a}_{i,j})$.
11:        **if** $\bar{x}_{i+1,j} - \underline{x}_{i+1,j} < -\epsilon$ **then return** `infeasible_flag`   ▷ MIQP subproblem infeasible
12:      **end for**
13:    **end if**
14: **end for**
15: **for** $i = N-1, \dots, 0$ **do**   ▷ Backward domain propagation
16:    **for** $j = 1, \dots, (n_x + n_u)$ **do**   ▷ State dynamic constraints: $x_{i+1} = a_i + F_i z_i$
17:      $\underline{z}_{i,j} \leftarrow \min\{z_{i,j} : x_{i+1} - a_i - F_i z_i = 0, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\}$
18:      $\bar{z}_{i,j} \leftarrow \max\{z_{i,j} : x_{i+1} - a_i - F_i z_i = 0, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\}$
19:      **if** $\bar{z}_{i,j} - \underline{z}_{i,j} < -\epsilon$ **then return** `infeasible_flag`   ▷ MIQP subproblem infeasible
20:    **end for**
21: **end for**

Output: Updated bound values $[\underline{z}_i^+, \bar{z}_i^+]$, $i \in \{0, \dots, N\}$.

Lemma 2

The updated bound values $\underline{z}_{i,j}^+$, $\bar{z}_{i,j}^+$ that are computed by Alg. 3 are a conservative approximation of optimization-based bound tightening, i.e., the following relationships hold

$$\underline{z}_{i,j}^+ = \max_{k \in \{1,\dots,n_c\}} \left(\min\{z_{i,j} : \underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \right) \tag{17a}$$

$$\leq \min\{z_{i,j} : \underline{c}_i \leq E_i z_i \leq \bar{c}_i, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \quad \leq \min\{z_{i,j} : (1b) - (1e)\}, \tag{17b}$$

where $E_{i,k,:}$ denotes the kth row of the matrix E_i, and similarly for the updated upper bound values

$$\bar{z}_{i,j}^+ = \min_{k \in \{1,\dots,n_c\}} \left(\max\{z_{i,j} : \underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \right) \tag{18a}$$

$$\geq \max\{z_{i,j} : \underline{c}_i \leq E_i z_i \leq \bar{c}_i, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \quad \geq \max\{z_{i,j} : (1b) - (1e)\}. \tag{18b}$$

Proof. The relationships in (17b) follow directly from the following inequalities

$$\min\{z_{i,j} : \underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \leq \min\{z_{i,j} : \underline{c}_i \leq E_i z_i \leq \bar{c}_i, \;\; \underline{z}_i \leq z_i \leq \bar{z}_i\} \leq \min\{z_{i,j} : (1b) - (1e)\},$$

which holds for each $k \in \{1, \dots, n_c\}$, due to the fact that an optimal objective value of a minimization problem only decreases after removing one or multiple constraints. The relationships in (18b) can be proved by a similar argument. □

A compact notation is used in Alg. 3 for matrices E_i^+, E_i^- which contain all positive and negative elements of the matrix $E_i = E_i^+ + E_i^-$, respectively. Line 10 of Alg. 3 includes rounding up $\lceil \underline{d} \rceil$ and rounding down $\lfloor \bar{d} \rfloor$ of the lower and upper bound values, respectively, if $z_{i,j}$ is an integer or binary optimization variable.

Algorithm 3 Single-row approximation of optimization-based bound tightening for variable $z_{i,j}$

Input: Bound values $[\underline{z}_i, \bar{z}_i]$, $E_i = [C_i \; D_i]$, $i \in \{0, \dots, N\}$, $j \in \{1, \dots, n_z\}$ and MIQP of the form (1).

1: $\tilde{\bar{c}} \leftarrow \bar{c}_i - \left(E_i^+ \underline{z}_i + E_i^- \bar{z}_i \right)$ and $\tilde{\underline{c}} \leftarrow \underline{c}_i - \left(E_i^+ \bar{z}_i + E_i^- \underline{z}_i \right)$.

2: $\underline{d} \leftarrow -\infty$ and $\bar{d} \leftarrow \infty$.

3: **for** $k = 1, \dots, n_c$ **do** ⊳ Affine inequality constraints: $\underline{c}_i \leq E_i z_i \leq \bar{c}_i$

4:     **if** $E_{i,k,j} > \epsilon$ **then**

5:         $\underline{d}_k \leftarrow \frac{\tilde{\underline{c}}_k + E_{i,k,j} \bar{z}_{i,j}}{E_{i,k,j}}$ and $\bar{d}_k \leftarrow \frac{\tilde{\bar{c}}_k + E_{i,k,j} \underline{z}_{i,j}}{E_{i,k,j}}$.

6:     **else if** $E_{i,k,j} < -\epsilon$ **then**

7:         $\underline{d}_k \leftarrow \frac{\tilde{\bar{c}}_k + E_{i,k,j} \bar{z}_{i,j}}{E_{i,k,j}}$ and $\bar{d}_k \leftarrow \frac{\tilde{\underline{c}}_k + E_{i,k,j} \underline{z}_{i,j}}{E_{i,k,j}}$.

8:     **end if**

9: **end for**

10: **if** $z_{i,j}$ is integer **then** $\underline{d} \leftarrow \lceil \underline{d} \rceil$ and $\bar{d} \leftarrow \lfloor \bar{d} \rfloor$. ⊳ Rounding integer variable bounds

11: $\underline{z}_{i,j}^+ \leftarrow \max(\underline{z}_{i,j}, \max_k(\underline{d}_k))$ and $\bar{z}_{i,j}^+ \leftarrow \min(\bar{z}_{i,j}, \min_k(\bar{d}_k))$.

Output: $\underline{z}_{i,j}^+ = \max_{k \in \{1, \dots, n_c\}} \left(\min\{z_{i,j} : \underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}, \; \underline{z}_i \leq z_i \leq \bar{z}_i \} \right)$,

$\bar{z}_{i,j}^+ = \min_{k \in \{1, \dots, n_c\}} \left(\max\{z_{i,j} : \underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}, \; \underline{z}_i \leq z_i \leq \bar{z}_i \} \right)$.

4.3 | Trivial Constraints and Dual Fixings

Algorithm 4 illustrates how single-row bounding for each of the affine inequality constraints can be used to detect an MIQP subproblem to be infeasible (see Line 3), for example, if

$$\min\{E_{i,k,:} z_i : \quad \underline{z}_i \leq z_i \leq \bar{z}_i \} = E_{i,k,:}^+ \underline{z}_i + E_{i,k,:}^- \bar{z}_i > \bar{c}_{i,k}, \tag{19}$$

where $E_{i,k,:}$ denotes the kth row of the matrix E_i, and similarly for the lower bound of $\underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}$, $k \in \{1, \dots, n_c\}$. In addition, single-row bounding can be used to detect lower and/or upper bounds of an affine inequality constraint to be redundant (see Line 5-6), for example,

$$\bar{c}_{i,k} \geq \max\{E_{i,k,:} z_i : \quad \underline{z}_i \leq z_i \leq \bar{z}_i \} = E_{i,k,:}^+ \bar{z}_i + E_{i,k,:}^- \underline{z}_i \quad \Rightarrow \quad \bar{c}_{i,k} \leftarrow \infty \quad \text{(upper bound redundant)}, \tag{20}$$

and similarly for the lower bound of each affine inequality constraint $\underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}$, $k \in \{1, \dots, n_c\}$. Finally, if an optimization variable does not enter the state dynamic equality constraints (see Line 9 of Alg. 4) and if the variable enters each of the non-redundant inequality constraints with a coefficient of the same sign, then the variable can be fixed to the lower (Line 10) or the upper bound (Line 11) value, depending on the sign of the constraint coefficients and the sign in the cost function. Lines 7-13 of Alg. 4 form a tailored block-sparse variant of *dual fixing*, which is described more generally in [43].

4.4 | Constraint Coefficient Strengthening

In *coefficient strengthening*, we aim to modify coefficients of the affine inequality constraints (1d) to tighten the convex QP relaxation without affecting the integer-feasible MIQP solutions. We present a block-sparse variant of coefficient strengthening in Alg. 5, as described more generally in [48]. Let us define constraint domination as follows:

Definition 3 (see [43,48])
Given two inequality constraints $e_1^\top z \leq f_1$ and $e_2^\top z \leq f_2$, where $e_1^\top$ and $e_2^\top$ denote row vectors, $z \in \mathbb{R}^n$ for which $z_j \in \mathbb{Z}$, $\forall j \in \mathcal{I}$ and $\underline{z} \leq z \leq \bar{z}$, then $e_1^\top z \leq f_1$ *dominates* $e_2^\top z \leq f_2$ if

$$\{z \in \mathbb{R}^n \mid \underline{z} \leq z \leq \bar{z}, \; e_1^\top z \leq f_1 \} \quad \subset \quad \{z \in \mathbb{R}^n \mid \underline{z} \leq z \leq \bar{z}, \; e_2^\top z \leq f_2 \}, \tag{21}$$

i.e., the feasible region of the convex QP relaxation for $e_1^\top z \leq f_1$ is strictly contained in the feasible region for $e_2^\top z \leq f_2$, given the variable bound values $\underline{z} \leq z \leq \bar{z}$.

Using the latter definition, the overall aim of coefficient strengthening is to reformulate one or multiple affine inequality constraints (1d) as constraints that *dominate* the original inequality constraints, without removing any feasible MIQP solution. We demonstrate the idea of coefficient strengthening based on following simple illustrative example.

Algorithm 4 Block-sparse redundant inequality constraint detection and dual fixings

---

Input: Bound values $[\underline{z}_i, \bar{z}_i]$, $E_i = \begin{bmatrix} C_i & D_i \end{bmatrix}$, $[\underline{c}_i, \bar{c}_i]$, $i \in \{0, \dots, N\}$ and MIQP of the form (1).

1: **for** $i = 0, \dots, N$ **do**
2:      $\tilde{\bar{c}} \leftarrow \bar{c}_i - \left(E_i^+ \underline{z}_i + E_i^- \bar{z}_i \right)$ and $\underline{\tilde{c}} \leftarrow \underline{c}_i - \left(E_i^+ \bar{z}_i + E_i^- \underline{z}_i \right)$. $\triangleright$ Inequality constraints: $\underline{c}_i \leq E_i z_i \leq \bar{c}_i$
3:      **if** $\min_k(\tilde{\bar{c}}_k) < -\epsilon \ \vee \ \max_k(\underline{\tilde{c}}_k) > \epsilon$ **then return** `infeasible_flag` $\triangleright$ MIQP subproblem infeasible
4:      $\tilde{\bar{c}} \leftarrow \bar{c}_i - \left(E_i^+ \bar{z}_i + E_i^- \underline{z}_i \right)$ and $\underline{\tilde{c}} \leftarrow \underline{c}_i - \left(E_i^+ \underline{z}_i + E_i^- \bar{z}_i \right)$.
5:      $\bar{c}_{i,k} \leftarrow \infty$ **if** $\tilde{\bar{c}}_k > -\epsilon$, $\forall k = 1, \dots, n_\mathrm{c}$ $\triangleright$ Upper bound constraint redundant
6:      $\underline{c}_{i,k} \leftarrow -\infty$ **if** $\underline{\tilde{c}}_k < \epsilon$, $\forall k = 1, \dots, n_\mathrm{c}$ $\triangleright$ Lower bound constraint redundant
7:      **for** $j = 1, \dots, n_\mathrm{u}$ **do**
8:          $\tilde{D} \leftarrow \left[\{D_{i,k,j}\}_{\forall k : \bar{c}_{i,k} < \epsilon_{\max}} \quad \{-D_{i,k,j}\}_{\forall k : \underline{c}_{i,k} > -\epsilon_{\max}} \right]$. $\triangleright$ Non-redundant inequality constraints
9:          **if** $\|B_{i,:,j}\| < \epsilon \ \wedge \ \|H_{i,:,n_\mathrm{x}+j}\| < \epsilon \ \wedge \ \|\bar{u}_{i,j} - \underline{u}_{i,j}\| > \epsilon$ **then**
10:              **if** $\min_k(\tilde{D}_k) > -\epsilon \ \wedge \ r_{i,j} > -\epsilon$ **then** $\bar{u}_{i,j} \leftarrow \underline{u}_{i,j}$ $\triangleright$ Fixing auxiliary variable to lower bound
11:              **if** $\max_k(\tilde{D}_k) < \epsilon \ \wedge \ r_{i,j} < \epsilon$ **then** $\underline{u}_{i,j} \leftarrow \bar{u}_{i,j}$ $\triangleright$ Fixing auxiliary variable to upper bound
12:          **end if**
13:      **end for**
14: **end for**

Output: Updated bound values $[\underline{z}_i, \bar{z}_i]$ and $[\underline{c}_i, \bar{c}_i]$, $i \in \{0, \dots, N\}$.

Example 4

Let us consider an MIQP with a continuous variable $x \in \mathbb{R}$, which is bounded $1 \leq x \leq 3$, and a binary variable $\delta \in \{0, 1\}$. Coefficient strengthening for an inequality constraint $2 \leq x + 100\,\delta$ then results in a dominating constraint $2 \leq x + \delta$, i.e.,

$$\{x \in \mathbb{R}, \delta \in \mathbb{R} \mid 1 \leq x \leq 3, \ 0 \leq \delta \leq 1, \ 2 \leq x + \delta\} \quad \subset \quad \{x \in \mathbb{R}, \delta \in \mathbb{R} \mid 1 \leq x \leq 3, \ 0 \leq \delta \leq 1, \ 2 \leq x + 100\,\delta\}. \quad (22)$$

Mixed-integer inequality constraints of the form in (22) are common, e.g., when using a "big-M" formulation[55] in MIOCPs, such that coefficient strengthening can be used to automatically reduce the large coefficient value $M > 0$ and tighten the convex QP relaxations. Algorithm 5 describes a systematic approach to strengthen the coefficients of each affine inequality constraint $\underline{c}_{i,k} \leq E_{i,k,:} z_i \leq \bar{c}_{i,k}$, $k \in \{1, \dots, n_\mathrm{c}\}$, with respect to each integer or binary optimization variable $u_{i,j}$, $\forall j \in \mathcal{I}_i$ in (1e).

4.5 | Binary Variable Probing

The general idea of probing is to select a binary variable, which is set tentatively to zero or one in order to derive further variable fixings and/or tightened inequality constraints, see[43,48]. For example, let $u_{i,j} \in \{0, 1\}$ be a binary variable, and let us define the lower and upper bounds $\underline{z}_i^0 \leq z_i \leq \bar{z}_i^0$ for $i = 0, \dots, N$ that have been deduced from setting $u_{i,j} = 0$ using one or multiple iterations of presolve operations, e.g., using the forward-backward domain propagation in Alg. 2. Similarly, the lower and upper bounds $\underline{z}_i^1 \leq z_i \leq \bar{z}_i^1$ for $i = 0, \dots, N$ have been deduced from setting $u_{i,j} = 1$. Our tailored implementation of binary variable probing in Alg. 6 is based on following observations:

- If both $u_{i,j} = 0$ and $u_{i,j} = 1$ leads to an infeasible problem $\Rightarrow$ problem is infeasible

- If $u_{i,j} = 0$ leads to infeasible problem $\Rightarrow$ $u_{i,j} = 1$ and $\underline{z}_{i,k} = \underline{z}_{i,k}^1$, $\bar{z}_{i,k} = \bar{z}_{i,k}^1$, $i = 0, \dots, N$, $k = 1, \dots, (n_\mathrm{x} + n_\mathrm{u})$

- If $u_{i,j} = 1$ leads to infeasible problem $\Rightarrow$ $u_{i,j} = 0$ and $\underline{z}_{i,k} = \underline{z}_{i,k}^0$, $\bar{z}_{i,k} = \bar{z}_{i,k}^0$, $i = 0, \dots, N$, $k = 1, \dots, (n_\mathrm{x} + n_\mathrm{u})$

- Bound values updated for each variable $\underline{z}_{i,k} = \min\{\underline{z}_{i,k}^0, \underline{z}_{i,k}^1\}$ and $\bar{z}_{i,k} = \max\{\bar{z}_{i,k}^0, \bar{z}_{i,k}^1\}$, $i = 0, \dots, N$, $k = 1, \dots, (n_\mathrm{x} + n_\mathrm{u})$

- If $\exists k : \underline{z}_{i,k}^0 = \bar{z}_{i,k}^0$ and $\underline{z}_{i,k}^1 = \bar{z}_{i,k}^1$ $\Rightarrow$ $z_{i,k}$ can be substituted as $z_{i,k} = \bar{z}_{i,k}^0 + (\bar{z}_{i,k}^1 - \bar{z}_{i,k}^0)\,u_{i,j}$

For simplicity, the last observation is not included in Alg. 6. Binary variable probing is a relatively simple strategy that can be very effective at reducing the B&B search tree, but it can become computationally expensive. Therefore, a limit on the number of probing iterations, $n < n_\mathrm{probing}$ (see Line 5 and Line 11), and/or a timeout is needed to ensure computational efficiency.

Algorithm 5 Block-sparse inequality constraint coefficient strengthening

---

Input: Bound values $[\underline{z}_i, \bar{z}_i]$, $E_i = \begin{bmatrix} C_i & D_i \end{bmatrix}$, $[\underline{c}_i, \bar{c}_i]$, $i \in \{0, \dots, N\}$ and MIQP of the form (1).

1: **for** $i = 0, \dots, N$ **do**
2:      $\tilde{\bar{c}} \leftarrow \bar{c}_i - \left(E_i^+ \bar{z}_i + E_i^- \underline{z}_i \right)$ and $\tilde{\underline{c}} \leftarrow \left(E_i^+ \underline{z}_i + E_i^- \bar{z}_i \right) - \underline{c}_i$.   ▷ Inequality constraints: $\underline{c}_i \le E_i z_i \le \bar{c}_i$
3:      **for** $j = 1, \dots, n_\mathrm{u}$ **and if** $u_{i,j}$ is integer $\wedge \|\bar{u}_{i,j} - \underline{u}_{i,j}\| > \epsilon$ **then do**
4:          **for** $k = 1, \dots, n_\mathrm{c}$ **do**
5:              **if** $\bar{c}_{i,k} < \epsilon_{\max} \wedge \underline{c}_{i,k} < -\epsilon_{\max}$ **then**
6:                  $\tilde{d} \leftarrow \tilde{\bar{c}}_k + |D_{i,k,j}| \left(\bar{u}_{i,j} - \underline{u}_{i,j} \right)$.
7:                  **if** $\tilde{d} > \epsilon \wedge D_{i,k,j} > \epsilon$ **then**
8:                      $D_{i,k,j} \leftarrow D_{i,k,j} - \tilde{d}$ and $\bar{c}_{i,k} \leftarrow \bar{c}_{i,k} - \tilde{d}\,\bar{u}_{i,j}$.   ▷ Strengthening of coefficient $D_{i,k,j}$
9:                  **else if** $\tilde{d} > \epsilon \wedge D_{i,k,j} < -\epsilon$ **then**
10:                     $D_{i,k,j} \leftarrow D_{i,k,j} + \tilde{d}$ and $\bar{c}_{i,k} \leftarrow \bar{c}_{i,k} + \tilde{d}\,\underline{u}_{i,j}$.   ▷ Strengthening of coefficient $D_{i,k,j}$
11:                  **end if**
12:              **else if** $\underline{c}_{i,k} > -\epsilon_{\max} \wedge \bar{c}_{i,k} > \epsilon_{\max}$ **then**
13:                  $\tilde{d} \leftarrow \tilde{\underline{c}}_k + |D_{i,k,j}| \left(\bar{u}_{i,j} - \underline{u}_{i,j} \right)$.
14:                  **if** $\tilde{d} > \epsilon \wedge D_{i,k,j} > \epsilon$ **then**
15:                     $D_{i,k,j} \leftarrow D_{i,k,j} - \tilde{d}$ and $\underline{c}_{i,k} \leftarrow \underline{c}_{i,k} - \tilde{d}\,\underline{u}_{i,j}$.   ▷ Strengthening of coefficient $D_{i,k,j}$
16:                  **else if** $\tilde{d} > \epsilon \wedge D_{i,k,j} < -\epsilon$ **then**
17:                     $D_{i,k,j} \leftarrow D_{i,k,j} + \tilde{d}$ and $\underline{c}_{i,k} \leftarrow \underline{c}_{i,k} + \tilde{d}\,\bar{u}_{i,j}$.   ▷ Strengthening of coefficient $D_{i,k,j}$
18:                  **end if**
19:              **end if**
20:          **end for**
21:      **end for**
22: **end for**

Output: Updated constraint bound values $[\underline{c}_i, \bar{c}_i]$ and matrices D_i, $i \in \{0, \dots, N\}$.

4.6 | Exact Block-sparse Presolve Procedure

Finally, we summarize our tailored block-sparse presolve procedure for optimal control structured MIQPs in Alg. 7, which should be called in each node of the B&B method (see Figure 1) before solving the convex relaxation. Each iteration of Alg. 7 includes the following block-sparse presolve operations:

1. Update variable bound values $[\underline{z}_i, \bar{z}_i]$, using forward-backward domain propagation in Alg. 2 (Line 3).

2. Update bound values $[\underline{z}_i, \bar{z}_i]$ and $[\underline{c}_i, \bar{c}_i]$, using redundant inequality constraint detection in Alg. 4 (Line 4).

3. Update bound values $[\underline{c}_i, \bar{c}_i]$ and matrices D_i, using constraint coefficient strengthening in Alg. 5 (Line 5).

4. If probing is enabled and allowed (see Line 6 of Alg. 7), use binary variable probing in Alg. 6 (Line 7).

The block-sparse presolve procedure in Alg. 7 is an iterative procedure that typically requires multiple iterations, because each operation may result in a tightening of a continuous or discrete variable bound or constraint that in turn may result in further tightenings in the subsequent iterations. The iterative procedure terminates immediately if any of the presolve operations detects an infeasibility. Alternatively, our termination condition on Line 2 of Alg. 7 is based on whether a particular measure of progress is sufficient or not. For example, progress can be measured by the number of variables that are fixed and/or the amount by which continuous or discrete variable bounds are tightened from one iteration to a next. The presolve procedure continues as long as one or multiple variable bounds are tightened sufficiently from one iteration to the next, i.e., if `sufficient_progress` is true on Line 2 of Alg. 7. However, after a minimum number of iterations, the algorithm may terminate if no new variable is fixed in the latest iteration, to avoid performing an excessive number of iterations for incremental tightening of the continuous variable bounds. Due to the relatively high computational cost of binary variable probing in Alg. 6, probing is performed only when the amount of progress that is made by other presolve operations is not sufficient and a maximum number of iterations has not been reached, see Line 6 of Alg. 7. The overall goal of the presolve procedure is that the total time spent for removing variables and

---

Algorithm 6 Block-sparse binary variable probing procedure

Input: Bound values $[\underline{z}_i, \bar{z}_i]$, $i \in \{0, \ldots, N\}$ and MIQP of the form (1).

1: **for** $i = 0, \ldots, N$ **do**
2:     **for** $j = 1, \ldots, n_u$ **and if** $u_{i,j}$ is binary $\wedge \|\bar{z}_{i,n_x+j} - \underline{z}_{i,n_x+j}\| > \epsilon$ **then do**
3:         $[\underline{z}^0, \bar{z}^0] \leftarrow [\underline{z}, \bar{z}]$ and $\bar{z}^0_{i,n_x+j} \leftarrow \underline{z}^0_{i,n_x+j}$.   ▷ Probing variable $u_{i,j}$ to lower bound
4:         `infeasible_0` $\leftarrow$ `false` and $n \leftarrow 0$.
5:         **while** `sufficient_progress` $\wedge$ `!infeasible_0` $\wedge$ $n < n_{\text{probing}}$ **do**
6:             Update bound values $[\underline{z}^0_i, \bar{z}^0_i]$, $i \in \{0, \ldots, N\}$ using forward-backward process in Alg. 2.
7:             `infeasible_0` $\leftarrow$ `infeasible_flag` for $[\underline{z}^0, \bar{z}^0]$ and $n \leftarrow n + 1$.
8:         **end while**
9:         $[\underline{z}^1, \bar{z}^1] \leftarrow [\underline{z}, \bar{z}]$ and $\underline{z}^1_{i,n_x+j} \leftarrow \bar{z}^1_{i,n_x+j}$.   ▷ Probing variable $u_{i,j}$ to upper bound
10:        `infeasible_1` $\leftarrow$ `false` and $n \leftarrow 0$.
11:        **while** `sufficient_progress` $\wedge$ `!infeasible_1` $\wedge$ $n < n_{\text{probing}}$ **do**
12:            Update bound values $[\underline{z}^1_i, \bar{z}^1_i]$, $i \in \{0, \ldots, N\}$ using forward-backward process in Alg. 2.
13:            `infeasible_1` $\leftarrow$ `infeasible_flag` for $[\underline{z}^1, \bar{z}^1]$ and $n \leftarrow n + 1$.
14:        **end while**
15:        **if** `infeasible_0` $\wedge$ `infeasible_1` **then**
16:            **return** `infeasible_flag`   ▷ QP subproblem infeasible
17:        **else if** `infeasible_0` **then**
18:            $[\underline{z}, \bar{z}] \leftarrow [\underline{z}^1, \bar{z}^1]$.   ▷ Fix variable $u_{i,j}$ to upper bound and update all values
19:        **else if** `infeasible_1` **then**
20:            $[\underline{z}, \bar{z}] \leftarrow [\underline{z}^0, \bar{z}^0]$.   ▷ Fix variable $u_{i,j}$ to lower bound and update all values
21:        **else**
22:            $[\underline{z}, \bar{z}] \leftarrow [\min\{\underline{z}^0, \underline{z}^1\}, \max\{\bar{z}^0, \bar{z}^1\}]$.   ▷ Update all bound values
23:        **end if**
24:     **end for**
25: **end for**

Output: Updated bound values $[\underline{z}_i, \bar{z}_i]$, $i \in \{0, \ldots, N\}$.

---

constraints and strengthening of bounds and coefficients, is considerably smaller than the reduction in B&B solution time that is achieved by applying the presolve in each of the nodes.

Remark 3

Other presolving techniques such as cut generation are used in general-purpose MIQP solvers, e.g., see[44]. However, in the present paper, we refrain from using cut generation techniques for two reasons. First, we assume that global cuts can be generated offline and included in the MIQP formulation (1). Second, standard cutting plane techniques will usually produce inequalities that couple variables across stages, which is undesirable for our block-sparsity exploiting implementation.

5 | HEURISTIC PRESOLVE METHOD FOR MIXED-INTEGER OPTIMAL CONTROL

Exact presolve techniques tighten constraints or prune a subset of decision variables by fixing them to the optimal values, e.g., using domain propagation, coefficient strengthening, and probing as discussed in Section 4. For MIQPs, the development of effective presolve methods in commercial solvers has proven crucial in accelerating computation times[43]. In this section, we present a *heuristic presolve* method, which was originally proposed in[24] to increase rate of feasibility for supervised learning, but the approach is generalized here as a heuristic for quickly computing a feasible but suboptimal MIQP solution. The heuristic can be used to compute an initial solution guess, i.e., an initial upper bound value that may result in additional pruning of nodes and therefore speedup the B&B algorithm (e.g., see Figure 1) or to implement a fast but suboptimal MI-MPC method.

Algorithm 7 Block-sparse presolve procedure for optimal control structured MIQP

Input: Optimal control structured MIQP subproblem of the form in (1), and `probing` $\in \{$`true`, `false`$\}$.

1: $n \leftarrow 0$.
2: **while** `sufficient_progress` $\wedge$ `!infeasible` $\wedge$ $n < n_{\max}$ **do**
3:     Update bound values $[\underline{z}_i, \bar{z}_i]$, $i \in \{0, \dots, N\}$ using forward-backward sweep in Alg. 2.
4:     Update bound values $[\underline{z}_i, \bar{z}_i]$ and constraint values $[\underline{c}_i, \bar{c}_i]$, $i \in \{0, \dots, N\}$ using Alg. 4.
5:     Update constraint bound values $[\underline{c}_i, \bar{c}_i]$ and matrices D_i, $i \in \{0, \dots, N\}$ using Alg. 5.
6:     **if** `!infeasible` $\wedge$ `probing` $\wedge$ `!sufficient_progress` $\wedge$ $l < l_{\max}$ **then**
7:         Update bound values $[\underline{z}_i, \bar{z}_i]$, $i \in \{0, \dots, N\}$ using binary variable probing in Alg. 6.
8:         `probing` $\leftarrow$ `false`.
9:     **end if**
10:     $n \leftarrow n + 1$.
11: **end while**

Output: Updated MIQP data with tightened constraints or subproblem is detected to be infeasible.

5.1 | Abstract Definition of a Single Presolve Step

We refer to the parametric MIQP in (1) as $\mathcal{P}(\theta)$, in which the problem parameters θ can include the current state estimate $\hat{x}_0$, and we use $\delta \in \mathbb{Z}^{N_\delta}$ to denote the discrete optimization variables in (1e). In addition, we use the compact notation $\mathcal{P}(\theta, \delta_{\mathcal{R}} = \hat{\delta})$ in the following abstract definition of a presolve step, to denote the MIQP (3) after fixing $\delta_i = \hat{\delta}_i, i \in \mathcal{R}$ for an index set $\mathcal{R}$.

Definition 5 (Single Presolve Step)
Given the problem $\mathcal{P}(\theta)$ and a set of integer values $\{\hat{\delta}_i\}_{i \in \mathcal{R}}$ for the index set $\mathcal{R} \subseteq \{1, \dots, N_\delta\}$, the presolve step computes

$$\{\texttt{flag}, \hat{\delta}^+, \mathcal{R}^+\} \leftarrow \text{Presolve}(\mathcal{P}(\theta), \hat{\delta}, \mathcal{R}), \tag{23}$$

resulting in updated integer values $\{\hat{\delta}_i^+\}_{i \in \mathcal{R}^+}$ for the index set $\mathcal{R}^+ \subseteq \{1, \dots, N_\delta\}$, for which following conditions are satisfied:

1. The new set of indices includes at least the original set, i.e., $\mathcal{R} \subseteq \mathcal{R}^+$.

2. Problem $\mathcal{P}(\theta, \delta_{\mathcal{R}^+} = \hat{\delta}^+)$ is infeasible or unbounded, i.e., `flag = False`, only if $\mathcal{P}(\theta, \delta_{\mathcal{R}} = \hat{\delta})$ is infeasible or unbounded.

3. Any feasible or optimal solution of $\mathcal{P}(\theta, \delta_{\mathcal{R}^+} = \hat{\delta}^+)$ can be mapped to a feasible or optimal solution of $\mathcal{P}(\theta, \delta_{\mathcal{R}} = \hat{\delta})$, and their objective values are identical.

A presolve routine applied to a root node in B&B corresponds to Definition 5 with $\mathcal{R} = \emptyset$. In general, presolve cannot prune all of the binary or integer decision variables, but it can often lead to a reduced problem that is significantly faster to solve. For example, the presolve step can be defined by the block-sparse implementation in Alg. 7.

5.2 | Iterative Procedure for Heuristic Presolve Method

One heuristic approach to compute a feasible but possibly suboptimal MIQP solution is to construct a solution guess $\hat{\delta}$ for all discrete variables δ in (1), followed by solving the convex QP that results from fixing each variable $\delta_i = \hat{\delta}_i$ for $i = 1, \dots, N_\delta$. The solution guess $\hat{\delta}$ can be constructed in multiple ways, for example, by shifting the optimal MIQP solution from the previous to the next time step or by rounding a solution of the convex QP relaxation at the root node to the nearest integer solution. Alternatively, supervised learning can be used to train a recurrent neural network architecture[24] to make accurate predictions of the optimal values for the discrete variables in the MIQP problem $\mathcal{P}(\theta)$. Given such a discrete solution guess $\hat{\delta}$, we present a heuristic presolve method that can be used to compute an updated solution guess $\hat{\delta}_{\text{PS}}$ with increased likelihood of finding a feasible and close to optimal MIQP solution by solving the resulting convex QP.

As illustrated in Figure 2, we propose to use a heuristic presolve method that aims to correct values in the discrete solution guess $\hat{\delta}$, given that the presolve operation can provide additional fixings that preserve feasibility and optimality; see Definition 5. Our key insight is that if presolve prunes a subset of the integer variables that can be fixed to the optimal values, then the discrete solution guess $\hat{\delta}$ only needs to be used to fix any remaining integer decision variables, reducing the risk of causing infeasibility. The proposed method can be used to improve any existing heuristic, e.g., when presolve prunes an integer variable δ_i for which

the heuristic made an incorrect prediction that would have resulted in an infeasible solution. Algorithm 8 describes the proposed heuristic presolve method given a candidate integer solution $\hat{\delta}$ for a particular vector of problem parameters θ. Let $\mathcal{R}$ be the index set of pruned decision variables (i.e., δ_i for $i \in \mathcal{R}$ are fixed), $|\mathcal{R}|$ is the number of pruned integer decision variables, and $\mathcal{R}, \hat{\delta}_{\mathrm{PS}}$ are initialized as empty sets (Line 1). The presolve step is then called (Line 3) and returns whether infeasibility was detected, an updated set of indices $\mathcal{R}$, and corresponding values $\hat{\delta}_{\mathrm{PS}}$. Algorithm 8 terminates if all of the integer variables have been fixed (Line 5) or if infeasibility is detected (Line 4). Otherwise, one of the remaining free integer variables δ_j for $j \in \{1, \ldots, N_\delta\} \setminus \mathcal{R}$ is selected and fixed to the value in the original solution guess, i.e., $\hat{\delta}_{\mathrm{PS},j} = \hat{\delta}_j$ (Line 6-8).
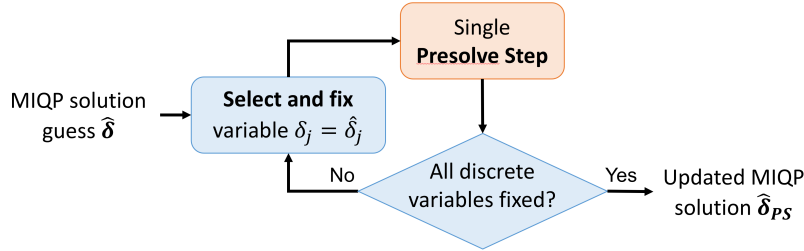
FIGURE 2 Heuristic presolve method to fix discrete optimization variables in MIQP.

In Algorithm 8, we denote `VarSelect` as the procedure to choose the integer variable δ_j to fix next, which is closely related to the variable selection policy in B&B routines. We will further illustrate the performance of the heuristic presolve method in Alg. 8 in combination with rounding of the relaxed QP solution at the root node to the nearest integer solution. In this case, the variable selection strategy `VarSelect` can select a remaining free integer variable, for which the relaxed QP solution is closest to integer feasible, i.e., the variable for which the difference with the value after rounding is smallest. Algorithm 8 is an iterative procedure, which calls our tailored presolve routine in Alg. 7 at each iteration (Line 3). At an increased computational cost, one can use the binary variable probing strategy in Alg. 6 specifically for the selected variable δ_j, see Line 7 of Alg. 8, in order to check whether the variable can be fixed based on probing before fixing it to the value in the solution guess. Alternatively, one could skip the presolve call when a particular limit on the computation time has been reached, i.e., such that all remaining free integer variables are fixed directly to their corresponding values in the solution guess (Line 8).

Algorithm 8 Iterative Procedure for Heuristic Presolve Method

---

Input: Candidate integer solution $\hat{\delta}$, problem parameters θ.

1: Initialize set of pruned integer variables $\mathcal{R} \leftarrow \emptyset$ and values $\hat{\delta}_{\mathrm{PS}} \leftarrow \emptyset$.
2: **for** $i \in \{1, \ldots, N_\delta\}$ **do**
3:      $\{\texttt{flag}, \hat{\delta}_{\mathrm{PS}}, \mathcal{R}\} \leftarrow \text{Presolve}(\mathcal{P}(\theta), \hat{\delta}_{\mathrm{PS}}, \mathcal{R})$ in Alg. 7.
4:      **if** $\texttt{flag} == \texttt{False}$ **then return** $(\texttt{False}, \hat{\delta}_{\mathrm{PS}})$.
5:      **if** $|\mathcal{R}| == N_\delta$ **then return** $(\texttt{True}, \hat{\delta}_{\mathrm{PS}})$.
6:      $j \leftarrow \texttt{VarSelect}(\{1, \ldots, N_\delta\} \setminus \mathcal{R})$.
7:      Optional: perform binary variable probing in Alg. 6 specifically for δ_j.
8:      **if** δ_j not fixed by probing **then** fix integer variable $\hat{\delta}_{\mathrm{PS},j} = \hat{\delta}_j$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{j\}$.
9: **end for**

Output: Updated integer solution guess $\hat{\delta}_{\mathrm{PS}}$.

---

6 | PRACTICAL CONSIDERATIONS FOR EMBEDDED MIXED-INTEGER MPC

In embedded real-time applications of mixed-integer MPC (MI-MPC), one needs to solve an MIQP (1) at each sampling instant under strict timing constraints. We can leverage the fact that we solve a sequence of similar problems, parametrized by the initial
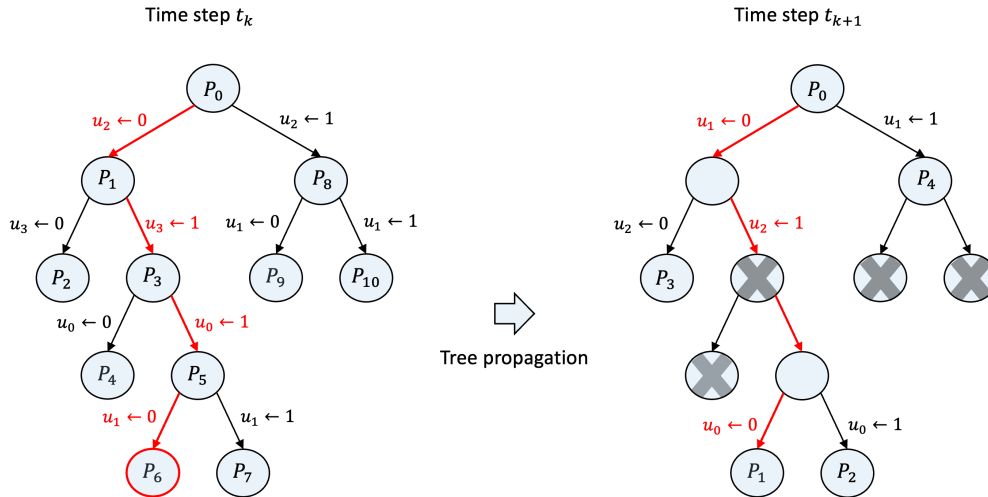
FIGURE 3 Illustration of the tree propagation technique for warm starting of the B&B method from one time step to the next in the MI-MPC algorithm: index i denotes the order in which each node P_i is solved.

state value $\hat{x}_0$, in order to warm-start the B&B algorithm. We refer to our warm-starting procedure as *tree propagation*, which was originally proposed in[29], because the main goal is to "propagate" the B&B tree forward by one time step.

6.1 | Branch-and-bound Tree Propagation for Warm Starting

The warm-starting procedure aims to use knowledge of one MIQP solution, i.e., the search tree after solving the problem, in order to improve the B&B search for the next MIQP[33,45]. Our approach is to store the path from the root to the leaf node where the optimal solution to the MIQP was found, as well as the order in which the variables are branched. This is based on the knowledge that the branching order is crucial for the efficiency of any B&B method[37]. We then perform a shifting and update of this path in order to obtain a *warm-started tree* to start our B&B search at the next time step. We illustrate this procedure in Figure 3, where the optimal path at the current time step is denoted by the sequence of nodes $P_0 \rightarrow P_1 \rightarrow P_3 \rightarrow P_5 \rightarrow P_6$. Let us consider a corresponding sequence of variables $u_2 \rightarrow u_3 \rightarrow u_0 \rightarrow u_1$ that we branched on in order to find the optimal solution at the leaf node P_6. After shifting by one time step, all branched variables in the first control interval can be ignored. Assuming the index refers to the time step in the control horizon, this results in a shifted and shorter path of variables $u_1 \rightarrow u_2 \rightarrow u_0$.

After obtaining a new state estimate $\hat{x}_0$, we execute the block-sparse presolve routine in Alg. 7 and we solve the convex QP relaxation corresponding to the root node. Given the warm-started tree and the relaxed solution at the root node, we remove nodes that correspond to branched variables that are already integer feasible in the solution at the root node. The shifted MIQP solution from the previous time step can be used to fix all integer variables at the current time step and solve the resulting convex QP, which may result in a feasible solution and therefore an upper bound for the B&B method. Alternatively or in addition, the heuristic presolve method in Alg. 8 can be used to compute a feasible solution and possibly a tighter B&B upper bound. We proceed by solving all the leaf nodes on the warm-started path. As we solve both children of a node on this path, we do not have to solve the parent node itself and therefore we reduce computations by solving fewer QP relaxations. We process the warm-started tree in the order depicted by the index of each node in Fig. 3, after which we resume normal procedure of the B&B method.

We can additionally shift and re-use the pseudo-cost information from one MPC time step to the next, in order to make better branching decisions without the need for computationally expensive strong branching. The propagation of pseudo-costs can be coupled with an update of the reliability parameters in order to discount relatively old pseudo-cost information, which may help to avoid increasing the worst-case computation time by making a bad branching decision. The reliability number may be reduced for each variable from one time step to the next, in order to force strong branching for variables that have not been branched on in a sufficiently long time. In addition, nodes can be removed from the warm-started path in case they correspond to branched variables for which there is no pseudo-cost information or it is not sufficiently reliable, in an attempt to avoid bad branching decisions in the B&B method. Finally, these warm-started pseudo-costs can also be used to re-order the warm-started tree to improve the branching order and hopefully result in smaller B&B search tree sizes.

6.2 | Embedded Software Implementation for MI-MPC

We refer to the proposed MI-MPC algorithm as `BB-ASIPM`, since it combines a B&B method with tailored block-sparse presolve techniques and `ASIPM` to solve the convex QP relaxations. At each control time step, it solves an MIQP where the B&B tree and the pseudo-cost information are warm-started. The B&B strategy and the tailored block-sparse presolve methods, the warm-start and heuristic branching techniques, as well as the `ASIPM` solver have all been implemented in self-contained C code, which allows for real-time implementations on embedded control hardware. In addition to the presented techniques to speed up a B&B method for MI-MPC, multiple heuristic techniques can be used to achieve real-time feasibility. For example, an upper bound on the number of B&B iterations can be imposed to ensure a maximum computation time and to allow the MI-MPC controller to use a feasible but suboptimal solution instead. If an integer-feasible solution has been found, e.g., based on warm-starting in Fig. 3 or the heuristic presolve method in Alg. 8, a B&B method can provide a bound on its suboptimality. Alternatively or in addition, an integer horizon can be introduced, i.e., integer feasibility could be enforced on the first $M < N$ stages in (1e). The latter can drastically reduce computation times at the cost of an approximation error as discussed in [31, Sec. 4.2]. However, the investigation of such additional heuristics is beyond the scope of the present paper.

7 | CASE STUDIES: MIXED-INTEGER MPC SIMULATION RESULTS

Let us illustrate the computational performance of `BB-ASIPM` against multiple state-of-the-art MIQP solvers for two numerical case studies of mixed-integer optimal control applications. In addition, we will illustrate the performance of the heuristic presolve method to compute a feasible but possibly suboptimal solution in real time. The two benchmark problems include: (1) mobile robot motion planning with obstacle avoidance constraints and (2) an underactuated cart-pole with soft contacts. All computation times in Section 7.1 are obtained on a MacBook Pro (16-inch, 2019), with 2.4 GHz 8-Core Intel Core i9 processor. The numerical results in Section 7.2 are obtained using hardware-in-the-loop (HIL) simulations on a dSPACE Scalexio rapid prototyping unit, with a DS6001 Processor Board for real-time processing [56].

7.1 | Mixed-integer Optimal Control for Motion Planning with Obstacle Avoidance

The first case study concerns the time-optimal motion planning, using a simple kinematic model for a mobile robot, taking into account collision avoidance constraints based on an MIQP formulation that is similar to the benchmark example from [24].

7.1.1 | Problem Formulation

We formulate an MIOCP of the form in (1), where the state vector is defined as $x = [p^{\mathrm{x}} \ p^{\mathrm{y}} \ v^{\mathrm{x}} \ v^{\mathrm{y}} \ g^{\mathrm{s}}]^{\top}$, i.e., $n_{\mathrm{x}} = 5$, including the 2D position $(p^{\mathrm{x}}, p^{\mathrm{y}})$ and velocity vector $(v^{\mathrm{x}}, v^{\mathrm{y}})$ of the robot and including a binary state $g^{\mathrm{s}} \in \{0, 1\}$ that indicates whether the goal has been reached. The discrete-time state dynamics in (1b) read as

$$
x_{i+1} = \begin{bmatrix} p^{\mathrm{x}}_{i+1} \\ p^{\mathrm{y}}_{i+1} \\ v^{\mathrm{x}}_{i+1} \\ v^{\mathrm{y}}_{i+1} \\ g^{\mathrm{s}}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_{\mathrm{s}} & 0 & 0 \\ 0 & 1 & 0 & T_{\mathrm{s}} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p^{\mathrm{x}}_i \\ p^{\mathrm{y}}_i \\ v^{\mathrm{x}}_i \\ v^{\mathrm{y}}_i \\ g^{\mathrm{s}}_i \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ T_{\mathrm{s}} & 0 & 0 \\ 0 & T_{\mathrm{s}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a^{\mathrm{x}}_i \\ a^{\mathrm{y}}_i \\ \delta^{\mathrm{g}}_i \end{bmatrix},
\tag{24}
$$

where the control inputs are $u = [a^{\mathrm{x}} \ a^{\mathrm{y}} \ \delta^{\mathrm{g}}]^{\top}$, i.e., $n_{\mathrm{u}} = 3$, including the 2D acceleration vector $(a^{\mathrm{x}}, a^{\mathrm{y}})$ and an auxiliary binary variable $\delta^{\mathrm{g}}_i \in \{0, 1\}$ that determines whether the goal is reached at the time step $i \in \{1, \dots, N\}$ in the prediction horizon. The inequality constraints (1d) include requirements on reaching the goal state as

$$
x^{\mathrm{goal}} - M (1 - g^{\mathrm{s}}_i) \le x_{i,1:4} \le x^{\mathrm{goal}} + M (1 - g^{\mathrm{s}}_i),
\tag{25}
$$

where $M > 0$ is sufficiently large and $x_{i,1:4} = [p^{\mathrm{x}}_i \ p^{\mathrm{y}}_i \ v^{\mathrm{x}}_i \ v^{\mathrm{y}}_i]^{\top}$, which ensures the implication $g^{\mathrm{s}}_i = 1 \implies x_{i,1:4} = x^{\mathrm{goal}}$ and we additionally enforce that $\sum_{i=0}^{N-1} \delta^{\mathrm{g}}_i = 1$, and therefore $g^{\mathrm{s}}_N = 1$ holds. As illustrated in Figure 4, we include collision avoidance
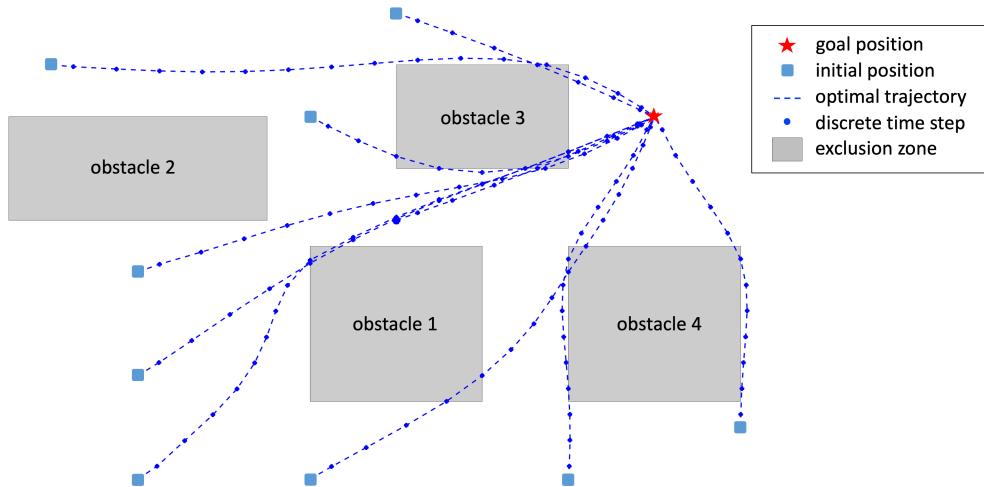
constraints for rectangular obstacle shapes that are aligned with the axes for simplicity,

$$p_j^{x,min} - M\delta_j^{o,1} \leq p_i^x \leq p_j^{x,min} + M(1 - \delta_j^{o,1}),$$
$$p_j^{x,max} - M(1 - \delta_j^{o,2}) \leq p_i^x \leq p_j^{x,max} + M\delta_j^{o,2},$$
$$p_j^{y,max} - M(1 - \delta_j^{o,4}) \leq p_i^y \leq p_j^{y,min} + M(1 - \delta_j^{o,3}), \quad (26)$$
$$p_j^{x,min} - M(1 - \delta_j^{o,3} - \delta_j^{o,4}) \leq p_i^x \leq p_j^{x,max} + M(1 - \delta_j^{o,3} - \delta_j^{o,4}),$$
$$\delta_j^{o,1} + \delta_j^{o,2} + \delta_j^{o,3} + \delta_j^{o,4} = 1,$$

where the bounds $[p_j^{x,min} \ p_j^{x,max}]$ and $[p_j^{y,min} \ p_j^{y,max}]$ define the rectangular avoidance zone for each obstacle $j = 1, \ldots, n_{obs}$. As can be observed in Figure 4, the obstacle avoidance constraints in (26) are enforced pointwise in time, and each exclusion zone includes appropriate margins to account for the physical dimension of the mobile robot and to account for discretization errors. The cost function 1a for this numerical case study reads as

$$\sum_{i=0}^{N-1} \left(i\,\delta_i^g + \|u_{i,1:2}\|_R^2 \right) + \sum_{i=0}^{N} \|x_{i,1:4} - x^{goal}\|_Q^2, \quad (27)$$

where the first term accounts for minimizing the time step at which the goal is reached and minimizing the acceleration inputs $u_{i,1:2} = [a_i^x \ a_i^y]^\top$, and the second term accounts for minimizing the state error with respect to the goal state.


FIGURE 4 Illustration of the MIP-based motion planning case study: the optimal trajectories for a horizon length $N = 20$ are depicted in blue, given four rectangular obstacle shapes, multiple initial positions and a fixed goal position. The obstacle avoidance constraints are enforced pointwise in time, i.e., for each discrete time step of the trajectories, and the obstacle dimensions include appropriate margins to account for discretization errors and to account for the physical shape of the mobile robot.

7.1.2 | MIQP Solver: Numerical Results

We illustrate the computational performance of the proposed MIQP solver by solving multiple MIOCPs for motion planning with obstacle avoidance for a varying number of obstacles n_{obs}, a varying horizon length N and varying initial state values $\hat{x}_0$. In particular, the number of obstacles is selected to be in the range $n_{obs} \in [1, 4]$ and the horizon length $N \in [6, 20]$. The MIOCP for each horizon length and for each number of obstacles is solved for 200 different initial state values, sampled from a uniform distribution where $0 \leq p_0^x \leq 10$ and $0 \leq p_0^y \leq 10$, excluding the avoidance regions. In addition to our proposed BB-ASIPM solver, the MIOCPs are solved using the state-of-the-art solvers GUROBI[41], MOSEK[42], GLPK[49], Cbc[50], and Matlab's intlinprog. Because the solvers GLPK, Cbc, and intlinprog do not provide direct support for the solution of MIQPs, we present numerical results for solving a mixed-integer linear programming (MILP) approximation with these particular software tools instead. Figure 5 shows the average computation time of the resulting MIOCP solutions for 200 randomly sampled initial

state values, for each of the numbers of obstacles $n_{obs} \in [1, 4]$ and for each horizon length $N \in [6, 20]$. As expected, the average computation time for motion planning grows rapidly with the number of obstacles n_{obs} and with the horizon length N, due to the increasing number of binary optimization variables $n_{bin} = N(1 + 4n_{obs})$ in the considered MIOCP formulation.
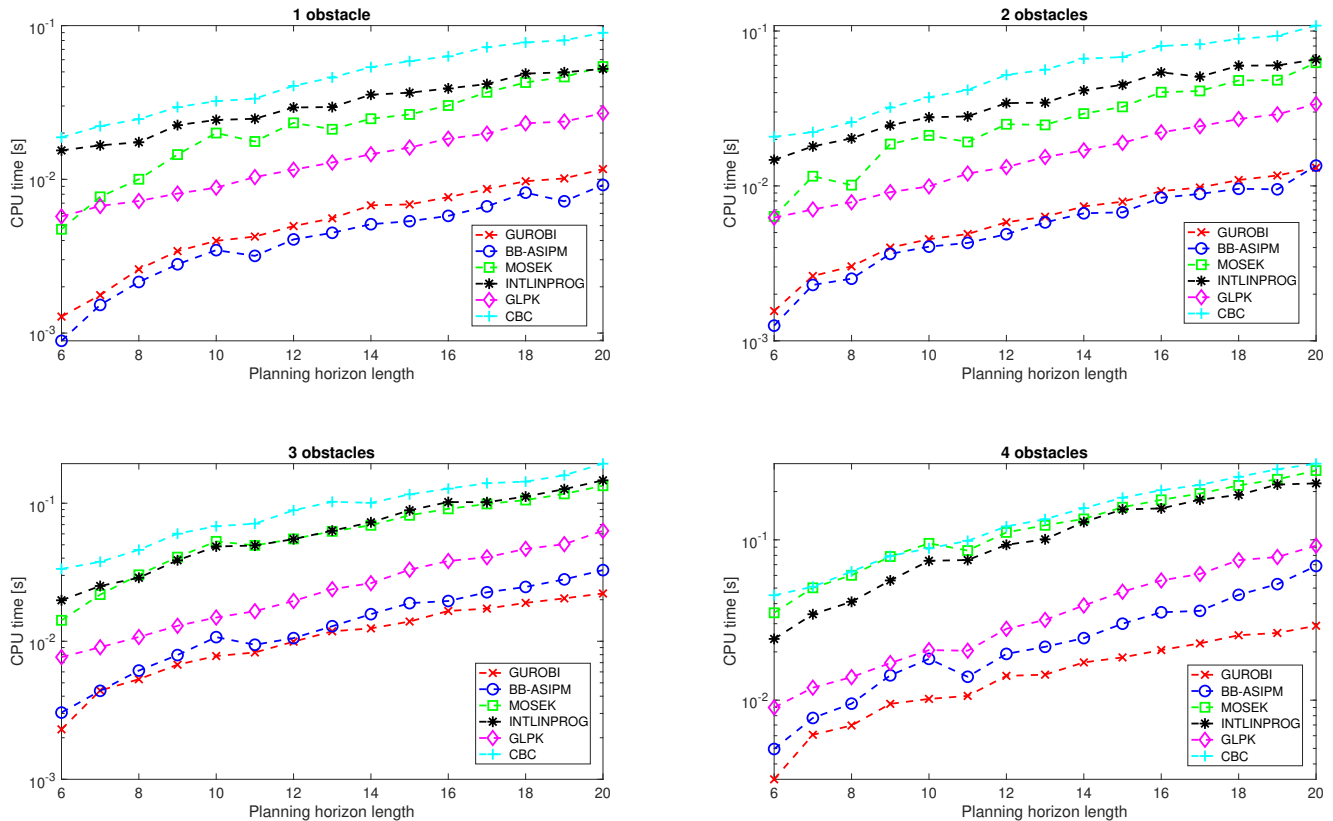
FIGURE 5 Average computation times for MIP-based motion planning case study for 1, 2, 3 and 4 obstacles, using a varying horizon length $N \in [6, 20]$, and using solvers GUROBI, MOSEK, intlinprog, GLPK, Cbc and the proposed BB-ASIPM solver. Note that each of the solvers computes the globally optimal solution of the MIP.

For this particular case study, the numerical results in Figure 5 show that our proposed BB-ASIPM is the fastest solver for all horizon lengths $N \in [6, 20]$, when $n_{obs} = 1$ and $n_{obs} = 2$. The state-of-the-art commercial solver GUROBI becomes faster for $n_{obs} \geq 3$ obstacles and it will likely continue to outperform the BB-ASIPM solver for larger problem dimensions, because GUROBI includes many advanced cutting plane techniques and heuristics[41] that are not implemented in the BB-ASIPM solver. However, BB-ASIPM remains relatively competitive and it outperforms the other state-of-the-art solvers even for larger MIOCPs with up to $n_{bin} = 340$ binary optimization variables for $N = 20$ and $n_{obs} = 4$, in this particular case study. These results confirm that our proposed BB-ASIPM solver can be competitive with state-of-the-art MIP solvers, even though the software implementation of BB-ASIPM is relatively compact and self-contained such that it can be executed on an embedded microprocessor for real-time applications of mixed-integer optimal control, e.g., for real-time motion planning with obstacle avoidance constraints. Instead, state-of-the-art optimization tools, such as GUROBI typically cannot be used on embedded control hardware with limited computational resources and with limited memory[10]. Some results of using BB-ASIPM on the dSPACE Scalexio rapid prototyping unit will be illustrated further in Section 7.2.

Remark 4
Advanced presolve and heuristic options have been activated for each of the state-of-the-art software tools, resulting in fair

computational comparisons. However, because the C code implementation of the BB-ASIPM solver is a single-threaded process, the computation times in this paper are based on single-threaded processing for all MIP solvers. Similar to state-of-the-art B&B methods, parallel processing could be used to speed up computations in the BB-ASIPM solver, by solving QP relaxations corresponding to multiple nodes in the B&B tree simultaneously. This remains outside the scope of the present paper.

7.1.3 │ Heuristic Presolve Method: Numerical Results

Table 1 shows the average computation time of using GUROBI, MOSEK, or BB-ASIPM to solve the MIOCPs for 200 randomly sampled initial state values, for each of the numbers of obstacles $n_{\mathrm{obs}} \in \{1, 3, 5, 7\}$ and for different horizon lengths $N \in \{6, 12, 18\}$. In particular, Table 1 also reports the average computation time for the Heuristic-presolve technique in Algorithm 8. The candidate binary solution $\hat{\delta}$, which is an important input that strongly affects the performance of Algorithm 8, is computed by rounding the relaxed QP solution at the root node to the nearest integer solution for the numerical results in Table 1. Since the Heuristic-presolve method cannot guarantee to find the globally optimal solution, and it may even fail to find a feasible solution of the MIQP, Table 1 reports the rate of infeasibility and suboptimality as

$$\text{Infeasibility} = 100 \, \frac{n_{\mathrm{fail}}}{n_{\mathrm{sample}}} [\%], \qquad \text{Suboptimality} = 100 \, \frac{J - J^\star}{J^\star} [\%], \tag{28}$$

where $n_{\mathrm{sample}} = 200$, $J^\star$ is the globally optimal objective value for a particular MIQP and J is the objective value of a feasible but potentially suboptimal solution to the same MIQP, i.e., $J \geq J^\star$. Similar to Figure 5, the numerical results in Table 1 show that BB-ASIPM can outperform GUROBI for relatively small problems, but the GUROBI solver is faster for larger problem dimensions. In addition, Table 1 illustrates how the proposed Heuristic-presolve technique can be used to compute a suboptimal solution faster than the time needed to solve the same MIQP with either GUROBI, MOSEK or BB-ASIPM. The rate of infeasibility and suboptimality of Heuristic-presolve is small for most cases of this particular problem, but it can be observed that these rates generally increase with the MIQP problem dimensions. It has been shown recently in [24] that the performance of Heuristic-presolve can be improved, i.e., the rate of infeasibility and suboptimality can be decreased, by using supervised learning techniques. Specifically, the work in [24] describes a framework based on deep recurrent neural networks in combination with the Heuristic-presolve in Algorithm 8. The details of this are outside the scope of the present paper.

TABLE 1 Average computation times for MIP-based motion planning case study for 1, 3, 5 and 7 obstacles, with a varying horizon length $N \in \{6, 12, 18\}$, and using solvers GUROBI, MOSEK, BB-ASIPM and the proposed Heuristic-presolve technique in Algorithm 8. We additionally report the rate of infeasibility and suboptimality in (28) for Heuristic-presolve.

	GUROBI	MOSEK	BB-ASIPM	Heuristic-presolve		
	Time [ms]	Time [ms]	Time [ms]	Time [ms]	Infeasibility [%]	Suboptimality [%]
$N = 6$, $n_{\mathrm{obs}} = 1$	1.2 ms	4.1 ms	0.8 ms	0.7 ms	0.0 %	0.0 %
$N = 6$, $n_{\mathrm{obs}} = 3$	2.3 ms	13.6 ms	2.9 ms	1.4 ms	0.0 %	0.1 %
$N = 6$, $n_{\mathrm{obs}} = 5$	3.2 ms	34.7 ms	5.4 ms	2.4 ms	0.0 %	1.1 %
$N = 6$, $n_{\mathrm{obs}} = 7$	4.3 ms	46.8 ms	8.0 ms	3.6 ms	0.0 %	2.4 %
$N = 12$, $n_{\mathrm{obs}} = 1$	4.2 ms	22.2 ms	3.8 ms	2.1 ms	0.0 %	0.2 %
$N = 12$, $n_{\mathrm{obs}} = 3$	10.7 ms	75.3 ms	14.4 ms	5.7 ms	0.0 %	0.2 %
$N = 12$, $n_{\mathrm{obs}} = 5$	13.6 ms	126.2 ms	23.5 ms	8.5 ms	0.0 %	2.6 %
$N = 12$, $n_{\mathrm{obs}} = 7$	22.2 ms	220.9 ms	43.3 ms	15.4 ms	0.0 %	4.8 %
$N = 18$, $n_{\mathrm{obs}} = 1$	7.1 ms	35.8 ms	6.4 ms	3.1 ms	0.5 %	0.0 %
$N = 18$, $n_{\mathrm{obs}} = 3$	18.8 ms	140.3 ms	27.9 ms	8.5 ms	1.2 %	0.1 %
$N = 18$, $n_{\mathrm{obs}} = 5$	28.4 ms	264.8 ms	55.8 ms	15.8 ms	1.3 %	4.6 %
$N = 18$, $n_{\mathrm{obs}} = 7$	37.5 ms	333.6 ms	91.3 ms	26.0 ms	1.9 %	6.5 %
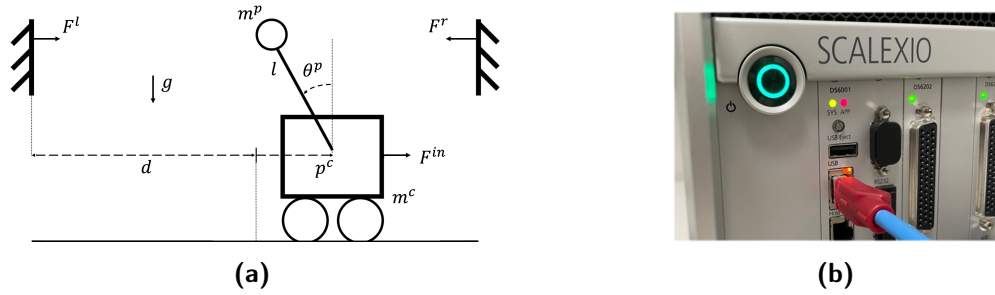
FIGURE 6 Illustration of the stabilization of an inverted pendulum on top of a cart with two soft walls that result in contact forces in (a), and the dSPACE Scalexio rapid prototyping unit for hardware-in-the-loop simulations in (b).

7.2 | Mixed-integer MPC for Stabilization of Inverted Pendulum with Soft Contacts

As a second case study, we consider MI-MPC for an inverted pendulum with soft contacts as shown in Fig. 6a, which is representative of underactuated, multi-contact control problems [57,58]. Although MIOCPs form an attractive framework for modeling such problems, controllers that can react and plan online for such systems are rarely real-time feasible or near-optimal [17].

7.2.1 | Problem Formulation

The cart-pole system with soft walls is often used as a benchmark problem for mixed-integer optimal control, e.g., see [45,23]. This standard hybrid control problem results in an MIQP of the form in (1), which is solved at each sampling time step of the MI-MPC controller. Given the current state of the cart-pole system $\hat{x}_0$, the aim is to use the horizontal forces of the cart to effectively regulate an inverted pendulum towards the unstable equilibrium at the origin. The state vector is defined as $x = [p^c\ \theta^p\ v^c\ \omega^p]^\top$, i.e., $n_x = 4$, including the position p^c and velocity v^c of the cart, and the angle θ^p and angular velocity ω^p of the pendulum. The force input applied to the cart is $F^{in} \in \mathbb{R}$ and $F^l, F^r \in \mathbb{R}$ are the contact forces transmitted by the soft walls on the left and right of the cart-pole system, see Fig. 6a. The discrete-time state dynamics in (1b) read as

$$
x_{i+1} = \begin{bmatrix} p^c_{i+1} \\ \theta^p_{i+1} \\ v^c_{i+1} \\ \omega^p_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & g\frac{m^p}{m^c} & 1 & 0 \\ 0 & g\frac{m^c+m^p}{m^c\,l} & 0 & 1 \end{bmatrix} \begin{bmatrix} p^c_i \\ \theta^p_i \\ v^c_i \\ \omega^p_i \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m^c} & 0 & 0 \\ \frac{1}{m^c\,l} & -\frac{1}{m^p\,l} & \frac{1}{m^p\,l} \end{bmatrix} \begin{bmatrix} F^{in}_i \\ F^l_i \\ F^r_i \end{bmatrix}, \tag{29}
$$

which are linearized around the nominal angle of the pendulum $\theta^p = 0$, and where g is the gravitational acceleration, l is the length of the pendulum, and m^c, m^p are the mass of the cart and pole, respectively. We use the same parameter values as in [45], i.e., $g = 10$, $l = 1$, and $m^c = m^p = 1$. The inequality constraints (1d) include the following bounds on states and inputs

$$
-d \le p^c_i \le d, \quad -\frac{\pi}{10} \le \theta^p_i \le \frac{\pi}{10}, \quad -1 \le v^c_i, \omega^p_i, F^{in}_i \le 1, \tag{30}
$$

where $d = 0.5$ is half the distance between the two walls (see Fig. 6a). A linear approximation for the position of the tip of the pole is $p^t_i = p^c_i - l\,\theta^p_i$, resulting in the relative distances $s^l_i = -d - p^t_i$ and $s^r_i = p^t_i - d$ (each relative distance is positive in case of penetration), and we additionally define the time derivatives $\dot{s}^l_i = -v^c_i + l\,\omega^p_i$ and $\dot{s}^r_i = v^c_i - l\,\omega^p_i$ with respect to the left and right wall, respectively. We use the piecewise-linear definition of the contact forces

$$
F^j_i = \begin{cases} \kappa s^j_i + \nu \dot{s}^j_i & \text{if } s^j_i \ge 0 \text{ and } \kappa s^j_i + \nu \dot{s}^j_i \ge 0, \\ 0 & \text{otherwise}, \end{cases} \tag{31}
$$

for each of the soft walls $j \in \{l, r\}$, where $\kappa = 100$ is the stiffness and $\nu = 10$ is the damping in the contact model. As described further in [45], two binary indicator variables can be used to model each of the piecewise-linear functions in (31), resulting in a total of $n_{bin} = 4N$ binary optimization variables in the considered MIOCP formulation.

7.2.2 | MIQP Solver: Simulation Results on dSPACE Scalexio Hardware

We illustrate the performance of the embedded `BB-ASIPM` solver using closed-loop simulations on the dSPACE Scalexio rapid prototyping unit, see Fig. 6b, where the MI-MPC controller solves the MIOCP at each control time step with a sampling time

period $T_s^{mpc} = 0.1$ s and $N = 8$ control intervals. The cart-pole system is simulated using the linearized dynamics in (29) with a smaller discretization time $T_s = 0.05$ s, and the state feedback includes zero-mean Gaussian sensor noise $w_i \sim \mathcal{N}(0, \sigma_x^2)$ with covariance matrix $\sigma_x^2 = \text{diag}(\sigma_{p^c}^2, \sigma_{\theta^p}^2, \sigma_{v^c}^2, \sigma_{\omega^p}^2)$ and standard deviation values $\sigma_{p^c} = 10^{-2}, \sigma_{\theta^p} = 10^{-3}, \sigma_{v^c} = 10^{-2}$ and $\sigma_{\omega^p} = 10^{-3}$. Figure 7 shows the resulting closed-loop MI-MPC trajectories for stabilization of the inverted pendulum with soft contacts, with the initial state values $p_0^c = 0.25, \theta_0^p = -0.05, v_0^c = 0$ and $\omega_0^p = 0$, and using the BB-ASIPM solver on the dSPACE Scalexio rapid prototyping unit. For this particular set of initial state values, it can be seen from Figure 7 that the tip of the pendulum makes contact with the right wall before making contact with the left wall, after which the MI-MPC controller stabilizes the cart-pole system towards the unstable equilibrium at the origin. The behavior of the MI-MPC controller strongly depends on the cost function and the horizon length in the MIOCP but, since the cart-pole system is just an illustrative case study, a thorough tuning process remains outside the scope of the present paper. Figure 7 also shows the computation times and the number of B&B iterations at each control time step. Finally, Table 2 shows the average and worst-case results for the computation times, number of B&B iterations and total number of IPM iterations to solve the convex QP relaxations in the BB-ASIPM solver, using four different sets of initial state values. It can be observed that the worst-case computation times of the BB-ASIPM solver on the dSPACE Scalexio rapid prototyping unit remain below the sampling time period of $T_s^{mpc} = 0.1$ s.
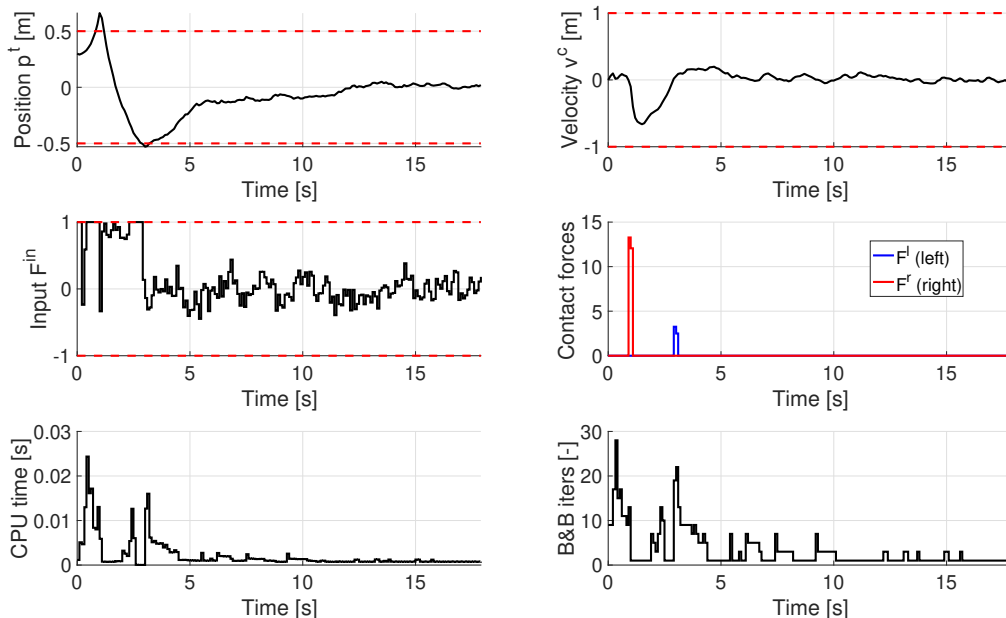
FIGURE 7 Closed-loop MI-MPC trajectories of simulations for stabilization of the inverted pendulum with soft contacts, with initial state values $p_0^c = 0.25$ and $\theta_0^p = -0.05$, and using the BB-ASIPM solver on dSPACE Scalexio rapid prototyping unit. The red dashed lines in the upper left plot indicate the positions at which the tip of the pole penetrates the soft walls, i.e., when $p_i^t \leq -d$ or $p_i^t \geq d$, corresponding to times when the contact forces in (31) can be nonzero from the left or right wall, respectively.

8 | CONCLUSIONS AND OUTLOOK

We proposed a solver for mixed-integer optimal control problems aimed at achieving a real-time implementation of mixed-integer model predictive control (MI-MPC) on embedded platforms based on tailored presolve methods, both exact techniques and heuristics, and using a tailored solver for convex relaxations. In particular, we provided an overview of recent work on the efficient implementation of infeasibility detection and early termination of an active-set based interior point method (ASIPM) for convex quadratic programming. We then proposed a novel collection of block-sparse presolve techniques to efficiently remove decision variables, and to remove or tighten inequality constraints in mixed-integer quadratic programming (MIQP). In addition,

TABLE 2 Average and worst-case computational results of the `BB-ASIPM` solver using four different MI-MPC simulations for stabilization of the inverted pendulum with soft contacts on the dSPACE Scalexio rapid prototyping unit.

Initial state value $\hat{x}_0$		CPU time [ms]		B&B iters [-]		IPM iters [-]	
		mean	max	mean	max	mean	max
$p_0^c = 0.25$	$\theta_0^p = -0.05$	2.1 ms	24.4 ms	3.2	28	39.2	569
$p_0^c = 0.00$	$\theta_0^p = -0.05$	2.3 ms	53.7 ms	3.2	48	43.4	1223
$p_0^c = 0.00$	$\theta_0^p = 0.05$	2.3 ms	59.5 ms	2.9	53	46.7	1406
$p_0^c = -0.25$	$\theta_0^p = 0.05$	1.8 ms	27.8 ms	2.4	25	34.4	693

we showed how the tailored presolve routine can be used in a novel iterative heuristic approach to compute feasible but possibly suboptimal MIQP solutions. Based on a self-contained C code implementation of a branch-and-bound (B&B) method, in combination with these tailored presolve techniques and ASIPM, the resulting `BB-ASIPM` solver can be used to implement MI-MPC on embedded microprocessors. We presented benchmarking results for the `BB-ASIPM` algorithm compared against state-of-the-art MIQP solvers, including `GUROBI`, `MOSEK`, `GLPK`, `Cbc`, and Matlab's `intlinprog`, based on a case study of mobile robot motion planning with obstacle avoidance constraints. Finally, we demonstrated the computational performance of the `BB-ASIPM` solver on a dSPACE Scalexio rapid prototyping unit, using a second case study of stabilization for an underactuated cart-pole with soft contacts. Future work involves the validation of the `BB-ASIPM` solver in real-world experiments using the proposed MI-MPC method for online trajectory generation, e.g., in combination with a standard MPC controller for reference tracking.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. Mayne D, Rawlings J. *Model Predictive Control*. Nob Hill . 2013.

2. Bemporad A, Morari M. Control of systems integrating logic, dynamics, and constraints. *Automatica* 1999; 35: 407–427.

3. Marcucci T, Tedrake R. Mixed-Integer Formulations for Optimal Control of Piecewise-Affine Systems. In: Proc. ACM Int. Conf. on Hybrid Systems: Computation and Control. ; 2019: 230–239.

4. Chen C, Culbertson P, Lepert M, Schwager M, Bohg J. TrajectoTree: Trajectory Optimization Meets Tree Search for Planning Multi-Contact Dexterous Manipulation. In: IEEE/RSJ Int. Conf. on Intelligent Robots & Systems. ; 2021: 8262–8268.

5. Walsh A, Di Cairano S, Weiss A. MPC for Coupled Station Keeping, Attitude Control, and Momentum Management of Low-Thrust Geostationary Satellites. In: Proc. American Control Conference (ACC). ; 2016: 7408–7413.

6. Landry B, Deits R, Florence PR, Tedrake R. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In: Proc. IEEE Conf. on Robotics and Automation. ; 2016: 1469-1475.

7. Sahin YE, Quirynen R, Di Cairano S. Autonomous Vehicle Decision-Making and Monitoring based on Signal Temporal Logic and Mixed-Integer Programming. In: Proc. American Control Conference (ACC). ; 2020: 454-459.

8. Pia A, Dey S, Molinaro M. Mixed-integer quadratic programming is in NP. *Mathematical programming* 2017; 162: 225–240.

9. Nemhauser GL, Wolsey LA. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience . 1988.

10. Di Cairano S, Kolmanovsky IV. Real-time optimization and model predictive control for aerospace and automotive applications. In: Amer. Control Conf. ; 2018: 2392–2409.

11. Kirches C. *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*. PhD thesis. Uni. Heidelberg, 2010.

12. Sager S, Bock H, Diehl M. Solving Mixed-integer Control Problems by Sum Up Rounding With Guaranteed Integer Gap. *SIAM Journal on Control and Optimization* 2008.

13. Achterberg T, Berthold T. Improving the feasibility pump. *Discrete Optimization* 2007; 4(1): 77–86.

14. Diamond S, Takapoui R, Boyd S. A general system for heuristic minimization of convex functions over non-convex sets. *Optimization Methods and Software* 2018; 33(1): 165-193.

15. Naik VV, Bemporad A. Embedded mixed-integer quadratic optimization using accelerated dual gradient projection. *IFAC-PapersOnLine* 2017; 50(1): 10723–10728.

16. Stellato B, Geyer T, Goulart PJ. High-Speed Finite Control Set Model Predictive Control for Power Electronics. *IEEE Transactions on Power Electronics* 2017; 32(5): 4007-4020.

17. Marcucci T, Deits R, Gabiccini M, Bicchi A, Tedrake R. Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In: IEEE-RAS Int. Conf. on Humanoid Robotics. ; 2017: 31-38.

18. Bengio Y, Lodi A, Prouvost A. Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operations Research* 2021; 290(2): 405–421.

19. Karg B, Lucia S. Deep learning-based embedded mixed-integer model predictive control. In: Proc. European Control Conference (ECC). ; 2018: 2075-2080.

20. Masti D, Bemporad A. Learning binary warm starts for multiparametric mixed-integer quadratic programming. In: Proc. European Control Conference (ECC). ; 2019: 1494-1499.

21. Bertsimas D, Stellato B. Online Mixed-Integer Optimization in Milliseconds. *INFORMS Journal on Computing* 2022; 34(4): 2229-2248.

22. Löhr Y, Klauco M, Fikar M, Mönnigmann M. Machine Learning Assisted Solutions of Mixed Integer MPC on Embedded Platforms. *IFAC-PapersOnLine* 2020; 53(2): 5195-5200. 21st IFAC World Congress.

23. Cauligi A, Culbertson P, Schmerling E, Schwager M, Stellato B, Pavone M. CoCo: Online Mixed-Integer Control Via Supervised Learning. *IEEE Robotics and Automation Letters* 2022; 7(2): 1447-1454.

24. Cauligi A, Chakrabarty A, Di Cairano S, Quirynen R. PRISM: Recurrent Neural Networks and Presolve Methods for Fast Mixed-integer Optimal Control. In: . 168 of *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*. Proceedings of Machine Learning Research (PMLR); 2022: 34–46.

25. Borrelli F, Baotić M, Bemporad A, Morari M. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica* 2005; 41(10): 1709-1721.

26. Oberdieck R, Pistikopoulos EN. Explicit hybrid model-predictive control: The exact solution. *Automatica* 2015; 58: 152-159.

27. Floudas CA. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press . 1995.

28. Axehill D, Hansson A. A mixed integer dual quadratic programming algorithm tailored for MPC. In: Decision and Control, 2006 45th IEEE Conference on. IEEE. ; 2006: 5693–5698.

29. Hespanhol P, Quirynen R, Di Cairano S. A Structure Exploiting Branch-and-Bound Algorithm for Mixed-Integer Model Predictive Control. In: Proc. European Control Conference (ECC). ; 2019: 2763-2768.

30. Quirynen R, Di Cairano S. PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control. *Optimal Control Applications and Methods* 2020; 41(6): 2282-2307.

31. Frick D, Domahidi A, Morari M. Embedded optimization for mixed logical dynamical systems. *Computers & Chemical Engineering* 2015; 72: 21–33.

32. Axehill D, Hansson A. A dual gradient projection quadratic programming algorithm tailored for model predictive control. In: IEEE Conference on Decision and Control. ; 2008: 3057-3064.

33. Bemporad A, Naik VV. A Numerically Robust Mixed-Integer Quadratic Programming Solver for Embedded Hybrid Model Predictive Control. *IFAC-PapersOnLine* 2018; 51(20): 412-417. Proc. 6th IFAC NMPC Conf.

34. Stellato B, Naik VV, Bemporad A, Goulart P, Boyd S. Embedded mixed-integer quadratic optimization using the OSQP solver. In: European Control Conference. ; 2018: 1536-1541.

35. Balcan MF, Dick T, Sandholm T, Vitercik E. Learning to Branch. In: Proceedings Int. Conf. on Machine Learning. ; 2018: 344–353.

36. Gerdts M. Solving mixed-integer optimal control problems by branch&bound: a case study from automobile test-driving with gear shift. *Optimal Control Appl. and Methods*; 26(1): 1-18.

37. Achterberg T, Koch T, Martin A. Branching rules revisited. *Operations Research Letters* 2005; 33(1): 42–54.

38. Le Bodic P, Nemhauser G. An abstract model for branching and its application to mixed integer programming. *Mathematical Programming* 2017; 166(1-2): 369–405.

39. Luo ZQ, Ma WK, So AMC, Ye Y, Zhang S. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine* 2010; 27(3): 20–34.

40. Axehill D, Vandenberghe L, Hansson A. Convex relaxations for mixed integer predictive control. *Automatica* 2010; 46(9): 1540 - 1545.

41. Gurobi Optimization, LLC . Gurobi Optimizer Reference Manual. 2022. Accessed 8/15/22, www.gurobi.com.

42. MOSEK ApS . *The MOSEK optimization toolbox for MATLAB manual.* 2017.

43. Achterberg T, Bixby RE, Gu Z, Rothberg E, Weninger D. Presolve Reductions in Mixed Integer Programming. *INFORMS Journal on Computing* 2019; 32(2): 473 – 506.

44. Achterberg T, Wunderling R. *Mixed Integer Programming: Analyzing 12 Years of Progress*: 449–481; Berlin, Heidelberg: Springer Berlin Heidelberg . 2013.

45. Marcucci T, Tedrake R. Warm Start of Mixed-Integer Programs for Model Predictive Control of Hybrid Systems. *IEEE Transactions on Automatic Control* 2021; 66(6): 2433-2448.

46. Frey J, Di Cairano S, Quirynen R. Active-Set based Inexact Interior Point QP Solver for Model Predictive Control. *IFAC-PapersOnLine* 2020; 53(2): 6522-6528. 21st IFAC World Congress.

47. Liang J, Di Cairano S, Quirynen R. Early Termination of Convex QP Solvers in Mixed-Integer Programming for Real-Time Decision Making. *IEEE Control Systems Letters* 2021; 5(4): 1417-1422.

48. Savelsbergh MWP. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing* 1994; 6(4): 445-454.

49. GLPK . GLPK (GNU Linear Programming Kit). 2022. Accessed 8/15/22, www.gnu.org/software/glpk/#documentation.

50. Cbc . Cbc (Coin-or branch and cut). 2022. Accessed 8/15/22, www.github.com/coin-or/Cbc.

51. Wright SJ. *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM Publications . 1997.

52. Shahzad A, Kerrigan EC, Constantinides GA. A fast well-conditioned interior point method for predictive control. In: IEEE Conference on Decision and Control (CDC). ; 2010: 508-513.

53. Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge: University Press . 2004.

54. Todd MJ. Detecting infeasibility in infeasible-interior-point methods for optimization. In: Foundations of computational mathematics. Cambridge Univ. Press; 2004: 157–192.

55. Trespalacios F, Grossmann IE. Improved Big-M reformulation for generalized disjunctive programs. *Computers and Chemical Engineering* 2015; 76: 98-103.

56. dSPACE . Scalexio Product Page. 2022. Accessed 8/15/22, www.dspace.com/en/inc/home/products/hw/simulator_hardware/scalexio.cfm.

57. Aydinoglu A, Preciado VM, Posa M. Contact-aware controller design for complementarity systems. In: Proc. IEEE Conf. on Robotics and Automation. ; 2020: 1525-1531.

58. Mordatch I, Todorov E, Popović Z. Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Transactions on Graphics* 2012; 31(4): 1–8.