Overall Complexity Certification of a Standard Branch and Bound Method for Mixed-Integer Quadratic Programming

Shamisa Shoja, Daniel Arnström and Daniel Axehill

LINKÖPING UNIVERSITY

Overall Complexity Certification of a Standard Branch and Bound Method for Mixed-Integer Quadratic Programming

Shamisa Shoja, Daniel Arnström, and Daniel Axehill

Abstract— This paper presents a method to certify the computational complexity of a standard Branch and Bound method for solving Mixed-Integer Quadratic Programming (MIQP) problems defined as instances of a multi-parametric MIQP. Beyond previous work, not only the size of the binary search tree is considered, but also the exact complexity of solving the relaxations in the nodes by using recent results from exact complexity certification of active-set QP methods. With the algorithm proposed in this paper, a total worst-case number of QP iterations to be performed in order to solve the MIQP problem can be determined as a function of the parameter in the problem. An important application of the proposed method is Model Predictive Control for hybrid systems, that can be formulated as an MIQP that has to be solved in real-time. The usefulness of the proposed method is successfully illustrated in numerical examples.

I. INTRODUCTION

The main motivation for this work is model predictive control (MPC) for discrete-time hybrid systems. MPC is a model-based control strategy with the aim to design an optimal controller for multi-variable constrained systems, [1]. In MPC, a finite-horizon optimal control problem is solved at each sampling time, starting at the current state. Measurements are used to update the problem at the next sample, and the optimization is repeated over the shifted horizon.

Hybrid systems arise naturally in applications where physical principles interact with discrete events. There are different models for hybrid systems, the one that will be considered in this work is Mixed-Logical Dynamical (MLD) systems, [2]. In these models, both real-valued variables and binary-valued variables exist in the optimization problem to be solved online in each sample. The result is an optimization problem in the form of a Mixed-Integer Quadratic Program (MIQP), which are non-convex problems known to be $\mathcal{NP}$-hard [3], and hence potentially more challenging to solve than the LP or QP required for linear MPC.

The MIQPs can in MPC either be solved in real-time online or be formulated as a multi-parametric MIQP (mp-MIQP) and solved offline parametrically for a range of states. Once the problems have been solved offline, the necessary computations online are reduced to evaluating a look-up table, [4]–[6]. However, it is very challenging to solve and compute an efficient data structure for a parametric solution of an mp-MIQP. Furthermore, the memory requirement for the look-up table of the solution in the embedded system is potentially large. For these reasons, the online approach is often the only realistic solution. For that approach to also be considered reliable for more critical applications, a priori guarantees for that the computational requirements of the problem at hand do not exceed the hardware capabilities are desirable. Similar work for LPs and QPs can be found in [7] and [8]–[10], respectively. The first work in that direction for MIQPs can be found in [11]. The strategy to achieve this, and to obtain a significantly more practically useful complexity analysis than the conservative classical one, is to consider a specific set of MIQP problems encoded as an mp-MIQP. Each MIQP problem to be solved online is an instance of the mp-MIQP for a fixed parameter. Furthermore, beyond the traditional algebraic theoretical analysis, a certification algorithm is developed and used offline to analyze the complexity of all problems that might be requested to be solved online.

In this work, the optimization problem online is to be solved using a branch-and-bound-based MIQP solver. To compute the solution of an MIQP with integer variables modeled as binaries, a straightforward approach is to enumerate all possible combinations of binary variables and solve a QP for each such combination. As the number of binary variables increases, however, the computational effort for this approach grows exponentially. An alternative solution strategy commonly offering a remedy to this problem is the branch and bound (B&B) method [11]–[14], in which a sequence of relaxed QP problems are ordered and solved in a binary search tree to find an optimal mixed-integer solution. In the worst case, this method still requires the solution of exponentially many QPs, however, the complexity observed in practice is often significantly lower than explicit enumeration. The objective with the work in this paper is to continue the work in [11] where useful sufficiently exact complexity guarantees are computed a priori for branch-and-bound focusing on the size of the binary search tree. The main contribution in this work is to also analyze the complexity of the QP problems for the relaxations in the nodes of the binary search tree. This is done using recently presented state-of-the-art methods for exact complexity certification for active-set QP solvers, [8], [10], where the exact bound on the worst-case number of iterations required by an active-set QP solver is determined. Hence, the focus in this work is on the overall *worst-case complexity* in a B&B-based MIQP solver, in terms of the accumulated worst-case number of QP iterations in the subproblems necessary to solve in order to compute the optimal mixed-integer solution. This knowledge enables us to compute relevant complexity

S. Shoja, D. Arnström, and D. Axehill are with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. (Email: {shamisa.shoja, daniel.arnstrom, daniel.axehill}@liu.se)

bounds on the worst-case complexity of MIQP solvers based on B&B, which is of significant importance in the context of real-time MPC for hybrid systems.

The organization of the paper is as follows. Section II introduces the mp-MIQP problem formulation. Some background theory and the B&B algorithm for solving MIQPs are revisited in Section III. The main contribution of the paper, i.e., an algorithm for certification of MIQP, is presented in Section IV. Finally, numerical experiments are provided in Section V to illustrate the result of the proposed algorithm.

II. PROBLEM FORMULATION

For an MLD system, by considering the state variables as parameter vector θ and control actions as optimization variables x, the hybrid MPC problem can be cast into an mp-MIQP problem as follows [2], [11],

$$\mathcal{P}_{\text{MI}}(\theta): \quad \begin{aligned} \min_{x} \quad & \frac{1}{2}x^T H x + f^T x + \theta^T f_\theta^T x, & \text{(1a)} \\ \text{s.t.} \quad & Ax \leqslant b + W\theta, & \text{(1b)} \\ & x_i \in \{0,1\}, \quad \forall i \in \mathcal{B} & \text{(1c)} \end{aligned}$$

where the decision variable consists of n_c continuous and n_b binary variables, i.e., $x = [x_c^T, x_b^T]^T \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$, $n = n_c + n_b$, and the parameter vector is $\theta \in \Theta_0 \subset \mathbb{R}^{n_\theta}$. The set Θ_0 is assumed to be polyhedral. The MIQP problem, is given by $H \in \mathbb{S}_{++}^n$, $f \in \mathbb{R}^n$, $f_\theta \in \mathbb{R}^{n \times n_\theta}$, and the feasible set is determined by $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $W \in \mathbb{R}^{m \times n_\theta}$. Since the n_b optimization variables indexed by the set $\mathcal{B}$ are binary-valued, the problem is no longer convex and is known to be $\mathcal{NP}$-hard [3].

Throughout this paper, the notation $\{\mathcal{D}^i\}_{i=1}^N$ denotes a finite collection $\{\mathcal{D}^1, \dots \mathcal{D}^N\}$ of N elements. When N is unimportant, we use the notation $\{\mathcal{D}^i\}_i$ instead.

III. OPTIMIZATION PRELIMINARIES

In this section, some useful optimization preliminaries to be able to understand B&B is briefly discussed.

A. Quadratic Programming

The QP problem considered in this work is given by,

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T H x + f^T x, & \text{(2a)} \\ \text{s.t.} \quad & Ax \leqslant b, & \text{(2b)} \\ & A_\mathcal{E} x = b_\mathcal{E} & \text{(2c)} \end{aligned}$$

where $x \in \mathbb{R}^n$, $f \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $b_\mathcal{E} \in \mathbb{R}^p$ and matrices $H \in \mathbb{S}_{++}^n$, $A \in \mathbb{R}^{m \times n}$, and $A_\mathcal{E} \in \mathbb{R}^{p \times n}$. There are several methods to compute the solution to (2), such as active-set methods, interior-point methods, and various gradient methods, [15]. In this paper the (dual) active-set method in [16] is applied, since this type of method is known to be very suitable in branch-and-bound [17]. Even though this work does not currently consider that aspect, the active-set methods' warm starting capabilities are of great interest in branch and bound. Another important reason for this choice of method is that there exist recently presented certification methods for this solver, [8], [10].

Active constraints is an important concept in this type of methods which defines whether a constraint holds with equality, e.g., constraint i in (1b) is active if $A_i x(\theta) = b_i + W_i\theta$, where subscript i denotes the ith row of a matrix or vector. The optimal active set, denoted by $\mathcal{A}$, is defined as the set containing the indices of all constraints active at the optimal solution.

The idea of the active-set method is to iteratively make steps towards the optimal solution by solving a sequence of equality constrained QPs, in which some of the inequality constraints are imposed as active. These constraints are said to be included in the working set. See [15] for a detailed description of the active-set method. In this work, the number of equality constrained QPs that the active-set solver solves to find the optimal solution of the QPs will be called the iteration number. In [10], it is shown how this number can be found a priori and also how the relevant sequences of working-set changes can be computed as a function of the parameters in an mp-QP.

B. Mixed-Integer Quadratic Programming

Consider the following MIQP problem,

$$\mathcal{P}_{\text{MI}}(\bar{\theta}): \quad \begin{aligned} \min_{x} \quad & \frac{1}{2}x^T H x + \bar{f}^T x, & \text{(3a)} \\ \text{s.t.} \quad & Ax \leqslant \bar{b}, & \text{(3b)} \\ & x_i \in \{0,1\}, \quad \forall i \in \mathcal{B} & \text{(3c)} \end{aligned}$$

Each instance obtained from the original problem (1) by fixing the parameters to a specific value $\bar{\theta}$ will be a problem in the form in (3), with $\bar{f} = f + f_\theta\bar{\theta}$ and $\bar{b} = b + W\bar{\theta}$. The decision variables, $H, A,$ and $\mathcal{B}$ in (3) are defined as in (1).

In this work, the branch and bound (B&B) method is used to solve the optimization problem in (3). In the B&B search tree, the binary constraints are relaxed into interval constraints forming a so-called relaxation given by,

$$\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1): \quad \begin{aligned} \min_{x} \quad & \frac{1}{2}x^T H x + \bar{f}^T x, & \text{(4a)} \\ \text{s.t.} \quad & Ax \leqslant \bar{b}, & \text{(4b)} \\ & 0 \leqslant x_i \leqslant 1, \quad \forall i \in \mathcal{B}, & \text{(4c)} \\ & x_i = 0, \forall i \in \mathcal{B}_0, \ x_i = 1, \forall i \in \mathcal{B}_1 & \text{(4d)} \end{aligned}$$

that is a convex QP problem in the form in (2), where $\mathcal{B}_0, \mathcal{B}_1 \subseteq \mathcal{B}$ and $\mathcal{B}_0 \cap \mathcal{B}_1 = \emptyset$.

The procedure of B&B is as follows. Starting in the root node, all binary constraints (3c) in $\mathcal{P}_{\text{MI}}(\bar{\theta})$ are relaxed to (4c) resulting in a fully relaxed QP problem $\mathcal{P}(\bar{\theta}, \emptyset, \emptyset)$ which is solved. At the next level in the tree, one of the relaxed binary-constrained variables is fixed to 0 and 1, resulting into two new subproblems that form two new nodes in the B&B tree which are called the children of the parent node. The procedure is repeated and new nodes are explored further down in the tree. In the bottom of the tree the leaf nodes are found in which all binary constraints have been fixed.

An important property of B&B is that the relaxation in a certain node gives a lower bound on the optimal integer solution for the subtree below that node, whereas an integer

feasible solution provides an upper bound on the value function valid in the entire tree. To use the result from the relaxations to prune parts of the search tree from explicitly being explored is a fundamental idea in B&B. When the solution to a relaxation in a node is shown to satisfy one of the following cut conditions, then all nodes in the subtree below that node can be shown to be of no use and can therefore be disregarded, i.e., the tree is cut in that node [13]:

1) The relaxation is *infeasible*. Adding constraints will not change that. Hence, the entire subtree below that node is infeasible.
2) The optimal objective function value of the relaxation is *greater* than the one of the best known integer solution so far. Adding constraints will not decrease the objective function value. Hence, a better objective function value cannot be found in the subtree below that node.
3) The solution to the relaxation is *integer feasible*. Adding constraints will not decrease the objective function value. Hence, an optimal solution for the entire subtree below that node has already been found.

In this work, we assign the value function of an infeasible relaxed problem to infinity. Therefore, the first and second cut conditions can be tested simultaneously. In comparing the value function with the upper bound, if the comparison holds, that is if the node has an objective function value that is worse than the best known integer solution, the node will be cut. The integer feasibility cut condition is tested by looking at the active set, such that if the binary constraints (3c) are active, i.e., all the binary decision variables are either 0 or 1, then the solution is integer feasible and the tree can be pruned and the upper bound is updated if improved. If none of the cut conditions hold, the node is split into two new subproblems by fixing a relaxed binary variable indexed by k to 0 and 1, forming two new subproblems $\mathcal{P}(\bar{\theta}, \mathcal{B}_0 \cup \{k\}, \mathcal{B}_1)$ and $\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1 \cup \{k\})$, respectively. These two subproblems are inserted in the sorted list $\mathcal{T}$, implementing a priority queue, to be analyzed as the tree is explored. The priority used in $\mathcal{T}$ is determined by the choice of the tree exploration strategy (e.g., depth-first) and whether the left or right branch should be explored first.

Algorithm 1 summarizes the B&B method for solving problem (3). In this algorithm, the sorted list $\mathcal{T}$ stores the subproblems to be solved in a user-defined exploration order, $\bar{x}$ is the best integer feasible solution so far, and $\bar{J}$ is its corresponding objective function value (the upper bound). Moreover, x and J are the associated optimal solution and objective function value of the relaxation, respectively. As mentioned earlier, if the solution of a subproblem turns out to be infeasible, we set $J = \infty$.

The QP relaxation (4) is solved using an active-set method as outlined in Section III-A. Of particular interest in this work is the number of QP iterations required to solve a relaxation, and the sum of all such QP iterations over the entire B&B tree for each choice of the parameter.

Algorithm 1 Branch and Bound for MIQP

Input: MIQP problem $\mathcal{P}_{\mathrm{MI}}(\bar{\theta})$ for $\bar{\theta}$ given
Output: $\bar{J}, \bar{x}$
1: $\bar{J} \leftarrow \infty, \bar{x} \leftarrow$ void
2: $\mathcal{T} \leftarrow \emptyset$
3: Push $\mathcal{P}(\bar{\theta}, \emptyset, \emptyset)$ to $\mathcal{T}$
4: **while** $\mathcal{T} \neq \emptyset$ **do**
5:    Pop $\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1)$ from $\mathcal{T}$
6:    $J, x \leftarrow$ Solve $\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1)$
7:    **if** $J \geq \bar{J}$ **then**
8:      There exists no feasible solution to $\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1)$ which is better than $\bar{x}$
9:    **else if** all binary variables are active **then**
10:      Better integer feasible solution has been found
11:      $\bar{J} \leftarrow J$
12:      $\bar{x} \leftarrow x$
13:    **else**
14:      Select $k : k \in \mathcal{B}, k \notin (\mathcal{B}_0 \cup \mathcal{B}_1)$
15:      Push $\mathcal{P}(\bar{\theta}, \mathcal{B}_0 \cup \{k\}, \mathcal{B}_1)$ and $\mathcal{P}(\bar{\theta}, \mathcal{B}_0, \mathcal{B}_1 \cup \{k\})$ to $\mathcal{T}$
16:    **end if**
17: **end while**

IV. COMPLEXITY CERTIFICATION OF MIQP

In this section, an algorithm that analyzes the B&B method in Algorithm 1 for all values of the parameter θ is presented. As mentioned, in hybrid MPC, these parameters could be system states or/and reference signals. The main idea behind the complexity certification method is to keep track of the complexity measure of interest parametrically as a function of θ. An interpretation of this is running Algorithm 1 parametrically and partitioning Θ_0 into regions containing parameters which yield for example the same working-set changes when employing the active-set method in [10] as the inner solver in B&B. In particular, the framework to be presented is able to rigorously analyze and upper bound the computational complexity for the process, in terms of e.g., the size of the B&B tree or the total number of QP iterations.

Algorithm 2 represents the proposed parametric B&B-based MIQP certification algorithm with the aim of certifying the solution process of the problem in (1) in detail. An important property of the algorithm is that it iteratively divides and explores the parameter space based on the poly-hedral partition from a QP certification algorithm. It therefore receives the mp-MIQP problem (1) and the parameter region Θ_0 as inputs and outputs a partitioning of the parameter space $\{\Theta^i\}_{i=1}^{N_f}$ together with the corresponding complexity measure $\kappa^i(\theta)$ for solving all the encountered QP relaxations (4) for $\theta \in \Theta^i$. In addition to the usual sorted list $\mathcal{T}$ in B&B, there are two additional lists of tuples in Algorithm 2. One is the list $\mathcal{S}$ holding regions in the parameter space where the analysis has not yet terminated. It contains tuples $(\Theta, \kappa, \mathcal{T}, \bar{\mathbb{J}}(\theta))$, where Θ is the associated polyhedral parameter set, $\mathcal{T}$ is the normal list (cf. Algorithm 1) of B&B to store subproblems, κ is the complexity measure for $\theta \in \Theta$, and $\bar{\mathbb{J}}(\theta)$ is a collection of the upper bounds on the value

function. Another related list is $\mathcal{F}$ which contains regions in the parameter space where the process has terminated, i.e., $\mathcal{F}$ holds the final partition. It consists of a tuple with the information (Θ, κ) for every terminated region with the associated complexity measure.

For what follows, the following definition is required.

Definition 1: A set of polyhedra $\{\mathcal{R}^i\}_{i=1}^N$ is a *polyhedral partition* of Θ if $\mathring{\mathcal{R}}^i \cap \mathring{\mathcal{R}}^j = \emptyset, i \neq j$, and $\cup_{i=1}^N \mathcal{R}^i = \Theta$, where $\mathring{\mathcal{R}}^i$ denotes the interior of the region $\mathcal{R}^i$.

At each iteration of Algorithm 2, a region Θ will be popped from $\mathcal{S}$. If the list $\mathcal{T}$ associated to that region is empty, the exploration of the tree has been completed for all parameters in Θ and is, hence, added to the final partition $\mathcal{F}$ (Step 8). Otherwise a node in $\mathcal{T}$ will be selected and the corresponding mp-QP relaxation will be solved and certified over Θ in Step 11. The function QPCERT here is any complexity certification algorithm for QPs, e.g. [10], which takes an mp-QP problem $\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1)$ and a parameter region Θ as inputs and returns a polyhedral partition $\{\Theta^j\}_{j=1}^N$ of Θ. In each region j, the function provides the computational complexity κ^j for solving $\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1)$ for all parameters in Θ^j, as well as the explicit solution of the mp-QP relaxation including the optimal active set $\mathcal{A}^j$ and the value function $J^j(\theta)$ which is piecewise quadratic over the polyhedral region. If $\mathcal{P}(\theta)$ is infeasible in Θ^j, we encode that by $J^j(\theta) = \infty$.

After the mp-QP relaxation has been processed, the sorted list $\mathcal{T}$ is updated in the CUTCONDEVAL procedure for each new region based on the optimal active set $\mathcal{A}^j$ and the value function J^j. The update consists of testing the three cut conditions described in Section III-B *parametrically*. If none of these conditions are invoked for Θ^j, two new nodes are created according to standard B&B (based on $\mathcal{A}^j$ and the binaries that are fixed in the current node) and added to $\mathcal{T}^j$.

The B&B algorithms in this work satisfy the following assumptions.

Assumption 1: The tree exploration strategy is depth-first in Algorithms 1 and 2. Moreover, the branch variable order is fixed (e.g., parameter independent) and known beforehand, and the order in which the 0 and 1 branch will be explored is fixed and known beforehand in both algorithms.

The proposed certification method is formally analyzed in the following subsections as follows. The spatial decomposition employed in Algorithm 2 is described in Section IV-A. Properties of the parametric B&B search tree are formalized in Section IV-B and finally, the main contributions of the proposed algorithm are stated in Theorems 1 and 2 in Section IV-C.

Algorithm 2 Certification of Branch and Bound for MIQP

Input: mp-MIQP problem $\mathcal{P}_{\text{MI}}(\theta)$, and Θ_0
Output: Partition $\mathcal{F} = \{(\Theta^i, \kappa^i)\}_{i=1}^{N_f}$
 1: $\kappa_0 \leftarrow 0$, $\bar{\mathbb{J}}_0 \leftarrow \{\infty\}$
 2: $\mathcal{T} \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$
 3: Push $\mathcal{P}(\theta, \emptyset, \emptyset)$ to $\mathcal{T}$
 4: Push $(\Theta_0, \kappa_0, \mathcal{T}, \bar{\mathbb{J}})$ to $\mathcal{S}$
 5: **while** $\mathcal{S} \neq \emptyset$ **do**
 6:    Pop $(\Theta, \kappa, \mathcal{T}, \bar{\mathbb{J}})$ from $\mathcal{S}$
 7:    **if** $\mathcal{T} = \emptyset$ **then**
 8:       Push (Θ, κ) to $\mathcal{F}$ and **goto** Step 5
 9:    **end if**
10:    Pop $\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1)$ from $\mathcal{T}$
11:    $\{(\Theta^j, \kappa^j, \mathcal{A}^j, J^j)\}_{j=1}^N \leftarrow \text{QPCERT}(\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1), \Theta)$
12:    **for** $j \in \{1, \ldots, N\}$ **do**
13:       $\mathcal{T}^j, \bar{\mathbb{J}}^j \leftarrow \text{CUTCONDEVAL}(\mathcal{T}, \Theta^j, \mathcal{A}^j, J^j, \bar{\mathbb{J}})$
14:       Push $(\Theta^j, \kappa + \kappa^j, \mathcal{T}^j, \bar{\mathbb{J}}^j)$ to $\mathcal{S}$
15:    **end for**
16: **end while**

18: **procedure** CUTCONDEVAL($\mathcal{T}, \Theta, \mathcal{A}, J, \bar{\mathbb{J}}$)
19:    **if** $\exists \bar{J} \in \bar{\mathbb{J}} : J(\theta) \geq \bar{J}(\theta), \forall \theta \in \Theta$ **then**
20:       There exists no θ for which a feasible solution
      of $\mathcal{P}$ provides better solution
21:    **else if** all binary variables are in $\mathcal{A}$ **then**
22:       Potentially better integer feasible solution has
      been found
23:       $\bar{\mathbb{J}}(\theta) \leftarrow \bar{\mathbb{J}}(\theta) \cup \{J(\theta)\}$
24:    **else**
25:       Select $k : k \in \mathcal{B}, k \notin (\mathcal{B}_0 \cup \mathcal{B}_1)$
26:       Push $\mathcal{P}(\theta, \mathcal{B}_0 \cup \{k\}, \mathcal{B}_1)$ and $\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1 \cup \{k\})$
      to $\mathcal{T}$
27:    **end if**
28:    **return** $\mathcal{T}, \bar{\mathbb{J}}$
29: **end procedure**

A. Decomposition of the parameter space

As mentioned above, an interpretation of Algorithm 2 is that it performs iterations of Algorithm 1 parametrically. As a result, the parameter space will be partitioned depending on, for example, how the exploration of the B&B tree differs for different parameters, but also the certification of the relaxations. We clarify how this partitioning occurs and some of its properties in this subsection.

At Step 11, after processing a node, the node is divided spatially into N subproblems with corresponding regions Θ^j by the QP certification algorithm, each holding a copy of the B&B-list $\mathcal{T}$. The CUTCONDEVAL procedure, which tests the pruning conditions parametrically, is then individually applied for each local tree of the N subproblems.

The infeasibility and integer feasibility cut conditions can directly be applied to the output from QPCERT $(\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1), \Theta)$, given by $J^j(\theta) = \infty$ and the active set $\mathcal{A}^j$ for the region, respectively, with no additional spatial partitioning. In the dominance cut condition however, if it is performed exactly, further spatial partitioning is im-

posed by $J^j(\theta) \geq \bar{J}(\theta)$, additionally splitting the node spatially into two new subproblems with finer regions. To make the certification more tractable by maintaining the polyhedral structure of the partition, at the price of conservatism, the spatial partitioning originating from the dominance cut can be avoided by only cutting if the condition is satisfied for the entire region, i.e., $J^j(\theta) \geq \bar{J}(\theta)$, $\forall \theta \in \Theta^j$, similarly to what was done in [11]. Note that the framework is in principle capable of exact certification, however, for the MIQP case the price for that is that the geometry becomes non-polyhedral and will in general include regions described by quadratic functions. To investigate the details of the less tractable exact case is therefore decided to be beyond the scope of this work. The price for not considering the exact case is that some conservatism will be included in the certification result and multiple upper bounds related to what in parametric programming for MIQPs is known as "overlaps" have to be introduced. Note that, however, the result from Algorithm 2 is guaranteed to *upper* bound the online complexity. The end result of the proposed algorithm is many local B&B trees, which represent all possible B&B trees that are used to compute the solution of the problem for all parameter values online.

In the following, we ensure that the partitioning made in Algorithm 2 is correct, i.e., the generated regions cover the entire Θ_0 and have no overlap among themselves.

Lemma 1 (Maintenance of complete partition): At an arbitrary iteration of Algorithm 2, the union of regions in $\mathcal{F}$ and $\mathcal{S}$ forms a partition of Θ_0.

Proof: We will prove the lemma by induction. Let $\{\Theta_S^i\}_i$ and $\{\Theta_{\mathcal{F}}^i\}_i$ denote the regions contained in $\mathcal{S}$ and $\mathcal{F}$, respectively, at the start of an iteration. Similarly, let $\{\Theta_{S'}^i\}_i$ and $\{\Theta_{\mathcal{F}'}^i\}_i$ denote the regions contained in $\mathcal{S}$ and $\mathcal{F}$, respectively, at the end of the iteration. Finally, let $\mathcal{T}$ and Θ denote the list and region, respectively, that are selected from $\mathcal{S}$ at the start of the iteration.

If $\mathcal{T} = \emptyset$ we get $\{\Theta_{\mathcal{F}'}^i\}_i = \{\Theta_{\mathcal{F}}^i\}_i \cup \{\Theta\}$ and $\{\Theta_{S'}^i\}_i = \{\Theta_S^i\}_i \setminus \{\Theta\}$ after Step 8 is executed. Clearly $\{\Theta_{\mathcal{F}'}^i\}_i$ and $\{\Theta_{S'}^i\}_i$ will form a partition of Θ_0 if $\{\Theta_{\mathcal{F}}^i\}_i$ and $\{\Theta_S^i\}_i$ do, since we have the same underlying regions (only a single region Θ has been moved from $\mathcal{S}$ to $\mathcal{F}$ during the iteration).

Next, consider the case when $\mathcal{T} \neq \emptyset$. Then the regions in $\mathcal{F}$ remain unchanged and we will partition Θ into $\{\Theta^i\}_i$ in Step 11. Hence, we get $\{\Theta_{S'}^i\}_i = (\{\Theta_S^i\}_i \setminus \{\Theta\}) \cup \{\Theta^i\}_i$. Then, since $\cup_i \Theta^i = \Theta$ from the properties of QPCERT, it follows that $\cup_i \Theta_{S'}^i = \cup_i \Theta_S^i$. Moreover, since $\Theta^i \subseteq \Theta$ $\forall i$ and that $\mathring{\Theta}^i \cap \mathring{\Theta}^j = \emptyset$ if $i \neq j$ (both, again, from the properties of QPCERT), all regions in $\{\Theta_{S'}^i\}_i$ and $\{\Theta_{\mathcal{F}'}^i\}_i$ are disjoint if the regions $\{\Theta_S^i\}_i$ and $\{\Theta_{\mathcal{F}}^i\}_i$ are. In conclusion, the regions in $\mathcal{S}$ and $\mathcal{F}$ at the end of an iteration will form a partition of Θ_0 if the regions in $\mathcal{S}$ and $\mathcal{F}$ do so at the start of the iteration (completing the induction step).

For the base case we have that $\mathcal{F} = \emptyset$ and $\mathcal{S}$ only contains Θ_0 at the start of the algorithm, which trivially forms a partition of Θ_0. ∎

As a special case, Lemma 1 ensures that the entire region of interest Θ_0 will be considered before termination.

Corollary 1 (Complete partition at termination): Assume that Algorithm 2 has terminated with the partition $\{(\Theta^i, \kappa^i)\}_i$. Then $\cup_i \Theta^i = \Theta_0$ and $\mathring{\Theta}^i \cap \mathring{\Theta}^j = \emptyset$ for $i \neq j$. That is, Θ_i is a partition of Θ_0.

Proof: Follows from Lemma 1 and that Algorithm 2 terminates when $\mathcal{S} = \emptyset$. ∎

B. Properties of the search tree

As mentioned in Section IV-A, the parametric analysis of the dominance cut is performed approximately in this work since the cost of an exact analysis would be quadratic regions. To ensure that Algorithm 2 still provides correct upper complexity bounds, despite this approximation, we review some fundamental properties of the B&B tree below, which we then use in Section IV-C to prove the correctness of the proposed method.

Let a *node* be characterized by a tuple $\eta \triangleq (\mathcal{B}_0, \mathcal{B}_1)$, where $\mathcal{B}_0$ and $\mathcal{B}_1$ are defined in (4d). The following definition is required to formalize the properties of the parametric B&B.

Definition 2: The node $\eta = (\mathcal{B}_0, \mathcal{B}_1)$ is a *descendant* to node $\hat{\eta} = (\hat{\mathcal{B}}_0, \hat{\mathcal{B}}_1)$, denoted by $\eta \in \mathcal{D}(\hat{\eta})$, if $\mathcal{B}_0 \supseteq \hat{\mathcal{B}}_0$ and $\mathcal{B}_1 \supseteq \hat{\mathcal{B}}_1$.

Lemma 2: Let $J_\eta^*(\theta)$ denote the value function for the mp-QP relaxation in node η and let $\delta \in \mathcal{D}(\eta)$, then $J_\eta^*(\theta) \leq J_\delta^*(\theta)$.

Proof: It directly follows from standard arguments from B&B that the feasible set for the mp-QP relaxation in node δ is a subset of the feasible set for the mp-QP relaxation in node η, since some additional equality constraints have been added in δ. ∎

Let $\mathbb{B}_\eta(\theta)$ be the set of nodes which have yielded an integer feasible solution up until node η is processed, given an ordering of $\mathcal{T}$. Moreover, let $\bar{J}_\eta(\theta)$ denote the lowest (i.e., best) upper bound found up until η is processed, i.e.,

$$\bar{J}_\eta(\theta) \triangleq \min_{\tilde{\eta} \in \mathbb{B}_\eta(\theta)} J_{\tilde{\eta}}^*(\theta) \tag{5}$$

The following lemma shows that if the dominance cut condition is invoked for node η, the best upper bound $\bar{J}_\eta(\theta)$ will not be changed (in particular not improved) by processing any descendant of η.

Lemma 3: Assume that $J_\eta^*(\theta) \geq \bar{J}_\eta(\theta)$ in a node η. Then the dominance cut condition will be satisfied in that node and

$$\min_{\tilde{\eta} \in \mathbb{B}_\eta(\theta) \cup \mathcal{D}(\eta)} J_{\tilde{\eta}}^*(\theta) = \bar{J}_\eta(\theta) \tag{6}$$

Proof: We have that

$$\min_{\tilde{\eta} \in \mathbb{B}_\eta(\theta) \cup \mathcal{D}(\eta)} J_{\tilde{\eta}}^*(\theta) = \min \left\{ \min_{\tilde{\eta} \in \mathbb{B}_\eta(\theta)} J_{\tilde{\eta}}^*(\theta), \min_{\delta \in \mathcal{D}(\eta)} J_\delta^*(\theta) \right\}$$

$$= \min \left\{ \bar{J}_\eta(\theta), \min_{\delta \in \mathcal{D}(\eta)} J_\delta^*(\theta) \right\}$$

$$= \bar{J}_\eta(\theta)$$

where the last equality follows from $J_\delta^*(\theta) \geq J_\eta^*(\theta) \geq \bar{J}_\eta(\theta)$ $\forall \delta \in \mathcal{D}(\eta)$, i.e., from Lemma 2 and the premise. ∎

Hence, processing descendants of a node for which the dominance cut was invoked would not change the best known upper bound. As a result, the exploration of, and application

of cut conditions in, the rest of the tree will not be affected if $\mathcal{T} \cup D(\eta)$ is used instead of $\mathcal{T}$ after the dominance cut is invoked in node η. We formalize this notion in the following corollary.

Corollary 2 (Futility of processing dominated descendants): Consider two cases of the list $\mathcal{T}$ after the dominance cut is invoked when processing node η,

1) $\mathcal{T}$ is used (normal operation)
2) $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{D}(\eta)$ is used (redundant addition)

Let $\mathbb{B}^1$ be the set of *all* nodes explored during case 1 and let $\mathbb{B}^2$ be the set of *all* nodes explored during case 2. Then $\mathbb{B}^2 \setminus \mathbb{B}^1 = \mathcal{D}(\eta)$. That is, exploring the nodes in $\mathcal{D}(\eta)$ does not affect the subsequent exploration of nodes in the tree.

Proof: The only thing that might alter the exploration is if the upper bound is changed by processing additional nodes (which could prune nodes that would otherwise be explored). However, from Lemma 3 processing any descendants of η can never change the best upper bound, which in turn means that further exploration is unaffected by processing any $\delta \in \mathcal{D}(\eta)$. ∎

C. Properties of the certification algorithm

So far we have shown the parametric behavior of Algorithm 2 and the properties of the search tree. In this subsection, the properties of the proposed certification algorithm are derived in the following theorems.

Theorem 1: Assume Assumption 1 holds. Let $\mathbb{B}(\theta)$ denote the set of all nodes explored to solve the problem in (3) for a fixed θ using Algorithm 1. Moreover, let $\hat{\mathbb{B}}_k(\theta)$ denote all nodes explored in Algorithm 2 for region Θ_k before terminating. Then $\mathbb{B}(\theta) \subseteq \hat{\mathbb{B}}_k(\theta)$, $\forall \theta \in \Theta_k$.

Proof: First, for the analysis to be meaningful, we stress that it is implicitly assumed that QPCERT certifies the QP solver correctly. In particular, this means that the active set and feasibility is correctly identified for each parameter value. As a result, the correctly identified active set and infeasibility directly leads to correctly invoked integer feasibility and infeasibility cuts, respectively. What remains is the dominance cut condition, which will now be analyzed in more detail. Consider the processing of node η on region Θ in Algorithm 2 and assume that a conservative upper bound $\tilde{J}(\theta)$ on Θ is used for the dominance cut (i.e., $\tilde{J}(\theta) \geq \bar{J}_\eta(\theta), \forall \theta \in \Theta$). Then we have the following cases that could occur in Step 19 of Algorithm 2:

1) $J_\eta^*(\theta) \geq \tilde{J}(\theta) \ \forall \theta \in \Theta \rightarrow$ dominance cut invoked correctly $\forall \theta \in \Theta$.
2) $\exists \tilde{\theta} \in \Theta$ such that $J_\eta^*(\tilde{\theta}) < \tilde{J}(\theta)$
   a) $J_\eta^*(\theta) < \bar{J}_\eta(\theta) \ \forall \theta \in \Theta \rightarrow$ dominance cut dismissed correctly for all $\theta \in \Theta$.
   b) Otherwise dominance cut is dismissed incorrectly for all $\theta \in \Theta^\epsilon \triangleq \{\theta \in \Theta : J_\eta^*(\theta) \geq \bar{J}_\eta(\theta)\}$

The most critical situation is in 2b), where we might explore descendants to η in the certification algorithm, while these would be pruned in Algorithm 1 pointwise. The possible extra nodes in the analysis are added to $\mathcal{T}^k$ according to Assumption 1 and consequently, when a new node is popped from the list in Step 10 of Algorithm 2 to be analyzed, it is either a relevant one that would be solved in the online solver for a specific parameter in the corresponding region, or a redundant one which has been cut away in Algorithm 1. For parameters with redundant nodes where case 2b) in the proof of Theorem 1 occurs, we have that $J_\eta^*(\theta) \geq \bar{J}_\eta(\theta)$. According to Corollary 2, the extra exploration of descendants to η will not affect the exploration of future nodes. Therefore, $\hat{\mathbb{B}}_k(\theta) \supseteq \mathbb{B}(\theta)$, $\forall \theta \in \Theta_k$. ∎

Theorem 2: Assume Assumption 1 holds. Let $\kappa(\theta)$ be a complexity measure for solving the QP relaxation $\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1)$ returned by QPCERT $(\mathcal{P}(\theta, \mathcal{B}_0, \mathcal{B}_1), \Theta)$ and that it satisfies $\kappa^j = \kappa(\theta)$, $\forall \theta \in \Theta^j \subseteq \Theta$, $\forall j$. Moreover, let $\{\Theta^i, \kappa_{tot}^i\}_{i=1}^{N_{\mathcal{F}}}$ be the collection of tuples in $\mathcal{F}$ returned by Algorithm 2 and let $\kappa_{tot}^*(\theta)$ be the total complexity for solving all QP relaxations encountered in Algorithm 1 for a specific $\theta \in \Theta_0$. Then $\kappa_{tot}^i \geq \kappa_{tot}^*(\theta)$, $\forall \theta \in \Theta^i$.

Proof: Based on the result from Theorem 1, the QP relaxations considered in Algorithm 2 for a single point θ will be a superset of the QP relaxations that would be solved in Algorithm 1. This in turn means that the total complexity $\{\kappa_{tot}^i\}_i$ returned by Algorithm 2 will be an upper bound on the total complexity of Algorithm 1 (pointwise), i.e., $\kappa_{tot}^i \geq \kappa_{tot}^*(\theta)$, $\forall \theta \in \Theta^i$. ∎

Remark 1: There is a freedom in the choice of complexity measure and Algorithm 2 is not restricted to a specific such measure. If the QP certification in [10] is employed in Step 11, then the complexity measure κ^j will be the iteration number, that is the total number of linear system of equations that have to be solved. In other words, κ^j will be the number of equality constrained QPs that an active-set method requires to solve to reach the optimal solution. Another alternative is to fix κ^j to 1. Then the complexity measure will be the total number of nodes required to be explored in the B&B tree, i.e., the number of relaxations (as in [11]). Another important choice is if κ^j holds the number of floating point operations (flops) for solving the relaxations, given by the QP certification algorithm. It then results in the total number of operations performed to solve all the relaxations in B&B.

V. NUMERICAL EXPERIMENTS

In this section, the proposed algorithm is tested in numerical experiments to illustrate its usefulness. To experimentally verify the analysis result, the upper bound on the complexity from Algorithm 2 in a given θ is compared with the result of the online solver implementing Algorithm 1. In the experiment, the complexity measure $\kappa(\theta)$ is chosen as the number of QP iterations. That is, for each value of $\theta \in \Theta_0$, the presented end result is the upper bound of the accumulated number of QP iterations from all relaxations necessary to solve in order to solve the mixed-integer problem for that particular value of θ. In the experiments, the QP certification algorithm from [10] was used to certify the relaxations in Step 11 of Algorithm 2.

For the evaluation, ten random mp-MIQPs have been generated with $n_c = 2$, $n_b = 4$, $m = 6$, and $n_\theta = 2$. The

mp-MIQPs have the following form

$$H = \bar{H} \times \bar{H}^T, \qquad f \sim \mathcal{N}(0,1), \qquad f_\theta \sim \mathcal{N}(0,1),$$
$$A \sim \mathcal{U}([0,1]), \qquad b \sim \mathcal{U}([1,2]), \qquad W \sim \mathcal{U}([0,2]). \qquad (7)$$

where $\bar{H} \sim \mathcal{N}(0,1)$, i.e., the Hessian matrices have been chosen such that they are symmetric and positive definite. Furthermore, the parameter set considered is $-1 \leq \theta \leq 1$.

Note that in the experiments, the conservative comparison is performed in the dominance cut in Algorithm 2 and as a result the partition will be polyhedral. The computation time required for the complexity certification of each one of the ten random examples are listed in Table I, when executed on an Intel® Core 1.8 GHz i7-8565U CPU. Furthermore, the worst-case overall QP iteration count κ_{max} of terminated regions is provided in Table I. All numerical experiments were implemented in MATLAB and Gurobi 9.0.2 [18] was used to solve the potentially indefinite QP optimization problem in Step 19 of Algorithm 2.

TABLE I
RESULTS FROM A PROOF-OF-CONCEPT MATLAB IMPLEMENTATION OF ALGORITHM 2 FOR 10 RANDOMLY GENERATED MP-MIQPS. κ_{max} IS THE WORST-CASE ACCUMULATED ITERATION NUMBER, AND t_{cert} DENOTES THE COMPUTATION TIME FOR THE CERTIFICATION IN SECONDS.

$mp - MIQP$	$t_{cert}[s]$	κ_{max}
# 1	12	27
# 2	231	41
# 3	97	51
# 4	332	51
# 5	143	46
# 6	173	38
# 7	399	48
# 8	572	47
# 9	327	47
# 10	785	58

The resulting partitioning of the parameter space based on the total number of iterations for one of the examples of random mp-MIQPs is shown in Fig. 1, where parameters in a region that share the same number of iterations are illustrated using the same color.

In order to verify the result from the certification, the worst-case iteration bounds obtained by the presented method were compared with results from Monte-Carlo (MC) simulations, i.e., applying Algorithm 1 to the same MIQP problems (3) for some specific parameters from the parameter set. To have the same complexity measure also online, the dual-active set method in [16] was applied to solve the QP relaxations in Step 6 of Algorithm 1, and for each evaluated parameter value, the accumulated number of QP iterations in all relaxations necessary to solve was recorded.

Fig. 2 demonstrates the result of solving the online MIQP problem using Algorithm 1 for 10000 samples on a deterministic grid in the parameter space, for the same example as in Fig. 1. The Chebychev centers of regions in Fig. 1 were also included as sample points to guarantee that there was at least one sample from each region. In
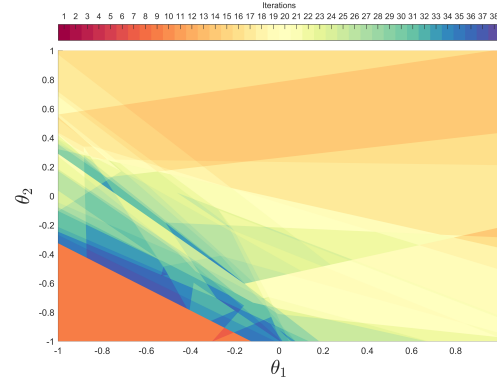

Fig. 1. Results from Algorithm 2 applied to a random example. The colors illustrate the accumulated number of QP iterations for a specific θ.

the experiments, the performed iterations by the QP solver have also been considered in the case when the problem is infeasible [10]. As the figures indicate, the result from the complexity certification coincides with the online algorithm in all sample points, despite that the conservative upper bound is used in Algorithm 2. The conclusions from the experiments are that the quality of the upper bound is clearly practically useful and the bound seems to be close to tight.
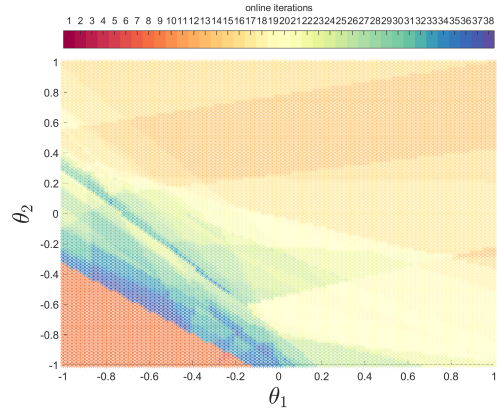

Fig. 2. The accumulated number of QP iterations for 10000 samples specified by * in the parameter space derived by applying Algorithm 1 to the same example as in Fig. 1. Points with the same color share the same number of complexity numbers.

To illustrate the amount of conservativeness that can be expected when the bound is not tight, two more experiments have been performed. In the first case, we increased the number of binary decision variables and inequality constraints to 8 ($n_b = m = 8$), with the same number of real-valued variable and parameters as the first run and a random mp-MIQP was generated in the form of (7) with the appropriate dimensions, and Algorithm 2 was applied to certify the problem. The maximum total iteration count in the worst region here was recorded as $\kappa_{max} = 126$. To verify the result, the online MIQP problem was solved in 10000 sample points on a deterministic grid in the parameter space. The results verified that the certification algorithm did never

underestimate the number of performed QP iterations online, as expected. In 7% of the analyzed points, the analysis turned out not to be exact and was conservative. In the second experiment, the dimension of parameters increased to $n_\theta = 3$ and another mp-MIQP was generated randomly with the variables similar to the first experiment dimension-wise. Here, the recorded overestimation difference between the certification and online results when running Algorithm 1 for 1000 parameters taken deterministically from the parameter space as well as 8400 Chebychev centers of the terminated regions was 6%. The differences originate from the dominance cut condition in Step 19 of Algorithm 2, which as described in Section IV-A, is somewhat conservative. However, it is considered highly useful in most applications.

VI. CONCLUSION

In this paper, complexity certification of a standard branch and bound method for MIQPs has been addressed. An algorithm for computing a useful upper bound of the worst-case computational complexity for solving any possible MIQP that can arise from a specific parameter in a polyhedral parameter set has been presented. Compared to the previous work, the current work uses recent algorithms for exact complexity certification of active-set QP methods, enabling also taking into account the complexity originating from solving the relaxations in the nodes. Even though the main focus in this work has been a complexity measure in terms of accumulated number of QP iterations, there is a freedom in this choice and alternatives such as flops and the size of the search tree can in principle be considered. Results from numerical experiments illustrate that the result is in general somewhat conservative but still practically useful. In fact, in many of the considered examples, the computed bound is tight. In future work, MILPs will be considered, the details of exact non-conservative application of the dominance cut condition will be studied, and more general branching-rules will be considered.

ACKNOWLEDGMENT

REFERENCES

[1] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[3] L. A. Wolsey, *Integer programming*. John Wiley & Sons, 2020.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[5] V. Dua, N. A. Bozinis, and E. N. Pistikopoulos, "A multiparametric programming approach for mixed-integer quadratic engineering problems," *Computers & Chemical Engineering*, vol. 26, no. 4-5, pp. 715–733, 2002.

[6] D. Frick, A. Domahidi, and M. Morari, "Embedded optimization for mixed logical dynamical systems," *Computers & Chemical Engineering*, vol. 72, pp. 21–33, 2015.

[7] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1524–1534, 2011.

[8] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[9] ——, "Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs," *Automatica*, vol. 100, pp. 29–37, 2019.

[10] D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, 2022.

[11] D. Axehill and M. Morari, "Improved complexity analysis of branch and bound for hybrid mpc," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 4216–4222.

[12] W. Zhang, "Branch-and-bound search algorithms and their computational complexity." USC/Information Sciences Institute, Tech. Rep., 1996.

[13] S. Almér and M. Morari, "Efficient online solution of multi-parametric mixed-integer quadratic problems," *International Journal of Control*, vol. 86, no. 8, pp. 1386–1396, 2013.

[14] D. Axehill, T. Besselmann, D. M. Raimondo, and M. Morari, "A parametric branch and bound approach to suboptimal explicit hybrid mpc," *Automatica*, vol. 50, no. 1, pp. 240–246, 2014.

[15] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[16] D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL' updates," *IEEE Transactions on Automatic Control*, 2022.

[17] D. Axehill and A. Hansson, "A mixed integer dual quadratic programming algorithm tailored for mpc," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 5693–5698.

[18] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com