2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Exact Complexity Certification of Start Heuristics in Branch-and-Bound Methods for Mixed-Integer Linear Programming

Shamisa Shoja and Daniel Axehill*

***Abstract*— Model predictive control (MPC) with linear performance measure for hybrid systems requires the solution of a mixed-integer linear program (MILP) at each time instance. A well-known method to solve MILP problems is branch-and-bound (B&B). To enhance the performance of B&B, start heuristic methods are often used, where they have shown to be useful supplementary tools to find good feasible solutions early in the B&B search tree, hence, reducing the overall effort in B&B to find optimal solutions. In this work, we extend the recently-presented complexity certification framework for B&B-based MILP solvers to also certify computational complexity of the start heuristics that are integrated into B&B. Therefore, the exact worst-case computational complexity of the three considered start heuristics and, consequently, the B&B method when applying each one can be determined offline, which is of significant importance for real-time applications of hybrid MPC. The proposed algorithms are validated by comparing against the corresponding online heuristic-based MILP solvers in numerical experiments.**

I. Introduction

Model predictive control (MPC) is a common technique to control hybrid systems in which both continuous and discrete dynamics interact [1]. For hybrid MPC with a linear performance measure, the optimization problem in question can be recast as a mixed-integer linear programming (MILP) problem that has to be solved at each sampling time subject to state and control signal constraints [2], which makes it important to have an efficient and reliable solver at hand. These problems can either be solved in real-time online, or be formulated as a multi-parametric MILP (mp-MILP) and solved offline parametrically for a range of states and reference signals [3]–[5], where the online effort is then reduced to retrieving the solution from a look-up table. The complexity of the pre-computed solution, however, grows exponentially with the dimensions of the problem, and for large problems, the required memory to store the solutions becomes potentially large, and computing an efficient data structure to implement the look-up table becomes computationally demanding. This makes the online approach a relevant option. For that approach to also be considered reliable for real-time applications, a priori guarantees for that the computational requirements do not exceed the hardware capabilities are desirable. Complexity certification frameworks for MIQPs and MILPs have been recently presented in [6]–[9].

A well-known method for solving MILPs is branch-and-bound (B&B) where a sequence of convex relaxations are solved in a binary search tree [10]. B&B is a so-called *complete* procedure, where it is guaranteed to find an optimal solution for every problem instance in finite time, if it exists. It is, however, a computationally expensive method and its worst-case complexity grows exponentially in the problem size. To enhance the performance of B&B, heuristic methods are often employed in high-performance solvers. Heuristics are *incomplete* methods, that is, they do not guarantee finding a feasible solution. However, they have shown to be highly relevant tools, e.g, in finding feasible solutions at early stages of the B&B procedure, reducing the workload of B&B by pruning more nodes [11], [12]. Furthermore, these are known to be important tools in commercial codes such as Gurobi and CPLEX. This paper, hence, aims to extend the complexity certification framework in [8] to exactly certify the computational complexity of B&B-based MILP solvers which also include heuristics. To that end, parametric versions of heuristics for mp-MILPs are required, which have not been considered in the literature before. Thus, the main contribution of this work is to present and analyze parametric versions of commonly used start heuristics known from high-performing MILP solvers, with the purpose of certifying the computational complexity of their online counterparts. In particular, three start heuristics, i.e., rounding (RENS), diving, and (objective) feasibility pump, are considered in this work. The proposed certification algorithms are then integrated into the B&B certification framework to compute the overall computational complexity of B&B schemes for MILP. The computational complexity can be measured in terms of, e.g., LP iterations, flops, or the required number of B&B nodes for the entire B&B scheme including the heuristics. This has a significant value for real-time applications such as hybrid MPC.

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

II. Problem Formulation

We consider an mp-MILP problem in the form,

$$\mathcal{P}_{\text{mpMILP}}(\theta): \quad \min_{x} \quad c^T x, \tag{1a}$$
$$\text{s.t.} \quad Ax \leqslant b + W\theta, \tag{1b}$$
$$x_k \in \{0,1\}, \ \forall k \in \mathcal{B} \tag{1c}$$

where $x = [x_c^T, x_b^T]^T \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ denotes the vector of $n = n_c + n_b$ continuous and binary decision variables, and $\theta \in \Theta_0 \subset \mathbb{R}^{n_\theta}$ is the parameter vector. The parameter set Θ_0 is assumed to be polyhedral. The constants that define the mp-MILP are given by $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $W \in \mathbb{R}^{m \times n_\theta}$. Moreover, $\mathcal{B}$ is the index set of binary variables. The problem (1) is non-convex and is known to be $\mathcal{NP}$-hard [10].

Definition 1: Let $\hat{F} \in \mathbb{R}^{n \times n_\theta}$ and $\hat{G} \in \mathbb{R}^n$. Then, the affine function $\hat{x}(\theta) = \hat{F}\theta + \hat{G}$, $\forall \theta \in \hat{\Theta} \subseteq \Theta_0$, where $\hat{\Theta}$ is a polyhedral region, is called

- infeasible for (1) if it does not satisfy (1b) and (1c),
- LP-feasible for (1) if it satisfies (1b), and $0 \leqslant \hat{x}_k \leqslant 1, \forall k \in \mathcal{B}$,
- mixed-integer for (1) if it satisfies (1c),
- integer-feasible for (1) if it satisfies (1b) and (1c).

Relaxing the integrality constraints (1c) to interval constraints results in the following convex relaxation in the form of an mp-LP problem,

$$\begin{aligned} & \min_{x} \quad c^T x, && \text{(2a)} \\ \mathcal{P}_{\text{mpLP}}(\theta): \quad & \text{s.t.} \quad Ax \leqslant b + W\theta, && \text{(2b)} \\ & \quad 0 \leqslant x_k \leqslant 1, \ \ \forall k \in \mathcal{B}, && \text{(2c)} \\ & \quad x_k = 0, \forall k \in \mathcal{B}_0, \ x_k = 1, \forall k \in \mathcal{B}_1 && \text{(2d)} \end{aligned}$$

where $\mathcal{B}_0, \mathcal{B}_1 \subseteq \mathcal{B}$ and $\mathcal{B}_0 \cap \mathcal{B}_1 = \emptyset$ are the index sets of binary variables fixed to 0 and 1, respectively.

To solve MILPs, a standard B&B method is used in this work [8]. In B&B, a sequence of relaxations is ordered and solved in a binary search tree, where each node in the tree corresponds to a convex relaxation [10]. A node containing a relaxation defined in (2) is here denoted $\eta(\theta) \triangleq (\mathcal{B}_0, \mathcal{B}_1, \theta)$, where $\mathcal{B}_0$ and $\mathcal{B}_1$ are defined in (2d) and are node dependent. An important concept when solving a relaxation is the *active set*, denoted $\mathcal{A}$, which is the index set of all active constraints at the optimal solution [13]. Note that a constraint is said to be *active* if it holds with equality.

For a specific parameter $\bar{\theta} \in \Theta_0$, the problem (1) simplifies to the following (non-parametric) MILP problem,

$$\begin{aligned} & \min_{x} \quad c^T x, && \text{(3a)} \\ \mathcal{P}_{\text{MILP}}(\bar{\theta}): \quad & \text{s.t.} \quad Ax \leqslant \bar{b}, && \text{(3b)} \\ & \quad x_k \in \{0,1\}, \ \ \forall k \in \mathcal{B} && \text{(3c)} \end{aligned}$$

where $\bar{b} = b + W\bar{\theta}$. The decision variables and other problem matrices are defined similarly as in (1). The LP relaxation of (3) is obtained by relaxing the integrality constraints (3c) into (2c) and (2d). A node containing a relaxation of (3) in the B&B tree is here denoted $\eta = (\mathcal{B}_0, \mathcal{B}_1)$ for a specific $\bar{\theta}$.

III. Heuristics for MILP

The heuristics considered in this work are procedures that try to compute feasible solutions computationally inexpensively. There is a wide variety of heuristics proposed in the literature for MILPs [11], [12]. The focus in this paper is on *start heuristic* methods, where the goal is to find a feasible solution, e.g., after solving the first relaxation (root node). Finding feasible solutions early can result in a useful upper bound that can help to prune some nodes in the B&B search tree, reducing the tree size and the overall effort. Three commonly-used start heuristics, i.e., *relaxation enforced neighborhood search (RENS)*, *diving*, and *feasibility pump (FP)* methods are considered here. Algorithms 1–3 present these methods, respectively. As inputs, they take a problem formulation, an LP-feasible solution $\underline{x}$ and the corresponding active set $\mathcal{A}$, and some extra information depending on the heuristic method. They then output a possibly found integer-feasible solution $\hat{x}$ and the accumulated complexity measure κ^h, e.g., the total number of LP iterations, flops, or nodes. We review these methods for the MILP problem (3) for a specific $\bar{\theta} \in \bar{\Theta}$ in the following subsections, where $\bar{\Theta}$ is a polyhedral region, and we refer the reader to [11] for extensive details.

A. RENS for MILP

After obtaining an optimal (LP-feasible) solution $\underline{x}$ of a relaxation, e.g., in the root node, the rounding methods, such as *simple rounding* and *RENS*, try to round relaxed binary variables up or down, such that the resulting variable $\hat{x}$ becomes integer feasible. This paper considers RENS, but it can be extended to other rounding methods as well.

The idea of RENS, presented in [11], is to solve a sub-MILP problem $\mathcal{P}_{\text{RENS}}(\underline{x})$ that is created from the original MILP (3) plus the following additional constraints,

$$x_k = \underline{x}_k, \qquad \forall k \in \mathcal{B} \setminus \bar{\mathcal{B}} \tag{4}$$

where $\bar{\mathcal{B}}$ is the index set of integrality constraints (2c) that are not active (see Step 1 of Algorithm 1), that is, the *candidate list* of relaxed binary variables. The additional constraints (4) in $\mathcal{P}_{\text{RENS}}(\underline{x})$ fixes all relaxed binary variables x_k, $\forall k \in \mathcal{B}$, for which the LP-feasible solution $\underline{x}_k$ is binary. Algorithm 1 outlines this heuristic method. The SOLVEMILP procedure at Step 6 is the B&B-based MILP solver that solves the sub-MILP and returns the possibly found integer-feasible solution $\hat{x}$ with the required complexity measure κ^h.

As the sub-MILP $\mathcal{P}_{\text{RENS}}(\underline{x})$ might be expensive to solve, some limits, e.g., on the total number of nodes, can be added to the the B&B-based MILP solver (at Step 6). Moreover, this heuristic is utilized if a sufficiently small number of binary variables remain to be fixed, i.e., when $|\bar{\mathcal{B}}| \leq r|\mathcal{B}|$, where $|.|$ denotes the cardinality of a set, and $r \in (0,1)$ is a user-defined parameter [11]. In this way, the risk that the sub-MILP problem is harder than the original MILP (3) can be reduced.

Algorithm 1 RENS heuristic for online B&B

Input: $\mathcal{P}_{\text{MILP}}$, $\underline{x}$, $\mathcal{A}$
Output: $\hat{x}, \kappa^h$
1: $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
2: **if** $|\bar{\mathcal{B}}| > r|\mathcal{B}|$ **then**
3: **return** NULL, 0
4: **else**
5: Formulate $\mathcal{P}_{\text{RENS}}(\underline{x})$ from the original $\mathcal{P}_{\text{MILP}}$ (3) subject to the additional constraints (4)
6: $\hat{x}, \kappa^h \leftarrow$ SOLVEMILP($\mathcal{P}_{\text{RENS}}(\underline{x})$)
7: **return** $\hat{x}, \kappa^h$

B. Diving for MILP

Starting from an optimal (LP-feasible) solution $\underline{x}$ of the current relaxation, e.g., in the root node, the diving method iteratively fixes relaxed binary variables to promising values and solves the obtained relaxations. The procedure is terminated once infeasibility is detected or an integer-feasible solution is found. Thereby, this method resembles a depth-first node selection strategy of a promising root-leaf path by "diving" in the B&B tree until a hopefully good feasible solution is found. The general principle of a diving heuristic is shown in Algorithm 2. Here, l denotes the loop iteration and SOLVELP($\bar{\eta}$) is a procedure for solving LPs,

e.g., the dual simplex method in [13]. This procedure returns a possibly found LP-feasible solution $\underline{x}$, the complexity measure for solving the LP problem κ (e.g., the LP iteration), and the corresponding active set $\mathcal{A}$. To control the time and effort the heuristic gets, some limits, e.g., on the accumulated complexity measure $\kappa^h_{\max}$ and/or loop iterations $l_{\max}$, have been added to the algorithm. If the procedure succeeds in finding an integer-feasible solution, $\hat{x} \neq$ NULL as well as κ^h result from Algorithm 2.

Algorithm 2 DIVING heuristic for online B&B

Input: $\eta = (\mathcal{B}_0, \mathcal{B}_1), \underline{x}, \mathcal{A}$
Output: $\hat{x}, \kappa^h$
$\kappa^h \leftarrow 0$, $l \leftarrow 0$, $\hat{x} \leftarrow$ NULL
while ($\kappa^h < \kappa^h_{\max}$ and $l < l_{\max}$) **do**
  $l \leftarrow l + 1$
  $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
  if $\bar{\mathcal{B}} = \emptyset$ **then** $\hat{x} \leftarrow \underline{x}$ **break**
  Select $k : k \in \bar{\mathcal{B}}$
  $\bar{\eta} \leftarrow (\mathcal{B}_0 \cup \{k\}, \mathcal{B}_1)$ or $\bar{\eta} \leftarrow (\mathcal{B}_0, \mathcal{B}_1 \cup \{k\})$
  $\underline{x}, \kappa, \mathcal{A} \leftarrow$ SOLVELP$(\bar{\eta})$
  $\kappa^h \leftarrow \kappa^h + \kappa$
  if $\bar{\eta}$ is infeasible **then break**
  if $\underline{x}$ is integer feasible **then** $\hat{x} \leftarrow \underline{x}$, **break**
return $\hat{x}, \kappa^h$

C. Feasibility pump for MILP

The idea of the FP method is to construct two sequences of points that hopefully converge to a feasible solution [14]. One sequence contains LP-feasible solutions that are not necessarily integer-feasible, and the other contains mixed-integer solutions that are not necessarily LP-feasible. Starting from an optimal solution $\underline{x}$ of a relaxation, e.g., the root node, these points are generated by iteratively rounding the LP-solution $\underline{x}$ to a mixed-integer solution $\hat{x}$ (rounding step), and then finding an LP-feasible solution closest to $\hat{x}$ in terms of the Manhattan distance (pumping step), which is then used as the new $\underline{x}$ in the next iteration. The distance of two vectors is defined as follows.

Definition 2: The L_1-distance of two vectors $x, \hat{x} \in \mathbb{R}^n$ with respect to the set $\mathcal{B}$ is defined as,

$$\Delta(x, \hat{x}) = \sum_{k \in \mathcal{B}} |x_k - \hat{x}_k| \tag{5}$$

For MILPs in the form of (3), where integer variables are modeled using binaries, the nonlinear function (5) can be reformulated as follows [11],

$$\Delta(x, \hat{x}) = \sum_{k \in \mathcal{B}: \hat{x}_k = 0} x_k + \sum_{k \in \mathcal{B}: \hat{x}_k = 1} 1 - x_k \tag{6}$$

To find the LP-feasible solution $\underline{x}$ closest to $\hat{x}$, the following problem is then solved at the pumping step of FP,

$$\begin{aligned} & \min_{x} && \Delta(x, \hat{x}) && \text{(7a)} \\ \mathcal{P}_{\text{FP}}(\hat{x}): \ & \text{s.t.} && Ax \leqslant \bar{b}, && \text{(7b)} \\ & && 0 \leqslant x_k \leqslant 1, \ \forall k \in \mathcal{B} && \text{(7c)} \end{aligned}$$

A modified version of FP, the so-called objective feasibility pump (OFP), has been introduced in [15] to also take the original objective function into account. In OFP, the objective function of (7) is replaced with a convex combination of (6) and the original objective function as follows,

$$\Delta^\alpha(x, \hat{x}) = (1 - \alpha)\Delta(x, \hat{x}) + \alpha \frac{\sqrt{|\hat{c}|}}{\|c\|} c^T x \tag{8}$$

where $\alpha \in [0, 1]$, $c \in \mathbb{R}^n \setminus \{0\}$, and $\hat{c}$ is the objective function vector of (6). At each iteration l, α_l is decreased as $\alpha_l = \phi \alpha_{l-1}$, where $\phi \in [0, 1)$ is a fixed factor and $\alpha_0 \in [0, 1]$. To find a solution $\underline{x}$ closest to $\hat{x}$ in the OFP, $\mathcal{P}^\alpha_{\text{FP}}(\hat{x})$ created from (7), where (8) is substituted for (7a), is instead solved.

Note that, if $\alpha_0 = 0$, the original FP is achieved. Algorithm 3 shows the OFP heuristic. The ROUND procedure here simply rounds the relaxed binary variables $\hat{x}_k$, $\forall k \in \bar{\mathcal{B}}$, to the closest binary values. The possibly found integer-feasible solution $\hat{x} \neq$ NULL and the accumulated complexity measure κ_h are outputted from this algorithm.

Algorithm 3 OBJECTIVEFEASIBILITYPUMP heuristic for online B&B

Input: $\eta = (\mathcal{B}_0, \mathcal{B}_1), \underline{x}, \mathcal{A}, \phi, \alpha_0$
Output: $\hat{x}, \kappa^h$
$\kappa^h \leftarrow 0$, $l \leftarrow 0$, $\hat{x} \leftarrow$ NULL
while ($\kappa^h < \kappa^h_{\max}$ and $l < l_{\max}$) **do**
  $l \leftarrow l + 1$
  $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
  $\hat{x} \leftarrow$ ROUND$(\underline{x}, \bar{\mathcal{B}})$
  if $\hat{x}$ is integer feasible **then break**
  $\alpha_l = \phi\, \alpha_{l-1}$
  $\underline{x}, \kappa, \mathcal{A} \leftarrow$ SOLVELP$(\mathcal{P}^{\alpha_l}_{\text{OFP}}(\hat{x}))$
  $\kappa^h \leftarrow \kappa^h + \kappa$
  if $\underline{x}$ is integer feasible **then** $\hat{x} \leftarrow \underline{x}$, **break**
return $\hat{x}, \kappa^h$

procedure ROUND$(\underline{x}, \bar{\mathcal{B}})$
  $\hat{x} \leftarrow \underline{x}$
  while $\bar{\mathcal{B}} \neq \emptyset$ **do**
    Select $k : k \in \bar{\mathcal{B}}$, $\bar{\mathcal{B}} \leftarrow \bar{\mathcal{B}} \setminus \{k\}$
    if $\hat{x}_k \leq 0.5$ **then**
      $\hat{x}_k = 0$ ▷ round down
    else
      $\hat{x}_k = 1$ ▷ round up
  return $\hat{x}$

IV. COMPLEXITY CERTIFICATION OF START-HEURISTIC-BASED B&B METHODS

In order to extend the certification framework presented in [8] to include the heuristics, parametric versions of these methods are required. This extension is the main contribution of this work. This section introduces the start heuristics for mp-MILPs, with the main idea of computing the complexity measure $\kappa(\theta)$ as a function of θ.

Algorithms 4–6 extend the heuristic methods reviewed in Section III for mp-MILPs, respectively. These algorithms iteratively divide and explore the parameter space based on the polyhedral partition from, e.g., certifying MILPs or LPs.

As inputs, they take a problem formulation, an LP-feasible solution $\underline{x}(\theta)$ and the corresponding active set $\mathcal{A}$, a single polyhedral region $\bar{\Theta}$, and some extra information depending on the heuristic method. Note that $\mathcal{A}$ associated to the solution $\underline{x}(\theta)$ is fixed inside $\bar{\Theta}$. They then output a partition $\{\Theta^j\}_j$ of $\bar{\Theta}$ in which each region Θ^j is associated with a possibly-found explicit integer-feasible solution $\hat{x}^j(\theta)$ and the accumulated complexity measure κ^{h_j}. These algorithms are discussed in more detail in Sections IV-A–IV-C, and are integrated into B&B in Section IV-D.

A. RENS for mp-MILP

Algorithm 4 certifies the complexity of the RENS Algorithm 1 for all parameters taken from a polyhedral region $\bar{\Theta}$. Similar to Algorithm 1, if $|\bar{\mathcal{B}}| \leq r|\mathcal{B}|$, then the parametric sub-mpMILP $\mathcal{P}_{\text{RENS}}(\underline{x}(\theta))$ is formed and certified by the SOLVEMILPCERT procedure $\forall\theta \in \bar{\Theta}$. The SOLVEMILPCERT is the complexity certification framework for MILPs, e.g., the one presented in [8], that takes an mp-MILP problem and a polyhedral region $\bar{\Theta}$ and returns a polyhedral partition $\{\Theta^j\}_j$ along with the explicit (integer-feasible) solution $\hat{x}^j(\theta)$ and the accumulated complexity measure κ^{h_j}. A list of tuples containing the terminated regions. i.e., $\mathcal{F}_h = \{(\Theta^j, \hat{x}^j(\theta), \kappa^{h_j})\}_j$, is outputted by Algorithm 4, where $\hat{x}^j(\theta) \neq$ NULL if an integer-feasible solution has been found in Θ^j.

Algorithm 4 RENS heuristic for parametric B&B

Input: $\mathcal{P}_{\text{mpMILP}}$, $\underline{x}(\theta), \mathcal{A}, \bar{\Theta}$
Output: $\mathcal{F}_h = \{(\Theta^j, \hat{x}^j(\theta), \kappa^{h_j})\}_j$
1: $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
2: **if** $|\bar{\mathcal{B}}| > r|\mathcal{B}|$ **then**
3: **return** $\mathcal{F}_h = \{(\bar{\Theta}, \text{NULL}, 0)\}$
4: **else**
5: Formulate $\mathcal{P}_{\text{RENS}}(\underline{x}(\theta))$ from the original $\mathcal{P}_{\text{mpMILP}}$ (1) subject to the additional constraints (4)
6: $\{(\Theta^j, \hat{x}^j, \kappa^{h_j})\}_{j=1}^{N} \leftarrow$ SOLVEMILPCERT($\mathcal{P}_{\text{RENS}}(\underline{x}(\theta))$, $\bar{\Theta}$)
7: **return** $\mathcal{F}_h = \{(\Theta^j, \hat{x}^j, \kappa^{h_j})\}_j$

B. Diving for mp-MILP

Algorithm 5 illustrates the complexity certification of the diving heuristic method, where it applies Algorithms 2 for all parameters. Two lists of tuples are used in this algorithm to store information. One is the list $\mathcal{S}_h$ to hold regions in the parameter space that the analysis has not yet terminated. Each tuple in $\mathcal{S}_h$ consists of a polyhedral region Θ, an (LP-feasible) explicit solution $\underline{x}(\theta)$, the accumulated complexity measure κ^h, the active set $\mathcal{A}$, and the loop iteration l, for $\theta \in \Theta$. The other list is $\mathcal{F}_h$ holding regions in the parameter space where the process has terminated. It contains tuples with the information $(\Theta, \hat{x}(\theta), \kappa^h)$ for every terminated region.

At each iteration of Algorithm 5, a popped region Θ (at Step 3) is terminated and added to $\mathcal{F}_h$ (at Step 21) if the limits on the accumulated complexity measure $\kappa^h_{\max}$ and/or loop $l_{\max}$ have been reached. Otherwise, the candidate list $\bar{\mathcal{B}}$ of relaxed binary variables is updated (Step 6) and an index $k \in \bar{\mathcal{B}}$ is selected and fixed to generate the next node (Step 11) if $\bar{\mathcal{B}} \neq \emptyset$. The selection rule for $k \in \bar{\mathcal{B}}$ at Step 10 and the selection of the 0- or 1-branch at Step 11 are here chosen identically to Steps 6 and 7 of the online Algorithm 2, respectively. The SOLVELPCERT then solves and certifies the relaxation in this node over Θ at Step 12. Note that SOLVELPCERT can be chosen as any complexity certification framework for LPs, e.g., [16], where it takes an mp-LP problem and a polyhedral region Θ and returns a polyhedral partition $\{\Theta^j\}_j$ of Θ each with the corresponding explicit (LP-feasible) solution $\underline{x}^j(\theta)$, the complexity measure κ^j, and the active set $\mathcal{A}^j$.

Algorithm 5 DIVING heuristic for parametric B&B

Input: $\eta(\theta) = (\mathcal{B}_0, \mathcal{B}_1, \theta), \underline{x}(\theta), \mathcal{A}, \bar{\Theta}$
Output: $\mathcal{F}_h = \{(\Theta^j, \hat{x}^j(\theta), \kappa^{h_j})\}_j$
1: Push $\{(\bar{\Theta}, \underline{x}(\theta), 0, \mathcal{A}, 0)\}$ to $\mathcal{S}_h$
2: **while** $\mathcal{S}_h \neq \emptyset$ **do**
3: Pop $(\Theta, \underline{x}(\theta), \kappa^h, \mathcal{A}, l)$ from $\mathcal{S}_h$
4: **if** ($\kappa^h < \kappa^h_{\max}$ and $l < l_{\max}$) **then**
5: $l \leftarrow l + 1$
6: $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
7: **if** $\bar{\mathcal{B}} = \emptyset$ **then**
8: Push $(\Theta, \underline{x}(\theta), \kappa^h)$ to $\mathcal{F}_h$
9: **else**
10: Select $k : k \in \bar{\mathcal{B}}$
11: $\bar{\eta}(\theta) \leftarrow (\mathcal{B}_0 \cup \{k\}, \mathcal{B}_1, \theta)$ or $\bar{\eta}(\theta) \leftarrow (\mathcal{B}_0, \mathcal{B}_1 \cup \{k\}, \theta)$
12: $\{(\Theta^j, \underline{x}^j, \kappa^j, \mathcal{A}^j)\}_{j=1}^{N_d} \leftarrow$ SOLVELPCERT($\bar{\eta}(\theta), \Theta$)
13: **for** $j \in \{1, \ldots, N_d\}$ **do**
14: **if** $\bar{\eta}(\theta)$ is infeasible $\forall\theta \in \Theta^j$ **then**
15: Push $(\Theta^j, \text{NULL}, \kappa^h + \kappa^j)$ to $\mathcal{F}_h$
16: **else if** $\underline{x}^j(\theta)$ is integer feasible $\forall\theta \in \Theta^j$ **then**
17: Push $(\Theta^j, \underline{x}^j(\theta), \kappa^h + \kappa^j)$ to $\mathcal{F}_h$
18: **else**
19: Push $(\Theta^j, \underline{x}^j(\theta), \kappa^h + \kappa^j, \mathcal{A}^j, l)$ to $\mathcal{S}_h$
20: **else**
21: Push $(\Theta, \text{NULL}, \kappa^h)$ to $\mathcal{F}_h$
22: **return** $\mathcal{F}_h$

After processing this node, for each j over a for-loop at Step 13, the new tuple $(\Theta^j, \underline{x}^j(\theta), \kappa^h + \kappa^j)$ is added to $\mathcal{F}_h$ if either infeasibility is detected (Step 15), or the solution turned out to be integer feasible (Step 17) in Θ^j. Otherwise, the new tuple (containing the extra information) is appended to $\mathcal{S}_h$ at Step 19 to be processed further in the loop, completing an iteration of Algorithm 5.

C. Feasibility pump for mp-MILP

Algorithm 6 presents the proposed complexity certification algorithm for the OFP heuristic shown in Algorithm 3 for all parameters taken from a polyhedral region $\bar{\Theta}$. Two lists of tuples $\mathcal{S}_h$ and $\mathcal{F}_h$ are again used to hold currently processed and terminated regions, respectively, in this algorithm, where each tuple in $\mathcal{S}_h$ also holds the parameter α_l (see (8)).

At each iteration of Algorithm 6, a popped region Θ (at Step 3) is terminated and added to $\mathcal{F}_h$ (at Step 20) if the limits on $\kappa^h_{\max}$ and/or $l_{\max}$ have been reached. Otherwise, the candidate list $\bar{\mathcal{B}}$ is updated (Step 6) and the parametric

LP-feasible solution $\underline{x}(\theta)$ is rounded by the ROUNDPARAM function (Step 7).

Algorithm 6 OBJECTIVEFEASIBILITYPUMP heuristic for parametric B&B

Input: $\eta(\theta) = (\mathcal{B}_0, \mathcal{B}_1, \theta), \underline{x}(\theta), \mathcal{A}, \phi, \alpha_0, \bar{\Theta}$
Output: $\mathcal{F}_h = \{(\Theta^j, \hat{x}^j(\theta), \kappa^{h_j})\}_j$

1: Push $\{(\bar{\Theta}, \underline{x}(\theta), 0, \mathcal{A}, 0, \alpha_0)\}$ to $\mathcal{S}_h$
2: **while** $\mathcal{S}_h \neq \emptyset$ **do**
3: Pop $(\Theta, \underline{x}(\theta), \kappa^h, \mathcal{A}, l, \alpha_l)$ from $\mathcal{S}_h$
4: **if** ($\kappa^h < \kappa^h_{\max}$ and $l < l_{\max}$) **then**
5: $l \leftarrow l + 1$
6: $\bar{\mathcal{B}} \leftarrow \{k \in \mathcal{B} \mid k \notin \mathcal{A}\}$
7: $\{\Theta^i, \hat{x}^i(\theta)\}_{i=1}^{N_r} \leftarrow$ ROUNDPARAM$(\underline{x}(\theta), \bar{\mathcal{B}}, \Theta)$
8: **for** $i \in 1, \ldots, N_r$ **do**
9: **if** $\hat{x}^i(\theta)$ is integer feasible $\forall \theta \in \Theta^i$ **then**
10: Push $(\Theta^i, \hat{x}^i(\theta), \kappa^h)$ to $\mathcal{F}_h$
11: **else**
12: $\alpha_l = \phi\, \alpha_{l-1}$
13: $\{(\Theta^j, \underline{x}^j(\theta), \kappa^j, \mathcal{A}^j)\}_{j=1}^{N_p} \leftarrow$ SOLVELPCERT$(\mathcal{P}^{\alpha_l}_{\text{FP}}(\hat{x}^i(\theta)), \Theta^i)$
14: **for** $j \in \{1, \ldots, N_p\}$ **do**
15: **if** $\underline{x}^j(\theta)$ is integer feasible $\forall \theta \in \Theta^j$ **then**
16: Push $(\Theta^j, \underline{x}^j(\theta), \kappa^h + \kappa^j)$ to $\mathcal{F}_h$
17: **else**
18: Push $(\Theta^j, \underline{x}^j(\theta), \kappa^h + \kappa^j, \mathcal{A}^j, l, \alpha_l)$ to $\mathcal{S}_h$
19: **else**
20: Push $(\Theta, \text{NULL}, \kappa^h)$ to $\mathcal{F}_h$
21: **return** $\mathcal{F}_h$
22:
23: **procedure** ROUNDPARAM$(\underline{x}(\theta), \bar{\mathcal{B}}, \tilde{\Theta})$
24: Push $\{(\tilde{\Theta}, \underline{x}(\theta), \bar{\mathcal{B}})\}$ to $\mathcal{S}_r$
25: **while** $\mathcal{S}_r \neq \emptyset$ **do**
26: Pop $(\Theta, \hat{x}(\theta), \bar{\mathcal{B}})$ from $\mathcal{S}_r$ ▷ $\hat{x}(\theta) = \hat{F}\theta + \hat{G}$
27: **if** $\bar{\mathcal{B}} \neq \emptyset$ **then**
28: Select $k : k \in \bar{\mathcal{B}}$, $\bar{\mathcal{B}} \leftarrow \bar{\mathcal{B}} \setminus \{k\}$
29: $\Theta_0 \leftarrow \{\theta \in \Theta | \hat{F}_k\theta + \hat{G}_k \leq 0.5\}$, ▷ $\hat{x}_k$ be round down
$\Theta_1 \leftarrow \{\theta \in \Theta | \hat{F}_k\theta + \hat{G}_k > 0.5\}$ ▷ $\hat{x}_k$ be round up
30: **if** $\Theta_1 = \emptyset$ **then** ▷ no further partitioning
31: $\hat{x}_k \leftarrow 0$, push $(\Theta, \hat{x}(\theta), \bar{\mathcal{B}})$ to $\mathcal{S}_r$
32: **else if** $\Theta_0 = \emptyset$ **then** ▷ no further partitioning
33: $\hat{x}_k \leftarrow 1$, push $(\Theta, \hat{x}(\theta), \bar{\mathcal{B}})$ to $\mathcal{S}_r$
34: **else** ▷ partition Θ to Θ_0 and Θ_1
35: $\hat{x}_k \leftarrow 0$, push $(\Theta_0, \hat{x}(\theta), \bar{\mathcal{B}})$ to $\mathcal{S}_r$
36: $\hat{x}_k \leftarrow 1$, push $(\Theta_1, \hat{x}(\theta), \bar{\mathcal{B}})$ to $\mathcal{S}_r$
37: **else**
38: Push $(\Theta, \hat{x}(\theta))$ to $\mathcal{F}_r$
39: **return** $\mathcal{F}_r$

In the ROUNDPARAM function, again, two list of tuples $\mathcal{S}_r$ and $\mathcal{F}_r$ are used to hold currently processed and terminated regions, respectively. Over a while loop at Step 25, for a popped region Θ, as long as $\bar{\mathcal{B}} \neq 0$, an index k of a relaxed binary variable is selected to be rounded and consequently removed from $\bar{\mathcal{B}}$ (Step 28). Note that the selection of k has to be done identically to the online Algorithm 3 at Step 16. Now, based on the variable $\hat{x}_k(\theta) = \hat{F}_k\theta + \hat{G}_k$ that should be rounded down or up, Θ is potentially split into two regions Θ_0 and Θ_1, respectively, at Step 29, where $\hat{x}_k(\theta)$ is fixed to 0, $\forall \theta \in \Theta_0$ (Step 31 or 35), and 1, $\forall \theta \in \Theta_1$ (Step 33 or 36). A partition $\{\Theta^i\}_{i=1}^{N_r}$ containing the rounded solutions $\hat{x}^i(\theta)$ is outputted from this procedure.

Returning to the main loop (Step 7), for each $i \in \mathbb{N}_{1:N_r}$, if the rounded solution $\hat{x}^i(\theta)$ is also integer feasible $\forall \theta \in \Theta^i$, this region is labeled as terminated and added to $\mathcal{F}_h$ at Step 10. Otherwise, the pumping step is performed in Θ^i by solving and certifying the following mp-LP problem by the SOLVELPCERT procedure,

$$\min_x \quad \Delta^\alpha(x(\theta), \hat{x}^i(\theta)) \tag{9a}$$

$$\mathcal{P}^\alpha_{\text{OFP}}(\hat{x}^i(\theta)) : \text{ s.t. } \quad Ax \leqslant b + W\theta, \tag{9b}$$

$$0 \leqslant x_k \leqslant 1, \ \ \forall k \in \mathcal{B} \tag{9c}$$

where $\Delta^\alpha(x(\theta), \hat{x}^i(\theta))$ given in (8) is an affine function of θ in Θ^i. This potentially further partitions Θ^i to polyhedral regions $\{\Theta^j\}_j$ at Step 13. Finally, for all $j \in \mathbb{N}_{1:N_p}$, if a resulting LP-feasible solution $\underline{x}^j(\theta)$ is also integer feasible, the OFP procedure terminates for Θ^j and adds it to $\mathcal{F}_h$ at Step 16. Otherwise, the new tuple is appended to $\mathcal{S}_h$ at Step 18 to be processed further, concluding an iteration of Algorithm 6.

D. Integration of the start heuristics into the B&B certification framework

We are now ready to integrate the parametric versions of the start heuristics into the complexity certification framework of B&B-based MILP solvers presented in [8]. In the following, we first show how the online Algorithms 1–3 are incorporated into the online B&B algorithm, and then integrate Algorithms 4–6 into the B&B certification framework.

Consider the online B&B Algorithm 1 in [8], and let STARTHEURIS denote any of the online start heuristics reviewed in Section III. Then, after processing a relaxation η (in the root node in this case) and obtaining a lower bound solution $\underline{x}$ with the corresponding active set $\mathcal{A}$ at Step 6 by the LP solver, Step 8 of Algorithm 1 in [8] (Step 8* below) that calls the EVALCUTCOND procedure (to evaluate the cut conditions) is replaced with the following extra steps,

if `root node` **then** ▷ call heuristic
$\hat{x}, \kappa^h \leftarrow$ STARTHEURIS $(\eta, \underline{x}, \mathcal{A})$
$\kappa^*_{\text{tot}} \leftarrow \kappa^*_{\text{tot}} + \kappa^h$
if $\hat{x} \neq$ NULL **then**
$\bar{x} \leftarrow \hat{x}, \ \ \bar{J} \leftarrow c^T\bar{x}$
$\mathcal{T}, \bar{x}, \bar{J} \leftarrow$ EVALCUTCOND$(\eta, \underline{x}, \underline{J}, \underline{\lambda}, \mathcal{A}, \bar{x}, \bar{J}, \mathcal{T})$ ▷ Step 8*

where if a node is the root node, a start heuristic is called and the best-known integer-feasible solution $\bar{x}$ and the upper bound $\bar{J}$ are updated if an integer-feasible solution $\hat{x} \neq$ NULL has been found by the heuristic. Also, κ^h is added to the accumulated complexity measure of B&B (κ^*_{tot}) to achieve the overall B&B complexity measure. The last step is Step 8 of the online B&B algorithm that is executed regardless of whether a node is the root node or not.

Now consider the B&B certification Algorithm 4 in [8]. Let STARTHEURISCERT denote any of the parametric start

heuristics introduced in Sections IV-A–IV-C. After processing a relaxation $\eta(\theta)$ (the root node in this case) at a region Θ, a partition $\{\Theta^j\}_{j=1}^N$ is obtained by the mp-LP certification procedure at Step 11, where each Θ^j is associated with the lower bound solution $\underline{x}^j(\theta)$ and the active set $\mathcal{A}^j$. Then for each individual region Θ^j (associated with the accumulated complexity measure κ_{tot}^j), Step 14 of Algorithm 4 in [8] (Step 14* below) that calls the EVALCUTCONDCERT procedure (to parametrically evaluate the cut conditions in Θ^j) is replaced with the following extra steps

---

if `root node` **then** ▷ call heuristic
$\{(\Theta^i, \hat{x}^i, \kappa^{h_i})\}_{i=1}^{N_h} \leftarrow$ STARTHEURISCERT$(\eta(\theta), \underline{x}^j(\theta), \mathcal{A}^j, \Theta^j)$
for $i = \{1, \dots, N_h\}$ **do**
$\kappa_{\text{tot}}^i \leftarrow \kappa_{\text{tot}}^j + \kappa^{h_i}$
$\bar{J}^i(\theta) \leftarrow \bar{J}(\theta)$
if $\hat{x}^i(\theta) \neq$ NULL **then**
$\bar{x}^i(\theta) \leftarrow \hat{x}^i(\theta), \quad \bar{J}^i(\theta) \leftarrow c^T \bar{x}^i(\theta)$
$\mathcal{S} \leftarrow$ EVALCUTCONDCERT$(\eta, \underline{J}^j, \underline{\lambda}^j, \mathcal{A}^j, \bar{J}^i, \mathcal{T}, \kappa_{tot}^i, \Theta^i, \mathcal{S})$
else
$\mathcal{S} \leftarrow$ EVALCUTCONDCERT$(\eta, \underline{J}^j, \underline{\lambda}^j, \mathcal{A}^j, \bar{J}, \mathcal{T}, \kappa_{tot}^j, \Theta^j, \mathcal{S})$
▷ Step 14*

---

where if a node is the root node, then Θ^j is potentially further partitioned to $\{\Theta^i\}_i$ by the start heuristic. For each $i \in \mathbb{N}_{1:N_h}$ then, the accumulated complexity measure for B&B κ_{tot}^i, $\forall \theta \in \Theta^i$, is updated. Next, the best-known integer-feasible solution $\bar{x}^i(\theta)$ and the upper bound $\bar{J}^i(\theta)$ are updated if an integer-feasible solution has been found in Θ^i. Finally, the EVALCUTCONDCERT is applied for Θ^i with the possibly new upper bound $\bar{J}^i(\theta)$. Otherwise, if the node is not the root node, then the heuristic is omitted and the last step which is Step 14 of Algorithm 4 in [8] will be executed $\forall \theta \in \Theta^j$.

Remark 1: A general piecewise affine function $\tau(\theta) : \Theta_0 \to \mathbb{R}$ for *triggering* a start heuristic can be applied in the online and certification algorithms to possibly also trigger a start heuristic at other stages of the B&B process, e.g., after reaching to some specific levels in the B&B tree. Note that, however, if a heuristic is called after an integer-feasible solution has been found (i.e., not at the root node) where $\bar{J} \neq \infty$, then $\bar{x}(\theta)$ and $\bar{J}(\theta)$ should only be updated for these parts of the parameter set where $\hat{x}(\theta)$ provides a better solution, cf. the dominance cut condition.

V. PROPERTIES OF THE CERTIFICATION ALGORITHMS

We now study the properties of the proposed complexity certification algorithms. Specifically, we show that the result of the certification Algorithms 4–6 are exact.

A. Properties of the RENS certification algorithm

This subsection shows that the parametric RENS method in Algorithm 4 coincides point-wise with the online Algorithm 1. In the following, the properties of the certification algorithm for RENS are investigated.

Lemma 1: The complexity measure $\kappa^h(\theta) : \bar{\Theta} \to \mathbb{N}$ returned by Algorithm 4 is polyhedral piecewise constant (PPWC), that is, $\kappa^{h_j}(\theta) = \kappa^{h_j}$, $\forall \theta \in \Theta^j \subseteq \bar{\Theta}$, $\forall j$.

Proof: The one and only step that partitions an inputted region $\bar{\Theta}$ in Algorithm 4 is performed at Step 6. As the certification algorithm presented in [8] is here used to certify the computational complexity of the sub-mpMILP $\mathcal{P}_{\text{RENS}}(\underline{x}(\theta))$, the desired result follows from Lemma 4 in [8] with replacing Θ_0 with $\bar{\Theta}$ here. ■

Theorem 1: Let $\kappa^{h*}(\theta)$ be the accumulated complexity measure returned by Algorithm 1 for a given $\theta \in \bar{\Theta}$. Moreover, let $\{\Theta^j, \hat{x}^j, \kappa^{h_j}\}_j$ be the collection of tuples in $\mathcal{F}_h$ returned by the Algorithm 4. Then $\kappa^{h_j} = \kappa^{h*}(\theta)$, $\forall \theta \in \Theta^j$, $\forall j$.

Proof: Consider Algorithms 1 and 4 for an arbitrary fixed $\theta \in \bar{\Theta}$. The one-to-one correspondence of Steps 1–5 of these algorithms follows from visual comparison. At Step 6, the sub-MILP is solved online for the fixed $\theta \in \bar{\Theta}$ in Algorithm 1, whereas it is certified $\forall \theta \in \bar{\Theta}$ in Algorithm 4. As the online B&B Algorithm 1 and the certification Algorithm 4 in [8] are here used as the SOLVEMILP and SOLVEMILPCERT procedures in Algorithms 1 and 4, respectively, the desired result follows from Corollary 1 in [8]. ■

B. Properties of the diving certification algorithm

This subsection shows that the parametric diving method in Algorithm 5 coincides point-wise with the online Algorithm 2. Since diving explores a possible root-leaf path of the B&B search tree with the depth-first node selection strategy, the online Algorithm 2 and the complexity certification Algorithm 5 can be seen as (simplified) special cases of the online B&B Algorithm 1 and the certification Algorithm 4 in [8], respectively. More precisely, the set of nodes $\mathcal{T}$ in the diving Algorithms 2 and 5 always includes just one node and, additionally, testing the dominance cut condition (i.e., Steps 12–13 of Algorithm 1, and Steps 18–22 of Algorithm 4 in [8]) are excluded. Consequently, the results in [8] can be immediately applied here, which are summarized in what follows while replacing Θ_0 with $\bar{\Theta}$.

Lemma 2: Assume Assumption 1 in [8] holds. The complexity measure $\kappa^h(\theta) : \bar{\Theta} \to \mathbb{N}$ returned by Algorithm 5 is PPWC, that is $\kappa^{h_j}(\theta) = \kappa^{h_j}$, $\forall \theta \in \Theta^j \subseteq \bar{\Theta}$, $\forall j$.

Proof: See [8], Lemma 4. ■

Theorem 2: Assume Assumption 1 in [8] holds. Let $\kappa^{h*}(\theta)$ be the accumulated complexity measure returned by Algorithm 2 for a given $\theta \in \bar{\Theta}$. Moreover, let $\{\Theta^j, \hat{x}^j, \kappa^{h_j}\}_j$ be the collection of tuples in $\mathcal{F}_h$ returned by Algorithm 5. Then $\kappa^{h_j} = \kappa^{h*}(\theta)$, $\forall \theta \in \Theta^j$, $\forall j$.

Proof: The proof follows from an analogous reasoning as in the proof of Corollary 1 in [8]. ■

C. Properties of the feasibility pump certification algorithm

This subsection shows that the parametric OFP in Algorithm 6 coincides point-wise with the online Algorithm 3. There are two sources of partitioning a region at each iteration of Algorithm 6; the ROUNDPARAM and SOLVELPCERT procedures. In the following, we consider the first one that is called at Step 7 over a single polyhedral region $\tilde{\Theta}$.

Lemma 3: The ROUNDPARAM procedure in Algorithm 6 has the following properties,

(i) Polyhedral region $\tilde{\Theta}$ *in* $\Rightarrow$ polyhedral partition $\{\Theta^i\}_i$ *out*, where *in* and *out* denote the input and output of the procedure.

(ii) After an execution of the ROUNDPARAM procedure in Algorithm 6, a returned $\hat{x}(\theta)$ coincides with the corresponding output $\hat{x}$ from the ROUND procedure in Algorithm 3 for any fixed $\theta \in \tilde{\Theta}$.

Proof: In each iteration of the while-loop in ROUNDPARAM, a popped region Θ at Step 26 is potentially partitioned by the hyperplane $\hat{x}_k(\theta) = \hat{F}_k\theta + \hat{G}_k$ into Θ_0 and Θ_1 at Step 29. As the inputted region $\tilde{\Theta}$ is polyhedral and is split by a hyperplane, the resulting non-empty regions Θ_0 and Θ_1, stored in $\mathcal{S}_r$, are polyhedral. As the stored non-empty regions in $\mathcal{S}_r$ are all polyhedral, and there is no overlap or holes between them, they form a polyhedral partition of $\tilde{\Theta}$ in $\mathcal{S}_r$, consequently in $\mathcal{F}_r$, confirming item (i).

Now consider an iteration of ROUNDPARAM for an arbitrary fixed $\theta \in \tilde{\Theta}$. For the fixed θ in Algorithm 6, as long as there exists a relaxed binary variable in $\hat{x}(\theta)$, its index $k \in \bar{\mathcal{B}}$ is selected and removed from $\bar{\mathcal{B}}$ (Step 28), identically to Step 16 in Algorithm 3. Now, depending on the value of $\hat{x}_k(\theta) = \hat{F}_k\theta + \hat{G}_k$, if $\hat{x}_k(\theta) \leq 0.5$, it is rounded down (at Step 31, $\forall\theta \in \Theta$, or at Step 35, $\forall\theta \in \Theta_0$), identically to Step 18 in Algorithm 3. Otherwise, if $\hat{x}_k(\theta) > 0.5$, it is rounded up (at Step 33, $\forall\theta \in \Theta$, or at Step 36, $\forall\theta \in \Theta_1$), identically to Step 20 in Algorithm 3. The modified $\hat{x}(\theta)$ is stored in $\mathcal{S}_r$ and popped again at Step 26 and the procedure is repeated. When all binary variables are fixed to either 0 or 1, $\hat{x}_k(\theta)$ is pushed to the final list $\mathcal{F}_r$ at Step 38. This extra step does clearly not change $\hat{x}(\theta)$, but stores and returns it at Step 39, while it is returned immediately at Step 21 in Algorithm 3. That is, an identical rounded solution is outputted by both algorithms for θ. As θ was chosen arbitrarily, it holds $\forall\theta \in \tilde{\Theta}$, confirming item (ii). ■

In the following, we confirm that the result of the certification Algorithm 6 for the OFP method is exact.

Lemma 4: Assume Assumption 1 in [8] holds. The complexity measure $\kappa^h(\theta) : \bar{\Theta} \to \mathbb{N}$ returned by Algorithm 6 is PPWC, that is $\kappa^{h_j}(\theta) = \kappa^{h_j}$, $\forall\theta \in \Theta^j \subseteq \bar{\Theta}$, $\forall j$.

Proof: From Assumption 1 in [8], κ^j returned by the SOLVELPCERT procedure is constant in Θ^j, so is the accumulated complexity measure $\kappa^h + \kappa^j$, $\forall\theta \in \Theta^j$, at any iteration of Algorithm 6. This in turn means that $\kappa^h(\theta)$ is a PWC function of θ over all regions at all iterations in Algorithm 6, and since all regions are polyhedral, $\kappa^h(\theta)$ returned by the algorithm is PPWC. ■

Theorem 3: Assume Assumption 1 in [8] holds. Let $\kappa^{h*}(\theta)$ be the accumulated complexity measure returned by Algorithm 3 for a given $\theta \in \bar{\Theta}$. Moreover, let $\{\Theta^j, \hat{x}^j, \kappa^{h_j}\}_j$ be the collection of tuples in $\mathcal{F}_h$ returned by Algorithm 6. Then $\kappa^{h_j} = \kappa^{h*}(\theta)$, $\forall\theta \in \Theta^j$, $\forall j$.

Proof: From Assumption 1 in [8], the SOLVELPCERT function correctly certifies the LP relaxations. That is, at an arbitrary iteration in Algorithm 6, the resulting $\kappa^j(\theta)$ at Step 13 is identical to the returned κ at Step 8 in Algorithm 3 for a fixed θ in that region. The accumulated complexity measure $\kappa^h(\theta)$ is then updated for each Θ^j at Step 16 and 18, identically to Step 9 in Algorithm 3. This, in turn, implies that the accumulated complexity measure returned by Algorithm 6 is identical to $\kappa^{h*}(\theta)$ from Algorithm 3 for any $\theta \in \bar{\Theta}$. ■

D. Properties of the start-heuristic-based B&B certification framework

This subsection shows that the B&B certification framework integrating the parametric start heuristics coincides point-wise with the online B&B algorithm integrating the online start heuristics. In the following, the properties of the B&B certification algorithm including heuristics are investigated.

Lemma 5: Assume Assumption 1 in [8] holds. Let STARTHEURISCERT denote any of the certification procedures in Algorithms 4–6 integrated into Algorithm 4 in [8]. Then, the complexity measure $\kappa_{tot}(\theta) : \Theta_0 \to \mathbb{N}$ returned by the B&B certification Algorithm 4 in $\mathcal{F}$ is PPWC, that is $\kappa^i_{tot}(\theta) = \kappa^i_{tot}$, $\forall\theta \in \Theta^i \subseteq \Theta_0$, $\forall i$.

Proof: First, consider an iteration of the B&B certification framework where a heuristic is called. From Assumption 1 and Lemma 4 in [8], κ^j_{tot} is constant for a resulting polyhedral region Θ^j at Step 13 of Algorithm 4 in [8]. Moreover, from Lemmas 1, 2, and 4, κ^{h_i} returned by the three start heuristics in the final list $\mathcal{F}_h$ is constant $\forall\theta \in \Theta^i$. Now consider the extra steps of integrating a heuristic into the B&B certification algorithm shown in Section IV-D. For any $i \in \mathbb{N}_{1:N_h}$, the overall accumulated complexity measure κ^i_{tot} computed by $\kappa^j_{\text{tot}} + \kappa^{h_i}$ is again constant in the polyhedral region Θ^i. This implies that $\kappa_{\text{tot}}(\theta)$ is PPWC at any iteration of Algorithm 4 in [8] that a heuristic is called. If, on the other hand, a heuristic is not called, the result follows from Lemma 4 in [8]. Therefore, at any iteration of Algorithm 4 in [8] integrated heuristics, $\kappa_{\text{tot}}(\theta)$ is PPWC. ■

Corollary 1: Assume Assumption 1 in [8] holds. Let $\kappa^*_{tot}(\theta)$ be the accumulated complexity measure returned by Algorithm 1 in [8] integrating the STARTHEURIS procedure for a given $\theta \in \Theta_0$. Moreover, let $\{\Theta^i, \kappa^i_{tot}\}_i$ be the collection of tuples in $\mathcal{F}$ returned by Algorithm 4 in [8] integrating the STARTHEURISCERT procedure. Then $\kappa^i_{tot} = \kappa^*_{tot}(\theta)$, $\forall\theta \in \Theta^i$.

Proof: Consider the extra steps of integrating the start heuristics into the B&B online and certification algorithms shown in Section IV-D for an arbitrary fixed $\theta \in \Theta^i$. If a heuristic is called for θ, the accumulated complexity measures $\kappa^*_{\text{tot}}(\theta)$ and κ^i_{tot} are updated identically in both algorithms, respectively, (see Section IV-D). As θ is chosen arbitrary, it holds $\forall\theta \in \Theta^i$, and $\forall i$. The rest of the proof follows from Corollary 1 in [8] and Theorems 1–3 for the three start heuristics. ■

VI. NUMERICAL EXPERIMENTS

To illustrate the use of the proposed methods, the B&B certification Algorithm 4 in [8] integrating Algorithms 4–6 was tested on 25 randomly generated mp-MILPs. The elements of the coefficients of (1) with $n = 8$, $n_b = 4$, $n_\theta = 3$, and $m = 14$, were generated according to,

$$c_i \sim \mathcal{N}(0,1),\ A_{ij} \sim \mathcal{N}(0,1),\ b_i \sim \mathcal{U}([0,2]),\ W_{ij} \sim \mathcal{N}(0,1)$$

where the considered parameter set is $\Theta_0 = \{\theta \in \mathbb{R}^3 | -1 \leq \theta \leq 1\}$. The complexity measures in the experiments are considered as the number of LP iterations and the size of the B&B tree. Moreover, the SOLVELPCERT procedure for the dual simplex algorithm falls into the category of complexity

certification algorithms covered by the framework in [16], and in particular Section VI-B therein. Furthermore, the next variable to branch on in diving has been selected as the lowest index in the set in the online and certification algorithms. Moreover, $r = 0.5$, $\alpha_0 = 1$, and $\phi = 0.9$ are chosen for the RENS and OFP methods, respectively. The numerical experiments were implemented in Julia and executed on an Intel® Core 1.8 GHz i7-8565U CPU.

The obtained accumulated complexity measures from Algorithm 4 in [8] integrated with the start heuristics in Section IV individually, for all 25 mp-MILPs, were compared with results from Monte-Carlo (MC) simulations. That is, the online B&B Algorithm 1 in [8], integrated with each one of the online start heuristics in Section III, was applied to MILP instances defined by 1000 parameter samples taken on a deterministic grid in Θ_0. The Chebychev centers of the resulting regions in the final partition were also considered as extra parameter samples. The results from all 75 experiments (25 mp-MILPs, each with the 3 start heuristics) are that the resulting complexity measures from the certification and the online algorithms when applying the three start heuristics coincide for all samples in all problems, with no discrepancy, confirming the validity of the proposed certification method.

As a demonstration of the usefulness of the proposed start-heuristic-based certification algorithm, the performance effect of individual heuristics when certifying the complexity of generated mp-MILPs is studied and reported in Table I for the 100 experiments. The values in Table I show,

- $I_{\text{gm}}^{\text{wc}}$ and $I_{\text{gm}}^{\text{avg}}$: the geometric mean of the worst-case and average number of LP iterations, respectively
- $N_{\text{gm}}^{\text{wc}}$ and $N_{\text{gm}}^{\text{avg}}$: the geometric mean of the worst-case and average number of B&B nodes, respectively
- $t_{\text{gm}}^{\text{cert}}$: the geometric mean of the execution time of the certification algorithm in seconds

of 25 experiments with 4 different settings compared to the default setting (with no heuristic), thereby value 1 in the first row of Table I. From this table, all heuristics have decreased the worst-case and the average complexity measures. OFP might take longer to be certified offline, possibly due to further partitioning of the space.

TABLE I: The geometric mean of the worst-case and average complexity measures and the certification time for 25 random mp-MILPs for the three heuristics w.r.t. the default setting.

Heuristic	$I_{\text{gm}}^{\text{wc}}$	$I_{\text{gm}}^{\text{avg}}$	$N_{\text{gm}}^{\text{wc}}$	$N_{\text{gm}}^{\text{avg}}$	$t_{\text{gm}}^{\text{cert}}[s]$
None	1	1	1	1	1
RENS	0.948	0.931	0.83	0.8	1
Diving	0.954	0.961	0.872	0.863	0.66
OFP	0.987	0.963	0.975	0.92	1.59

As a result, the proposed method provides a tool to analyze the performance gain from a start heuristic. Moreover, as each heuristic method has some user-defined parameters, the method in this work could help in identifying high-performing values of those.

VII. Conclusion

In this paper, complexity certification of three commonly used start heuristics integrated in B&B has been considered. Such knowledge is essential to have for real-time applications such as hybrid MPC, where hard real-time requirements have to be fulfilled. The main contribution of this work is certification algorithms that can a priori offline compute the exact worst-case computational complexity, e.g., the total number of LP iterations or nodes, performed by a standard B&B-based MILP solver also when these heuristics are incorporated. Furthermore, the theoretical overall computational complexity is characterized and shown to be PPWC. The algorithms have been successfully verified in numerical examples. Moreover, as an application, it is illustrated that the result from this work can be used to determine whether a heuristic method helps in speeding up the B&B process, and exactly to what extent. In future work, the complexity certification of improvement heuristics will also be considered. Moreover, techniques to use this framework to optimize the online performance will be investigated, and it will be considered how to bound the computation *time*.

Acknowledgment

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

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

[2] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[3] V. Dua and E. N. Pistikopoulos, "An algorithm for the solution of multiparametric mixed integer linear programming problems," *Annals of Operations Research*, vol. 99, no. 1, pp. 123–139, 2000.

[4] A. Bemporad, F. Borrelli, M. Morari *et al.*, "Model predictive control based on linear programming – the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.

[5] 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.

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

[7] S. Shoja, D. Arnström, and D. Axehill, "Overall complexity certification of a standard branch and bound method for mixed-integer quadratic programming," in *Proceedings of 2022 American Control Conference (ACC)*, 2022, pp. 4957–4964.

[8] ——, "Exact complexity certification of a standard branch and bound method for mixed-integer linear programming," in *Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*, 2022, pp. 6298–6305.

[9] S. Shoja and D. Axehill, "Exact complexity certification of suboptimal branch-and-bound algorithms for mixed-integer linear programming," in *Proceedings of the 22nd IFAC World Congress*, 2023.

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

[11] T. Berthold, "Primal heuristics for mixed integer programs," Master's thesis, Technische Universität Berlin, 2006.

[12] T. Achterberg, "Constraint integer programming," Ph.D. dissertation, Technische Universität Berlin, 2007.

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

[14] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," *Mathematical Programming*, vol. 104, pp. 91–104, 2005.

[15] T. Achterberg and T. Berthold, "Improving the feasibility pump," *Discrete Optimization*, vol. 4, no. 1, pp. 77–86, 2007.

[16] D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2758–2770, 2022.