

Mind the Gap: Online Control Allocation for Multirotors with Low-speed Deadbands

Ahmed Ali^{*1}, Fiorella Romano^{*1,2}, Chiara Gabellieri¹, Pieter van Goor^{1,3}, Fabio Ruggiero², Antonio Franchi^{1,4}

Abstract— In this work, we present a novel optimization-based input allocation strategy for fully-actuated multirotor aerial vehicles with N propellers that explicitly respects box constraints on the control inputs in both positive and negative actuation values while accounting for a deadband around zero. The resulting optimization problem belongs to the fundamental class of NP-hard global optimization problems, which can be recast as a Mixed-Integer Linear Program (MILP). We show that feasible solutions to this formulated MILP can efficiently be computed using a standard branch-and-bound algorithm. Building on that algorithm, the proposed method also robustly handles cases in which the desired allocation is infeasible by introducing several fallback instances. The method is experimentally validated on an octo-rotor platform, demonstrating the effectiveness of the proposed approach and its superior performance compared to the conventional QP-based allocation method.

I. INTRODUCTION

Fully-actuated Multirotor Aerial Vehicles (MAVs) represent an emerging class of aerial robots capable of decoupled force and moment generation, allowing for executing independent translational and rotational maneuvers [1]. Over the past decade, progress in this relatively recent technology has enabled deployment in a vast range of industrial applications. However, these advances have, in turn, also raised new challenges in system design, modeling, and control [2].

One such challenge is the development of a strategy to map the desired wrench, generated by a control scheme, to the actuator speeds while respecting their hardware limits. This problem is commonly referred to as control or input allocation. Straightforward inversion of the allocation matrix is always possible, provided that it is full-rank, for square systems, such as the ODAR vehicle [3] and the hexarotor [4] where the authors solved the allocation problem taking

the inverse of this matrix, this approach is generally not applicable to a large class of fully-actuated MAVs since they typically possess more actuators than the dimension of the wrench space, resulting in a redundant system [5].

Several works have addressed the control allocation problem in the literature, often in the context of implementing specific control strategies [6]. In the absence of actuator constraints, or when the desired motion is guaranteed to remain sufficiently slow such that actuator limits are never reached, the min-norm solution, obtained via the Moore-Penrose pseudoinverse of the allocation matrix, can provide a very computationally-efficient approach, which is the method adopted in [7]–[10]. Alternatively, the works in [11], [12] use a weighted pseudoinverse to properly balance the distribution of control effort among the inputs. Although it does not impose constraints on propeller speeds, the authors in [13] show that weighting matrices can be utilized to accommodate the maximum tilt constraints of the actively-tilting propellers. Comparison of the performance of various pseudoinversion schemes in experiments is given in [14]. However, in practical applications, box-shaped actuator constraints are ubiquitous because lower bounds on propeller speeds are imposed by common sensor-less electronic speed controllers whose indirect rotation speed measurement using current sign inversion and a timer does not work properly below a certain speed threshold [15], rendering the unconstrained pseudoinverse solution generally insufficient.

With numerous efficient numerical solvers available [16], quadratic programming (QP) constitutes a convenient framework to deal with input constraints, when the feasible solutions live in a convex set, since it is well-suited for real-time implementation on reasonably-sized widely-available microcontrollers which typically require less than 0.5 ms to compute a solution [17]. This approach is followed by a large body of literature, representative examples include [18]–[21]. The method in [22] formulates a constrained convex optimization problem solving the exact allocation while taking into account the actuator dynamics and their bounds. Furthermore, exploiting redundancy through optimizing the projection on the nullspace of the allocation matrix to satisfy input constraints is explored in [23], [24].

An alternative methodology consists of incorporating the allocation problem directly into the control algorithm, such that the actuator commands are computed simultaneously with the control law itself. For instance, in [25], the authors present a discretely-linearized convex optimization frame-

¹ Robotics and Mechatronics Department, Electrical Engineering, Mathematics, and Computer Science (EEMCS) Faculty, University of Twente, 7500 AE Enschede, The Netherlands. ahmed.ali@utwente.nl, fiorellaromano@icloud.com, c.gabellieri@utwente.nl, schol@r-franchi.eu

² PRISMA Laboratory, Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy fabio.ruggiero@unina.it

³ School of Aerospace, Mechanical, and Mechatronic Engineering (AMME), Faculty of Engineering, University of Sydney, NSW, 2006, Australia. pieter.vangoor@sydney.edu.au

⁴Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy. schol@r-franchi.eu

This work was partially supported by the Horizon Europe research agreement no. 101120732 (AUTOASSESS), by NWO OTP AVIATOR, and by INVITALIA Project NEMESI (CUP C67G22000420008).

* The authors have equally contributed to the work, and both can be considered first authors.

work that solves for the propeller speeds while enforcing input constraints. Moreover, replacing the optimization formulation, an explicit reference governor (ERG) that modifies the reference signals such that actuator limits are not exceeded is proposed in [26]. Similarly, model predictive control (MPC) framework can explicitly consider actuator limits within its list of constraints, as demonstrated in [27]–[29].

The focus in all of those methods remains solely on *convex* feasible-solution sets. However, the feasible set shaped by the input constraints can become non-convex when each propeller input is restricted to two disjoint bounded intervals, corresponding to the two actuation directions and separated by a deadband around zero. This is the case when bidirectional propellers are used, as commonly done in fully-actuated multirotor designs [10], [30]. These vehicles may require some propellers to reverse their direction of rotation during flight. In each of these directions, the electronic speed controllers (ESC) may impose a minimum speed limit in addition to a maximum rotation speed. This creates a deadzone around zero velocity in the input space.

While [31] implements a nullspace optimization method to impose constraints of that nature commanding inputs away from the deadzone, this optimization is not guaranteed to find a global solution, since the search is restricted to a hysteresis set, i.e., a set within which some propellers assume fixed rotation directions for a prespecified time period. By seeking solutions only within that set at a given instant, it may exclude other feasible and lower-cost thrust distributions corresponding to different combinations of rotation directions among propellers. Moreover, due to this waiting period introduced by the hysteresis mechanism, the allocation problem may become infeasible even when the desired wrench is physically attainable. This method might also not be easy to generalize if the explicit nullspace span is difficult to compute or does not exhibit a convenient structure.

In contrast, our method directly minimizes switching in the cost function itself, does not depend on any special structure of the allocation matrix or its nullspace, and solves the full constrained allocation problem in real time, yielding a global solution and making it applicable to more general redundant systems. Therefore, this work addresses the constrained control allocation problem, handling a non-convex feasible set. The main contributions of this work are as follows.

- A formulation of the control allocation problem for redundantly actuated MAVs as a non-convex optimization problem that explicitly accounts for the saturation limits of bidirectional propellers in each direction of rotation.
- A reformulation of this non-convex optimization into an equivalent Mixed-Integer Linear Program (MILP) that also minimizes switching in the propeller rotation directions.
- A computationally-efficient procedure suitable for real-time implementation that solves this allocation problem, based on the Branch-and-Bound algorithm [32].

An experimental validation of the proposed allocation algorithm on an octarotor MAV platform is performed to validate the contributions above. The proposed allocation is compared against a QP-based allocation, showing considerable

improvement in the trajectory tracking accuracy.

The rest of the paper is organized as follows. Sec. II lays out the formulation of the non-convex optimization and its equivalent MILP problem, while Sec. III describes the proposed solution procedure. We devote Sec. IV to display the results of the flight experiments. Sec. IV concludes the paper by summarizing the results and giving some remarks on future work.

II. PROBLEM FORMULATION

This section introduces the formulation of the non-convex QP optimization allocation problem for a generic redundantly actuated MAV with $N > 6$ fixedly-attached propellers. Subsequently, an equivalent mixed-integer linear programming (MILP) formulation is described.

For ease of notation, and without loss of generality, we assume that all propellers share identical speed bounds denoted by $\underline{u}, \bar{u} \in \mathbb{R}$ for minimum and maximum speed limits, respectively. Under bidirectional operation, the admissible speed set for each propeller i is given by the union of two disjoint intervals in $\mathbb{R}$ corresponding to positive (counterclockwise) and negative (clockwise) rotation directions. Let the admissible speed interval in the positive direction be $[\underline{u}, \bar{u}] \subset \mathbb{R}$, hence the feasible speed set of propeller i is $S_i = -[\underline{u}, \bar{u}] \cup [\underline{u}, \bar{u}]$. For a number of N propellers, the admissible input set S^N is a union of 2^N disjoint convex sets and given by

$$\begin{aligned} S^N &:= \{ \mathbf{u} = \text{vec}(\mathbf{u}_i) \in \mathbb{R}^N : \underline{u} \leq |u_i| \leq \bar{u}, i = 1, \dots, N \} \\ &= \bigcup_{\mathbf{k} \in \{1, -1\}^N} (k_1[\underline{u}, \bar{u}] \times \dots \times k_N[\underline{u}, \bar{u}]) \subset \mathbb{R}^N \end{aligned} \quad (1)$$

where $\text{vec}(\ast)$ is a vector in $\mathbb{R}^N$ whose elements are $\ast$, $u_i = \omega_i |\omega_i| \in S_i$ with $\omega_i \in \mathbb{R}$ being the rotation speed of propeller i while k_i is the component of each sign vector $\mathbf{k} \in \{1, -1\}^N$ that corresponds to propeller i direction of rotation: $k_i = 1$ or $k_i = -1$ for counterclockwise or clockwise rotation, respectively. Figure 1 visualizes the topology of this input space for a trivial example of a two-propeller MAV. In this situation, $S^2 \subset \mathbb{R}^2$ becomes

$$\begin{aligned} S^2 &= \bigcup_{\mathbf{k} \in \{1, -1\}^2} (k_1[\underline{u}, \bar{u}] \times k_2[\underline{u}, \bar{u}]) \\ &= [\underline{u}, \bar{u}] \times [\underline{u}, \bar{u}] \cup [\underline{u}, \bar{u}] \times [-\bar{u}, -\underline{u}] \\ &\quad \cup [-\bar{u}, -\underline{u}] \times [\underline{u}, \bar{u}] \cup [-\bar{u}, -\underline{u}] \times [-\bar{u}, -\underline{u}] \end{aligned} \quad (2)$$

Now, we define the allocation problem.

Problem 1 (Non-convex QP). *Given the allocation matrix $\mathbf{A} \in \mathbb{R}^{6 \times N}$, mapping the propeller speeds to the desired wrench [5], and a desired wrench $\mathbf{w}_l \in \mathcal{W}$ at time instant l , where $\mathcal{W} \subseteq \mathbb{R}^6$ is the wrench space, find an admissible propeller speeds vector $\mathbf{u}_l \in S^N$, with S^N defined as in (1), such that it satisfies P.E, if feasible, or otherwise P.N*

- **P.E: Exact Allocation**

$$\begin{aligned} \min_{\mathbf{u}_l \in S^N} \quad & \frac{1}{2} \|\mathbf{u}_l\|_{2, \mathbf{Q}_u} + \frac{1}{2} \|\mathbf{u}_l - \mathbf{u}_{l-1}\|_{2, \mathbf{Q}_{\Delta u}} \\ \text{s.t.} \quad & \mathbf{w}_l = \mathbf{A} \mathbf{u}_l \end{aligned} \quad (3)$$

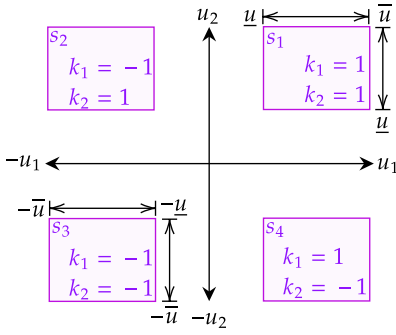


Fig. 1: Example of the admissible input set $S^2 \subset \mathbb{R}^2$ in case of $N = 2$ bidirectional propellers u_1 and u_2 : $S^2 = s_1 \cup s_2 \cup s_3 \cup s_4$. k_i indicates the rotation direction of propeller i inside convex subset s_j , with $j = 1, \dots, 2^N$.

• P.N: Minimum Error Allocation

$$\min_{\mathbf{u}_l \in S^N} \frac{1}{2} \|\mathbf{u}_l\|_{2, Q_u} + \frac{1}{2} \|\mathbf{u}_l - \mathbf{u}_{l-1}\|_{2, Q_{\Delta u}} + \frac{1}{2} \|\mathbf{w}_l - \mathbf{A} \mathbf{u}_l\|_{2, Q_w} \quad (4)$$

where $\|\cdot\|_{2, Q}$ denotes the Euclidean 2-norm of $\cdot$ weighted by an appropriately-sized positive-definite matrix Q .

This optimization in Problem 1 is non-convex due to the non-convexity of the admissible input set $S \subset \mathbb{R}^N$, which is the union of 2^N disjoint hyperrectangles in $\mathbb{R}^N$. For sufficiently small N , the problem can be solved in a brute force fashion [33]. In particular, all 2^N sign combinations are considered, each corresponding to a convex subset of S for which the resulting problem reduces to a convex quadratic program (QP). The optimal solution is then selected as the one that minimizes the objective function among all candidate solutions obtained by enumerating over all these 2^N subsets. This approach becomes computationally-intractable as N gets larger, making it unsuitable for real-time implementation.

Much faster algorithms to solve this class of problems exist if it can be transformed to a mixed-integer optimization. To this end, we define a binary vector $\mathbf{y} \in \{0, 1\}^N$, whose i -th component encodes the direction of rotation of the i -th propeller. Specifically, $y_i = 0$ corresponds to negative rotation, otherwise it represents a positive rotation. The sign variable k_i is mapped to y_i according to

$$k_i = 2y_i - 1, \quad k_i \in \{-1, 1\}. \quad (5)$$

Thus, similarly to $\mathbf{k}$, each instance of the vector $\mathbf{y}$ indexes a convex subset in S^N . Problem 1 can thus be recast as the following MILP that seeks, at each instant l , solutions of $\mathbf{u}_l$ in $\mathbb{R}^N$ and $\mathbf{y}_l \in \{0, 1\}^N$ subject to 0-1 integer constraints.

Problem 2 (MILP). Given the allocation matrix $\mathbf{A} \in \mathbb{R}^{6 \times N}$ and a desired wrench $\mathbf{w}_l \in \mathcal{W}$ at time instant l , find an admissible propeller speeds vector $\mathbf{u}_l \in \mathbb{R}^N$ and the associated vector $\mathbf{y}_l \in \{0, 1\}^N$ such that they satisfy P.E, if feasible, or otherwise P.N

• P.E: Exact Allocation

$$\begin{aligned} \min_{\mathbf{u}_l, \mathbf{y}_l} \quad & \frac{1}{2} \|\mathbf{u}_l\|_{2, Q_u} + \frac{1}{2} \|\mathbf{u}_l - \mathbf{u}_{l-1}\|_{2, Q_{\Delta u}} \\ & + \|\mathbf{y}_l - \mathbf{y}_{l-1}\|_{1, \lambda} \\ \text{s.t.} \quad & \mathbf{w}_l = \mathbf{A} \mathbf{u}_l, \\ & \underline{u} \mathbf{y}_l - \bar{u}(\mathbf{1}_N - \mathbf{y}_l) \leq \mathbf{u}_l \leq \bar{u} \mathbf{y}_l - \underline{u}(\mathbf{1}_N - \mathbf{y}_l) \end{aligned} \quad (6)$$

• P.N: Minimum Error Allocation

$$\begin{aligned} \min_{\mathbf{u}_l, \mathbf{y}_l} \quad & \frac{1}{2} \|\mathbf{u}_l\|_{2, Q_u} + \frac{1}{2} \|\mathbf{u}_l - \mathbf{u}_{l-1}\|_{2, Q_{\Delta u}} \\ & + \|\mathbf{y}_l - \mathbf{y}_{l-1}\|_{1, \lambda} + \frac{1}{2} \|\mathbf{w}_l - \mathbf{A} \mathbf{u}_l\|_{2, Q_w} \\ \text{s.t.} \quad & \underline{u} \mathbf{y}_l - \bar{u}(\mathbf{1}_N - \mathbf{y}_l) \leq \mathbf{u}_l \leq \bar{u} \mathbf{y}_l - \underline{u}(\mathbf{1}_N - \mathbf{y}_l) \end{aligned} \quad (7)$$

where $\|\cdot\|_{1, \lambda}$ denotes the 1-norm weighted by a vector $\lambda \in \mathbb{R}_{\geq 0}^N$, whereas $\mathbf{1}_N$ signifies a vector of size N with all entries equal to 1.

The cost term on $\mathbf{y}$ in the objective penalizes the total number of jumps in $\mathbf{y}_l$ from the previous $\mathbf{y}_{l-1}$, weighted by the elements of λ which can be used to manage the distribution of chattering among the propellers. We focus our attention on solving Problem 2 in the next section.

III. PROPOSED ALLOCATION ALGORITHM

This section provides the algorithm used to compute the optimal solution to Problem 2, based on the Branch-and-Bound (B&B) search [32], and discusses its real-time implementation. We first recall the fundamental principles underlying the B&B algorithm. We conclude by constructing a tailored procedure that reduces the burden on the computational resources, making the proposed approach suitable for real-time deployment on MAVs platforms.

Problem 1 is an instance of non-convex quadratic programming with box constraints on the decision variables that form disjoint sets, which is well-known to belong to the class of NP-hard problems [34], meaning that, loosely speaking, there is no polynomial-time solution for them, rendering them intractable to solve for large inputs N . Although the time complexity to solve Problem 2 is exponential in N in the worst case, which is enumeration over all 2^N elements of the search space by brute force, the average complexity is shown to be much lower since the B&B algorithm can find the optimum without exploring all the search space [35], [36].

A. Background: Branch-and-Bound algorithm

The B&B algorithm, first appeared in [32], is briefly reviewed here to establish the necessary foundation for understanding the allocation procedure presented in the following. Interested readers are encouraged to refer to standard textbooks on integer programming, such as [33, Chapter 7], for a more comprehensive treatment.

The B&B algorithm computes the global optimum of Problem 2 by systematically exploring a search tree until a leaf node corresponding to a candidate integer-feasible solution is reached. At each node, three main operations are

performed after initialization: branching, which partitions the node's feasible set into smaller subproblems; node selection, which determines which child node to be explored first; and pruning, which eliminates subtrees that are infeasible or cannot contain a solution better than the current incumbent, i.e., best found solution of Problem 2.

The search tree originates at a root node corresponding to the continuous relaxation of Problem 2, obtained by allowing the binary decision variable $\mathbf{y}$ to take values in $\mathbb{R}^N$ and adding these bound constraints $\mathbf{0}_N \leq \mathbf{y} \leq \mathbf{1}_N$ to the existing ones. This reduces Problem 2 to a constrained convex QP. After solving it, its optimal objective value initializes the dual bound, denoted by J_D . If the $\mathbf{y}$ solution additionally is integer-feasible, i.e. all of its components live in $\{0, 1\}^N$, the objective provides an initial primal bound, symbolized by J_P , as well. Otherwise, the J_P is set to $+\infty$, indicating that no integer-feasible solution has yet been found, and then we start the iteration by branching out of this root node. At each iteration, the three operations performed are summarized as follows.

- **Branching:** If the solution $\mathbf{y}$ contains fractional components, branching is performed by selecting the component whose fractional value is closest to 0.5. Subsequently, two child nodes are generated by replacing the bound constraint on the selected component y_i existing in the parent's relaxed problem with the equality constraint $y_i = 0$ in one child node and $y_i = 1$ in the other. This shrinks the feasible set of the parent's relaxed problem by progressively enforcing integrality of $\mathbf{y}$. It is worth noting that different branching and node selection strategies exist [37], and their choice influences the size of the explored search tree and, consequently, the computational complexity required to obtain the optimal solution.
- **Node selection:** If neither of the two child nodes to the current one is pruned, one of them is selected to be explored next if it has the lowest dual bound J_D (cost). This best-bound strategy prioritizes the most promising sub-trees to search first, leading to minimizing the total number of nodes evaluated [33]. The tree can initially be traversed using a Depth-First Search (DFS), which has an $O(N)$ time complexity, i.e. the fastest descending in the tree by a sequential computer, until an integer-feasible solution is found. Then, node selection switches to best-bound strategy. This can rapidly yield an initial incumbent primal bound $J_P < +\infty$, boosting the pruning more efficiently in the next step.
- **Pruning:** After solving the relaxed problem at a given node, pruning criteria are applied to eliminate subtrees that cannot contain a better solution than the incumbent primal bound. Specifically, a node is pruned, if any of the following occurs:
 - the relaxed QP problem is infeasible;
 - the relaxed solution is integer-feasible, in which case its dual bound J_D is used to update the primal bound J_P only if it is lower than the incumbent primal bound J_P , and no further branching is required, making this node

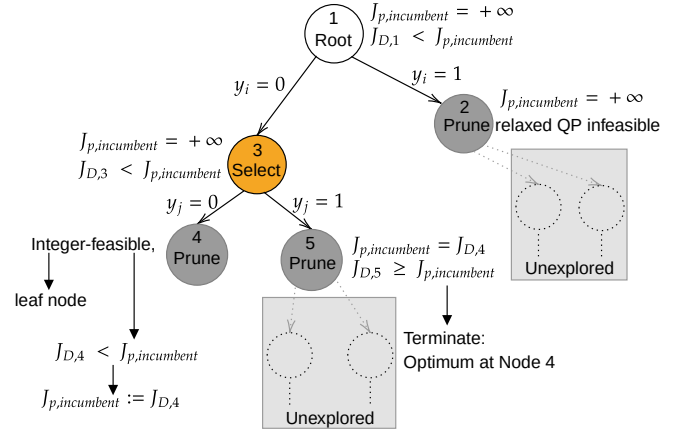


Fig. 2: Diagram illustrating a simple execution of our implemented version of B&B, explained in Sec. III-A. The maximum size of any branch is N . This tree is initially traversed using DFS until $J_p := J_{D,4} < +\infty$, an incumbent, is found, hence the node numbering, then switches to best-bound strategy.

a leaf;

- its dual bound J_D is greater than or equal to the incumbent primal bound, indicating that no better integer-feasible solution can exist within that sub-tree.

The algorithm terminates when no active nodes remain, i.e. all explored nodes have either yielded leafs or been pruned, at which point the incumbent primal bound corresponds to the globally optimal solution to Problem 2. Figure 2 depicts a simple search tree of the algorithm.

B. Proposed Solution

In the worst-case scenario, finding the global optimum with the B&B algorithm would require enumerating all 2^N nodes in the search tree. For an Octo-rotor with $N = 8$ propellers, this entails solving $2^8 = 256$ QPs per control cycle at any given instant, taking about 130 ms, assuming a QP takes about 0.5 ms at most [17], causing the computational load to be incompatible with the frequency at which the control loop is typically executed (500 – 200 Hz) on flight autopilot boards. For this reason, any real-time implementation must have an exception handling mechanism to accommodate this extreme case in which the algorithm exceeds a prescribed time threshold dictated by the real-time constraint.

Here, we propose a solution procedure that invokes the B&B algorithm only when necessary and incorporates additional measures to handle cases in which the algorithm takes excessive time to converge. We detail the aforementioned procedure in Algorithm 1 to solve the allocation problem, including the fallback mechanisms employed when Problem 2 becomes infeasible or when some time thresholds denoted by $T_{P,E} > 0$ and $T_{P,N} > 0$ are exceeded. The optimization at time l is warm-started using the previous input $\mathbf{u}_{l-1}$ and set selector vector $\mathbf{y}_{l-1}$. Accordingly, the root node of the B&B algorithm is initialized with these values whereas each child node is warm-started using the optimal solution of its respective parent node. The rationale behind this is that the desired wrench typically evolves smoothly over time, making

Algorithm 1 Allocation Procedure at Instant l

Input: desired wrench w_l , previous commanded input u_{l-1} , previous commanded set vector y_{l-1} and it is measured current value y_l^{meas} from the ESC board.
Output: u_l and y_l : (u_l, y_l)
Solve allocation by pseudoinverse:

$$u_l^{\text{pinv}} \leftarrow A^\dagger w_l$$

if $u_l^{\text{pinv}}, u_{l-1} \in S^N$ **and** $y_l^{\text{meas}} = y_{l-1}$ **then**

$$(u_l, y_l) \leftarrow (u_l^{\text{pinv}}, \frac{1}{2}(\text{sign}(u_l^{\text{pinv}}) + \mathbf{1}_N))$$

return (u_l, y_l)

end if

Solve Prob. 2 (P.E) using B&B in Sec. III-A: ($u_l^{P.E}, y_l^{P.E}$)

if global $u_l^{P.E}$ feasible found before time $T_{P.E}$ **then**

$$(u_l, y_l) \leftarrow (u_l^{P.E}, y_l^{P.E})$$

return (u_l, y_l)

else if threshold $T_{P.E}$ reached **then**

$$(u_l, y_l) \leftarrow (u_{l,\text{inc}}^{P.E}, y_{l,\text{inc}}^{P.E}) \text{ \%local incumbent at } T_{P.E}$$

return (u_l, y_l)

end if

Solve Prob. 2 (P.N) by B&B in Sec. III-A: ($u_l^{P.N}, y_l^{P.N}$)

if global $u_l^{P.N}$ feasible found before time $T_{P.N}$ **then**

$$(u_l, y_l) \leftarrow (u_l^{P.N}, y_l^{P.N})$$

return (u_l, y_l)

else if threshold $T_{P.N}$ reached **then**

$$(u_l, y_l) \leftarrow (u_{l,\text{inc}}^{P.N}, y_{l,\text{inc}}^{P.N}) \text{ \%local incumbent at } T_{P.N}$$

return (u_l, y_l)

end if

Solve exact convex QP1 with only max constraints: u_l^{QP1}

$$\min_{u_l^{QP1}} \frac{1}{2} \|u_l^{QP1}\|_{2, Q_u} + \frac{1}{2} \|u_l^{QP1} - u_{l-1}\|_{2, Q_{\Delta u}}$$

$$\text{s.t. } w_l = A u_l^{QP1}, \quad -\bar{u} \leq u_l^{QP1} \leq \bar{u}$$

if feasible **then**

$$(u_l, y_l) \leftarrow (u_l^{QP1}, \frac{1}{2}(\text{sign}(u_l^{QP1}) + \mathbf{1}_N))$$

return (u_l, y_l)

end if

Solve min-err. convex QP2 with only max constraints: u_l^{QP2}

$$\min_{u_l^{QP2}} \frac{1}{2} \|u_l^{QP2}\|_{2, Q_u} + \frac{1}{2} \|u_l^{QP2} - u_{l-1}\|_{2, Q_{\Delta u}}$$

$$+ \frac{1}{2} \|w_l - A u_l^{QP2}\|_{2, Q_w}$$

$$\text{s.t. } -\bar{u} \leq u_l^{QP2} \leq \bar{u}$$

if feasible **then**

$$(u_l, y_l) \leftarrow (u_l^{QP2}, \frac{1}{2}(\text{sign}(u_l^{QP2}) + \mathbf{1}_N))$$

return (u_l, y_l)

else if infeasible **then**

$$\text{return } (u_l, y_l) \leftarrow (u_{l-1}, y_{l-1})$$

end if

the new optimal input likely to be close to the previous one. This can accelerate the convergence of the B&B algorithm by providing effective initial guesses to its relaxed-QPs.

At each sampling instant l , Algorithm 1 begins by computing the pseudoinverse solution. Before accepting it, it verifies whether it is admissible and whether the previously commanded input set has been successfully realized by the hardware. If these conditions are not satisfied, a timer $T_{P.E}$ is activated, during which the MILP solver attempts to compute the exact allocation P.E in Prob 2. If a global incumbent is

found while $T_{P.E}$ is still running, it terminates. Otherwise, if $T_{P.E}$ is on but P.E is infeasible, $T_{P.E}$ is replaced by a second timer $T_{P.N}$ for solving the non-exact allocation P.N in Prob 2. Similarly, if a global solution is found within $T_{P.N}$, it is returned. If P.N is found infeasible in the $T_{P.N}$ period, the procedure continues by solving an exact or min-error convex QPs, disregarding the deadzone, and denoted, respectively, by QP1 and QP2 in Alog. 1. In all cases, if either $T_{P.E}$ or $T_{P.N}$ is exceeded without a global incumbent, the algorithm returns the best incumbent solution available at that time, corresponding to a locally optimal allocation. Thus, the role of the timers is to limit the time allowed for the global feasibility check of the MILP problem. If all methods are infeasible, the previous solution is reapplied.

IV. EXPERIMENTAL VALIDATION

We carry out an experiment that is well-representative of the target application scenario. The octo-rotor robot used in the experiments, shown in Fig. 4, is an omnidirectional MAV with eight fixedly-tilted propellers and a rigidly-attached end-effector, for which modeling and trajectory tracking controller are extensively studied in [25], [30].

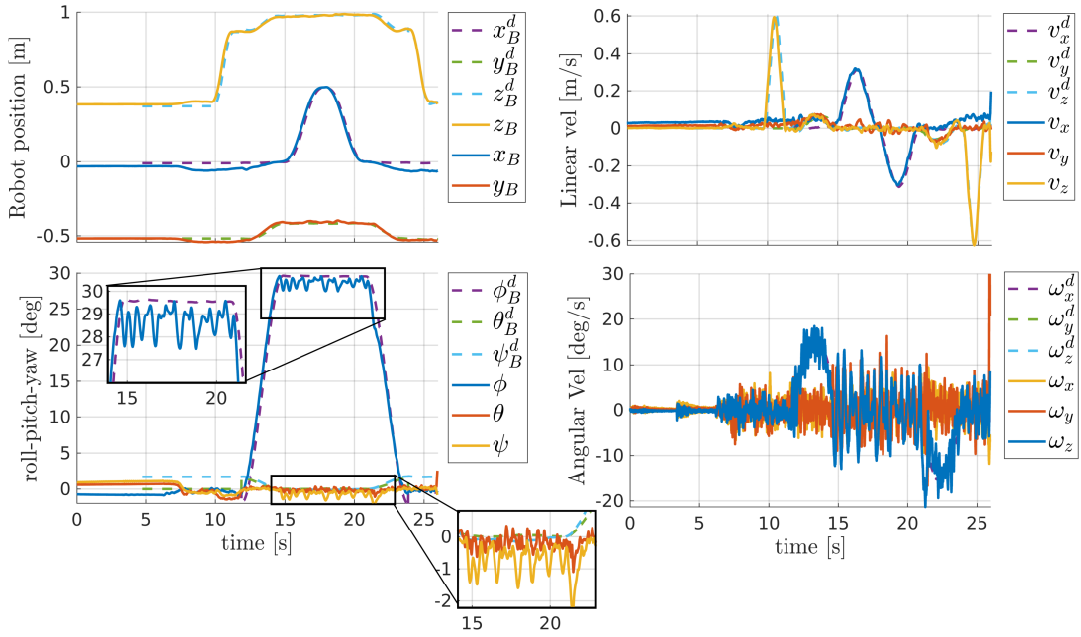
The experimental setup relies on several components. The trajectory controller, implemented in C++ as a GenoM3 component running on the Paparazzi Chimera v1.1 autopilot board, takes the state as an input whose estimation comes from a standard Unscented Kalman Filter (UKF). It fuses full pose readings from a motion capture system (OptiTrack) and measurements from the accelerometer and gyroscope sensors in the IMU of the autopilot. A reference trajectory obtained via a constant-snap minimum-time spline interpolation between waypoints is sent from a ground station equipped with intel i5-10210U CPU, having four cores and running at 1.60 GHz, to the onboard tracking controller. The controller outputs the desired wrench to the allocator, whose task is to compute the desired propeller speeds to be tracked by the propellers' ESCs. Two different allocators are used: the proposed one and the baseline.

The baseline allocator solves this convex QP-based allocation

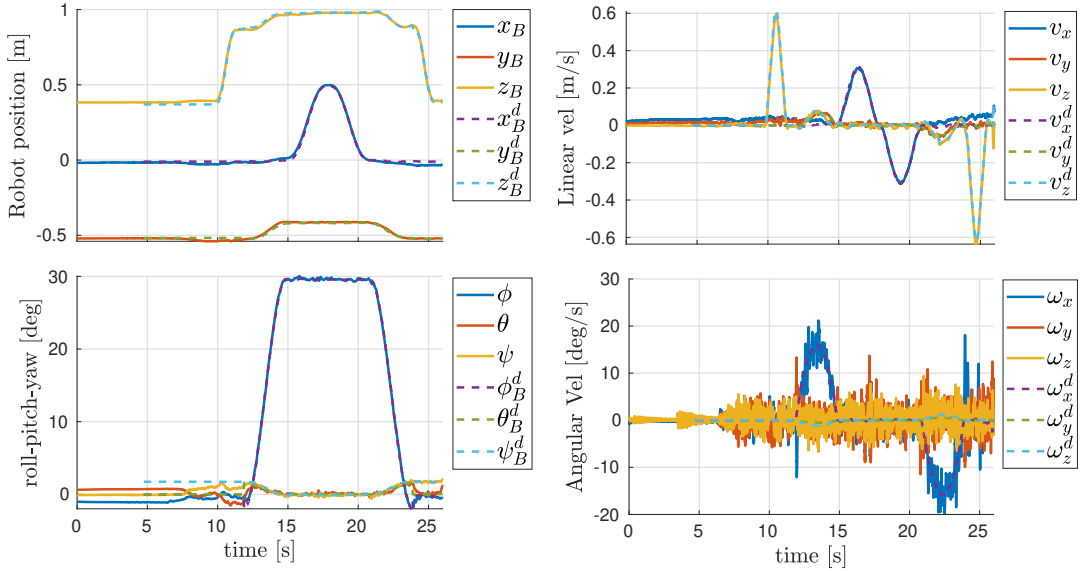
$$\begin{aligned} \min_{u_l^{QP}} \quad & \frac{1}{2} \|u_l^{QP}\|_{2, Q_u} + \frac{1}{2} \|u_l^{QP} - u_{l-1}\|_{2, Q_{\Delta u}} \\ \text{s.t.} \quad & w_l = A u_l^{QP}, \quad -\bar{u} \leq u_l^{QP} \leq \bar{u}. \end{aligned} \quad (8)$$

The baseline is deadzone-agnostic, so it can command propellers to lie in the deadzone for a relatively long duration. Both allocation methods are tested under identical initial conditions, desired trajectory, software and hardware settings, and controller tuning. All nodes solve convex QPs using the ProxQP solver [17], implemented with the Eigen library in GenoM3, and each QP is warm-started using the active-set method. Videos of the experiments are available at: <https://youtu.be/fkAZkyIDCJk>.

The results of the experiments are displayed in Fig. 3, where the state evolution along with the desired trajectory of the MAV during experiment are plotted. Observe that when the propeller speeds calculated by the baseline allocator (see



(a) Baseline: conventional QP allocation



(b) Proposed allocation (Algorithm 1)

Fig. 3: Time evolution of the states and their desired trajectory during a combined rotation and translation maneuver. x_B, y_B, z_B are the X-, Y-, and Z- coordinates of the robot's position in the inertial frame; ϕ, θ, ψ are the roll, pitch, and yaw angles of the robot, respectively. The superscript ^d indicates desired quantities. The proposed method significantly reduces attitude fluctuations compared to conventional QP allocation. The flight starts at 5 [s].

Fig. 3a), there is a considerable amount of chattering in the attitude component, which leads to violent vibrations in the mechanical chassis of the robot that are also visible in the provided video. The vibrations occur around 15 s, when the roll angle is commanded at 30 deg. Such vibrations can have a detrimental effect on achieving contact-based physical interaction with the end-effector when it is desired to slide along this trajectory or to establish static contact at the shown attitude in Fig. 4. Moreover, inaccuracies in position tracking ensue. In contrast, our method in Fig 3b yields an

improved smoother tracking by removing these variations in the attitude and providing a tighter tracking of the position trajectory.

To provide an insight into why our method enhances the tracking, Fig. 5 illustrates the propeller speeds obtained via the two allocation methods. As it can be noted, the amount of fast direction switching in the conventional QP-based method is substantial. Furthermore, it commands speeds within the deadzone, defined by the band between $-\underline{u}$ and $\underline{u}$, which is not realizable by the ESCs. These two phenomena



Fig. 4: The octarotor used in experiments during the 30-degree roll maneuver, where the QP-based allocation causes chattering and unrealizable speeds.

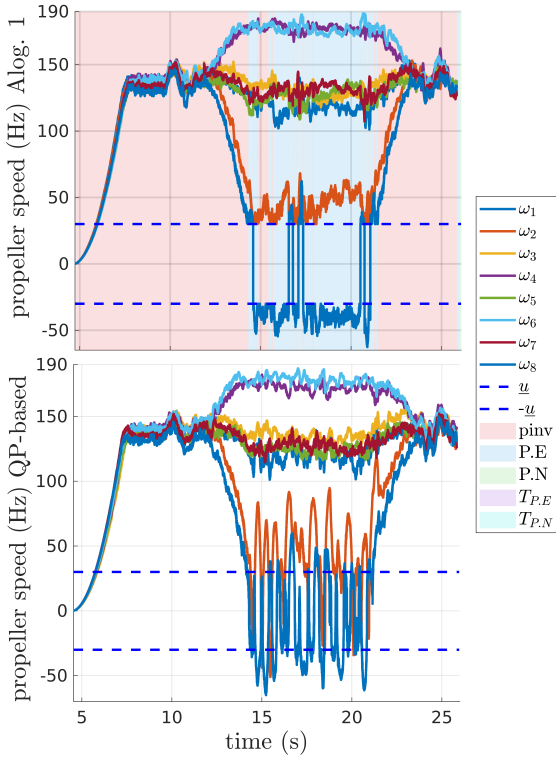


Fig. 5: Comparison between the commanded propeller speed between the two methods. Our method is on top, and the baseline is at the bottom. Unlike with conventional QP, minimized switching and no commanded speeds in the deadzone occur with the proposed allocation. Different color bands are assigned to the durations for which each subroutine in Algorithm 1 is active. Note that almost 100% of the time only 2 subroutines are active: pinv and P.E.

combined, namely fast switching and unrealizable speeds, result in the poor tracking seen in Fig. 3a. On the other hand, the proposed method minimizes switching, requiring only eight sufficiently-spaced switches of one propeller, and avoids requesting propeller speeds within this deadzone from the ESCs.

The computational cost of running our method is statistically evaluated in Fig. 6. While the exact allocation MILP

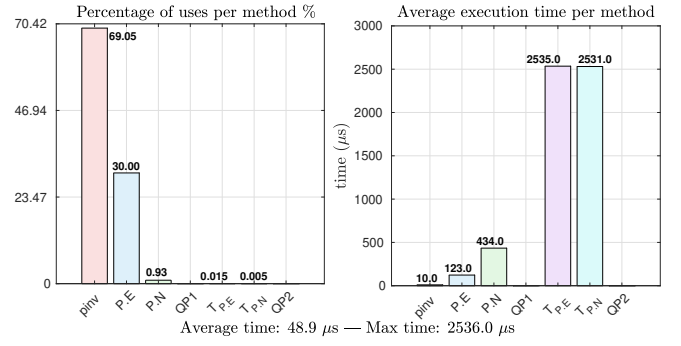


Fig. 6: Comparison of Algorithm 1 subroutines in terms of call frequency and average execution time. $T_{P,E} = 2.5[m.s]$ and $T_{P,N} = 2.5[m.s]$ combined are hit for 0.02% of the flight time.

(P.E) in Algorithm 1 takes on average around 0.125 ms to solve, the min-error MILP (P.N) converges to a solution after about 0.5 ms. This is expected since MILP (P.N) of Problem 2 has a larger admissible set of solutions due to the absence of the equality constraints, consequently requiring expanding more nodes to find the optimum. Both times are comparable to that of the baseline case which can take around 0.5 ms to solve the QP [16]. It is worth noting that the MILP allocation in Algorithm 1 is only executed when the almost instantaneous pseudoinverse solution is not admissible. The MILP time thresholds are reached only for 0.02% of the flight time.

V. CONCLUSION

We proposed an optimization-based formulation of the control allocation problem for fully-actuated MAVs equipped with bidirectional propellers that explicitly enforces the minimum and maximum saturation limits of each propeller in both directions of rotation. The resulting feasible set is shown to be the union of disjoint intervals. We recast the non-convex QP as a MILP, in which integer variables appear in both the cost functional and the constraints. Experimental results demonstrated that, compared to conventional QP-based allocation, the proposed allocator significantly reduces fluctuations in propeller speed commands when operating near the actuator dead-zones dictated by the hardware, improving trajectory tracking performance. At the same time, the computational time remains comparable to that of a standard QP-based scheme.

Future work will focus on extending the experimental validation to more aggressive flight scenarios, generalizing the allocation framework to treat the case of actively-tilted propellers with position and rate constraints, and developing a formal feasibility analysis of the proposed optimization.

REFERENCES

- [1] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli, "Fully actuated multirotor uavs: A literature review," *IEEE Robotics & Automation Magazine*, vol. 27, no. 3, pp. 97–107, 2020.
- [2] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2021.

- [3] S. Park, J. Her, J. Kim, and D. Lee, "Design, modeling and control of omni-directional aerial robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1570–1575.
- [4] R. Rashad, P. Kuipers, J. Engelen, and S. Stramigioli, "Design, modeling, and geometric control on se (3) of a fully-actuated hexarotor for aerial interaction," *arXiv preprint arXiv:1709.05398*, 2017.
- [5] M. Hamandi, F. Usai, Q. Sablé, N. Staub, M. Tognon, and A. Franchi, "Design of multirotor aerial vehicles: A taxonomy based on input allocation," *The International Journal of Robotics Research*, vol. 40, no. 8-9, pp. 1015–1044, 2021.
- [6] T. M. Blaha, E. J. J. Smeur, and B. D. W. Remes, "A survey of optimal control allocation for aerial vehicle control," *Actuators*, vol. 12, no. 7, 2023.
- [7] J. Xu, D. S. D'Antonio, and D. Saldaña, "Modular multirotors: From quadrotors to fully-actuated aerial vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 19328–19339, 2025.
- [8] J. Spaans, S. Gilbert, K. A. Stol, and S. Al-Zubaidi, "System identification for fully-actuated uav control allocation," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 193–200.
- [9] M. Ryll, D. Bicego, M. Giurato, M. Lovera, and A. Franchi, "Fast-hex—a morphing hexarotor: design, mechanical implementation, control and experimental validation," *IEEE/ASME transactions on mechatronics*, vol. 27, no. 3, pp. 1244–1255, 2021.
- [10] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," vol. 2016-June, 2016, Conference paper, p. 3261 – 3266.
- [11] P. Segui-Gasco, Y. Al-Rihani, H.-S. Shin, and A. Savvaris, "A novel actuation concept for a multi rotor uav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1, pp. 173–191, 2014.
- [12] M. Allenspach, K. Bodie, M. Brunner, L. Rinsoz, Z. Taylor, M. Kamel, R. Siegwart, and J. Nieto, "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1305–1325, 2020.
- [13] Y. Yang, X. Yu, Z. Li, and M. V. Basin, "A new overactuated multirotor: Prototype design, dynamics modeling, and control," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 8, pp. 9449–9459, 2024.
- [14] M. Kirchengast, M. Steinberger, and M. Horn, "Control allocation under actuator saturation: An experimental evaluation," *IFAC-PapersOnLine*, vol. 51, no. 25, pp. 48–54, 2018, 9th IFAC Symposium on Robust Control Design ROCOND 2018.
- [15] A. Franchi and A. Mallet, "Adaptive closed-loop speed control of bldc motors with applications to multi-rotor aerial vehicles," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 5203–5208.
- [16] F. Stark, J. Middelberg, D. Mronga, S. Vyas, and F. Kirchner, "Benchmarking different qp formulations and solvers for dynamic quadrupedal walking," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 14412–14418.
- [17] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "Prox-qp: Yet another quadratic programming solver for robotics and beyond," in *RSS 2022 - Robotics: Science and Systems*, New York, United States, June 2022.
- [18] E. Dyer, S. Sirouspour, and M. Jafarinasab, "Energy optimal control allocation in a redundantly actuated omnidirectional uav," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5316–5322.
- [19] M. Tognon and A. Franchi, "Omnidirectional aerial vehicles with unidirectional thrusters: Theory, optimal design, and control," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2277–2282, 2018.
- [20] M. Santos, L. Honório, A. P. G. M. Moreira, M. Silva, and V. F. Vidal, "Fast real-time control allocation applied to over-actuated quadrotor tilt-rotor," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 3, p. 65, 2021.
- [21] J. C. Monteiro, F. Lizarralde, and L. Hsu, "Optimal control allocation of quadrotor uavs subject to actuator constraints," in *2016 American Control Conference (ACC)*, 2016, pp. 500–505.
- [22] D. A. Santos and J. A. Bezerra, "On the control allocation of fully actuated multirotor aerial vehicles," *Aerospace Science and Technology*, vol. 122, p. 107424, 2022.
- [23] Y. Su, P. Yu, M. J. Gerber, L. Ruan, and T.-C. Tsao, "Nullspace-based control allocation of overactuated uav platforms," *IEEE robotics and automation letters*, vol. 6, no. 4, pp. 8094–8101, 2021.
- [24] M. Ryll, H. H. Bühlhoff, and P. R. Giordano, "A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 540–556, 2015.
- [25] Y. Aboudorra, C. Gabellieri, R. Brantjes, Q. Sablé, and A. Franchi, "Modelling, analysis, and control of omnimorph: an omnidirectional morphing multi-rotor uav," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 1, p. 21, 2024.
- [26] B. Convens, K. Merckaert, M. M. Nicotra, R. Naldi, and E. Garone, "Control of fully actuated unmanned aerial vehicles with actuator saturation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12715–12720, 2017, 20th IFAC World Congress.
- [27] J. Li, J. Sugihara, and M. Zhao, "Servo integrated nonlinear model predictive control for overactuated tiltable-quadrotors," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8770–8777, 2024.
- [28] Z. Shayan, J. Cristobal, M. Izadi, A. Yazdanshenas, M. Naderi, and R. Faiegghi, "Nonlinear model predictive control of tiltrotor quadrotors using feasible control allocation," *Journal of Intelligent & Robotic Systems*, vol. 111, no. 2, pp. 1–16, 2025.
- [29] M. Brunner, K. Bodie, M. Kamel, M. Pantic, W. Zhang, J. Nieto, and R. Siegwart, "Trajectory tracking nonlinear model predictive control for an overactuated mav," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5342–5348.
- [30] R. Veenstra, A. Ali, C. Gabellieri, and A. Franchi, "6d physical interaction with an omnidirectional aerial robot," *Drones*, vol. 10, no. 2, 2026.
- [31] D. Brescianini and R. D'Andrea, "An omni-directional multirotor vehicle," *Mechatronics*, vol. 55, pp. 76–93, 2018.
- [32] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
- [33] L. A. Wolsey, *Integer programming*. John Wiley & Sons, 2020.
- [34] S. Burer and A. N. Letchford, "On nonconvex quadratic programming with box constraints," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 1073–1089, 2009.
- [35] N. Thakoor, V. Devarajan, and J. Gao, "Computation complexity of branch-and-bound model selection," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 1895–1900.
- [36] A. Basu, M. Conforti, M. Di Summa, and H. Jiang, "Complexity of branch-and-bound and cutting planes in mixed-integer optimization," *Mathematical Programming*, vol. 198, no. 1, pp. 787–810, 2023.
- [37] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016.