2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Efficient Solution of Mixed-Integer MPC Problems for Obstacle Avoidance Using Hybrid Zonotopes

Joshua A. Robbins, Sean B. Brennan, and Herschel C. Pangborn

Abstract— **Model predictive control (MPC) is a powerful approach for autonomous vehicle motion planning. MPC can account for both the vehicle dynamics and obstacle avoidance constraints by iteratively solving constrained optimization problems. Due to the non-convexities associated with obstacle avoidance constraints, these optimization problems often take the form of mixed-integer programs, which are challenging to solve online in embedded applications. This paper presents a mixed-integer quadratic program (MIQP) solution strategy for obstacle avoidance MPC formulations based on a *hybrid zonotope* set representation of the obstacle avoidance constraints. The structure of the hybrid zonotope constraints is exploited within both a branch-and-bound mixed-integer solver and an interior point method QP solver. For applications such as automated driving, where the obstacle-free space can be represented using an occupancy grid, the QP solution time is not strongly affected by the complexity of the obstacle map. For the examples considered in this paper, using 5 and 10 step MPC horizons, the proposed MIQP solver with hybrid zonotope constraints found the optimal solution 2-6 times faster on average than general-purpose commercial solvers and up to 13 times faster than MIQP formulations using H-rep constraints.**

I. Introduction

Obstacle avoidance is a fundamental problem in vehicle autonomy, for which many different solution approaches exist [1]. This paper focuses on model predictive control (MPC) strategies for autonomous vehicle (AV) motion planning subject to obstacle avoidance constraints. MPC explicitly accounts for the AV dynamics and constraints, and can be formulated to guarantee obstacle avoidance subject to disturbances or model uncertainty [2]. However, MPC formulations for obstacle avoidance often require solving non-convex optimization problems, which can be computationally expensive. This has limited the practicality of applying MPC to embedded systems.

Broadly, obstacle avoidance MPC formulations use one of the following solution strategies: 1) approximate, local, or heuristic solutions, 2) simplification of the problem domain, or 3) global optimization. Approximate solution methods are often more computationally tractable than global solution methods at the expense of optimality. One example is sequential convex programming [3]–[5], which iteratively solves convex approximations of the original non-convex problem. Sequential convex programming converges to a local minimum but will not in general find the global minimum [6]. Other heuristic solution methods relevant to obstacle avoidance optimization problems have been proposed [7]–[10].

Sampling-based methods, such as model predictive path integral (MPPI) control, are also used to find approximate local solutions to non-convex MPC problems [11], [12]. MPPI approximates the optimal trajectory from a weighted sum of sample trajectories that are generated in the neighborhood of the optimal trajectory from the previous time step. This can present challenges when the globally optimal trajectory for the MPC problem changes significantly between time steps, as in the case of a large disturbance or an obstacle suddenly entering the field of view [13], [14].

Many authors simplify the problem domain in their MPC formulations. In [15], the authors construct convex safe regions for autonomous driving based on a list of different traffic scenarios. Others have used optimization-based methods to compute large convex regions of the obstacle-free space [16], [17]. Additional approaches to domain simplification include obstacle clustering [18] and the use of a neural network to classify which obstacle avoidance constraints are likely to be relevant to the MPC problem [19]. Methods that seek inner approximations of the obstacle-free space can be overly conservative, while methods that seek to remove irrelevant constraints can be insufficiently conservative.

The last family of solution strategies for obstacle avoidance MPC is global optimization over the non-convex constraints. Global optimization methods are not vulnerable to poor quality solutions, as in the case of local or approximate solution methods, and do not suffer from being overly conservative, as in the case of domain convexification methods. The primary challenge with global optimization methods is computational. Mixed-integer program (MIP) formulations of the optimization problem are most frequently used [2], [18], [20]–[24]. However, MIPs are NP-hard [25], and for sufficiently complex environments it is often intractable to solve MIPs online at update rates relevant to AV applications.

Custom mixed-integer solvers for obstacle avoidance problems have been proposed to address these challenges. In [26], the authors devise a branch-and-bound solver where branching corresponds to a clockwise versus counter-clockwise obstacle avoidance decision. In [27], a branch-and-bound method is introduced where branches are generated based on detected collisions in the optimal solutions to the relaxed sub-problems. An approximate branch-and-bound solver for path planning problems with uncertainty is presented in [28].

Advanced set representations—namely zonotopes and their extensions—have also been used to reduce the complexity of the resulting MIP. These representations describe

Joshua A. Robbins, Sean B. Brennan, and Herschel C. Pangborn are with the Department of Mechanical Engineering, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: `jrobbins@psu.edu`, `sbrennan@psu.edu`, `hcpangborn@psu.edu`).

This research was supported by Peraton.

sets implicitly and have found significant use in reachability analysis of dynamic systems [29]–[32]. In [33], zonotope obstacle approximations are used to simplify a hyperplane arrangement representation of the obstacle-free space. Polynomial zonotopes and hybrid zonotopes have been proposed for use as exact representations of the obstacle-free space [34], [35]. In both cases, MPC solution times were reduced as compared to more traditional obstacle avoiding MPC formulations when using general-purpose solvers.

This paper leverages the properties of an advanced set representation within a solution methodology for obstacle avoidance MPC formulations. Hybrid zonotopes are used to represent the obstacle-free region of the space and paired with a custom mixed-integer quadratic program (MIQP) solver tailored to exploit properties of hybrid zonotopes. The MIQP solver implements a new branch-and-bound algorithm that uses a notion of constraint region "reachability" to greatly reduce the number of infeasible branches that are generated. This solver generates a series of quadratic program (QP) sub-problems, which are solved with a multi-stage primal-dual interior point method that exploits the box constraint structure of hybrid zonotope inequality constraints to reduce the number of Cholesky factorizations that need to be computed. Obstacle-free spaces made up of repeating zonotopes can be represented as hybrid zonotopes with structure that is especially exploitable within the QP sub-problems. Numerical results demonstrate that the proposed MIQP solver often outperforms commercial solvers and halfspace representations (H-reps) of obstacle avoidance constraints.

II. Preliminaries

A. Notation

Unless otherwise stated, scalars are denoted by lowercase letters, vectors by boldface lowercase letters, matrices by uppercase letters, and sets by calligraphic letters. $\mathbf{0} = \begin{bmatrix}0 & \cdots & 0\end{bmatrix}^T$ and $\mathbf{1} = \begin{bmatrix}1 & \cdots & 1\end{bmatrix}^T$ denote vectors consisting entirely of zeroes and ones, respectively, of appropriate dimensions. I denotes the identity matrix.

B. MPC Formulation

This paper considers the following MPC formulation:

$$\min_{\mathbf{x}_k,\mathbf{u_k}} \sum_{k=0}^{N-1} \left[(\mathbf{x}_k - \mathbf{x}_k^r)^T Q_k (\mathbf{x}_k - \mathbf{x}_k^r) + \mathbf{u}_k^T R_k \mathbf{u}_k\right] + (\mathbf{x}_N - \mathbf{x}_N^r)^T Q_N (\mathbf{x}_N - \mathbf{x}_N^r) \ , \tag{1a}$$

$$\text{s.t. } \forall k \in \mathcal{K} = \{0, \cdots, N-1\} \ :$$

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \ , \tag{1b}$$

$$\mathbf{y}_k = H\mathbf{x}_k, \ \mathbf{y}_N = H\mathbf{x}_N, \ \mathbf{x}_0 = \mathbf{x}(0) \ , \tag{1c}$$

$$\mathbf{x}_k, \mathbf{x}_k^r \in \mathcal{X}, \ \mathbf{x}_N, \mathbf{x}_N^r \in \mathcal{X}_N, \ \mathbf{u}_k \in \mathcal{U} \ , \tag{1d}$$

$$\mathbf{y}_k, \mathbf{y}_N \in \mathcal{F} \ . \tag{1e}$$

Here, $\mathbf{x}_k$, $\mathbf{x}_k^r$, $\mathbf{u}_k$, and $\mathbf{y}_k$ are the AV state, reference, input, and position, respectively, associated with time step k. A and B are the linear dynamics matrices and H extracts the AV position from $\mathbf{x}_k$. $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ are the feasible state and input sets, respectively, $\mathcal{X}_N \subseteq \mathbb{R}^{n_x}$ is the terminal constraint set, and $\mathcal{F} \subset \mathbb{R}^{n_p}$ is the non-convex obstacle-free space. Eq. (1) is solved over a receding horizon, such that the current state $\mathbf{x}_0$ is provided to the MPC controller at each update time and the optimal input for the first step $\mathbf{u}_0$ is applied to the system.

Let $\mathcal{O} \subset \mathbb{R}^{n_p}$ represent the obstacle space and $\mathcal{V} \subset \mathbb{R}^{n_p}$ the feasible position domain (e.g., the limits of the "map" in which the AV operates). The following assumptions are used throughout the rest of this paper.

Assumption 1: Q_k, R_k, and Q_N are diagonal $\forall k \in \mathcal{K}$.

Assumption 2: $\mathcal{X}$, $\mathcal{U}$, $\mathcal{X}_N$, and $\mathcal{V}$ are polytopes.

Assumption 3: $\mathcal{O}$ is a union of polytopes.

Assumption 1 is required for the structural exploitation of the hybrid zonotope constraints described in Sec. III-B.2. Diagonal cost matrices are common in practice and can often be achieved under a change of variables if necessary. Assumptions 2 and 3 are typical for MPC formulations and obstacle avoidance problems.

C. Common Mixed-Integer Obstacle Avoidance Constraint Representations

By Assumption 3, the obstacle space can be expressed using H-rep by

$$\mathcal{O} = \bigcup_{j=1}^{n_O} \{\mathbf{y} \mid A_{Oj}\mathbf{y} \leq \mathbf{b}_{Oj}\} \ . \tag{2}$$

The obstacle-free space is $\mathcal{F} = \mathcal{V} \setminus \mathcal{O}$. By Assumptions 2 and 3, a convex partition of $\mathcal{F}$ can be constructed such that

$$\mathcal{F} = \bigcup_{j=1}^{n_F} \{\mathbf{y} \mid A_{Fj}\mathbf{y} \leq \mathbf{b}_{Fj}\} \ . \tag{3}$$

One common method used to embed $\mathcal{F}$ in a mixed-integer program is hyperplane arrangements [2], [20]. Using the "Big-M" method [36] to capture conditional logic with scalar $M \gg 1$, a hyperplane arrangement representation of $\mathcal{F}$ is

$$\mathcal{F} = \left\{ \mathbf{y} \left| \begin{array}{c} A_{Oj}\mathbf{y} \geq \mathbf{b}_{Oj} - MI_{n_j}\mathbf{d}_{Oj}, \ \mathbf{1}^T\mathbf{d}_{Oj} \leq n_j - 1 \\ \mathbf{d}_{Oj} \in \{0,1\}^{n_j}, \ j \in \{1, \cdots, n_O\} \end{array} \right. \right\} , \tag{4}$$

where n_O is the number of obstacles and n_j is the number of hyperplanes associated with the j^{th} obstacle.

Alternatively, the union of H-rep polytopes in (3) can be described using the Big-M method [20] as

$$\mathcal{F} = \left\{ \mathbf{y} \left| \begin{array}{c} A_{Fj}\mathbf{y} + M\left(\sum_{i=1}^{n_F} \delta_{ji} d_{Fi}\right)\mathbf{1} \leq \mathbf{b}_{Fj} + M\mathbf{1} \\ \mathbf{1}^T\mathbf{d}_F = 1 \ , \mathbf{d}_F \in \{0,1\}^{n_F}, \ j \in \{1, \cdots, n_F\} \end{array} \right. \right\} , \tag{5}$$

where δ_{ji} is the Kronecker delta. The elements of the binary variable vector $\mathbf{d}_F$ correspond to polytopes rather than hyperplanes. $\mathbf{1}^T\mathbf{d}_F = 1$ is a "choose one" constraint.

D. Hybrid Zonotope Obstacle Avoidance Constraints

Next, the *zonotope*, *constrained zonotope*, and *hybrid zonotope* set representations are briefly reviewed. A set $\mathcal{Z} \subset \mathbb{R}^n$ is a zonotope if $\exists \ G_c \in \mathbb{R}^{n \times n_g}$, $\mathbf{c} \in \mathbb{R}^n$ such that

$$\mathcal{Z} = \{G_c\boldsymbol{\xi}_c + \mathbf{c} \mid \boldsymbol{\xi}_c \in \mathcal{B}_\infty^{n_g}\} \ , \tag{6}$$

where $\mathcal{B}_\infty^{n_g} = \{\xi_c \in \mathbb{R}^{n_g} \mid \|\xi_c\|_\infty \leq 1\}$ is the infinity-norm ball. Zonotopes are convex, centrally symmetric sets [37].

A set $\mathcal{Z}_C \subset \mathbb{R}^n$ is a constrained zonotope if $\exists\ G_c \in \mathbb{R}^{n \times n_g}$, $\mathbf{c} \in \mathbb{R}^n$, $A_c \in \mathbb{R}^{n_c \times n_g}$, $\mathbf{b} \in \mathbb{R}^{n_c}$ such that

$$\mathcal{Z_C} = \{G_c \boldsymbol{\xi}_c + \mathbf{c} \mid \boldsymbol{\xi}_c \in \mathcal{B}_\infty^{n_g},\ A_c \boldsymbol{\xi}_c = \mathbf{b}\}\ . \tag{7}$$

Constrained zonotopes can represent any polytope and can be constructed from a halfspace representation [30, Thm 1].

Hybrid zonotopes extend (7) by including binary factors $\boldsymbol{\xi}_b$. A set $\mathcal{Z_H} \subset \mathbb{R}^n$ is a hybrid zonotope if in addition to G_c, $\mathbf{c}$, A_c, and $\mathbf{b}$, $\exists\ G_b \in \mathbb{R}^{n \times n_b}$, $A_b \in \mathbb{R}^{n_c \times n_b}$ such that

$$\mathcal{Z_H} = \left\{ \begin{bmatrix} G_c & G_b \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_c \\ \boldsymbol{\xi}_b \end{bmatrix} + \mathbf{c} \left| \begin{array}{c} \begin{bmatrix} \boldsymbol{\xi}_c \\ \boldsymbol{\xi}_b \end{bmatrix} \in \mathcal{B}_\infty^{n_g} \times \{0,1\}^{n_b} \\ \begin{bmatrix} A_c & A_b \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_c \\ \boldsymbol{\xi}_b \end{bmatrix} = \mathbf{b} \end{array} \right. \right\} . \tag{8}$$

The factors $\boldsymbol{\xi}_b$ in this paper are constrained to the set $\{0,1\}^{n_b}$ rather than $\{-1,1\}^{n_b}$ as used in [32]. These forms are equivalent and one can easily convert between them with linear algebra manipulations. Analogously, constrained zonotopes for which $\boldsymbol{\xi}_c = \begin{bmatrix} \boldsymbol{\xi}_{cc}^T & \boldsymbol{\xi}_{cb}^T \end{bmatrix}^T$, $\boldsymbol{\xi}_{cc} \in [-1,1]^{n_{gc}}$, $\boldsymbol{\xi}_{cb} \in [0,1]^{n_{gb}}$, $n_{gc} + n_{gb} = n_g$, are equivalent to (7).

Hybrid zonotopes can represent unions of convex polytopes [32] and as such can be used to represent $\mathcal{F}$. In [35], the authors generate a hybrid zonotope representation of $\mathcal{F}$ by taking the complement of a hybrid zonotope representation of $\mathcal{O}$. In this paper, for the case of general polytopic obstacles, an algorithm for converting from a collection of polytopes in vertex representation (V-rep) to a single hybrid zonotope is used for its reduced complexity as compared to hybrid zonotope complements [38, Thm 5]. The V-rep partition of $\mathcal{F}$ is generated using the efficient Hertel and Mehlhorn algorithm [39], and the zonoLAB toolbox [40] is used to generate and represent the hybrid zonotope representation of $\mathcal{F}$. The resulting hybrid zonotope has dimensions $n_g = 2n_v$, $n_b = n_F$, and $n_c = n_v + 2$ with n_v the total number of vertices and n_F the number of polytopes in the V-rep partition of $\mathcal{F}$. In Sec. IV, a different method of constructing a hybrid zonotope representation of $\mathcal{F}$ is presented for occupancy grid obstacle avoidance, where $n_c = 1$, $n_b = n_F$, and $n_g = 2$.

E. Multi-Stage MIQP

Based on Assumptions 2 and 3, any of the obstacle avoidance constraint representations can be used to write Eq. (1) as the multi-stage MIQP

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{k=0}^{N} \frac{1}{2} \mathbf{z}_k^T P_k \mathbf{z}_k + \mathbf{q}_k^T \mathbf{z}_k\ , \tag{9a}$$

$$\text{s.t. } \mathbf{0} = C_k \mathbf{z}_k + D_{k+1} \mathbf{z}_{k+1} + \mathbf{c}_k,\ \forall k \in \mathcal{K}\ , \tag{9b}$$

$$G_k \mathbf{z}_k \leq \mathbf{w}_k,\ \forall k \in \mathcal{K} \cup N\ , \tag{9c}$$

with $\mathbf{z}_k = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{u}_k^T & \boldsymbol{\alpha}_{ck}^T & \boldsymbol{\alpha}_{bk}^T \end{bmatrix}^T$. Vectors $\boldsymbol{\alpha}_{ck}$ and $\boldsymbol{\alpha}_{bk}$ denote continuous and integer variables, respectively, needed to represent $\mathcal{F}$. $\boldsymbol{\alpha}_{ck}$ may also be used to represent slack variables. Eq. (9b) accounts for the system dynamics and initial condition constraints as shown in [41].

Algorithm 1 General branch-and-bound framework

```
Result: optimal solution $\mathbf{z}, j$ to MIP $j = f(\mathbf{z})$
1:  nodes, j ← root, $+\infty$
2:  **while** nodes **not** empty **do**
3:      **pop** node **from** nodes
4:      $j_{node}, \mathbf{z}_{node} \leftarrow$ **solve** node
5:      **if** $j_{node} > j$ **or** infeasible(node) **then continue**
6:      **else if** integer-feasible($\mathbf{z}_{node}$) **then**
7:          **if** $j_{node} < j$ **then**
8:              $j, \mathbf{z} \leftarrow j_{node}, \mathbf{z}_{node}$
9:              **prune** nodes($j_{node} > j$) **from** nodes
10:         **else continue**
11:     **else append** branch(node) **to** nodes
```

To represent (1e) in (9), hyperplane arrangements (4) result in $\boldsymbol{\alpha}_{bk} \in \mathbb{Z}^{n_{hpb}}$ and introduce $3n_{hpb}$ inequality constraints in (9c), where $n_{hpb} = \sum_{j=1}^{n_O} n_j$. H-rep (5) results in $\boldsymbol{\alpha}_{bk} \in \mathbb{Z}^{n_F}$ and introduces $2n_F + \sum_{j=1}^{n_F} n_{Hj}$ inequality constraints in (9c), where $\mathbf{b}_{Fj} \in \mathbb{R}^{n_{Hj}}$. Hybrid zonotopes (8) result in $\boldsymbol{\alpha}_{ck} \in \mathbb{R}^{n_g}$, $\boldsymbol{\alpha}_{bk} \in \mathbb{Z}^{n_b}$, introduce $n_c + n_p$ equality constraints in (9b), and introduce $2(n_g + n_b)$ inequality constraints in (9c). In all cases, (9c) includes the binary integer variable constraint $\mathbf{0} \leq \boldsymbol{\alpha}_{bk} \leq \mathbf{1}$.

In contrast with hyperplane arrangements and unions of H-rep polytopes, hybrid zonotopes do not employ the Big-M method. This can improve the numerical conditioning of (9) and eliminates the challenge of needing to select an appropriate value for M [36]. Furthermore, all of the obstacle avoidance inequality constraints using a hybrid zonotope MPC formulation are box constraints (i.e., of the form $\mathbf{l} \leq \mathbf{z}_k \leq \mathbf{u}$). This structure can be exploited as described below.

III. MIQP Solver for Obstacle Avoidance MPC

This section presents a solution strategy for MIQPs arising in obstacle avoidance MPC formulations. This strategy is designed to exploit the structure of a hybrid zonotope representation of the obstacle-free space $\mathcal{F}$. Sec. III-A describes a novel branch-and-bound mixed-integer solver that branches along "reachable" combinations of polytopic regions of $\mathcal{F}$ to reduce the complexity of the mixed-integer search. Sec. III-B describes an interior point QP solver that efficiently computes Newton steps by exploiting the box constraint form of the hybrid zonotope inequality constraints.

A. Mixed-Integer Solver

A branch-and-bound mixed-integer solver is used to find the optimal selection of binary variables. Branch-and-bound methods solve mixed-integer programs to global optimality by searching along a tree of "relaxed" convex sub-problems. The root "node" of the tree is constructed by turning all integer variables into continuous ones and restricting them to the appropriate domain (e.g., $z_i \in \{0,1\} \rightarrow z_i \in [0,1]$). Subsequent nodes are created by applying constraints to the relaxed integer variables (e.g., $z_i = 0$ and $z_i = 1$). The optimal objective for a node is a lower bound on the objective of that node's children. Algorithm 1 is a general branch-and-bound framework with $\min_{\mathbf{z}} f(\mathbf{z})$ a generic MIP [25].

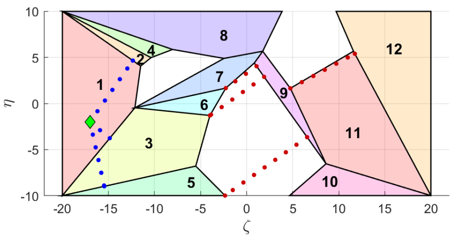

Fig. 1. Obstacle avoidance map with convex sub-regions displayed around white obstacles. The current AV position is displayed using a green diamond and the blue dots show point-to-region reachability for regions 2, 3, and 5. The red dots show region-to-region reachability for regions 3, 5, 6, and 12 from region 9. Referencing Definition 1, the red and blue dots are spaced at a an interval of d_{max} for time steps less than k. At time step k, the spacing is $d_r - (k-1)d_{max}$.

Branch-and-bound algorithm performance can vary significantly based on how nodes are selected and how branches are generated (lines 3 and 11 in Algorithm 1 respectively). A review of common node selection and branching strategies is presented in [42]. The branch-and-bound algorithm proposed here implements a branching strategy for obstacle avoidance problems for which the obstacle-free space is partitioned into convex sub-regions as in the case of halfspace and hybrid zonotope representations. At each time step of the MPC problem, the position of the AV is constrained to lie in one of these sub-regions. Integer variables are used to select which sub-region constraint is active at each time step.

The proposed "reachability"-constrained branching operation significantly reduces the number of infeasible branches that are generated, simplifying the mixed-integer search. This novel branching operation is then paired with branching variable selection and node selection algorithms similar to existing approaches in the literature that reduce the total number of nodes that need to be solved.

1) Branching Constraints: To reduce the size of the branch-and-bound integer variable tree, the branching operation is constrained to only generate sequences of convex sub-region selections that are "reachable" from each other and from the AV's starting position. We define this notion of reachability as follows.

Definition 1: A region r_i is k steps reachable from a point $\mathbf{y}_0$ (or another region r_j) if $\lfloor (d_r/d_{max}) \rfloor \geq k$, where d_r is the length of the shortest line segment from $\mathbf{y}_0$ (or r_j) to r_i and d_{max} is the maximum distance that the AV can travel in a single time step of the MPC problem.

This is illustrated in Fig. 1, where region 3 is 2 steps reachable from the AV's current position (indicated by the green diamond), and region 3 is 5 steps reachable from region 9 and vice-versa. Algorithm 2 shows how the set of reachable regions is computed. Lines 6 and 9 require the solution of a QP. In this implementation, the "region" case QPs are solved offline because the obstacle map is assumed to be fully known and not changing, while the "point" case QPs are solved at the start of every MPC iteration. Reachability is only re-computed for regions that were reachable within the MPC horizon in the previous iteration and regions that are 1 step reachable from those regions.

Algorithm 3 gives the branching operation (line 11 in Algorithm 1) for this reachability-constrained branch-and-bound algorithm. Line 6 in Algorithm 3 is implemented by adding the equality constraint $\mathbf{s}_r^T \mathbf{z}_{k_b} - 1 = 0$ to $\mathbf{c}_{k_b}$ and C_{k_b} or D_{k_b} as appropriate in the QP relaxation of (9). $\mathbf{s}_r$ selects the binary variable corresponding to the sub-region constraint and has i^{th} element $s_{ri} = \delta_{ri}$ with δ_{ri} the Kronecker delta.

Algorithm 2 Function to compute reachable regions

```
Input: input "region" or "point"
       number of time steps $n_t$
       max distance AV can travel in a time step $d_{max}$
Result: list of reachable regions
 1: function REACHABLE(input, $n_t$, $d_{max}$)
 2:     regions ← empty
 3:     for region$_i$ in all regions do
 4:         switch input do
 5:             case region
 6:                 $\mathbf{y}_0^*, \mathbf{y}_i^* \leftarrow \text{argmin}_{\mathbf{y}_0, \mathbf{y}_i} \left( ||\mathbf{y}_0 - \mathbf{y}_i||_2^2 \mid \mathbf{y}_0 \in \text{region}, \mathbf{y}_i \in \text{region}_i \right)$
 7:             case point
 8:                 $\mathbf{y}_0^* \leftarrow$ point
 9:                 $\mathbf{y}_i^* \leftarrow \text{argmin}_{\mathbf{y}_i} \left( ||\mathbf{y}_0^* - \mathbf{y}_i||_2^2 \mid \mathbf{y}_i \in \text{region}_i \right)$
10:         distance ← $||\mathbf{y}_0^* - \mathbf{y}_i^*||_2$
11:         if distance $\leq d_{max} \cdot n_t$ then
12:             append region$_i$ to regions
13:     return regions
```

Algorithm 3 Reachability-constrained branching

```
Input: branching node and time step (node$_b$, $k_b$)
       current AV position $\mathbf{y}_0$
       max distance AV can travel in a time step $d_{max}$
Result: updated nodes list "nodes"
1: regions ← REACHABLE($\mathbf{y}_0$, $k_b$, $d_{max}$)
2: for region, $k$ ← constraints(node$_b$) do
3:     regions ← regions \ [all regions \ REACHABLE(region, $|k - k_b|$, $d_{max}$)]
4: for region in regions do
5:     node ← node$_b$
6:     append (region, $k_b$) to constraints(node)
7:     append node to nodes
```

2) Branching Variable Selection: In this formulation, branching variable selection corresponds to selecting an MPC time step at which to impose sub-region constraints. Algorithm 4 does this using logic similar to that proposed in [27]. The branching node may be identified as integer feasible, triggering the corresponding conditional in Algorithm 1. An H-rep description of the obstacle-free space (5) is used to check whether $\mathbf{y}_k$ belongs to region$_k$. H_z is defined such that $\mathbf{y}_k = H_z \mathbf{z}_k$ referencing (1) and (9).

3) Node Selection: Algorithm 5 gives a *depth-then-best* node selection strategy [43]. A tie-breaker is implemented based on the node solution $\mathbf{z}_{node}$. $z_{node_{br}}$ denotes the value of the relaxed binary variable corresponding to the most recently applied constraint pair (region, k_b).

Algorithm 4 Collision-detecting branching variable selection

```
Input: solved node "node$_b$"
Result: branching time step $k_b$ or node$_b$ is integer feasible
 1: $k$, cons ← 0, empty
 2: while $k \le N$ do
 3:     $\mathbf{z}_k$ ← node$_b$ solution at time step $k$
 4:     if k not in constraints(node$_b$) then
 5:         find region$_k$ such that $H_z \mathbf{z}_k$ in region$_k$
 6:         if region$_k$ is empty then break
 7:         append (region$_k$, $k$) to cons
 8:     $k \leftarrow k+1$
 9: if $k \le N$ then $k_b \leftarrow k$
10: else
11:     append cons to constraints(node$_b$)
12:     integer-feasible(node$_b$) ← true
```

B. QP Solver

A custom implementation of Mehrotra's primal-dual interior point method [44], [45] is used to solve the QP sub-problems generated by the mixed-integer solver (Sec. III-A). As described in [46] and [41], MPC QPs have a multi-stage structure that can be exploited in interior point solvers. Interior point methods solve a large linear system to compute the Newton step at each iteration. This is typically the costliest operation in the solver. Multi-stage algorithms break this linear system into a series of smaller linear systems that can be solved much more efficiently. With respect to the QP sub-problems, a contribution of this paper is the recognition that the Newton step solution strategy provided in [41] can be accelerated by using constrained zonotopes to represent any polytopic constraints.

1) Efficient Solution of Multi-Stage QPs: Here, the procedure to solve for the Newton step, following [41], is described. For conciseness, Mehrotra's algorithm is not presented here, and the interested reader is directed to [45]. Consider a QP relaxation of (9) with $\boldsymbol{\alpha}_{bk}$ relaxed from $\mathbb{Z}^{n_b} \to \mathbb{R}^{n_b}$. In primal-dual interior point methods, the Newton step is given by the solution to the linear system

$$\begin{bmatrix} P & C^T & G^T & 0 \\ C & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta \mathbf{z} \\ \Delta \boldsymbol{\nu} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_C \\ \mathbf{r}_E \\ \mathbf{r}_I \\ \mathbf{r}_S \end{bmatrix} , \tag{10}$$

where

$$P = \text{blkdiag}\left(\begin{bmatrix} P_0 & \cdots & P_N \end{bmatrix}\right) , \tag{11a}$$

$$C = \begin{bmatrix} C_0 & D_1 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & C_{N-1} & D_N \end{bmatrix} , \tag{11b}$$

$$G = \text{blkdiag}\left(\begin{bmatrix} G_0 & \cdots & G_N \end{bmatrix}\right) , \tag{11c}$$

$$\mathbf{r}_C = P\mathbf{z} + \mathbf{q} + C^T \boldsymbol{\nu} + G^T \boldsymbol{\lambda} , \tag{11d}$$

$$\mathbf{r}_E = C\mathbf{z} + \mathbf{c} , \tag{11e}$$

$$\mathbf{r}_I = (G\mathbf{z} - \mathbf{w}) + \mathbf{s} , \tag{11f}$$

$$\mathbf{r}_S = S\boldsymbol{\lambda} . \tag{11g}$$

Algorithm 5 *Depth-then-best* node selection

```
Input: nodes list "nodes"
Result: node selected for branching node$_b$
1: if $j = +\infty$ then
2:     depth-nodes ← $\text{argmax}_{nodes}$[depth(nodes)]
3:     node$_b$ ← BestNode(depth-nodes)
4: else node$_b$ ← BestNode(nodes)
5: function BestNode(nodes)
6:     best-nodes ← $\text{argmin}_{nodes}$ [$j_{node}$(nodes)]
7:     return $\text{argmax}_{best-nodes}$[$z_{node_{br}}$(best-nodes)]
```

Here, $\mathbf{z}$ are the primal variables, $\boldsymbol{\nu}$ and $\boldsymbol{\lambda}$ are the dual variables for the equality and inequality constraints, respectively, and $\mathbf{s}$ are the inequality constraint slack variables. S and Λ are diagonal matrices constructed from $\mathbf{s}$ and $\boldsymbol{\lambda}$.

Solving (10) directly generally requires an LU decomposition, which has a cost of $(2/3)n^3$ flops where n is the dimension of the linear system matrix. Although sparse linear algebra techniques can be used to reduce the computational cost, solving (10) remains the primary computational challenge in primal-dual interior point methods.

For multi-stage problems, (10) can be written concisely as

$$Y \Delta \boldsymbol{\nu} = \boldsymbol{\beta} , \tag{12}$$

with

$$Y = C \Phi^{-1} C^T \in \mathcal{S}_{>0} , \tag{13a}$$

$$\boldsymbol{\beta} = \mathbf{r}_E - C\Phi^{-1}\left(\mathbf{r}_C + G^T S^{-1} \Lambda \mathbf{r}_I - G^T S^{-1} \mathbf{r}_S\right) , \tag{13b}$$

and $\mathcal{S}_{>0}$ denoting the set of positive definite matrices. Y has block-banded structure given by

$$Y_{k,k} = C_{k-1}\Phi_{k-1}^{-1}C_{k-1}^T + D_k \Phi_k^{-1} D_k^T \quad \forall k \in \{1, \cdots, N\} , \tag{14a}$$

$$Y_{k,k+1} = D_k \Phi_k^{-1} C_k^T \ \forall k \in \{1, \cdots, N-1\} , \tag{14b}$$

$$Y_{k+1,k} = Y_{k,k+1}^T ,\ Y_{kj} = 0 \ \forall j \notin \{k-1, k, k+1\} , \tag{14c}$$

and $\Phi = \text{blkdiag}\left(\begin{bmatrix} \Phi_0 & \cdots & \Phi_N \end{bmatrix}\right)$ with blocks

$$\Phi_k = P_k + G_k^T S_k^{-1} \Lambda_k G_k . \tag{15}$$

Matrices S_k and Λ_k correspond to the inequality constraints at time step k.

To construct the Y_{kj} matrices, it is necessary to first compute Cholesky factorizations of the $(N+1)$ Φ_k matrices. Y_{kj} can then be constructed by substitution. Each Φ_k factorization has a cost of $(1/3)n_{in_k}^3$ flops where n_{in_k} is the number of inequality constraints for the kth MPC time step.

Once the Y matrix has been constructed, it is factorized to solve (12). This requires N Cholesky factorizations at a cost of $(1/3)n_{eq_k}^3$ flops where n_{eq_k} is the number of equality constraints in the C_k matrix.

2) Efficient QP Solution with Constrained Zonotope Constraints: When using hybrid zonotopes to represent the obstacle-free space, QP relaxations of (9) have constrained zonotope constraints as defined in (7). Constraints of the form $\mathbf{y}_k \in \mathcal{Z}_C$ with $\mathcal{Z}_C$ a constrained zonotope can be exploited when computing the Cholesky factorizations of the

Φ_k matrices during the Newton step computation. As pointed out in [41], the Φ_k matrices are diagonal for the case of diagonal quadratic cost P_k and box constraints G_k. For the constraint $\mathbf{y}_k \in \mathcal{Z}_\mathcal{C}$, the inequality constraints are always box constraints.

By Assumptions 1 and 2, the MPC formulation (1) can be constructed so that Φ_k is diagonal using a hybrid zonotope representation of $\mathcal{F}$. If $\mathcal{X}$ and $\mathcal{U}$ are not already box constraints, then they can be represented as constrained zonotopes to achieve diagonal Φ_k. In the QP solver implemented here, the Φ_k are inverted when needed rather than computing their Cholesky factors.

The computational penalty in the QP solver for using constrained zonotopes is the inclusion of additional equality constraints and primal optimization variables in (9) as described in Sec. II-E. The equality constraints will increase the computational cost of factorizing the $Y_{k,k}$ matrices. The $\boldsymbol{\xi}_{ck}$ and relaxed $\boldsymbol{\xi}_{bk}$ will increase the overall size of the linear system in (10). In the multi-stage formulation however, the operations with the worst scaling (Cholesky factorizations at $(1/3)n^3$ flops) are in terms of n_{in_k} and n_{eq_k}. As such, the addition of the $\boldsymbol{\xi}_{ck}$ and $\boldsymbol{\xi}_{bk}$ variables does not significantly affect the QP solution time for large problem sizes.

IV. Repeated Zonotope Obstacle Maps with Application to Automated Driving

For the QP solver described in Sec. III-B, the computational benefits to using a hybrid zonotope obstacle-free space representation are most significant when the number of hybrid zonotope equality constraints is minimal. This is the case for obstacle maps where the obstacle-free space $\mathcal{F}$ can be represented as a repeating zonotope.

Consider a template zonotope $\mathcal{Z}$ as defined in (6) with $G_c = G_0$ and $\mathbf{c} = \mathbf{0}$. Referencing (8), an obstacle-free space $\mathcal{F}$ consisting of repeated instances of this zonotope is given by the hybrid zonotope with the following parameters:

$$G_c = G_0,\ G_b = \begin{bmatrix}\mathbf{c}_{b\,1} & \cdots & \mathbf{c}_{b\,n_g}\end{bmatrix},\ \mathbf{c} = 0, \tag{16a}$$
$$A_c = \mathbf{0}^T,\ A_b = \mathbf{1}^T,\ \mathbf{b} = 1\ . \tag{16b}$$

The $\mathbf{c}_{bi}$ denote the centers of each repeated zonotope.

The hybrid zonotope described in (16) has only a single equality constraint. Referring back to Sec. III-B.2, this implies that the operations with the worst scaling in the QP solver will have a small, fixed increase in computational cost due to the hybrid zonotope constraints that is independent of the map complexity.

Repeated zonotope maps arise naturally in automated driving applications with occupancy grids [47]. The template zonotope in this case has the form $G_0 = \text{diag}(\begin{bmatrix}d_\zeta/2 & d_\eta/2\end{bmatrix})$ with d_ζ and d_η giving the dimensions of a cell in the occupancy grid. This technique could be used to efficiently represent complex roadway traffic patterns [48].

V. Numerical Results

This section compares the proposed obstacle avoidance MPC solution strategy to several general-purpose MIQP solvers in simulation. These examples additionally demonstrate the advantages of using a hybrid zonotope representation of the obstacle-free space as compared to a halfspace representation (see Sec. II-C). Hyperplane arrangements are not considered because they cannot be accommodated by the proposed mixed-integer solver. The complexity of hyperplane arrangements is discussed in Sec. II-E. These representations have general polytopic inequality constraints and would not be amenable to acceleration in the QP solver.

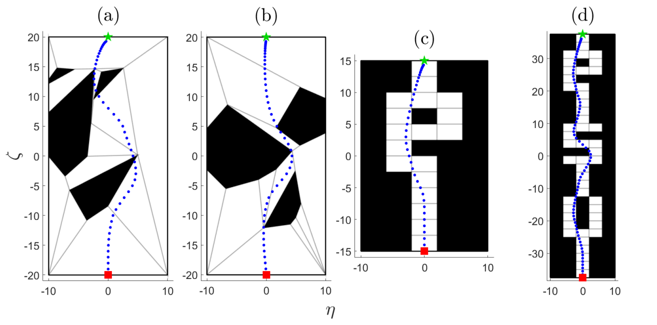

Fig. 2. Simulation results for a 10 step horizon. The red square denotes the start point and the green star is the reference $\mathbf{x}^r$. The discrete time trajectory of the double integrator model is given by the blue dots. The black regions are obstacles and the gray lines show the convex partition of the free space. Map (a) has 14 sub-regions, (b) has 12, (c) has 18, and (d) has 48.

An AV with constrained double-integrator motion dynamics is simulated. The MPC formulation (1) is specified with

$$\mathbf{x}_k = \begin{bmatrix}\zeta_k & \dot{\zeta}_k & \eta_k & \dot{\eta}_k\end{bmatrix}^T,\ \mathbf{u}_k = \begin{bmatrix}\ddot{\zeta} & \ddot{\eta}\end{bmatrix}^T, \tag{17a}$$
$$A = \begin{bmatrix}1 & \Delta t & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & \Delta t\\ 0 & 0 & 0 & 1\end{bmatrix},\ B = \begin{bmatrix}0 & 0\\ \Delta t & 0\\ 0 & 0\\ 0 & \Delta t\end{bmatrix}, \tag{17b}$$
$$\mathcal{X} = \mathcal{X}_N = \left\{\mathbf{x}_k \middle| \begin{bmatrix}-100 & -1 & -100 & -1\end{bmatrix}^T \leq \mathbf{x}_k \leq \begin{bmatrix}100 & 1 & 100 & 1\end{bmatrix}^T\right\}, \tag{17c}$$
$$\mathcal{U} = \left\{\mathbf{u}_k \middle| \begin{bmatrix}-1 & -1\end{bmatrix}^T \leq \mathbf{u}_k \leq \begin{bmatrix}1 & 1\end{bmatrix}^T\right\}, \tag{17d}$$
$$\mathcal{V} = \left\{\mathbf{y}_k \middle| \begin{bmatrix}\zeta_- & \eta_-\end{bmatrix}^T \leq \mathbf{y}_k \leq \begin{bmatrix}\zeta_+ & \eta_+\end{bmatrix}^T\right\}, \tag{17e}$$
$$Q_k = 0_{4\times4},\ R_k = 10I_4,\ Q_N = \text{diag}([10\ 0\ 10\ 0]), \tag{17f}$$

ζ_-, η_-, ζ_+, and η_+ are given by the map bounds in Fig. 2. Position constraints in (17c) ensure that the Φ_k matrices in Sec. III-B are full rank. A time step of $\Delta t = 1$ is used. The reference is taken to be a fixed value $\mathbf{x}_k^r = \mathbf{x}^r$. To ensure feasibility of the MPC optimization problem, softening is applied to the obstacle avoidance constraints (1e) with a slack cost of $1e6 \cdot \sigma_i^2$ for each required slack variable σ_i.

Simulations using two types of obstacle maps are given. The first case (maps (a) and (b)) considers randomly generated polytopic obstacles with a convex partitioning of $\mathcal{F}$ as described in Sec. II-C. The second case (maps (c) and (d)) considers repeated zonotope maps. In both cases, the full map is included in the MPC constraints. MPC horizons of $N = 5$, 10, and 15 time steps are examined. Simulation results for the 10 time step case are shown in Fig. 2.

The solution time of the proposed MIQP solver is compared to general-purpose commercial solvers Gurobi [49]

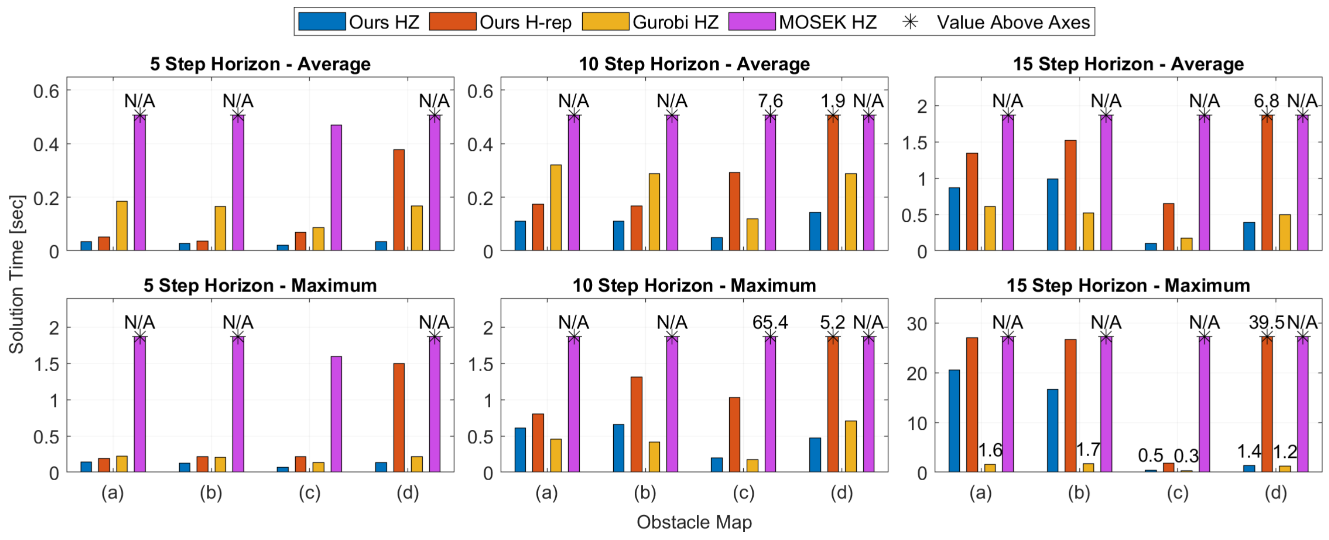

Fig. 3. MPC solution times for obstacle maps given in Fig. 2.

and MOSEK [50]. All three solvers were given the same MIQP using a hybrid zonotope (HZ) representation of $\mathcal{F}$ and configured to find the global optimum. Hybrid zonotope representations are compared to H-rep using the proposed MIQP solver. Fig. 3 plots the average and maximum solution times for each map, MPC horizon, and solution approach.

The proposed branch-and-bound solver (Sec. III-A) is implemented in MATLAB and modified from the open source miOSQP solver [43]. The proposed QP solver (Sec. III-B) is implemented in C++ using the Eigen linear algebra library [51]. Simulations were conducted on a desktop computer with a 2.1 GHz Intel i7 processor and 16 GB of RAM. All solvers were configured not to use parallelization. MPC solution times were averaged over 5 trials. A maximum of 100 seconds was allowed for an individual MPC solution. Cases where this limit was violated are marked as “N/A”.

When using the proposed MIQP solver, hybrid zonotope representations of the obstacle-free space consistently result in reduced solution times as compared to H-rep. This is most pronounced for repeated zonotope maps ((c) and (d)) due to the efficiency of the hybrid zonotope representation in those cases, as discussed in Sec. IV. As an example, for the 10 step horizon case, the average QP solution time is 1.7 ms for (c) and 3.3 ms for (d) using hybrid zonotopes. For H-rep, those times are 19.1 ms and 115.8 ms, respectively.

The proposed MIQP solver finds the optimal solution to the MPC obstacle avoidance problem faster than MOSEK for all tested cases. The solver is faster than Gurobi for the shorter MPC horizons but becomes slower for larger MPC horizons. This performance discrepancy can be attributed in part to the fact that the proposed branch-and-bound solver is implemented in MATLAB while Gurobi is entirely implemented in C. For maps (c) and (d) in these examples, 63-82% of the overall MIQP solution time was spent in the MATLAB-based branch-and-bound solver. Solution times for the proposed solver increase by up to an order of magnitude when moving from a 10 step horizon to a 15 step horizon. Future work will seek to improve scalability via parallelization, a more efficient code implementation using C++, and potentially further exploitation of the sub-region reachability constraints.

VI. Conclusion

This paper leverages hybrid zonotopes as a non-convex constraint representation for use in obstacle avoidance MPC problems and develops an MIQP solver to exploit the structure of these constraints. The proposed MIQP solver achieves better solution times than highly optimized commercial solvers for most of the test cases considered, particularly at shorter prediction horizons.

The proposed approach can be used to efficiently solve motion planning problems for autonomous vehicles without being subject to poor quality approximate or local solutions. These techniques can be applied to both general polytopic obstacle maps and to repeated zonotope maps (i.e., occupancy grids), which are commonly encountered in automated driving applications. Future work will include implementing the proposed solver fully in C++ and adding support for parallelization. Additionally, the applicability of this solver to automated driving problems will be validated in experiments or high-fidelity simulations using a representative embedded processor.

References

[1] A. Khairnar, S. Sivashangaran, and A. Eskandarian, “A comparison of motion planning methods for autonomous ground vehicle exploration and search,” in *ASME International Mechanical Engineering Congress and Exposition*, vol. 87639, p. V006T07A069, 2023.

[2] A. Richards and J. P. How, “Robust variable horizon model predictive control for vehicle maneuvering,” *International Journal of Robust and Nonlinear Control*, vol. 16, no. 7, pp. 333–351, 2006.

[3] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1917–1922, 2012.

[4] B. Alrifaee, J. Maczijewski, and D. Abel, “Sequential convex programming MPC for dynamic vehicle collision avoidance,” in *2017 IEEE Conference on Control Technology and Applications*, pp. 2202–2207, 2017.

[5] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, “Model predictive control of swarms of spacecraft using sequential convex programming,” *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[6] Q. T. Dinh and M. Diehl, “Local convergence of sequential convex programming for nonconvex optimization,” in *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*, pp. 93–102, 2010.

[7] L. Blackmore, M. Ono, and B. C. Williams, “Chance-constrained optimal path planning with obstacles,” *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[8] V. V. Naik and A. Bemporad, “Embedded mixed-integer quadratic optimization using accelerated dual gradient projection,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10723–10728, 2017.

[9] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone, “Learning mixed-integer convex optimization strategies for robot planning and control,” in *2020 59th IEEE Conference on Decision and Control*, pp. 1698–1705, 2020.

[10] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, “A simple effective heuristic for embedded mixed-integer quadratic programming,” *International journal of control*, vol. 93, no. 1, pp. 2–12, 2020.

[11] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Aggressive driving with model predictive path integral control,” in *2016 IEEE International Conference on Robotics and Automation*, pp. 1433–1440, 2016.

[12] G. Williams, A. Aldrich, and E. A. Theodorou, “Model predictive path integral control: From theory to parallel computation,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[13] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, “Robust model predictive path integral control: Analysis and performance guarantees,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[14] I. M. Balci, E. Bakolas, B. Vlahov, and E. A. Theodorou, “Constrained covariance steering based tube-MPPI,” in *2022 American Control Conference*, pp. 4197–4202, 2022.

[15] T. Brüdigam, M. Olbrich, D. Wollherr, and M. Leibold, “Stochastic model predictive control with a safety guarantee for automated driving,” *IEEE Transactions on Intelligent Vehicles*, 2021.

[16] S. Bogomolov, T. T. Johnson, D. M. Lopez, P. Musau, and P. Stankaitis, “Online reachability analysis and space convexification for autonomous racing,” in *Proceedings of the Fifth International Workshop on Formal Methods for Autonomous Systems*, pp. 95–112, 2023.

[17] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming,” in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pp. 109–124, 2015.

[18] V. A. Battagello, N. Y. Soma, and R. J. M. Afonso, “Trajectory planning with a dynamic obstacle clustering strategy using mixed-integer linear programming,” in *2021 American Control Conference*, pp. 3339–3344, 2021.

[19] H. Kim, S. H. Nair, and F. Borrelli, “Scalable multi-modal model predictive control via duality-based interaction predictions,” *arXiv preprint arXiv:2402.01116*, 2024.

[20] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, “Mixed-integer programming in motion planning,” *Annual Reviews in Control*, vol. 51, pp. 65–87, 2021.

[21] M. H. Maia and R. K. Galvao, “On the use of mixed-integer linear programming for predictive control with avoidance constraints,” *International Journal of Robust and Nonlinear Control*, vol. 19, no. 7, pp. 822–828, 2009.

[22] M. Kögel, M. Ibrahim, C. Kallies, and R. Findeisen, “Safe hierarchical model predictive control and planning for autonomous systems,” *International Journal of Robust and Nonlinear Control*, 2022.

[23] R. Deits and R. Tedrake, “Efficient mixed-integer planning for UAVs in cluttered environments,” in *2015 IEEE international conference on robotics and automation*, pp. 42–49, 2015.

[24] F. Stoican, T.-G. Nicu, and I. Prodan, “A mixed-integer MPC with polyhedral potential field cost for obstacle avoidance,” in *2022 American Control Conference*, pp. 2039–2044, 2022.

[25] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.

[26] A. Eele and A. Richards, “Path-planning with avoidance using nonlinear branch-and-bound optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 384–394, 2009.

[27] H. Fukushima, K. Kon, and F. Matsuno, “Model predictive formation control using branch-and-bound compatible with collision avoidance problems,” *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1308–1317, 2013.

[28] D. Strawser and B. Williams, “Approximate branch and bound for fast, risk-bound stochastic path planning,” in *2018 IEEE International Conference on Robotics and Automation*, pp. 7047–7054, 2018.

[29] M. Althoff, *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[30] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, “Constrained zonotopes: A new tool for set-based estimation and fault detection,” *Automatica*, vol. 69, pp. 126–136, 2016.

[31] N. Kochdumper and M. Althoff, “Sparse polynomial zonotopes: A novel set representation for reachability analysis,” *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2020.

[32] T. J. Bird, H. C. Pangborn, N. Jain, and J. P. Koeln, “Hybrid zonotopes: A new set representation for reachability analysis of mixed logical dynamical systems,” *Automatica*, vol. 154, p. 111107, 2023.

[33] D. Ioan, I. Prodan, F. Stoican, S. Olaru, and S.-I. Niculescu, “Complexity bounds for obstacle avoidance within a zonotopic framework,” in *2019 American Control Conference*, pp. 335–340, 2019.

[34] I. B. Nascimento, B. S. Rego, L. C. Pimenta, and G. V. Raffo, “NMPC strategy for safe robot navigation in unknown environments using polynomial zonotopes,” in *2023 62nd IEEE Conference on Decision and Control*, pp. 7100–7105, 2023.

[35] T. J. Bird and N. Jain, “Unions and complements of hybrid zonotopes,” *IEEE Control Systems Letters*, vol. 6, pp. 1778–1783, 2021.

[36] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese, “On mathematical programming with indicator constraints,” *Mathematical programming*, vol. 151, pp. 191–223, 2015.

[37] G. M. Ziegler, *Lectures on polytopes*, vol. 152. Springer Science & Business Media, 2012.

[38] J. A. Siefert, T. J. Bird, J. P. Koeln, N. Jain, and H. C. Pangborn, “Reachability analysis of nonlinear systems using hybrid zonotopes and functional decomposition,” *arXiv preprint arXiv:2304.06827*, 2024.

[39] J. O'Rourke, *Computational geometry in C*. Cambridge University Press, 1998.

[40] J. Koeln, T. J. Bird, J. Siefert, J. Ruths, H. Pangborn, and N. Jain, “zonoLAB: A MATLAB toolbox for set-based control systems analysis using hybrid zonotopes,” in *2024 American Control Conference*, pp. 2498–2505, 2024.

[41] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, “Efficient interior point methods for multistage problems arising in receding horizon control,” in *2012 IEEE 51st IEEE conference on decision and control*, pp. 668–674, 2012.

[42] M. Karamanov, *Branch and cut: an empirical study*. PhD thesis, Carnegie Mellon University, 2006.

[43] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, “Embedded mixed-integer quadratic optimization using the OSQP solver,” in *2018 European Control Conference*, pp. 1536–1541, 2018.

[44] S. Mehrotra, “On the implementation of a primal-dual interior point method,” *SIAM Journal on optimization*, vol. 2, no. 4, pp. 575–601, 1992.

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

[46] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.

[47] A. Elfes, “Using occupancy grids for mobile robot perception and navigation,” *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[48] S. Hoermann, M. Bach, and K. Dietmayer, “Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling,” in *2018 IEEE International Conference on Robotics and Automation*, pp. 2056–2063, 2018.

[49] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2023.

[50] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0*, 2019.

[51] G. Guennebaud, B. Jacob, *et al.*, “Eigen v3.” http://eigen.tuxfamily.org, 2010.