Combinatorial Disjunctive Constraints for Obstacle Avoidance in Path Planning

Raul Garcia[1], Illya V. Hicks[2] and Joey Huchette[3]

***Abstract*— We present a new approach for modeling avoidance constraints in 2D environments, in which waypoints are assigned to obstacle-free polyhedral regions. Constraints of this form are often formulated as mixed-integer programming (MIP) problems employing big-M techniques—however, these are generally not the strongest formulations possible with respect to the MIP's convex relaxation (so called *ideal* formulations), potentially resulting in larger computational burden. We instead model obstacle avoidance as combinatorial disjunctive constraints and leverage the *independent branching* scheme to construct small, ideal formulations. As our approach requires a biclique cover for an associated graph, we exploit the structure of this class of graphs to develop a fast subroutine for obtaining biclique covers in polynomial time. We also contribute an open-source Julia library named `ClutteredEnvPathOpt` to facilitate computational experiments of MIP formulations for obstacle avoidance. Experiments have shown our formulation is more compact and remains competitive on a number of instances compared with standard big-M techniques, for which solvers possess highly optimized procedures.**

I. INTRODUCTION

An essential task in many robotics applications is for an autonomous agent to find a path through a cluttered environment to safely arrive at a destination. Paths are often represented as a sequence of waypoints satisfying dynamic and obstacle avoidance constraints, and may or may not integrate time. Approaches to path planning can be broadly categorized into discrete searches on graphs or continuous optimization.

Discrete approaches typically utilize graphs to capture feasible paths among a set of waypoints within the environment [1], [2] or may construct a tree of possible paths given a successor set of available actions [3], [4], [5], [6]. Shortest paths between the start and goal positions can then computed via Dijkstra's algorithm or A^*. Obstacle avoidance is accomplished by appropriately prohibiting certain edges/nodes in the former or pruning the tree accordingly in the latter. While these approaches have worked well in certain instances, they tend to suffer from computational burden in complex environments and zig-zagging behavior [2], [3], [7], and they do not consider the optimality of the exact waypoint locations. Sampling-based algorithms such as Probabilistic Roadmaps [8] and Rapidly-exploring Random Trees [9] help relieve computational burden; however, their probabilistic completeness property implies sampling may need to grow large before a feasible path may be found.

On the other hand, continuous optimization avoids the challenges of exploring graphs and instead directly operates on the waypoint placements via continuous decision variables. This allows the construction of a more expressive objective function, and dynamic constraints can be employed to prevent infeasible or undesirable behavior. However, incorporating agent yaw and obstacle avoidance generally results in nonlinearity and/or nonconvexity, presenting challenges in computational efficiency, global optimality, and outright guaranteeing obstacle avoidance [10], [11].

The past two decades have seen a growing interest in mixed-integer programming (MIP) in the path planning community due to its ability to model nonlinearity and nonconvexity [7], [12], [13], [14], in conjunction with the availability of specialized solvers. This is particularly useful for modeling collision avoidance, in which we require an agent to lie in some region of obstacle-free safe terrain, typically nonconvex due to cavities brought forth by the obstacles. One approach to modeling obstacle avoidance is the direct assignment of waypoints (or trajectory pieces) to obstacle-free polyhedral regions in the configuration space, through the use of binary variables [15], [16]. That is, letting x represent a waypoint, we require

$$x \in \bigcup_{i=1}^{d} P^i, \tag{1}$$

where each $P^i \subseteq \mathbb{R}^n$ is a polyhedron. These are *disjunctive constraints* and can be modeled for each waypoint using MIP, potentially along other dynamic and/or logical constraints. Modeling this nonconvex domain has computational implications, however. The typical approach to formulating disjunctive constraints is through a big-M technique [14], [15]. While these contraints are simple to implement and reason about, they generally have weak convex relaxations, which can often result in poor computational performance.

In this work, we leverage the *independent branching* (IB) scheme introduced by Vielma and Nemhauser [17] and further expanded by Huchette and Vielma [18], in order to construct small, ideal formulations of combinatorial disjunctive constraints (CDC) for obstacle avoidance in 2D environments. We discuss conditions for when this approach may be employed, and, as this method requires a *biclique cover* of an associated graph, we develop a polynomial time algorithm for finding biclique covers on this class of graphs.

[1]Raul Garcia is with the Department of Computational Applied Mathematics & Operations Research, Rice University, Houston, TX 77005, USA rjgarcia@rice.edu

[2]Illya V. Hicks is with Faculty of Computational Applied Mathematics & Operations Research, Rice University, Houston, TX 77005, USA ivhicks@rice.edu

[3]Joey Huchette is with Google Research, Cambridge, MA 02142, USA jhuchette@google.com

Furthermore, we contribute an open-source Julia library named `ClutteredEnvPathOpt`[1] to facilitate computational experiments of MIP formulations for obstacle avoidance and demonstrate our framework through an application of footstep planning for a humanoid robot, largely adapted from the mixed-integer quadratically constrained quadratic program (MIQCQP) presented by Deits and Tedrake [15]. While our experiments have shown the big-M method to generally outperform our IB scheme approach, we hypothesize it is due to the specialized heuristics of MIP solvers. Moreover, we highlight that in a handful of instances our method was able to compute optimal paths 40-50% faster compared to big-M and remains competitive for many more. We hope our approach may become a favorable alternative in particular classes of instances.

This paper is organized as follows. In Section II, we provide background on MIP formulation theory and introduce the big-M and IB scheme approaches for disjunctive constraints. In Section III, we discuss conditions for when the IB scheme may be employed in 2D environments and present our algorithm for obtaining biclique covers on the class of graphs arising in this context. We present our computational results and an example of our framework in Section IV, before ending with a discussion on the observed performance.

II. PRELIMINARIES

A. Polyhedral Theory

A (bounded) $\mathcal{V}$-polyhedron is a set $P \subset \mathbb{R}^n$ that can be expressed as

$$P = \text{Conv}(V) \stackrel{\text{def}}{=} \left\{ \sum_{v \in V} \lambda_v v : \lambda \in \Delta^V \right\}$$

for some finite set of vectors $V \subset \mathbb{R}^n$, where $\Delta^V \stackrel{\text{def}}{=} \left\{ \lambda \in \mathbb{R}_+^{|V|} : \sum_{v \in V} \lambda_v = 1 \right\}$ is the standard simplex. In other words, it is the convex hull of the vectors V. We will commonly refer to bounded polyhedrons as *polytopes*.

A $\mathcal{H}$-polyhedron is a set $P \subset \mathbb{R}^n$ that can be expressed as

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

where A is an $m \times n$ matrix and b is a vector in $\mathbb{R}^m$. Without loss of generality, bounded polyhedra have both $\mathcal{V}$- and $\mathcal{H}$-representations.

A (binary) MIP formulation F for a constraint $x \in Q \subseteq \mathbb{R}^{n_1}$ is the composition of linear inequalities

$$\begin{aligned} R \stackrel{\text{def}}{=} \{(x, y, z) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times [0,1]^{n_3} : \\ Ax + By + Cz \leq d\} \end{aligned} \tag{2}$$

with (binary) integrality conditions

$$F \stackrel{\text{def}}{=} R \cap (\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \{0,1\}^{n_3}) \tag{3}$$

such that $\text{Proj}_x(F) = Q$. The variables y and z denote auxiliary continuous and binary variables, respectively, which are employed to handle the nonlinearity and/or nonconvexity of Q. R is referred to as the *LP relaxation*, or just *relaxation*, of F, since it relaxes the integrality conditions.

For computational purposes, we are interested in understanding both the strength of a given formulation, as well as in quantifying its size or complexity. A formulation is said to be *ideal* if each extreme point of the relaxation naturally satisfies the integrality conditions. This is the strongest possible MIP formulation with respect to the convex relaxation that one can expect; in particular, if an MIP formulation is ideal, it can be solved as a linear program (LP), foregoing the need for branching on fractional variables. On the other hand, one measure of the complexity of a formulation is the number of auxiliary continuous variables y and auxiliary binary variables z used, as well as the number of inequalities in the description of R.

B. Big-M Techniques

A standard approach to formulating polyhedral disjunctive constraints is the big-M method [14]. Letting $P^i = \{x \in \mathbb{R}^n : A^i x \leq b^i\}$, the disjunctive constraint (1) can be modeled by

$$A^i x \leq b^i z_i + M^i(1 - z_i) \qquad \forall i \in [d] \tag{4a}$$

$$\sum_{i=1}^{d} z_i = 1 \tag{4b}$$

$$z \in \{0,1\}^d, \tag{4c}$$

where $[d] \stackrel{\text{def}}{=} \{1, \ldots, d\}$ and $M^i, i \in [d]$, are vectors with sufficiently large entries. In particular, for each $i \in [d]$, we must take

$$M_k^i \geq \max_{x \in \Omega} (A^i x)_k, \tag{5}$$

for each row index k of the system $A^i x \leq b^i$ and $\Omega \stackrel{\text{def}}{=} \bigcup_{i=1}^d P^i$. M values can be computed efficiently via linear programming by maximizing over some polytope containing Ω.

Such formulations are not ideal in general; in fact, the values of M play a crucial role in the strength of a formulation [14]. Larger values increase the size of the convex relaxation, which generally results in a larger branch-and-cut search tree. Therefore, it is preferable to obtain values for M which are as tight as possible.

C. Combinatorial Disjunctive Constraints

When the disjunctive constraint (1) is a union of $\mathcal{V}$-polyhedra (e.g., Fig. 1), it suffices to consider only their extreme points. Let $J = \bigcup_{i=1}^d \text{ext}(P^i)$ denote the *ground set* and take $\mathcal{S} = \{\text{ext}(P^i)\}_{i=1}^d \subseteq 2^J$ as the collection of extreme points for each of the polyhedra.

Definition 1 (Definition 1, [18])**.** A *combinatorial disjunctive constraint* (CDC) induced by the set $\mathcal{S}$ is

$$\lambda \in \text{CDC}(\mathcal{S}) \stackrel{\text{def}}{=} \bigcup_{S \in \mathcal{S}} Q(S), \tag{6}$$

[1] https://github.com/raulgarcia66/ClutteredEnvPathOpt.jl

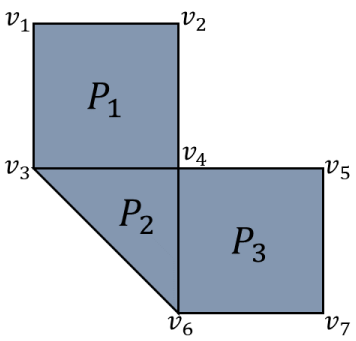

Fig. 1. A nonconvex region Ω defined by a union of polytopes.

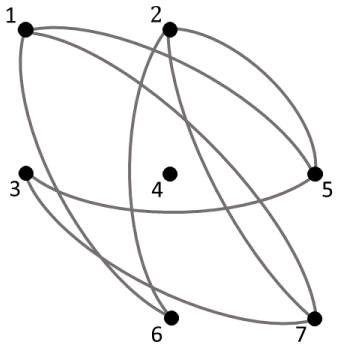

Fig. 2. Conflict graph of the polyhedral partition from Fig. 1. No edge has both endpoints in the same polytope.

where $Q(S) \stackrel{\text{def}}{=} \{\lambda \in \Delta^J : \lambda_{J\setminus S} = 0\}$ is the face that $S \subseteq J$ induces on the standard simplex.

In other words, Def. 1 states λ must obtain support over some $S \in \mathcal{S}$. A corresponding formulation for (1) is thus

$$x \in \left\{ \sum_{v \in J} \lambda_v v : \lambda \in \text{CDC}(\mathcal{S}) \right\}. \tag{7}$$

We highlight that one advantage of this approach is that we can construct a single, strong formulation for a given CDC($\mathcal{S}$) and use it for other instances whose combinatorial structures are sufficiently equivalent, e.g., path planning in similar environments (see [18] for a sufficient condition).

As with [18], we assume $\mathcal{S}$ is irredundant: there do not exist distinct $S, T \in \mathcal{S}$ such that $S \subseteq T$. We say that a set $S \subseteq J$ is a *feasible set* with respect to CDC($\mathcal{S}$) if $S \subseteq T$ for some $T \in \mathcal{S}$ and that it is an *infeasible set* otherwise. Furthermore, a *minimal infeasible set* does not contain any infeasible set as a proper subset.

Standard ideal formulations for (6) are $\mathcal{O}(|\mathcal{S}| + \sum_{S \in \mathcal{S}} |S|)$ in size [18], [19]. The *independent branching* (IB) scheme [17], [18], however, when one exists, is a logically equivalent way of expressing a CDC as a series of choices between two alternatives, whose formulation size on the order of the number of dichotomies.

Definition 2 (Definition 2, [18])**.** An *independent branching* scheme for CDC($\mathcal{S}$) is given by a family of sets (L^j, R^j) (where $L^j, R^j \subseteq J$) for $j \in [t]$, where

$$\text{CDC}(\mathcal{S}) = \bigcap_{j=1}^{t} \left(Q(L^j) \cup Q(R^j) \right). \tag{8}$$

Such an IB scheme is said to have *depth* t and that each $j \in [t]$ yields a corresponding *level*.

CDC($\mathcal{S}$) is thus equivalently represented by t constraints, each requiring a selection among two alternatives, and satisfies the condition that

$$T \subseteq J \text{ is a feasible set} \iff \forall j \in [t] : \; T \subseteq L^j \text{ or } T \subseteq R^j.$$

One can obtain IB schemes in an algorithmic manner based on a graphical characterization of CDC($\mathcal{S}$). To illustrate this, we define the *conflict graph* of a CDC($\mathcal{S}$) as $G^c_{\mathcal{S}} \stackrel{\text{def}}{=} (J, \bar{E}_{\mathcal{S}})$, where $\bar{E}_{\mathcal{S}} \stackrel{\text{def}}{=} \{(u, v) \in J^2 : u \neq v, \{u, v\} \text{ is an infeasible set}\}$ [18] (see Fig. 2). Additionally, we require the following notion from graph theory.

Definition 3 (Definition 4, [18])**.** A *biclique cover* of the graph $G = (J, E)$ is a collection of complete bipartite subgraphs $\left\{ G^j \stackrel{\text{def}}{=} (A^j \cup B^j, E^j) \right\}_{j=1}^{t}$ of G that covers all edges of G.

We will refer to the sets $\{(A^j, B^j)\}_{j=1}^t$ as a biclique cover.

PROPOSITION 1: *(Proposition 3, [18])*
If CDC($\mathcal{S}$) is IB-representable and $\{(A^j, B^j)\}_{j=1}^t$ is a biclique cover for $G^c_{\mathcal{S}}$, then the following is an ideal formulation for CDC($\mathcal{S}$):

$$\sum_{v \in A^j} \lambda_v \leq z_j \qquad \forall j \in [t] \tag{9a}$$

$$\sum_{v \in B^j} \lambda_v \leq 1 - z_j \qquad \forall j \in [t] \tag{9b}$$

$$(\lambda, z) \in \Delta^J \times \{0, 1\}^t. \tag{9c}$$

An intuitive understanding of this formulation is that the construction of infeasible sets is prevented by prohibiting the convex combination of vertices from minimal infeasible sets.

Consequently, the smallest depth at which an IB scheme can be constructed equates to the minimum biclique cover problem [20]. While the decision version of this problem is known to be NP-complete [21], we may instead, however, opt for biclique covers of suboptimal depth, provided they can be obtained efficiently. For example, the trivial biclique cover

$$A^v = \{v\}, \quad B^v = \{u \in J : \{u, v\} \in \bar{E}_{\mathcal{S}}\} \quad \forall v \in J$$

allows for a CDC formulation of depth $|J|$ [18]. Biclique covers may also be reduced in size by merging bicliques together, provided they satisfy the complete bipartite subgraph definition.

III. CONTRIBUTIONS

Many obstacle avoidance scenarios naturally give rise to combinatorial disjunctive constraints corresponding to polyhedral partitions in the plane. For this setting, we present:

1) A direct procedure for determining when a CDC admits an IB scheme;
2) A method for obtaining IB-representable polyhedral partitions;
3) A polynomial time algorithm for obtaining biclique covers on the class of conflict graphs that arise in these scenarios.

A. IB Scheme Representability

We begin by defining a polyhedral partition. Consider a (nonconvex) bounded region in the plane $\Omega \subset \mathbb{R}^2$ describing the obstacle-free space an agent may travel in. Ω can be partitioned into polyhedra $\{P^i\}_{i=1}^d$ such that $\bigcup_{i=1}^d P^i = \Omega$ and whose relative interiors do not overlap (see Fig. 1). Such a partition will not be unique in general and plays a significant role in representability and formulation size.

With a fixed partition, we describe the associated polyhedra in $\mathcal{V}$-form so that the combinatorial disjunctive constraint is characterized by $\mathcal{S} = \{\text{ext}(P^i)\}_{i=1}^d$ and $J = \bigcup_{S\in\mathcal{S}} S$. As with [18], we also forbid "internal vertices" by requiring that

$$v \in P^i \iff v \in \text{ext}(P^i) \quad \forall i \in [d], v \in J. \tag{10}$$

COROLLARY 1:
An IB scheme of a polyhedral partition in the plane exists if and only if each infeasible triplet is not minimal infeasible.

One can thus verify IB-representability by enumerating all infeasible triplets and determining if they are minimal infeasible.

Additionally, we present a procedure for partitioning an obstacle-free region Ω in a manner that is always IB-representable, satisfies the internal vertex condition, and which can be computed quickly without manual user input. A Delaunay triangulation is a triangulation of a given set of vertices (hence, a graph) satisfying certain properties, which we omit for brevity (see [22] for a formal definition). A constrained Delaunay triangulation (CDT) requires certain edges be included and, possibly, certain edges within a region be excluded. For the following theorem, we will refer to obstacles with no edges comprising the boundary of Ω as *internal* obstacles.

THEOREM 1:
Given a set of obstacles with no triangular internal obstacles, the partition of a (nonconvex) obstacle-free region Ω in the plane yielded from a constrained Delaunay triangulation, which includes the edges corresponding to the boundary of Ω and prohibits the region enclosed by obstacles from containing diagonal edges, always admits a pairwise IB scheme.

CDTs can be computed in $\mathcal{O}(n \log n)$ time (where n is the number of vertices) [22], and software packages such as *Triangle* [23] have readily available implementations.

B. Obtaining Biclique Covers

By exploiting the special structure of conflict graphs arising from polyhedral partitions in the plane, we present a polynomial time algorithm for obtaining biclique covers on this class of graphs. We say a graph is *planar* if, intuitively, it can be drawn without any of its edges crossing each other (see [24] for a formal definition). The regions enclosed by edges of the graph are referred to as *faces* and can be described in terms of the vertices of the edges that comprise the boundary. A *finite element graph* is any graph formed from a planar embedding of a planar graph by adding all possible diagonal edges to each face, or *element* (i.e., each face becomes a clique). We will relax this definition to allow some faces to remain unaltered.

The following theorem presents a special property of finite element graphs that we leverage in our biclique cover algorithm.

COROLLARY 2: *(Corollary 4, [24])*
Let G be any n-vertex finite element graph. Suppose no element of G has more than k boundary vertices. The vertices of G can be partitioned into three sets A, B, C such that no edge joins a vertex in A with a vertex in B, neither A nor B contains more than $2n/3$ vertices, and C contains no more than $4\lfloor k/2 \rfloor \sqrt{n}$ vertices.

Moreover, Lipton and Tarjan present an algorithm for finding a partition A, B, C in $\mathcal{O}(n)$ time [24].

We denote the conflict graph's complement as $G_{\mathcal{S}}$; it is easy to see that $G_{\mathcal{S}} = \{(u, v) \in J^2 : u \neq v, \{u, v\} \text{ is a feasible set}\}$, and, in particular, $G_{\mathcal{S}}$ will be a finite element graph. We can then observe that if we apply the separator algorithm [24] on $G_{\mathcal{S}}$ to obtain a partition of its vertices A, B, C, the complete bipartite graph induced by (A, B) (assuming neither set is empty) will form a biclique of $G_{\mathcal{S}}^c$. As there may remain uncovered edges in $G_{\mathcal{S}}^c$, we may likewise apply the separator algorithm to the subgraphs of $G_{\mathcal{S}}$ induced by $A \cup C$ and $B \cup C$, respectively, to obtain further bicliques. Applying the separator algorithm to $G_{\mathcal{S}}$ in a divide-and-conquer manner will thus yield a collection of bicliques forming a biclique cover for $G_{\mathcal{S}}^c$. We provide a high level summary in Algorithm 1.

Algorithm 1 Biclique cover algorithm

Require: $G_{\mathcal{S}}$ a finite element graph

1: Apply Lipton and Tarjan's separator algorithm for finite element graphs on $G_{\mathcal{S}}$ to obtain a partition A, B, C of the graph's vertices.
2: Apply a postprocessing procedure to ensure A and B are both nonempty (given $G_{\mathcal{S}}$ is not a complete graph). Insert the complete bipartite graph induced by (A, B) into the biclique cover.
3: Recursively apply this algorithm to the subgraphs of $G_{\mathcal{S}}$ induced by the vertices $A \cup C$ and $B \cup C$, respectively, until all subgraphs are complete graphs.

It is important to note that the planar separator algorithm does not guarantee both A and B be nonempty simultaneously. However, in such a case a postprocessing procedure can be applied to transfer vertices appropriately.

THEOREM 2:
Given a finite element graph G, the biclique cover algorithm terminates with a biclique cover for the complement graph G^c. Moreover, the complexity of this algorithm is $\mathcal{O}(n^4)$.

IV. COMPUTATIONAL RESULTS

To demonstrate our framework, we applied our approach to the specific case of footstep planning of a humanoid robot. In these problems, we seek to determine the precise x, y and θ coordinates of N footsteps en route to a goal pose, such that no step intersects any obstacle and every step is reachable relative to the previous. We apply much of the formulation techniques proposed by Deits and Tedrake [15]; however, while their implementation leveraged the big-M technique to formulate footstep assignment in $xyz\theta$-space, we are limited to applying the IB scheme framework in xy-space. Nonetheless, we assumed even terrain to forgo z and imposed constraints to limit the change in θ between steps. The final problem is an MIQCQP in which a walking humanoid robot is incentivized to reach a goal destination.

Using our library, `CluttEredEnvPathOpt`, we conducted experiments comprising of 69 original hypothetical scenarios of 2D obstacles contained within the unit square, which is without loss of generality since our workspace can always be mapped into the unit square. Each scenario contains 3 polyhedral obstacles of various shapes and sizes (which may lie on the boundary), and each obstacle contains at least 4 vertices, with the majority having between 4 and 6 (internal obstacles may not have 3 for IB-representability purposes). We solve each scenario with one, two and three obstacles at a time, as well as with three methods: 1) IB with the original biclique cover computed by our algorithm; 2) IB with the biclique cover obtained after applying a merging procedure to the original; 3) the big-M method (computed as described in Sec. II-B by maximizing over $[0, 1]^2$). All scenarios share the same parameters, e.g., 25 footsteps, and begin on the bottom left corner of the unit square, with the goal pose positioned on the top right.

In constructing the optimization problems, we took our obstacle-free space to be the unit square minus the obstacles and use a constrained Delaunay Triangulation to produce an IB-representable polyhedral partition. To decrease the number of disjunctions in our footstep assignment constraint, it is possible to merge polytopes by taking their convex hull, provided vertices do not become redundant and the internal vertex condition (10) remains satisfied. However, we found that merging the free-space polytopes may result in a partition that is no longer IB-representable. Therefore, we withheld from merging polytopes in our experiments.

All instances were instantiated with Julia 1.6.1 and solved with Gurobi 9.1.2 on a machine with an Intel Core i7 processor at 1.8 GHz and 8 GB of RAM. For practicality of footstep planning, we set a time limit of 5 minutes. Source code and obstacle files can be found on our library's Github.

To gain a better understanding of our path planning framework, we provide the following example (scenario 47 from our set). The obstacle-free space partition is shown in Fig. 3, and the associated conflict graph $G_{\mathcal{S}}^c$ and its complement $G_{\mathcal{S}}$ are shown in Figs. 4 and 5, respectively. In this example, there are 13 vertices and 15 "free" faces, and the biclique cover obtained by our algorithm is of depth 8 (down from 15 after merging), listed in Table I. We show an optimal footstep plan for this instance in Fig. 6. One can observe the solution opts to trim 6 steps and use 19 from the available 25. One may also notice the path appears to cut the corners of the obstacles, which is consistent with our formulation given we only consider the position of the footsteps when in contact with the ground. This potential shortcoming can be addressed via existing techniques from the literature [25].

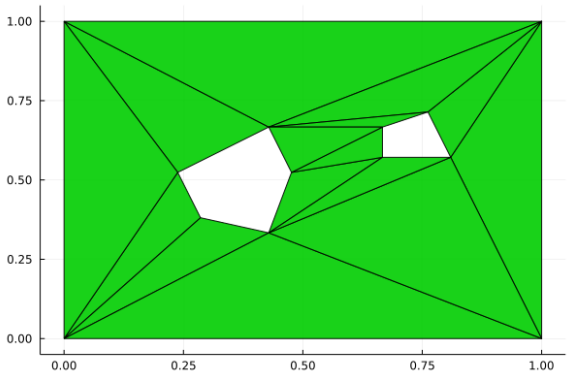

Fig. 3. Polyhedral partition from CDT.

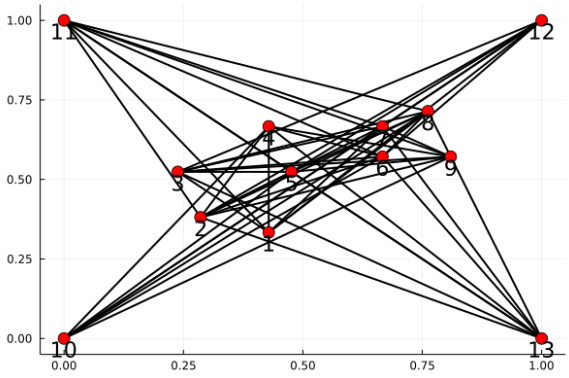

Fig. 4. Conflict graph $G_{\mathcal{S}}^c$.

Level	A	B
1	{3, 7}	{1, 9, 12}
2	{9, 12}	{3, 5, 10}
3	{4, 7}	{1, 9}
4	{2, 10, 13}	{4, 5, 6, 7, 8}
5	{3, 11}	{1, 5, 6, 7, 8, 9}
6	{1, 2}	{11, 12}
7	{2, 3, 4, 11}	{6, 9, 13}
8	{8, 12}	{1, 5, 6, 10}

TABLE I
BICLIQUE COVER OF CONFLICT GRAPH IN FIG. 4.

In Table II, we compare IB with the merged cover, IB with the original cover, and the big-M method, across solve times and footstep assignment formulation sizes. We can

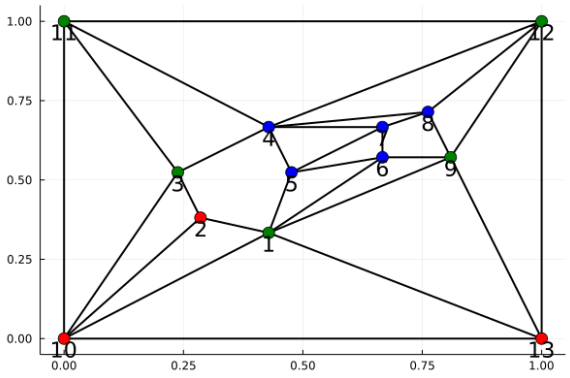

Fig. 5. G_{S}. The colored nodes in G_{S} indicate the vertex partition computed by the planar separator algorithm, with A red, B blue and C green.

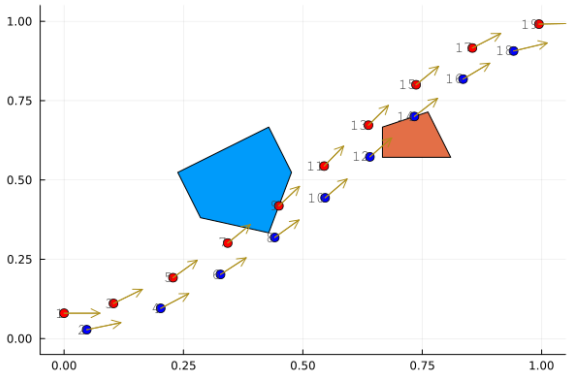

Fig. 6. An optimal solution computed via the IB framework.

see that despite having a larger amount of binary variables and general inequalities on average, the big-M method tends to outperform the IB formulations. We highlight that the standard deviations of solve times are larger for the IB approaches as well, reflective of the fact that solve times are competitive in some instances. In fact, our framework was able to solve problems 40-50% faster in 5 instances and was the fastest in 11 more. We can also observe the general trend of increasing solve times with increasing number of obstacles. This is expected: more obstacles result in a larger, more complex conflict graph for the IB case (and likely a larger biclique cover) and more free face inequality systems for the big-M approach. The sizes of these intrinsic properties along with biclique cover reduction statistics are summarized in Table III.

V. CONCLUSION

We have seen that although the IB scheme footstep assignment formulations are (individually) ideal and tend to have fewer binary variables and general inequalites than our respective big-M formulation, it is generally outperformed by the latter. There are a number of possible explanations for this observation.

We hypothesize the strongest factor in the performance discrepancy comes from Gurobi's internal optimization procedures, which include sophisticated branching variable selection techniques, cutting planes, presolve, and numerous heuristics, all of which can reduce the size of the branch-and-cut search tree [26], [27]. A good first direction would be to empirically investigate with other MIQCQP solvers.

Initial feasible solutions can also have a significant affect. The "nearer" an initial feasible solution is to the optimal solution, the fewer subproblems we can expect in a branch-and-cut procedure [27]. In a small set of experiments conducted to test this hypothesis, we observed shorter solve times for the IB approach than big-M when warm starting with near-optimal solutions computed from a variety of methods. Moreover, perusing the obstacle scenarios for patterns of performance, we found the IB approaches to be most often competitive (though not always) when the optimal path involved minimal meandering around obstacles, which we hypothesize is due to the feasibility of initial solutions. A direction of future work is to understand this phenomena experimentally and develop heuristic procedures for obtaining "good" warm starting solutions quickly.

Another factor is that the IB scheme introduces auxiliary continuous variables λ, in addition to the auxiliary binary variables z. In contrast, the big-M method only requires auxiliary binary variables. However, given the moderate size of our final formulations and modern solvers' capabilities for solving large-scale optimization problems [27], we do not expect this difference in continuous variables to play a significant role. We also highlight our program contains quadratic constraints, for which strong formulation theory may not apply exactly as for the linear case. However, Gurobi generally solves MIQCQPs by transforming them into linear programs and adding separating hyperplanes ad hoc [28].

Finally, as our approach remains competitive in some instances, we must be careful not to generalize our results to all scenarios, let alone all path planning MIP approaches, without further treatment. It would be beneficial to expand the obstacle scenario test sets, allowing for larger quantities of obstacles, obstacles with more vertices, more diverse start and goal positions, and more diverse configurations of obstacle placements. This would allow us to determine if there are particular settings for which our framework is favorable; for example, in scenarios with a large quantity of obstacles, the size of the biclique covers could remain significantly smaller than the number of free faces.

APPENDIX

We refer the reader to Garcia [29] for proofs to Corollary 1, Theorem 1 and Theorem 2.

ACKNOWLEDGMENTS

The authors would like to thank Miles Olson for his software contributions to `ClutteredEnvPathOpt`. The authors would also like to thank Dr. Lydia E. Kavraki and Carlos Quintero Pena for their robotics related input.

REFERENCES

[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communnications of the ACM*, vol. 22, no. 10, p. 560–570, Oct 1979. [Online]. Available: https://doi.org/10.1145/359156.359164

	1 Obstacle			2 Obstacles			3 Obstacles		
	IB	IB_orig	Big-M	IB	IB_orig	Big-M	IB	IB_orig	Big-M
Fastest	4	4	61	1	3	62	4	0	53
Timeouts	14	17	0	49	47	4	55	57	12
Solve Time Avg (s)	85.06	80.71	31.18	130.36	104.12	55.74	104.18	139.26	76.02
Solve Time Std (s)	72.11	71.41	34.54	77.18	72.22	47.22	85.12	94.27	59.1
Binary Var.	115.58	179.71	208.33	192.39	351.45	359.06	241.30	492.39	496.01
Cont. Var.	214.13	214.13	0.00	317.75	317.75	0.00	415.22	415.22	0.00
Inequalities	231.16	359.42	625.00	384.78	702.90	1077.17	482.61	984.78	1488.04

TABLE II

'FASTEST' COUNTS AND SOLVE TIME STATISTICS ONLY CONSIDER FULLY SOLVED PROBLEMS. FORMULATION SIZES ARE AVERAGED AMONG THE 69 TEST INSTANCES.

	1 Obstacle	2 Obstacles	3 Obstacles
Vertices	8.57	12.71	16.61
B.C. Original	7.19	14.06	19.70
B.C. Merged	4.62	7.69	9.65
B.C. Reduction (%)	35.00	44.75	50.73
Free Faces (F.F.)	8.33	14.36	19.84
F.F. Halfspaces	25.00	43.09	59.52

TABLE III

TEST INSTANCE AVERAGES OF INTRINSIC PROPERTIES.

[2] E. Masehian and G. Habibi, "Robot path planning in 3d space using binary integer programming," *International Journal of Mechanical Systems Science and Engineering*, vol. 1, no. 1, 01 2007.

[3] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, "Real-time replanning using 3d environment for humanoid robot," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 584–589.

[4] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2001, pp. 500–505.

[5] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, October 2003.

[6] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade, "Vision-guided humanoid footstep planning for dynamic environments," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 2005, pp. 13–18.

[7] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578820300754

[8] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept. Oct.*, vol. 98, no. 11, 1998.

[10] M. P Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. Pérez-D'Arpino, R. Deits, M. Dicicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller, "An architecture for online affordance-based perception and whole-body planning," *Journal of Field Robotics*, vol. 32, no. 2, p. 229–254, 10 2014.

[11] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[12] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference*, vol. 3, 2002, pp. 1936–1941.

[13] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *2001 European Control Conference (ECC)*, 2001, pp. 2603–2608.

[14] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Review*, vol. 57, no. 1, pp. 3–57, 2015.

[15] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 279–286.

[16] ——, "Efficient mixed-integer planning for uavs in cluttered environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 42–49.

[17] J. P. Vielma and G. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Mathematical Programming*, vol. 128, no. 1-2, pp. 49–72, 2011.

[18] J. Huchette and J. P. Vielma, "A combinatorial approach for small and strong formulations of disjunctive constraints," *Mathematics of Operations Research*, vol. 44, no. 3, pp. 793–820, May 2019.

[19] R. Jeroslow and J. Lowe, "Modelling with integer variables," *Mathematical Programming Study*, vol. 22, pp. 167–184, 1984.

[20] P. C. Fishburn and P. L. Hammer, "Bipartite dimensions and bipartite degrees of graphs," *Discrete Mathematics*, vol. 160, no. 1, pp. 127–148, 1996.

[21] J. Orlin, "Contentment in graph theory: Covering graphs with cliques," *Indagationes Mathematicae (Proceedings)*, vol. 80, no. 5, pp. 406–424, 1977.

[22] L. P. Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 7, no. 1, pp. 97–108, 1989.

[23] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," in *Applied Computational Geometry: Towards Geometric Engineering*, ser. Lecture Notes in Computer Science. Springer-Verlag, May 1996, vol. 1148, pp. 203–222.

[24] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *SIAM Journal on Applied Mathematics*, vol. 36, no. 2, pp. 177–189, 1979. [Online]. Available: https://doi.org/10.1137/0136016

[25] F. Stoican, E. Ingar Grøtli, I. Prodan, and C. Oară, "On corner cutting in multi-obstacle avoidance problems," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 185–190, 2015, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315025653

[26] "Mixed-integer programming (mip) – a primer on the basics." [Online]. Available: https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/

[27] M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer Berlin Heidelberg, 6 2009.

[28] "Gurobi qcp and socp optimizer overview." [Online]. Available: https://www.gurobi.com/events/gurobi-qcp-and-socp-optimizer-overview/

[29] R. Garcia, "A combinatorial disjunctive constraint approach to optimal footstep planning," Master's thesis, Rice University, 2023.