Template Matching and Decision Diagrams for Multi-Agent Path Finding

Jayanth Krishna Mogali$^{1(\boxtimes)}$, Willem-Jan van Hoeve$^{2[0000-0002-0023-753X]\star}$, and Stephen F. Smith$^{1[0000-0002-7053-3166]}$

^{1} Robotics Institute, Carnegie Mellon University, USA
^{2} Tepper School of Business, Carnegie Mellon University, USA
{jmogali,vanhoeve,ssmith}@andrew.cmu.edu

Abstract. We propose a polyhedral cutting plane procedure for computing a lower bound on the optimal solution to multi-agent path finding (MAPF) problems. We obtain our cuts by projecting the polytope representing the solutions to MAPF to lower dimensions. A novel feature of our approach is that the projection polytopes we used to derive the cuts can be viewed as 'templates'. By translating these templates spatio-temporally, we obtain different projections, and so the cut generation scheme is reminiscent of the template matching technique from image processing. We use decision diagrams to compactly represent the templates and to perform the cut generation. To obtain the lower bound, we embed our cut generation procedure into a Lagrangian Relax-and-Cut scheme. We incorporate our lower bounds as a node evaluation function in a conflict-based search procedure, and experimentally evaluate its effectiveness.

Keywords: MAPF · Projection Cuts · Template polytopes · Decision diagrams · Lagrangian relax and cut · Conflict based search

1 Introduction

Multi-agent path finding (MAPF) is the problem of finding paths for individual robots (agents), given a start and end vertex for each robot on some layout (graph), such that the paths are spatio-temporally conflict-free and an objective resembling travel costs is minimized. MAPF has found many applications in warehouse logistic systems [18] and robotics. MAPF is known to be NP-Hard to solve optimally on general graphs [20], nonetheless many techniques have been proposed and they come in different flavors.

Current approaches for MAPF can be broadly classified into search based methods [14, 16], and solution methods that rely on polyhedral techniques such as the integer programming formulation of [19], and the branch-cut-price method of [11]. A significant challenge for either of these approaches is in developing

* Partially supported by Office of Naval Research Grant No. N00014-18-1-2129 and National Science Foundation Award #1918102.

strong lower bounding techniques. Such techniques are needed to prune search regions that do not lead to an optimal solution. From the point of view of search based methods, this translates into developing strong admissible heuristics. For polyhedral techniques, strong lower bounds are typically obtained by developing cutting planes that tighten the linear description of the solution space.

In this paper, we propose a cut generation scheme that can be incorporated into search based methods for MAPF as well as polyhedral approaches. Incorporating cuts into techniques that use polyhedral approaches is common, but incorporating cuts into search based methods for MAPF is somewhat rare, and will be the focus of this paper.

Contributions. Main contributions of this work are 1) a new polyhedral approach for MAPF based on lower-dimensional 'templates' that can be translated spatio-temporally over the input graph, 2) the development of a cut generation scheme from these templates, which utilizes a decision diagram representation, 3) a Lagrangian Relax-and-Cut procedure to compute the lower bound, and 4) incorporating the resulting lower bound as a node evaluation function in a conflict-based search (CBS) procedure. Experimental evaluation shows that our lower bounds can be very effective when the MAPF problem is more constrained.

2 MAPF Problem Description

We consider the makespan-constrained version of the MAPF problem in this paper. We are given an undirected graph $G = (V, E)$, a set of $\mathbf{N}$ robots $\mathcal{R} = \{r_1, ..., r_{\mathbf{N}}\}$, and a makespan upper bound $\mathbf{T} \in \mathbb{Z}_+$, where $\mathbb{Z}_+$ represents the set of positive integers. Corresponding to each robot $r_i \in \mathcal{R}$, we are given a start vertex $s_i \in V$, and goal vertex $g_i \in V$. The task is to find a path for each robot, such that the robot paths do not conflict while minimizing the cumulative sum of path costs. A path p can be viewed as a function $p : \{0, 1, ..., \mathbf{T}\} \rightarrow V$, where $p(t)$ returns a vertex in V corresponding to time t. If $\mathcal{P} = \{p_1, ..., p_{\mathbf{N}}\}$ is a set of robot paths with 1 path for each robot, $\mathcal{P}$ is feasible to the MAPF problem iff:

1. $p_i(0) = s_i$ and $p_i(\mathbf{T}) = g_i$, $\forall i \in \{1, 2, ..., \mathbf{N}\}$.
2. For each robot $r_i \in \mathcal{R}$ and for all $t \in \{0, 1, ..., \mathbf{T} - 1\}$, we require $p_i(t) = p_i(t+1)$, or $(p_i(t), p_i(t+1)) \in E$. The robot either stays in its current vertex or moves to a neighbor.
3. To prevent vertex collisions, we require that $p_i(t) \neq p_j(t)$, for all pairwise distinct $i, j \in \{1, ..., \mathbf{N}\}$ and time $t \in \{0, 1, ..., \mathbf{T}\}$.
4. To prevent edge collisions, there should not exist a pair of robots r_i, r_j and time $t \in \{0, 1, ..., \mathbf{T} - 1\}$ such that, $p_i(t) = p_j(t + 1)$ and $p_i(t + 1) = p_j(t)$.

We refer to any path p_i satisfying 1 and 2 as a **start-end path** for robot r_i. The cost of start-end path p_i is given by $c_i(p_i) = \sum_{t=0}^{\mathbf{T}-1} c_i(p_i(t), p_i(t + 1))$, where

$$c_i(p_i(t), p_i(t + 1)) = \begin{cases} 0, & \text{if } p_i(t) = p_i(t + 1) = g_i \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

Equation (1) assigns a cost of 0 if the robot is waiting at its goal vertex, else assigns a cost of 1. The goal of MAPF is to find a set of conflict-free robot paths

$p_1, ..., p_{\mathbf{N}}$ that minimizes the objective $\sum_{i=1}^{\mathbf{N}} c_i(p_i)$. The cost function in (1) slightly differs from the sum of completion times used in [9, 12], where completion time is the earliest time the robot reaches its goal and remains stationary until time $\mathbf{T}$ at the goal vertex. We adopt the cost function shown in (1) to simplify the presentation of template construction presented in a later section.

3 Integer programming model for MAPF

We next provide a multi-commodity flow based Integer Programming (IP) model for the MAPF problem, similar to [19]. The IP model will be useful in deriving valid inequalities for the lower bounding procedure we propose in later sections. In the descriptions below, for any $n \in \mathbb{Z}_+$ we will use the notation $[n]$ to denote the set $\{1, 2, ..., n\}$.

The IP model will make use of the so-called "time expanded graph". The time expanded graph is an arc-weighted directed acyclic graph defined for each robot, where the nodes can be partitioned into $\mathbf{T} + 1$ layers, and arcs into $\mathbf{T}$ layers. We shall denote the time expanded graph for robot r_i by $F_i(N_i, A_i)$. Denote the nodes in layer $t \in \mathbb{Z}_+ \cup \{0\}$ by $N_i(t)$. Corresponding to each vertex $v \in V$, there exists a node in $N_i(t)$ if the shortest path from s_i to v in $G(V, E)$ passes through at most t edges, and the shortest path from v to g_i passes through at most $\mathbf{T} - t$ edges. With a slight abuse of notation, we shall denote the node corresponding to vertex $v \in V$ in $N_i(t)$ by v_i^t. Throughout this paper, if a node from any of the graphs in the set $\{F_i\}_{i \in [\mathbf{N}]}$ is specified, we will assume that the vertex, time and robot associated with that node can be deduced from our notation. Arcs in A_i connect nodes between adjacent layers, with the tail of the arc emanating from the node belonging to the lower indexed layer. Denote the arcs in level t by $A_i(t)$. If $u_i^t \in N_i(t)$ and $v_i^{t+1} \in N_i(t + 1)$, then there exists an arc $(u_i^t, v_i^{t+1}) \in A_i(t)$ iff $u = v$, or $(u, v) \in E$. We let $c_i(u_i^t, v_i^{t+1})$ denote the weight of arc (u_i^t, v_i^{t+1}), where $c_i(u_i^t, v_i^{t+1}) = 0$ if $u = v = g_i$, and 1 otherwise. There is a 1:1 correspondence between **start-end paths** for robot r_i and $s_i^0 - g_i^{\mathbf{T}}$ paths in $F_i(N_i, A_i)$.

In describing our IP model, we will make use of the following notation. For any node $v_i^t \in N_i(t)$, we denote $\delta_{F_i}^+(v_i^t)$ as the set of arcs in A_i whose tail is the node v_i^t, and $\delta_{F_i}^-(v_i^t)$ as the set of arcs in A_i whose head is the node v_i^t. For any vertex $u \in V$, we introduce the set $\overline{V}^t(u) = \{i \in [\mathbf{N}] | u_i^t \in N_i(t)\}$ for representing vertex collision constraints. For representing edge collision constraints, we define, for $(u, w) \in E$:

$$\overline{E}^t(u, w) = \{(i, j) \in [\mathbf{N}] \times [\mathbf{N}] | (u_i^t, w_i^{t+1}) \in A_i(t), (w_j^t, u_j^{t+1}) \in A_j(t), i \neq j\}$$

For each robot $r_i \in \mathcal{R}$, and each arc $a \in A_i$, we introduce a binary variable $x(a) \in \{0, 1\}$ to indicate whether robot r_i traverses arc a in a feasible solution to the MAPF problem. Let $|A| = \sum_{i \in [\mathbf{N}]} |A_i|$, where $|A_i|$ denotes cardinality of set A_i. The 0-1 IP formulation for the MAPF problem is:

$$\underset{x \in \{0,1\}^{|A|}}{\text{minimize}} \sum_{i=1}^{\mathbf{N}} \sum_{a \in A_i} c_i(a) x(a) \tag{2}$$

$$\text{s.t.} \sum_{a \in \delta^{+}_{F_i}(s_i^0)} x(a) = 1, \forall i \in [\mathbf{N}] \tag{3}$$

$$\sum_{a \in \delta^{-}_{F_i}(u_i^t)} x(a) = \sum_{a \in \delta^{+}_{F_i}(u_i^t)} x(a), \forall i \in [\mathbf{N}], \forall t \in [\mathbf{T}-1], \forall u_i^t \in N_i(t) \tag{4}$$

$$\sum_{i \in \overline{V}^t(u)} \sum_{a \in \delta^{-1}_{F_i}(u_i^t)} x(a) \leq 1, \forall u \in V, \forall t \in [\mathbf{T}] \tag{5}$$

$$x(u_i^t, w_i^{t+1}) + x(w_j^t, u_j^{t+1}) \leq 1, \ \forall (u,w) \in E, \forall t \in \{0,1,...,\mathbf{T}-1\}, \forall (i,j) \in \overline{E}^t(u,w) \tag{6}$$

Equations (3) and (4) (a.k.a flow balance constraints) ensure that a **start-end path** is chosen for every robot, while Eqns (5) and (6) prevent vertex and edge collisions respectively.

4 Lower bounds from Cut Generation

We provide lower bounds to the MAPF problem using a Lagrangian relax and cut (LRC) scheme [8] that makes use of a cut generation procedure. In this section we describe our cut generating procedure, which will later be incorporated into the Lagrangian relax and cut (LRC) scheme described in Sec. 7.

Let $\mathbf{P}$ denote the MAPF polytope as shown below:

$$\mathbf{P} = \mathbf{conv}(x \in \{0,1\}^{|A|} | x \text{ satisfies } (3) - (6)) \tag{7}$$

where **conv** denotes convex hull. Given a $\bar{x} \in \{0,1\}^{|A|}$ that violates some constraint in (5) - (6), we develop a cut generating procedure that outputs a cut $a^T x \leq b$ that strictly separates $\bar{x}$ from $\mathbf{P}$ i.e. $\max_{x \in \mathbf{P}} a^T x \leq b < a^T \bar{x}$.

The cuts generated by our procedure are a form of projection cuts. The idea is to select a subset of arcs $S \subset A$, where assume $|S| = n$. We will construct a polytope $P(S) \subset \mathbb{R}^n$ such that $\text{Proj}_{x(S)}(\mathbf{P}) \subseteq P(S)$, where $x(S)$ denotes the variables corresponding to the arcs in S, and $\text{Proj}_{x(S)}(\mathbf{P})$ is the orthogonal projection of $\mathbf{P}$ onto the space spanned by the variables in $x(S)$:

$$\text{Proj}_{x(S)}(\mathbf{P}) = \{y \in \mathbb{R}^n : \exists w \in \mathbb{R}^{|A|-n}, \text{s.t. } (y,w) \in \mathbf{P}\} \tag{8}$$

In order to generate a cut that separates $\bar{x}$ from $\mathbf{P}$ we will output a face of $P(S)$, which separates $\text{Proj}_{x(S)}(\bar{x})$ and $P(S)$. Clearly, if $P(S)$ is a tight relaxation for $\text{Proj}_{x(S)}(\mathbf{P})$, then the cut obtained will also be deep.

Different choices for S give rise to different $P(S)$, so different cuts separating $\bar{x}$ and $\mathbf{P}$ can be derived by varying S. From the perspective of projection cuts, the collision avoidance constraints in Eqns (5), (6) contain variables belonging to a small spatio-temporal neighbourhood. For instance, if $\bar{x}$ violates Eqn (6), then Eqn (6) can be viewed as a cut which separates $\text{Proj}_{x(S)}(\mathbf{P})$ and $\text{Proj}_{x(S)}(\bar{x})$, where $S = \{(u_i^t, w_i^{t+1}), (w_j^t, u_j^{t+1})\}$. In this work, we will typically choose larger spatio-temporal neighbourhoods for selecting the arcs in S. Consequently, the cuts that are generated by our approach tend to be deeper.

On selecting S for an infeasible $\bar{x}$: The choice of S for a given infeasible $\bar{x}$ will not be made arbitrarily. An arbitrary choice for S can lead to poor cuts, and computing a good approximation to $\text{Proj}_{x(S)}(\mathbf{P})$ is challenging. To specify S, we first parameterize S by the sets $\mathcal{R}(S), T(S), L(S)$. $\mathcal{R}(S) \subseteq [\mathbf{N}]$ is a set of indices of robots, $T(S) = \{\mathbf{l}, \mathbf{l+1}, \dots, \mathbf{u}\}$ is a discrete interval where $\mathbf{l}, \mathbf{u} \in \mathbb{Z}_+$ and $\mathbf{u} < \mathbf{T}$. For each $i \in \mathcal{R}(S)$ and each time $t \in T(S)$, let $L_i^t(S)$ denote a set of nodes in $N_i(t)$. Denote $L(S) = \underset{i \in \mathcal{R}(S), t \in T(S)}{\cup} L_i^t(S)$, S is defined as the set of all incoming and outgoing arcs associated with nodes in $L(S)$

$$S = \underset{i \in \mathcal{R}(S), t \in T(S)}{\cup} \; \underset{v_i^t \in L_i^t(S)}{\cup} \left(\delta^+_{F_i}(v_i^t) \cup \delta^-_{F_i}(v_i^t)\right) \tag{9}$$

W.l.o.g let $\bar{x}$ contain a conflict between robots r_1, r_2 at time t_c, and let us denote the set of nodes from N_1, N_2 involved in the conflict by Z_{cf}. Note that if the conflict is a vertex conflict, then Z_{cf} contains 2 nodes, while for an edge conflict Z_{cf} contains 4 nodes. We say that S is an appropriate selection for $\bar{x}$ iff $1, 2 \in \mathcal{R}(S)$, $t_c \in T(S)$ and $Z_{cf} \subseteq L(S)$ i.e. loosely speaking S contains arcs relevant to the conflict present in $\bar{x}$. We provide an example for S below.

Consider the set of robot paths for r_1 and r_2 shown in Fig. 1. Robot r_1 moves from location $(3, 3)$ at time 4 to $(4, 3)$ at time 5, while r_2 moves from $(4, 3)$ at time 4 to $(3, 3)$ at time 5, so we have an edge collision at time 4.

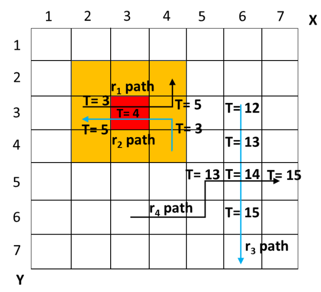


Fig 1. One possible choice of S for the edge conflict between r_1 and r_2 is: $\mathcal{R}(S) = \{1, 2\}$, and $T(S) = \{3, 4, 5\}$. For all $i \in [1, 2]$ and $\forall t \in T(S)$, $L_i^t(S)$ is set to nodes in $N_i(t)$ corresponding to all locations in the 3×3 grid centered at $(3, 3)$ (highlighted in yellow). S for our example can be obtained from the parameters specified by applying Eqn (9). Clearly the arcs in the edge conflict are present in S.

Defining $P(S)$: We retain as many relevant inequalities from the IP formulation in Sec. 3 for defining $P(S)$, thereby providing a tight relaxation for $\text{Proj}_{x(S)}(\mathbf{P})$.

$$P(S) = \mathbf{conv}\left(x(S) \in \{0,1\}^n | x \text{ satisfies } (11) - (14)\right) \tag{10}$$

$$\sum_{a \in \delta^-_{F_i}(v_i^t)} x(a) = \sum_{a \in \delta^+_{F_i}(v_i^t)} x(a), \; \forall i \in \mathcal{R}(S), \; \forall t \in T(S), \forall v_i^t \in L_i^t(S) \tag{11}$$

$$\sum_{j \in \overline{V}^t(v) \cap \mathcal{R}(S)} \; \sum_{a \in \delta^{-1}_{F_j}(v_j^t)} x(a) \leq 1, \; \forall t \in T(S), \forall i \in \mathcal{R}(S), \forall v_i^t \in L_i^t(S) \tag{12}$$

$$x(u_i^t, w_i^{t+1}) + x(w_j^t, u_j^{t+1}) \leq 1, \; \substack{\forall t \in \{l-1\} \cup T(S), \forall (i,j) \in \{(k,l) \in \mathcal{R}(S) \times \mathcal{R}(S) | k \neq l\}, \\ \forall (u,w) \in \{(p,q) \in E | (p_i^t, q_i^{t+1}), (q_j^t, p_j^{t+1}) \in S\}} \tag{13}$$

$$\sum_{a \in S \cap A_i(t)} x(a) \leq 1, \; \forall i \in \mathcal{R}(S), \forall t \in \{l-1\} \cup T(S) \tag{14}$$

Algorithm 1 PSGA for **CGLP**

```
1: Initialize: $k := 1$, $w^{(k)} = 0, \Delta = 0, \mathbf{z} = \text{Proj}_{x(S)}(\bar{x})$.
2: while Stopping criterion is not met do
3:     $v^{(k)} \in \arg \max_{v \in Vert(P(S))} v^T w^{(k)}$
4:     $w^{(k+1)} := \text{Proj}_{\|w\|_2 \leq 1} \left( w^{(k)} + \rho^{(k)}(\mathbf{z} - v^{(k)}) \right)$, where $\rho^{(k)} := \frac{1}{k}$
5:     if $H(w^{(k+1)}) > \max(0, \Delta)$ then
6:         $\Delta \leftarrow H(w^{(k+1)}), w^* \leftarrow w^{(k+1)}$
7:     $k := k + 1$
8: if $\Delta > 0$ then
9:     return $w^*$
```

Eqn (11) ensures flow balance for all nodes in $L(S)$. Eqn (12) prohibits vertex collisions on nodes in $L(S)$. Eqn (13) prohibits edge collisions over arcs in S. Eqn (14) ensures no two arcs present in S and belonging to the same arc layer in the time expanded graph of a robot are both simultaneously selected. Clearly Eqns (12) and (14) are implied from the MAPF IP formulation, while Eqns (11) and (13) are present in the MAPF IP formulation, hence $\text{Proj}_{x(S)}(\mathbf{P}) \subseteq P(S)$.

Separating $\bar{x}$ and $P(S)$: Recall, we assumed that $\bar{x}$ violates some constraint from the set of Eqns (5)-(6). Assuming we have selected an S that is appropriate for $\bar{x}$, then we know that some constraint in the set of Eqns (12)-(13) must be violated by $\bar{x}$. So by the strict hyper-plane separation theorem, we know that $\exists w \in \mathbb{R}^n$ such that $w^T z < w^T \text{Proj}_{x(S)}(\bar{x})$, $\forall z \in P(S)$. We can obtain such a w by solving the optimization problem **CGLP** in Eqn (15). $Vert(P(S))$ shown in Eqn (16) refers to the vertices of polytope $P(S)$.

$$\textbf{CGLP} : \max_{\|w\|_2 \leq 1} H(w), \text{ where } H(w) = w^T(\text{Proj}_{x(S)}(\bar{x})) - h(w), \text{ and} \tag{15}$$

$$h(w) = \max_{y \in P(S)} \{w^T y\} \iff \max_{v \in Vert(P(S))} \{w^T v\} \tag{16}$$

The objective in **CGLP** is a piece-wise concave function, so **CGLP** can be solved using the well known projected sub-gradient ascent (PSGA) method, shown in Alg. 1. For performing the maximization in line 3 of Alg. 1, we require the vertices of $P(S)$. Drawing inspiration from the works of [2, 7, 15], in Sec. 5 we propose a compact representation for the vertices of $P(S)$ in terms of a decision diagram. The compact representation will enable us to perform maximization in a reasonable amount of time, at least when $|\mathcal{R}(S)|, |L(S)|$ are not too large. We provide details on how cuts are utilized for obtaining lower bounds to the MAPF problem in Sec. 7, and how the bound is incorporated into a CBS procedure in Sec. 8. Sections 7 and 8 may be read independently of Sections 5 and 6.

5 Decision Diagram Representation

Borrowing notation from [7], we denote the decision diagram (DD) for $P(S)$ by $\mathcal{D}(S) = (\mathcal{U}, \mathcal{A}, f)$, where $\mathcal{U}$ represents a set of nodes, $\mathcal{A}$ represents arcs in a top-down multi-graph, f labels each an arc in $\mathcal{A}$ to some subset of arcs in S. $\mathcal{U}$ can be decomposed into $|T(S)| + 2$ layers $\mathcal{U}_0,\mathcal{U}_1, ...,\mathcal{U}_{|T(S)|+1}$, and $\mathcal{A}$ into $|T(S)| + 1$ layers $\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_{|T(S)|}$. $\mathcal{U}_0$ contains a single node **sr** called source and $\mathcal{U}_{|T(S)|+1}$ contains a single node **sk** called sink. The tail of any arc in layer j is connected to a node in $\mathcal{U}_j$ and its head to a node in $\mathcal{U}_{j+1}$.

To construct $\mathcal{D}(S)$ we will use the concept of a state transition diagram. The idea is to interpret each node in $\mathcal{U}$ as a state, a practice widely used for optimization using Decision Diagrams [5]. A state maps each $i \in \mathcal{R}(S)$ to a robot state. For $i \in \mathcal{R}(S)$, if robot r_i occupies location $v \in V$ at time $t \in T(S)$, then the state of robot r_i is defined as:

$$\begin{cases} v_i^t, \text{ if } v_i^t \in L_i^t(S) \\ \mathbf{o}, \text{ otherwise} \end{cases}$$

We motivate introducing **o** as a state for a robot with an example. Say robot r_1 traverses from location a at time 1 to location b at time 2 i.e r_1 traverses the arc $(a_1^1, b_1^2) \in A_1(1)$ (refer $A_i(t)$ notation from Sec. 3), and then traverses the arc $(c_1^2, d_1^3) \in A_1(2)$, then first observe that such a path for r_1 is infeasible to the MAPF problem. However, when $b_1^2, c_1^2 \notin L_1^2(S)$ recall that $P(S)$ does not enforce flow balance at the nodes b_1^2, c_1^2, and so $P(S)$ may contain a vertex corresponding to the infeasible MAPF path, since after-all $P(S)$ is only a relaxation to $\text{Proj}_{x(S)}(\mathbf{P})$. As we chose to adopt a state space representation for building $\mathcal{D}(S)$, introducing **o** allows us to interpret our example as: r_1 traverses state a_1^1 to state **o** using arc (a_1^1, b_1^2), and then transitions from **o** to d_1^3 using arc (c_1^2, d_1^3).

Observe that since $P(S)$ is parameterized with arc variables and any vertex of $P(S)$ corresponds to a series of arc traversals for the robots in $\mathcal{R}(S)$, we can interpret each vertex as a series of state transitions much the same way as we did earlier for the single robot. In constructing $\mathcal{D}(S)$, the overall aim is to ensure that there is a 1:1 correspondence between vertices in $P(S)$ and state transition paths in $\mathcal{D}(S)$. We next construct the nodes in $\mathcal{D}(S)$. Let us assume that $T(S) = \{\mathbf{l}, \mathbf{l+1}, ..., \mathbf{u}\}$. At time $t \in T(S)$, observe there are at most $\prod_{i\in\mathcal{R}(\mathcal{S})} (|L_i^t(S)| + 1)$ states for the robots. For each of those states at time $t \in T(S)$, we introduce a node in layer $t - \mathbf{l} + 1$ of $\mathcal{D}(S)$ i.e. $\mathcal{U}_{t-\mathbf{l}+1}$. For any node $u \in \mathcal{U}$, we shall use the notation $u[i]$ to denote the state of robot r_i in u. The node **sr** in layer $\mathcal{U}_0$ and node **sk** in layer $\mathcal{U}_{T(S)+1}$, both correspond to a state where $\mathbf{sr}[i] = \mathbf{sk}[i] = \mathbf{o}$, $\forall i \in \mathcal{R}(S)$. Some of the nodes (states) populated in $\mathcal{U}$, may contain vertex collisions. A node $u \in \mathcal{U}$ is said to contain a vertex collision iff $\exists i, j \in \mathcal{R}(S)$ such that both $u[i]$ and $u[j]$ are different from **o**, and $u[i], u[j]$ correspond to the same vertex in V. We remove all nodes that contain vertex collisions from $\mathcal{U}$, as such states can never be attained from any vertex of $P(S)$. Vertex collision checking with **o** was skipped because Eqn (12) has vertex collision constraints only for nodes in $L(S)$. The states for **sr**, **sk** was defined that way because Eqn (12) is unconcerned with vertex collisions at times $\mathbf{l} - 1, \mathbf{u} + 1$.

Before describing how to populate arcs in $\mathcal{D}(S)$, we take a brief detour. For the choice of parameters of S in Fig. 1, we show a portion of its corresponding

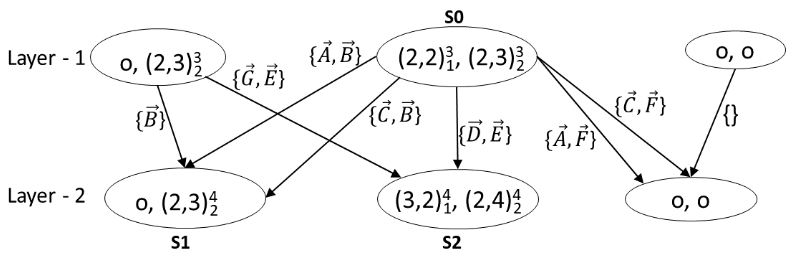


Fig. 2: A portion of the DD for the choice of S in Fig. 1 is shown above, where $\overrightarrow{A}= \big((2,2)^3_1,(2,1)^4_1\big)$, $\overrightarrow{B}= \big((2,3)^3_2,(2,3)^4_2\big)$, $\overrightarrow{C}= \big((2,2)^3_1,(1,2)^4_1\big)$, $\overrightarrow{D}= \big((2,2)^3_1,(3,2)^4_1\big)$, $\overrightarrow{E}= \big((2,3)^3_2,(2,4)^4_2\big)$, $\overrightarrow{F}= \big((2,3)^3_2,(1,3)^4_2\big)$, $\overrightarrow{G}= \big((3,1)^3_1,(3,2)^4_1\big)$.

DD in Fig. 2. Only a few states in layers $\mathcal{U}_1$,$\mathcal{U}_2$, and arcs in $\mathcal{A}_1$ with their labels are shown. For robot r_1 to transition from location $(2,2)$ at time 3 (denoted by state $(2,2)^3_1$) to $(3,2)$ at time 4, r_1 needs to transition with arc $\overrightarrow{D}$ (refer Fig. 2 for definition). Likewise, for r_2 to transition from state $(2,3)^3_2$ to $(2,4)^4_2$, r_2 needs to transition with arc $\overrightarrow{E}$. Since the transition between those states for r_1 and r_2 does not lead to a collision, we connect states **S0**, **S2** with an arc in $\mathcal{A}_1$ and label the arc with the set $\{\overrightarrow{D},\overrightarrow{E}\}$, see Fig. 2. From the definition of $L^4_1(S)$ provided in the example in Fig. 1, the reader can easily verify that r_1 can transition to state **o** at time 4 from $(2,2)^3_1$ using either $\overrightarrow{A}$ or $\overrightarrow{C}$. Consequently, note that r_1, r_2 can transition from **S0** to **S1** in 2 different ways as shown in Fig. 2.

Moving on from the example, we now provide a formal procedure to populate the arcs in $\mathcal{D}(S)$. Consider any node $v \in \mathcal{U}_k$ and any node $w \in \mathcal{U}_{k+1}$.To decide whether we should connect arcs from v to w, reduces to first determining whether w is a feasible state transition of v, and if yes, then determining all different ways in which the robots may transition from v to w. By repeating this process for all pairs of nodes in $\mathcal{U}$ occurring between consecutive layers, we can populate $\mathcal{A}$. To establish whether a state transition from v to w is feasible, for each $i \in \mathcal{R}(S)$ we will establish all the different ways in which robot r_i can transition from $v[i]$ to $w[i]$ by traversing some arc in S, and store this information in the set $h(v,w,i) \subset S$. One of the cases below will be applicable for populating $h(v,w,i)$:

1. If $k=0$ and $w[i] \in L^{\mathbf{1}}_i(S)$, then first observe that $v[i] = \mathbf{o}$ since v is the **sr** node. Robot r_i can use any one of the arcs from the set $\delta^-_{F_i}(w[i])$ to transition to $w[i]$, and so $h(v,w,i) = \delta^-_{F_i}(w[i])$.
2. If $1 \leq k < |T(S)|$, we consider separately the following 3 cases:
– If $v[i] \in L^{k+\mathbf{1}-1}_i(S)$, $w[i] \in L^{k+\mathbf{1}}_i(S)$ and $(v[i],w[i]) \in \delta^+_{F_i}(v[i])$, then clearly: $h(v,w,i) = \{(v[i],w[i])\}$.
– If $v[i] = \mathbf{o}$ and $w[i] \in L^{k+\mathbf{1}}_i(S)$, then :

$$h(v,w,i) = \{(p^{k+\mathbf{1}-1}_i, w[i]) \in \delta^-_{F_i}(w[i]) | p^{k+\mathbf{1}-1}_i \notin L^{k+\mathbf{1}-1}_i(S)\}$$

– If $v[i] \in L^{k+\mathbf{1}-1}_i(S)$ and $w[i] = \mathbf{o}$, then:

$$h(v,w,i) = \{(v[i], p^{k+\mathbf{1}}_i) \in \delta^+_{F_i}(v[i]) | p^{k+\mathbf{1}}_i \notin L^{k+\mathbf{1}}_i(S)\}$$

3. If $k = |T(S)|$ and $v[i] \in L_i^k(S)$, then $h(v,w,i) = \delta^+_{F_i}(v[i])$. Also note that w is **sk** node, and so $w[i] = \mathbf{o}$.
4. If $v[i] = w[i] = \mathbf{o}$, then $h(v,w,i) = \emptyset$ since there are no arcs in S that the robot can use to traverse between such a pair of states. However a state transition for the robot is still be feasible without using any arcs from S. For example, if at all times $t \in T(S)$ robot r_i does not use any arc in S in some start-end path which is feasible to the MAPF problem, then r_i is in state $\mathbf{o}$ at all $t \in T(S)$.

If for some $i \in \mathcal{R}(S)$, $v[i], w[i]$ are not both $\mathbf{o}$, and no arc could be found for $h(v,w,i)$ by analyzing 1, 2 and 3, then robot r_i cannot transition from $v[i]$ to $w[i]$. In such a case, we can safely conclude that the transition from v to w is not feasible, and so we do not need to insert any arc from v to w in $\mathcal{A}$. Assuming that this is not the case for v, w, we then proceed to check whether arcs can be added from v to w. As the function $h(v,w,i)$ corresponds to the different ways in which the robot r_i transitions from state $v[i]$ to $w[i]$, it is only natural that the elements of the set $H(v,w) = \prod_{i \in \mathcal{R}(S)} h(v,w,i)$ correspond to all the different ways in which robots can transition from v to w. Note that $H(v,w)$ is a set product, and so each element of $H(v,w)$ is itself a subset of S (includes the empty set). For any $i \in \mathcal{R}(S)$, note that each element of $H(v,w)$ contains at most one arc from A_i. Some elements of $H(v,w)$ may contain arcs from S, wherein robots transitioning using those arcs will result in an edge collision. We remove all those elements from $H(v,w)$ which will result in edge collisions. Corresponding to each element remaining in $H(v,w)$ after the previous edge collision filtration step, we add an arc a from v to w in $\mathcal{A}_k$ and label (cf. f function in definition of $\mathcal{D}(S)$) a by the arcs from S present in the element of $H(v,w)$.

There is a 1:1 correspondence between source-sink (**sr**−**sk**) paths in $\mathcal{D}(S)$ and vertices of $P(S)$, i.e. if x_v is a vertex of $P(S)$ and let $Q = \{a \in S | x_v(a) = 1\}$, then there is a **sr**−**sk** path in $\mathcal{D}(S)$ such that the labels occurring on the path coincide exactly with Q. Conversely for any **sr**−**sk** path, if $\bar{S} \subset S$ are labels occurring on the path, then there is a vertex x_v in $P(S)$ such that $x_v(\bar{S}) = 1, x_v(S \backslash \bar{S}) = 0$.

Computing line 3 in Alg. 1 using $\mathcal{D}(S)$: To perform the maximization in line 3, we can make use of the correspondence between vertices of $P(S)$ and $\mathbf{sr} - \mathbf{sk}$ paths in $\mathcal{D}(S)$. We assign a cost to each arc in $\mathcal{A}$ depending on the labels on the arc. For instance, if arc $a \in \mathcal{A}$ is labelled with b_1, b_2, where $b_1, b_2 \in S$, then we simply assign a cost of $w^{(k)}(b_1) + w^{(k)}(b_2)$ to a, where $w^{(k)}(b_i)$ is the value corresponding to b_i in vector $w^{(k)}$. If a is not labeled with any arc from S, then we assign a cost of 0. After setting costs to all arcs in $\mathcal{A}$ in the manner just described, obtaining the arg max vertex in line 3 is equivalent to obtaining any longest $\mathbf{sr} - \mathbf{sk}$ path in $\mathcal{D}(S)$. The computational effort needed to obtain the longest path is $\mathcal{O}(|\mathcal{A}|)$, since $\mathcal{D}(S)$ is a directed acyclic graph.

6 Templates for grids

As conflict locations and robots involved in conflicts vary, a different set of parameters for S may need to be chosen in order to generate a cut for each conflict.

Consequently, a different projection polytope ($P(S)$) needs to be built for each conflict, which is computationally expensive. When G is a 4 or 8-connectivity grid, the neighborhood relative to any location on the grid is same across all locations on the grid, a property that allows us to build *Templates*.

Let us denote the polytope $P(S)$ described in Fig. 1 by P_1. Now consider the vertex conflict for robots r_3 and r_4 at time 14 shown in Fig. 1. For this conflict, we can create a polytope $P(S_2)$ with parameters : $\mathcal{R}(S_2) = \{3, 4\}$, and $T(S_2) = \{13, 14, 15\}$. For all $i \in [3, 4]$ and $\forall t \in T(S_2)$, $L_i^t(S_2)$ is set to nodes in $N_i(t)$ corresponding to all locations in the 3×3 grid centered at $(6, 5)$. Clearly $P(S_2)$ can also output a cut for the conflict between r_3, r_4. While polytopes $P_1, P(S_2)$ lie in different dimensions, the facial structure of both polytopes are identical. If we substitute r_1 for r_3, r_2 for r_4, advance the interval $T(S_2)$ by 10 time units, and translate all locations in $L_{\cdot}^{\cdot}(S_2)$ by 3 units along the negative X-axis and by 2 units along the negative Y-axis, we get back all the parameters for S described in Fig. 1. Hence, we claim that both P_1 and $P(S_2)$ are manifestations of the same base template polytope.

While working with structured graphs such as grids, we can precompute a library of different templates, and use those templates to generate all cuts. By spatio-temporally shifting the parameters of the template about the conflict, multiple cuts can be generated using the same base template. While generating cuts with templates, some locations may be physically blocked on the grid, and so certain states in the DD representation of the template cannot be attained by the robots. In that case, we adjust the longest $\mathbf{sr} - \mathbf{sk}$ path computation procedure to avoid paths that pass through infeasible states.

7 Lagrangian Relax-and-Cut

Equipped with the cut generation oracle from the previous section, we will now describe a Non-Delayed Lagrangian Relax-and-Cut (LRC) procedure [13] to generate lower bounds to the MAPF problem. Consider the optimization problem shown in Eqn (17) obtained by omitting all collision constraints from the MAPF IP formulation shown in Sec. 3.

$$\min_{x \in \{0,1\}^{|A|}} \{c^T x | x \text{ satisfies Eqns } (3) - (4)\} \tag{17}$$

Notice, that the optimal solution to Eqn (17) (call it $\bar{x}$) consists of robot start-end paths, which potentially may contain vertex and edge conflicts. We can use our cut generation technique from the previous section, and generate cuts (denote the set of inequalities generated by $Cx \leq d$) that separate $\bar{x}$ from $\mathbf{P}$. LRC incorporates the cuts generated by solving the Lagrangian dual problem:

$$\max_{\lambda \geq 0} \min_{x \in \{0,1\}^{|A|}} \{c^T x + \lambda^T (Cx - d) | x \text{ satisfies Eqns } (3) - (4)\} \tag{18}$$

Eqn (18) is solved using the iterative PSGA procedure, where at each iteration λ is updated by using the solution to the inner minimization problem, which

Algorithm 2 Lagrangian relax and cut algorithm

```
1: Given: MAX_CUT_ITER, $c$
2: Output: Inequalities $Cx \leq d$, optimal Lagrangian multipliers $\lambda^*$, and upper
   bound(UB).
3: Initialize: $k = 0$, $C = \emptyset, d = \emptyset$, $\lambda = \emptyset$, UB $= \infty$
4: repeat
5:     $\bar{x}_k = \arg \min_{x \in \{0,1\}^{|A|}} \{c^T x + \lambda^T (Cx - d) \,|x \text{ satisfies Eqns } (3) - (4)\}$
6:     $\lambda := (\lambda + \rho_k (C\bar{x}_k - d))_+$
7:     if $\bar{x}_k$ contains conflicts then
8:         if $k <$ MAX_CUT_ITER then
9:             Generate cuts $E_k x \leq f_k$ separating $\bar{x}_k$ from $\mathbf{P}$
10:            Append $E_k x \leq f_k$ to $Cx \leq d$. Introduce Lagrangian multipliers for $E_k x \leq$
               $f_k$ initialized all to 1, and append it to the vector $\lambda$.
11:        Repair $\bar{x}_k$ to generate non-conflicting paths. If repair is successful and cost of
           repaired solution is less than UB, update UB.
12:    else
13:        if Cost($\bar{x}_k$) < UB then
14:            UB := Cost($\bar{x}_k$)
15:    $k := k + 1$
16: until Termination criterion is met
```

is a min-cost flow problem. Denote the optimal solution to the min-cost flow problem at iteration k of PSGA by $\bar{x}_k$. Note that $\bar{x}_k$ represent start-end paths for robots, and potentially contains conflicts. The key innovation of LRC is that, if $\bar{x}_k$ contains conflicts, cuts are generated to separate $\bar{x}_k$ and $\mathbf{P}$. Denote the cuts generated by $E_k x \leq f_k$. $E_k x \leq f_k$ is incorporated into the optimization problem by dualizing them with appropriate Lagrangian multipliers. In other words, we can think of this operation as expanding $Cx \leq d$ at each iteration by including $E_k x \leq f_k$. The motivation to include cuts at each iteration is to strengthen the dual bound by including inequalities (valid for $\mathbf{P}$) that are violated by the current dual solution. Deeper the included cut, greater will be the magnitude of the dual gradient term in line 6, and the hope is that ascent along the gradient will lead to a solution with a higher objective. After a few iterations, no more cuts are added, at which point LRC becomes a standard dual ascent scheme.

A schematic overview of the LRC scheme described above is shown in Alg. 2. Alg. 2 takes as input a positive integer MAX_CUT_ITER, to decide when to stop adding cuts. In line 6, the λ vector is updated with step size ρ. In our implementation, ρ_k was set to $\mathcal{O}(\frac{1}{k})$ up-to MAX_CUT_ITER iterations, after which the step size was set according to the scheme in [1], to accelerate convergence. At iteration k, if $\bar{x}_k$ contains conflicts, we apply a primal repair procedure to try and convert $\bar{x}_k$ into a conflict free solution, for generating an upper bound.

Our primal repair procedure identifies a maximal independent set (MIS) of non conflicting robots from the paths provided in $\bar{x}_k$, and fixes the path of the robots in the MIS to as they are in $\bar{x}_k$. The paths for the remaining robots are computed sequentially, where the path assigned to a robot is the shortest start-end path that does not collide with any path fixed previously to other robots. The order in which the robots are chosen for their path to be computed is determined dynamically using the rule in [17]. If the procedure fails to compute a path for robot r, then the primal repair procedure is unsuccessful for the current iteration.

8 An LRC-based Search Node Evaluation Function

In this section we will describe a new evaluation function for Conflict Based Search (CBS) that uses the output of the LRC procedure. We briefly describe only the relevant portions of CBS to our work. CBS performs a best first search on a search tree, where the most promising node among the previously unexplored nodes of the search tree is selected for exploration by applying an evaluation function. Each node in the search tree is characterized by a set of arcs that the robots are prohibited from using. An evaluation function takes as input any search tree node, and outputs a cost that does not overestimate the cost of the optimal solution to the MAPF problem with the added constraint that robots do not use any arcs prohibited in the search node. The node with the least evaluation cost is then selected. If the cost outputted by the evaluation function closely matches the true lower bound at every search tree node, then we should expect good search performance. Our goal is to improve existing evaluation functions.

Given a search node sn, let us denote the set of arcs that are prohibited in the node by $\bar{\mathcal{A}}_{sn} \subset \cup_{i\in[\mathbf{N}]} A_i$. We provide an evaluation function $\hat{f}_1(\cdot)$ based on Lagrangians. Before the search tree is created we apply Alg. 2, and let $\hat{C}x \leq \hat{d}$, $\hat{\lambda}$, and UB denote the outputs. $\hat{f}_1(sn)$ is computed as:

$$\hat{f}_1(sn) = \min_{\substack{x\in\{0,1\}^{|A|} \\ x(a)=0,\forall a\in\bar{\mathcal{A}}_{sn}}} \{c^T x + \hat{\lambda}^T(\hat{C}x - \hat{d}) | x \text{ satisfies Eqns } (3)-(4)\} \tag{19}$$

The validity of $\hat{f}_1(sn)$ as an evaluation function follows from the fact that $\hat{f}_1(sn)$ is a Lagrangian dual function where, $\hat{\lambda} \geq 0$ and the set of inequalities $\hat{C}x \leq \hat{d}$ is valid for the feasible region of sn. Note that $\hat{f}_1(sn)$ can be computed using any shortest path algorithm on the time expanded graphs, but with arc costs reflecting the objective shown in Eqn (19). Our evaluation function is similar to obtaining Lagrangian lower bounds in Constraint Programming [3, 4, 10].

We can combine our proposed evaluation function with any other evaluation function f previously proposed for the MAPF problem (see [9, 12]), by taking the maximum i.e. $\max(f(sn), \hat{f}_1(sn))$ to yield a stronger evaluation function than either just f or $\hat{f}_1$. If $\max(f(sn), \hat{f}_1(sn)) \geq UB$, we can omit descendants of sn in the remainder of the search procedure i.e. we can prune the node sn.

We designed another evaluation function $\hat{f}_2(\cdot)$ inspired directly from the minimum vertex cover (MVC) heuristic of [9] and WDG heuristic of [12]. Observe that the minimization in Eqn (19) can be performed independently for the robots. Let us re-write Eqn (19) as $\hat{f}_1(sn) = -\hat{d}^T\hat{\lambda} + \sum_{i\in[\mathbf{N}]} \hat{f}_1(sn, i)$. Similar to the approach in WDG heuristic, for each pair of robots we can solve a 2-Agent MAPF problem to obtain lower bounds on the sum of pairwise costs. In this work, we use Lagrangian arc costs to obtain the pairwise bounds. The arc costs are modified to the Lagrangian objective shown in Eqn (19). Say for robots r_i, r_j, the optimal cost to the 2 agent MAPF problem with Lagrangian arc costs (i.e., the cost of each arc is set to the arc's co-efficient in $c^T x + \hat{\lambda}^T \hat{C} x$) is denoted by $l_{ij}(\hat{\lambda}, sn)$, we propose a node evaluation function $\hat{f}_2(sn)$ as:

$$\begin{aligned} \hat{f}_2(sn) = \min_{y \in \mathbb{R}^{\mathbf{N}}} & -\hat{d}^T \hat{\lambda} + \sum_{i \in [\mathbf{N}]} y_i \\ \text{s.t. } & y_i + y_j \geq l_{ij}(\hat{\lambda}, sn), \forall i, j \in [\mathbf{N}] \times [\mathbf{N}] \text{ and } i < j \\ & y_i \geq \hat{f}_1(sn, i), \forall i \in [\mathbf{N}] \end{aligned} \tag{20}$$

9 Experimental Evaluation

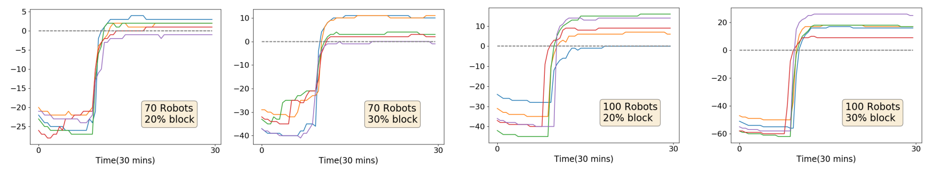
The primary goals of our experiments are to understand the additional value that our approach can bring to existing search methods, and how its performance is influenced by the characteristics of the problem.

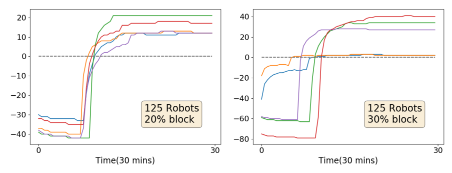
Experimental Setup As baseline search method, we implemented a state-of-the-art variant of conflict-based search, called DG [12], however our implementation does not include the run time reduction techniques and MDD merging technique proposed in [12]. When determining whether a pair of robots should share an edge in the conflict graph constructed in DG, we applied a two-agent MAPF solver to check this condition. As branching rule, we implement the rule proposed for ICBS [6], i.e., prioritizing cardinal over semi-cardinal over non-cardinal conflicts. We will refer to our implementation of DG as DG*. Our LRC approach implements DG with $\hat{f}_2(\cdot)$ as node evaluation function. We selected $\hat{f}_2(\cdot)$ over $\hat{f}_1(\cdot)$, because Eqn (20) implies $\hat{f}_2(sn) \geq \hat{f}_1(sn)$. We denote the combination of DG* with $\hat{f}_2(\cdot)$ by LR-DG*.

All experiments in this paper were carried out on an Intel 4 core i7-4790 processor running at 3.6 GHz with 16GB RAM, and the program was written in C++. All our experiments were conducted on 30×30 4-connectivity grids, where some % of the locations on the grid are randomly chosen and blocked. The start and end locations for the robots on the grid are also randomly assigned. The makespan constraint $\mathbf{T}$ was set to 3 more than the shortest time it took for all robots to reach their goal from start when collision constraints are omitted.

In all our experiments for comparing DG* with LR-DG*, we allocated a time limit of 30 minutes for both algorithms. The 30 minutes allocated to LR-DG* is further split as follows. A time limit of 10 minutes was allocated beyond which cuts are not added in Alg. 2 and MAX_CUT_ITER was set to 1000. Optimizing the Lagrangian multipliers using the accelerated step size update rule, in practice takes $\approx$ 2-4 minutes for the 100 robot instances. The remaining time was spent in performing conflict based search with $\hat{f}_2$ as evaluation function.

As all experiments were conducted on a 4-connectivity grid, we could precompute a template library. Our template library consisted of 64 templates with





Block \ Robots	30	50	70	100	125
10%	0.1	-0.1	-0.3	-0.6	1.5
20%	0	-0.4	1	7.9	14.7
30%	-0.3	1.7	5.8	14.8	17.8

Fig. 4: On Y-axis, we plot the difference in lower bounds outputted by LR-DG* and DG* with time for 5 instances. For each problem scenario, the table reports the difference in bounds averaged over 10 instances after 30 mins of simulation per instance.

each template parameterized by 3 robots, time horizon (cf. $T(S)$) of 5 time units, and $|L_i^t(S)|$ varied between 6 - 9 nodes. This translates roughly to 0.2 million arcs in the size of the DD per template.

Experimental Results To study the impact of problem characteristics on our algorithm performance, we considered problem scenarios with different blocked locations (%) and number of robots. For each such scenario, we record the progress of our solution method's lower bound with time. In Fig. 4, we graphically show the difference between LR-DG* and DG* lower bounds with time for different scenarios, and the results after 30 minutes are summarized in the table.

We first explain why the difference in bounds between the 2 methods looks like a step size function. Initially, the LR-DG* bound lags behind the DG* bound as it performs the cut addition phase of Alg. 2. In the cut addition phase, the LR-DG* lower bound is not improving much. On the other hand the MVC heuristic in DG* is able to quickly identify pairs of robots that are in some sense constraining one another, and by branching on their conflicts it is able to make rapid progress initially. Once LR-DG* enters into the Lagrangian multipliers optimization phase with accelerated step size update rule, we see a marked improvement in the LR-DG* bound.

From the table shown in Figure 4, one can observe the following trends in the gap between the lower bounds of the two methods. For a fixed number of robots, we see that as the block % increases, LR-DG* dominates over the DG* bound. Also, for a fixed block %, we observe that the gap between the 2 methods increases as the number of robots increases. The lower bound to the MAPF problem computed just after Lagrangian Relax and Cut phase i.e after Alg. 2 in many cases dominates the bound obtained from DG* after 30 minutes. For 100 robot problems and 20% blocked cells, on average (over 10 instances) we observed this gap to be 4.89, and for 30% blocked cells the gap was 11.7. These results clearly indicate that the cuts generated in the LRC phase are strong, and capable of generating strong lower bounds for the given objective.

We explain the results observed. It is clear that when not many collisions are expected between robots, then it is unlikely that the robots have to wait for another robot or take a longer route. Trying to raise the lower bound with cuts is unlikely to result in an improvement of the lower bound, which is why we see that when the the number of robots is few and/or blocked cells are also few, LR-DG* is unable to do any better than DG*. However, when the expected number of collisions is large, we see that LR-DG* generates strong bounds which reflects the strength of the cuts generated during the LRC phase. We explain this performance using an analogy. In environments which contain a lot of spatio-temporal bottleneck regions, i.e., local regions in time where many robots need to pass through to reach their goal, then by simply analyzing paths of robots within the bottleneck region, we may be able to infer facts such as, at least one robot must wait or take a longer route in order to pass through the bottleneck region without colliding. The strength of the inference improves as more robots are included in the analysis. Through the use of templates, our approach essentially focuses on a localized spatio-temporal neighbourhood. Our cut generating templates are able analyze all feasible paths through the neighbourhood at once for the robots parameterizing the template, thus going beyond pairwise analysis of robot paths, thereby able to output strong cuts.

Despite LR-DG* producing stronger lower bounds than DG*, in general we observed that LR-DG* was unable to prove optimality for any problem that DG* also could not. The results indicates a need for stronger lower bounds and a better primal heuristic than the one used in this work for proving optimality. For problems that were solved to optimality by both DG* and LR-DG*, we compared the number of search tree nodes expanded. For 30 robot problems, LR-DG* on average expanded 37% fewer nodes, however the % reduction in nodes across instances displayed high variance. On many problems, the fact that LR-DG* proved optimality during LRC phase itself has skewed the results. In general however, observe that since the cuts in the procedure have been generated at the root node, their utility diminishes as the depth of the search node increases. In future work, we will generate new cuts at each search node.

10 Conclusions

We proposed a new polyhedral approach for MAPF based on lower-dimensional polytopes called ‘templates’, which allows us to simultaneously analyze the paths of a number of robots within a spatio-temporal neighbourhood. We used decision diagrams to represent these templates and developed a cut generation scheme. The templates are translated spatio-temporally over the input graph to generate cuts for paths with conflicts. To obtain a lower bound, we embedded the cut generation into a Lagrangian Relax-and-Cut procedure. We incorporated the lower bound as a node evaluation function in a conflict-based search procedure. Our experimental results demonstrated that our lower bounds are particularly effective when the MAPF problem is very constrained due to large number of agents and\or fewer traversable paths on the input graph.

References

1. Baker, B.M., Sheasby, J.: Accelerating the convergence of subgradient optimisation. European Journal of Operational Research **117**(1), 136–144 (1999)
2. Becker, B., Behle, M., Eisenbrand, F., Wimmer, R.: Bdds in a branch and cut framework. In: International Workshop on Experimental and Efficient Algorithms. pp. 452–463. Springer (2005)
3. Benoist, T., Laburthe, F., Rottembourg, B.: Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In: CPAIOR. vol. 1, pp. 15–26 (2001)
4. Bergman, D., Cire, A.A., van Hoeve, W.J.: Improved constraint propagation via lagrangian decomposition. In: International Conference on Principles and Practice of Constraint Programming. pp. 30–38. Springer (2015)
5. Bergman, D., Cire, A.A., Van Hoeve, W.J., Hooker, J.: Decision diagrams for optimization, vol. 1. Springer (2016)
6. Boyarski, E., Felner, A., Stern, R., Sharon, G., Tolpin, D., Betzalel, O., Shimony, E.: Icbs: improved conflict-based search algorithm for multi-agent pathfinding. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
7. Davarnia, D., van Hoeve, W.J.: Outer approximation for integer nonlinear programs via decision diagrams. Mathematical Programming pp. 1–40 (2020)
8. Escudero, L.F., Guignard, M., Malik, K.: A lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships. Annals of Operations Research **50**(1), 219–237 (1994)
9. Felner, A., Li, J., Boyarski, E., Ma, H., Cohen, L., Kumar, T.S., Koenig, S.: Adding heuristics to conflict-based search for multi-agent path finding. In: Twenty-Eighth International Conference on Automated Planning and Scheduling (2018)
10. Khemmoudj, M.O.I., Bennaceur, H., Nagih, A.: Combining arc-consistency and dual lagrangean relaxation for filtering csps. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. pp. 258–272. Springer (2005)
11. Lam, E., Le Bodic, P., Harabor, D., Stuckey, P.J.: Branch-and-cut-and-price for multi-agent pathfinding. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), International Joint Conferences on Artificial Intelligence Organization. pp. 1289–1296 (2019)
12. Li, J., Boyarski, E., Felner, A., Ma, H., Koenig, S.: Improved heuristics for multi-agent path finding with conflict-based search. In: International Joint Conference on Artificial Intelligence. pp. 442–449 (2019)
13. Lucena, A.: Non delayed relax-and-cut algorithms. Annals of Operations Research **140**(1), 375–410 (2005)
14. Sharon, G., Stern, R., Felner, A., Sturtevant, N.R.: Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence **219**, 40–66 (2015)
15. Tjandraatmadja, C., van Hoeve, W.J.: Target cuts from relaxed decision diagrams. INFORMS Journal on Computing **31**(2), 285–301 (2019)
16. Wagner, G., Choset, H.: M*: A complete multirobot path planning algorithm with performance bounds. In: 2011 IEEE/RSJ international conference on intelligent robots and systems. pp. 3260–3267. IEEE (2011)
17. Wang, J., Li, J., Ma, H., Koenig, S., Kumar, T.: A new constraint satisfaction perspective on multi-agent path finding: Preliminary results. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2253–2255. International Foundation for Autonomous Agents and Multiagent Systems (2019)

18. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI magazine **29**(1), 9–9 (2008)
19. Yu, J., LaValle, S.M.: Planning optimal paths for multiple robots on graphs. In: 2013 IEEE International Conference on Robotics and Automation. pp. 3612–3617. IEEE (2013)
20. Yu, J., LaValle, S.M.: Structure and intractability of optimal multi-robot path planning on graphs. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)