Cost Splitting for Multi-Objective Conflict-Based Search

Cheng Ge[1*], Han Zhang[2*], Jiaoyang Li[3], Sven Koenig[2]

[1] Tsinghua University
[2] University of Southern California
[3] Carnegie Mellon University
gec19@mails.tsinghua.edu.cn, zhan645@usc.edu, jiaoyangli@cmu.edu, skoenig@usc.edu

Abstract

The Multi-Objective Multi-Agent Path Finding (MO-MAPF) problem is the problem of finding the Pareto-optimal frontier of collision-free paths for a team of agents while minimizing multiple cost metrics. Examples of such cost metrics include arrival times, travel distances, and energy consumption. In this paper, we focus on the Multi-Objective Conflict-Based Search (MO-CBS) algorithm, a state-of-the-art MO-MAPF algorithm. We show that the standard splitting strategy used by MO-CBS can lead to duplicate search nodes and hence can duplicate the search effort of MO-CBS. To address this issue, we propose two new splitting strategies for MO-CBS, namely cost splitting and disjoint cost splitting. Our theoretical results show that, when using either splitting strategy, MO-CBS maintains its completeness and optimality guarantees. Our experimental results show that disjoint cost splitting, our best splitting strategy, speeds up MO-CBS by up to two orders of magnitude and substantially improves its success rates in various settings.

Introduction

The Multi-Agent Path Finding (MAPF) problem is the problem of finding a set of collision-free paths for a team of agents. It is related to many real-world applications (Wurman, D'Andrea, and Mountz 2008; Morris et al. 2016). Solving it optimally is known to be NP-hard for various objective functions (Yu and LaValle 2013; Ma et al. 2016). In this paper, we study a variant of the MAPF problem called the Multi-Objective MAPF (MO-MAPF) problem (Ren, Rathinam, and Choset 2022). In MO-MAPF, a solution is a set of collision-free paths for all agents, and we consider multiple cost metrics for each solution. Many real-world applications of MAPF can be viewed as multi-objective optimization problems. For example, in multi-robot systems, some interesting cost metrics are travel distance, energy consumption, and risk. The objective of MO-MAPF is to find the Pareto-optimal frontier, that is, all solutions that are not dominated by any other solutions, where a solution Π dominates another solution Π' iff the cost of Π is no larger than the cost of Π' for every cost metric and the cost for at least one cost metric is smaller.

*These authors contributed equally to this work

Solving the MO-MAPF problem is quite different from solving classic MAPF as it tasks us with finding the Pareto-optimal frontier, which is different from the tasks of most, if not all, existing MAPF algorithms. There exist only a few MO-MAPF algorithms. MO-M* (Ren, Rathinam, and Choset 2021) generalizes M* (Wagner and Choset 2015), an algorithm for solving MAPF optimally, to MO-MAPF. However, experimental results (Ren, Rathinam, and Choset 2022) show that MO-M* is slower than Multi-Objective Conflict-Based Search (MO-CBS), the other MO-MAPF algorithm, in various domains.

MO-CBS (Ren, Rathinam, and Choset 2022) generalizes Conflict-Based Search (CBS) (Sharon et al. 2015), which is also an algorithm for solving MAPF optimally, to MO-MAPF. Here, conflicts refer to collisions between agents. In MAPF, CBS first finds a minimum-cost path for each agent without considering conflicts. To resolve a conflict, CBS splits into two subproblems, each with a new constraint imposed on either one of the conflicting agents to prevent the conflict from happening again. CBS replans for the constrained agent in each subproblem by finding a minimum-cost path for it that satisfies all constraints in the subproblem. CBS repeats the conflict resolution process until it finds a solution. In MO-MAPF, MO-CBS is similar to CBS and resolves conflicts by imposing new constraints on agents. One difference between them is that, when replanning for the constrained agent, MO-CBS finds all Pareto-optimal paths for it that satisfy the constraints and creates a new subproblem for each of them. The resulting subproblems have the same set of constraints and differ from each other only with respect to the path of the constrained agent. As we will show in this paper, these similar subproblems can lead to substantial duplicate search effort.

In this paper, we dramatically speed up MO-CBS by improving its splitting strategy. We propose two new splitting strategies for MO-CBS, namely cost splitting and disjoint cost splitting. Both cost splitting and disjoint cost splitting impose additional constraints on the costs of the paths for each constrained agent during splitting. Therefore, each subproblem is more constrained, and, when resolving conflicts in its following subproblems, MO-CBS generates fewer subproblems and hence has less duplicate search effort. Our theoretical results show that, when using either of the new splitting strategies, MO-CBS maintains its completeness and

optimality. Our experimental results show that the new splitting strategies substantially improve the success rates and runtimes of MO-CBS in various domains. For many MO-MAPF instances, the runtime speedups are more than $25\times$, and the maximum runtime speedup is more than $125\times$.

Terminology and Problem Definition

In this paper, we use boldface font to denote vectors, sets of vectors, or vector functions. We use v_i to denote the i-th component of vector or vector function $\mathbf{v}$. Given two vectors $\mathbf{v}$ and $\mathbf{v}'$ of the same length N, their addition is defined as $\mathbf{v} + \mathbf{v}' = [v_1 + v'_1, v_2 + v'_2 \dots v_N + v'_N]$, and their component-wise maximum is defined as $\mathbf{comax}(\mathbf{v}, \mathbf{v}') = [\max(v_1, v'_1), \max(v_2, v'_2) \dots \max(v_N, v'_N)]$. We say that $\mathbf{v}$ *weakly dominates* $\mathbf{v}'$, that is, $\mathbf{v} \preceq \mathbf{v}'$, iff $v_i \leq v'_i$ for all $i = 1, 2 \dots N$. We say that $\mathbf{v}$ *dominates* $\mathbf{v}'$, that is, $\mathbf{v} \prec \mathbf{v}'$, iff $\mathbf{v} \preceq \mathbf{v}'$ and there exists an $i \in \{1, 2 \dots N\}$ with $v_i < v'_i$. Given a set of vectors $\mathbf{S}$, we use $\mathbf{ND}(\mathbf{S}) = \{\mathbf{v} \in \mathbf{S} \mid \forall \mathbf{u} \in \mathbf{S}, \mathbf{u} \not\prec \mathbf{v}\}$ to denote the set of all undominated vectors in $\mathbf{S}$.

Property 1. *Given three vectors* $\mathbf{v}_1$, $\mathbf{v}_2$, *and* $\mathbf{u}$, *we have* $\mathbf{comax}(\mathbf{v}_1, \mathbf{v}_2) \preceq \mathbf{u}$ *iff* $\mathbf{v}_1 \preceq \mathbf{u}$ *and* $\mathbf{v}_2 \preceq \mathbf{u}$.

Define the *dominated region* $D(\mathbf{v}) = \{\mathbf{v}' \in \mathbb{R}^N \mid \mathbf{v} \preceq \mathbf{v}'\}$ of a vector $\mathbf{v}$ as the set of vectors that $\mathbf{v}$ weakly dominates. Property 1 shows that $D(\mathbf{comax}(\mathbf{v}_1, \mathbf{v}_2))$ is the intersection of $D(\mathbf{v}_1)$ and $D(\mathbf{v}_2)$. This geometric interpretation of $\mathbf{comax}$ is important for understanding some key steps of our new splitting techniques.

An MO-MAPF instance is defined by a weighted directed graph $G = \langle V, E \rangle$ and a set of m agents $\{a_1, a_2 \dots a_m\}$, each a_i associated with a start vertex $s_i \in V$ and a goal vertex $g_i \in V$. V is the set of vertices that the agents can stay at, and E is the set of directed edges that the agents can move along, each associated with an edge cost, which is a positive vector of length N. An edge pointing from $u \in V$ to $v \in V$ is denoted as $\langle u, v \rangle \in E$. Additionally, we allow a self-pointing edge at each vertex, which corresponds to an agent waiting at its current vertex.

In each timestep, an agent either moves along an edge (that is, moves to a neighboring vertex or waiting at its current vertex) or terminates at its goal vertex (that is, waits at the goal vertex forever and does not move again). Let N denote the number of cost metrics in an MO-MAPF instance. The cost of an action is the cost of the corresponding edge if the action is a move action along an edge and a zero vector of length N if it is a terminate action. A *path* π of an agent is a sequence of actions that leads it from its start vertex to its goal vertex and ends with a terminate action. Its *cost* $\mathbf{c}(\pi)$ is the sum of the costs of its actions. There are two types of conflicts. A *vertex conflict* happens when two agents stay at the same vertex simultaneously, and an *edge conflict* happens when two agents swap their vertices simultaneously. A *solution* Π is a set of conflict-free paths for all agents. Its *cost* $\mathbf{c}(\Pi)$ is the sum of the costs of its paths.

Given an MO-MAPF instance (respectively an MO-MAPF instance and an agent), a solution (respectively a path) is *Pareto-optimal* iff its cost is not dominated by the cost of any other solution (respectively path); the *Pareto-optimal frontier* is the set of all Pareto-optimal solutions (respectively paths); and a *cost-unique Pareto-optimal frontier* is a maximal subset of the Pareto-optimal frontier that does not contain any two solutions (respectively paths) of the same cost.

The task of the MO-MAPF problem is to find a cost-unique Pareto-optimal frontier (of solutions). We can easily realize different objectives by assigning different edge costs. For example, we can use *travel time*, a common MAPF cost metric, as the i-th cost metric by setting the i-th component of every edge cost to one.

Algorithmic Background

We review two existing algorithms on which we build our techniques, namely CBS for the MAPF problem and MO-CBS for the MO-MAPF problem.

CBS

CBS is a complete and optimal two-level MAPF algorithm. On the high level, CBS performs a best-first search on a *Constraint Tree* (CT). Each CT node n contains a set of constraints $n.constraints$ and a set of paths $n.paths$, one for each agent, that satisfy the constraints. A *vertex constraint* $\langle a_i, v, t \rangle$ prohibits agent a_i from using vertex v at timestep t, and an *edge constraint* $\langle a_i, u, v, t \rangle$ prohibits agent a_i from using edge $\langle u, v \rangle$ between timesteps t and $t + 1$. The cost of a CT node is the sum of the costs of its paths.

CBS starts with the root CT node, which has an empty set of constraints and a minimum-cost path for each agent that ignores conflicts. When expanding a CT node, CBS returns its paths as a solution if they are conflict-free. Otherwise, CBS picks a conflict to resolve: It splits the CT node into two child CT nodes and adds a constraint to each child CT node to prohibit either one or the other of the two conflicting agents from using the conflicting vertex or edge at the conflicting timestep. CBS then calls its low level to replan the path of the newly constrained agent in each child CT node. On the low level, CBS finds a minimum-cost path for the given agent that satisfies the constraints of the given CT node but ignores conflicts with the other agents.

MO-CBS

MO-CBS extends CBS to the MO-MAPF problem. Algorithm 1 shows its high level. MO-CBS first calls function INITIALIZATION to generate (potentially multiple) root CT nodes (Line 1). Specifically, MO-CBS calls its low level to find a cost-unique Pareto-optimal frontier Π_i for each agent a_i. For each combination of paths in $\Pi_1 \times \Pi_2 \dots \Pi_m$, MO-CBS generates a root CT node that has an empty set of constraints. It then inserts all root CT nodes into $Open$. In each iteration, a CT node with the lexicographically smallest cost is extracted from $Open$ for expansion (Line 4).

Similar to CBS, when expanding a CT node n whose paths are not conflict-free, MO-CBS picks a conflict to resolve (Line 9). The splitting strategy of MO-CBS is two-level. MO-CBS first splits a CT node into two subproblems, each with a new constraint on either one of the conflicting agents (Line 19). Then, for each constraint, MO-CBS calls its low level to replan a cost-unique Pareto-optimal frontier

Algorithm 1: MO-CBS with standard splitting

```
1  INITIALIZATION()
2  S ← ∅
3  while Open is not empty do
4      n ← Open.pop()
5      if ∃ Π ∈ S, c(Π) ⪯ c(n.paths) then continue
6      if n is conflict-free then
7          add n.paths to S
8          continue
9      conf ← a conflict in n.paths
10     children ← SPLIT(n, conf)
11     foreach n' ∈ children do
12         if ∃ Π ∈ S, c(Π) ⪯ c(n'.paths) then continue
13         add n' to Open
14 return S
15 Function SPLIT(n, conf):
16     children ← ∅
17     foreach a_i involved in conf do
18         cons ← the constraint imposed on a_i
19         C'_i ← n.constraints ∪ {cons}
20         Π'_i ← LOWLEVELSEARCH(a_i, C'_i)
21         foreach π_i ∈ Π'_i do
22             n' ← n
23             n'.constraints ← C'_i
24             n'.paths[i] ← π_i
25             add n' to children
26     return children
```

for the constrained agent (Line 20). For each path in it, MO-CBS generates a new child CT node (Lines 21-25). When expanding a CT node whose paths are conflict-free, MO-CBS has found a new solution and adds it to the solution set (Line 7). MO-CBS terminates when $Open$ is empty.

The low level of MO-CBS can be implemented with any algorithm that computes a cost-unique Pareto-optimal frontier for a given agent and given constraints. Ren, Rathinam, and Choset (2022) propose to use BOA* (Ulloa et al. 2020), a state-of-the-art single-agent bi-objective (where there are only two costs to minimize) search algorithm, and NAMOA*-dr (Pulido, Mandow, and Pérez-de-la Cruz 2015), a single-agent multi-objective search algorithm, as the low-level search algorithms for MO-MAPF instances with only two cost metrics and MO-MAPF instances with more than two cost metrics, respectively.

Example 1. *Figure 1 shows an MO-MAPF instance with two agents. In the beginning, a cost-unique Pareto-optimal frontier for agent a_1 contains paths $[A, C, D]$ and $[A, B, D]$ with costs $(2, 3)$ and $(3, 1.5)$, respectively. A cost-unique Pareto-optimal frontier for agent a_2 contains only one path $[E, F, D, G]$ with cost $(3, 3)$. Therefore, MO-CBS generate two root CT nodes n_1 and n_2. We assume that MO-CBS breaks ties in favor of CT nodes that are generated earlier when extracting a CT node from Open.*

1. *MO-CBS first expands CT node n_1 according to the lexicographical order and picks the vertex conflict between agents a_1 and a_2 at vertex D at timestep 2 to resolve. MO-CBS splits n_1 with vertex constraints $\langle a_1, D, 2\rangle$ and $\langle a_2, D, 2\rangle$. For agent a_1, MO-CBS calls its low-level search to find a cost-unique Pareto-optimal frontier that consists of three paths $[A, C, C, D]$, $[A, B, B, D]$, and $[A, I, B, D]$ and thus generates three child CT nodes n_3, n_4, and n_5, one for each path. For agent a_2, MO-CBS calls its low-level search to find a cost-unique Pareto-optimal frontier that consists of only one path $[E, F, F, D, G]$ and generates child CT node n_6.*
2. *MO-CBS expands CT node n_2 and, similar to the expansion of n_1, generates four CT nodes n_7, n_8, n_9, and n_{10}.*
3. *MO-CBS expands CT node n_3 and finds a Pareto-optimal solution with cost $(6, 7)$.*
4. *MO-CBS extracts and then prunes CT nodes n_6 and n_7 because their costs are both weakly dominated by the cost of the solution found in CT node n_3.*
5. *MO-CBS expands CT node n_4 and finds a Pareto-optimal solution with cost $(7, 5.5)$.*
6. *MO-CBS extracts and then prunes CT nodes n_8 and n_{10} because their costs are both weakly dominated by the cost of the solution found in CT node n_4.*
7. *MO-CBS expands CT node n_5 and finds a Pareto-optimal solution with cost $(8, 4.5)$.*
8. *MO-CBS extracts and then prunes CT node n_9 because its cost is weakly dominated by the cost of the solution found in CT node n_5.*

MO-CBS terminates and finds a cost-unique Pareto-optimal frontier with three solutions.

While we focus on minimizing the sum of costs in this paper, another interesting objective is to minimize the maximum cost over all agents (makespan). In fact, MO-CBS as well as the splitting strategies that we describe later work for makespan, too, if the costs of CT nodes are calculated appropriately.

Cost Splitting

The following example shows that MO-CBS sometimes generates identical CT nodes.

Example 2. *Consider CT nodes n_3 and n_7 in Figure 1b. Both CT nodes contain the same set of constraints and paths, which makes these two CT nodes indistinguishable from each other. (CT nodes n_4 and n_5 are indistinguishable from CT nodes n_8 and n_9, respectively, too.) CT nodes n_3 and n_7 are both conflict-free. However, for an MO-MAPF instance with more agents, such duplicate CT nodes can contain conflicts between other agents, and, to resolve these conflicts, MO-CBS duplicates the search effort in the trees rooted in these duplicate CT nodes.*

We now describe cost splitting, which generates fewer CT nodes than the standard splitting strategy and still retains the completeness and optimality guarantees of MO-CBS. For each CT node n, MO-CBS with cost splitting maintains a *cost lower bound* $n.\mathbf{lb} = [n.\mathbf{lb}_1, n.\mathbf{lb}_2 \dots n.\mathbf{lb}_m]$, that consists of m vectors of length N, one for each agent. It also modifies the INITIALIZATION and SPLIT functions in Algorithm 1.

Definition 1. *A path π_i of agent a_i is compatible with a CT node n iff:*

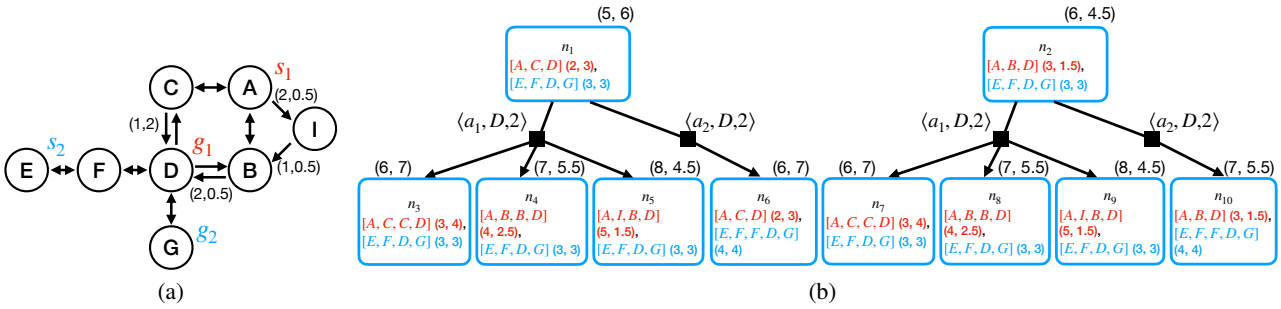

Figure 1: An example MO-MAPF instance with two agents. (a) shows the graph for this MO-MAPF instance. The self-pointing edges are not shown, and all of them have cost $(1, 1)$. Except for $\mathbf{c}(\langle B, D\rangle) = \mathbf{c}(\langle A, I\rangle) = (2, 0.5)$, $\mathbf{c}(\langle C, D\rangle) = (1, 2)$, and $\mathbf{c}(\langle I, B\rangle) = (1, 0.5)$, all shown edges have cost $(1, 1)$. s_i and g_i, $i \in \{1, 2\}$, denotes the start and goal vertices of agent a_i, respectively. (b) shows the CTs of MO-CBS for this MO-MAPF instance. Blue boxes represent CT nodes. The label inside each CT node denotes the name of the CT node, the paths for agents a_1 (in red) and a_2 (in blue), and the corresponding path costs inside the parentheses. For ease of presentation, we show each path as the sequence of traversed vertices. The label next to each CT node is its cost.

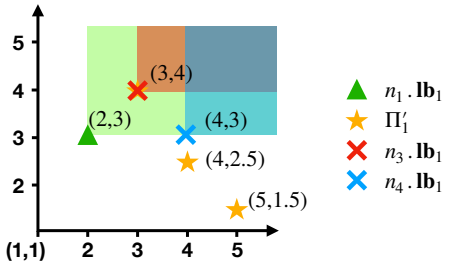

Figure 2: An example of cost splitting. Each marker corresponds to a vector of length two. The x- and y-axes correspond to the first and the second component of each vector, respectively.

1. $n.\mathbf{lb}_i \preceq \mathbf{c}(\pi_i)$ *and*

2. π_i *satisfies all constraints of CT node* n.

A solution is compatible with a CT node iff all its paths are compatible with the CT node.

Each vector in the cost lower bound $n.\mathbf{lb}_i$, $i = 1, 2 \ldots m$, specifies that n "considers" only solutions whose path cost for agent a_i is weakly dominated by (conceptually, "lower-bounded" by) $n.\mathbf{lb}_i$.

In INITIALIZATION, MO-CBS with cost splitting initializes the cost lower bound of each agent in each root CT node to the cost of its path. For example, the cost lower bounds of CT nodes n_1 and n_2 in Figure 1b are initialized to $[(2, 3), (3, 3)]$ and $[(3, 1.5), (3, 3)]$, respectively. The green triangle in Figure 2 represents $n_1.\mathbf{lb}_1$, and the green region in Figure 2 represents the dominated region of $n_1.\mathbf{lb}_1$. Conceptually, CT node n_1 only considers solutions whose path costs for agent a_1 are inside this green region.

Property 2. *Any solution is compatible with at least one root CT node.*

Algorithm 2: Cost splitting

```
1  Function SPLIT(n, conf):
2      children ← ∅
3      foreach a_i involved in conf do
4          cons ← the constraint imposed on a_i
5          C'_i ← n.constraints ∪ {cons}
6          Π'_i ← LOWLEVELSEARCH(a_i, C'_i)
7          LB_i ← ND({comax(n.lb_i, c(π)) | π ∈ Π'_i})
8          foreach lb ∈ LB_i do
9              n' ← n
10             n'.constraints ← C'_i
11             n'.lb_i ← lb
12             n'.paths[i] ← a path π ∈ Π'_i such that
                 comax(n.lb_i, c(π)) = lb
13             add n' to children
14     return children
```

Proof. Any combination of the cost-unique Pareto-optimal frontiers of all agents corresponds to a root CT node, so any solution is weakly dominated by the paths and thus the cost lower bounds of at least one root CT node. Since root CT nodes do not have any constraints, the statement holds. □

Algorithm 2 describes the SPLIT function of cost splitting. For each agent a_i involved in the conflict and the corresponding constraint $cons_i$, cost splitting calls its low level to replan a cost-unique Pareto-optimal frontier Π'_i that satisfies the constraints $n.constraints \cup \{cons_i\}$ (Line 6). It then computes a set of vectors (Line 7)

$$\mathbf{LB}_i = \mathbf{ND}(\{\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi)) \mid \pi \in \Pi'_i\}).$$

For each cost vector $\mathbf{lb} \in \mathbf{LB}_i$, cost splitting creates a child CT node whose cost lower bound for agent a_i is $\mathbf{lb}$ (Line 11), and whose path of agent a_i is the corresponding path in Π'_i (Line 12). We define the splitting strategy in this way because (1) it guarantees that MO-CBS does

not exclude any compatible solution during CT node splitting, as indicated by Property 3, and (2) it eliminates some duplicate search effort that the standard splitting strategy may spend, as shown in Example 3. Recall that $D(\mathbf{v})$ denotes the dominated region of a vector $\mathbf{v}$. Each vector in $\{\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi)) \mid \pi \in \Pi'_i\}$ corresponds to a dominated region that is the intersection between $D(n.\mathbf{lb_i})$ and $D(\pi)$ for some $\pi \in \Pi'_i$. For any solution compatible with n, its cost for agent a_i must lie in at least one of these dominated regions. This still holds after removing those dominated vectors (via the $\mathbf{ND}$ function).

Property 3. *Any solution that is compatible with a CT node is compatible with at least one of its child CT nodes.*

Proof. The two constraints for resolving a conflict ensure that every solution satisfies at least one of them (because, otherwise, the solution is not conflict-free). Therefore, we need to prove only that any solution Π that is compatible with CT node n and satisfies the new constraint imposed on agent a_i is compatible with at least one child CT node in $nodes_i$, where $nodes_i$ is the set of child CT nodes generated after replanning for agent a_i. Let π_i be the path of agent a_i in solution Π. Because every CT node in $nodes_i$ differs from CT node n only in the path, constraint, and cost lower bound of agent a_i, according to Definition 1, we only need to prove that there exists a child CT node in $nodes_i$ that π_i is compatible with.

Since Π'_i is a cost-unique Pareto-optimal frontier of agent a_i (Line 6), there exists a path $\pi^* \in \Pi'_i$ such that $\mathbf{c}(\pi^*) \preceq \mathbf{c}(\pi_i)$. With $n.\mathbf{lb}_i \preceq \mathbf{c}(\pi_i)$ (because Π is compatible with n) and Property 1, we have $\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi^*)) \preceq \mathbf{c}(\pi_i)$. Since $\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi^*)) \in \{\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi)) \mid \pi \in \Pi'_i\}$, there exists a vector in $\mathbf{LB}_i$ that weakly dominates $\mathbf{comax}(n.\mathbf{lb}_i, \mathbf{c}(\pi^*))$, which in turn weakly dominates $\mathbf{c}(\pi_i)$. Therefore, π_i is compatible with the corresponding child CT node. □

Example 3. *Consider CT node n_1 in Figure 1b. Recall that, in* INITIALIZATION*, the cost lower bounds of root CT nodes n_1 is initialized to $n_1.\mathbf{lb} = [(2,3),(3,3)]$. When replanning for agent a_1 with new constraint $\langle a_1, D, 2\rangle$, the cost-unique Pareto-optimal frontier consists of three paths $\pi_1^{(1)} = [A,C,C,D]$ (with cost $(3,4)$), $\pi_1^{(2)} = [A,B,B,D]$ (with cost $(4,2.5)$), and $\pi_1^{(3)} = [A,I,B,D]$ (with cost $(5,1.5)$). For these three paths, we have $\mathbf{comax}(n_1.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(1)})) = (3,4)$, $\mathbf{comax}(n_1.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(2)})) = (4,3)$, and $\mathbf{comax}(n_1.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(3)})) = (5,3)$, respectively. Since $(5,3)$ is weakly dominated by $(4,3)$, cost splitting does not generate a child CT node for $\pi_1^{(3)}$. Intuitively, if cost splitting generated a child CT node for $\pi_1^{(3)}$, it would correspond to CT node n_5, every path that is compatible with n_5 would also be compatible with n_9. Therefore, it is not necessary to generate n_5. Cost splitting generates two child CT nodes for only $\pi_1^{(1)}$ and $\pi_1^{(2)}$, which correspond to CT nodes n_3 and n_4 in Figure 1b. Note that the path of a_1 ($\pi_1^{(2)}$ with cost $(4,2.5)$) in CT node n_4 is not compatible with CT node n_4 because $n_4.\mathbf{lb}_1 = (4,3)$ does not weakly dominate $\mathbf{c}(\pi_1^{(2)}) = (4,2.5)$. As we will show later, this does not affect the optimality and completeness of MO-CBS. The red and blue crosses in Figure 2 represent $n_3.\mathbf{lb}_1$ and $n_4.\mathbf{lb}_1$, respectively. The red and blue regions in Figure 2 represent the dominated regions of $n_3.\mathbf{lb}_1$ and $n_4.\mathbf{lb}_1$, respectively. The dominated regions of $n_3.\mathbf{lb}_1$ and $n_4.\mathbf{lb}_1$ are both subsets of the dominated region of $n_1.\mathbf{lb}_1$. Cost splitting still needs to generate CT node n_4 to ensure that it does not miss a solution whose path cost for a_1 is weakly dominated by $(4,3)$ but not $(3,4)$.*

Consider CT node n_2, we have $\mathbf{comax}(n_2.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(1)})) = (3,4)$, $\mathbf{comax}(n_2.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(2)})) = (4,2.5)$, and $\mathbf{comax}(n_2.\mathbf{lb}_1, \mathbf{c}(\pi_1^{(3)})) = (5,1.5)$, which do not weakly dominates each others. Thus, cost splitting generates three child CT nodes for $\pi_1^{(1)}$, $\pi_1^{(2)}$, and $\pi_1^{(3)}$, which correspond to CT nodes n_7, n_8, and n_9 in Figure 1b. Overall, MO-CBS with cost splitting generates one fewer CT node (namely, CT node n_5) than MO-CBS with standard splitting.

Property 4. *When splitting a CT node, MO-CBS with cost splitting never generates more child CT nodes than MO-CBS with standard splitting.*

Proof. The property holds because, according to Line 7 of Algorithm 2, the size of $\mathbf{LB}_i$ (that is, the number of CT nodes that MO-CBS with cost splitting generates) is at most the size of Π'_i (that is, the number of CT nodes that MO-CBS with standard splitting generates). □

Disjoint Cost Splitting

In Example 3, any path of agent a_1 whose cost is weakly dominated by $(3,3)$ is compatible with both CT nodes n_1 (with $n_1.\mathbf{lb}_1 = (2,3)$) and n_2 (with $n_2.\mathbf{lb}_1 = (3,1.5)$). Hence, CT nodes n_1 and n_2 can still lead to the same solution and thus result in duplicate search effort. In this section, we describe our second splitting strategy for MO-CBS, disjoint cost splitting, which further reduces duplicate search effort of MO-CBS and ensures that, when splitting a CT node, any solution that is compatible with the CT node is compatible with exactly one of its child CT nodes.

In addition to maintaining a cost lower bound, MO-CBS with disjoint cost splitting also maintains a *cost upper bound* $n.\mathbf{UB}_i$ for each agent a_i in each CT node n, that consists of a set of cost vectors that are weakly dominated by $n.\mathbf{lb}_i$. It also modifies Definition 1 as well as the INITIALIZATION and SPLIT functions used by Algorithm 1 as follows.

Definition 2. *A path π_i of agent a_i is UB-compatible with a CT node n iff:*

1. *π_i is compatible with CT node n and*
2. *$\forall\, \mathbf{ub} \in n.\mathbf{UB}_i\ \mathbf{ub} \not\preceq \mathbf{c}(\pi_i)$.*

A solution is UB-compatible with a CT node iff all its paths are UB-compatible with the CT node.

In INITIALIZATION, MO-CBS with disjoint cost splitting finds a cost-unique Pareto-optimal frontier Π_i for each agent a_i and generates a root CT node for each combination of the paths in $\Pi_1 \times \Pi_2 \ldots \Pi_m$. Let $[\pi_i^1, pi_i^2 \ldots \pi_i^{|\Pi_i|}]$ denote the paths of Π_i sorted in a predetermined order. For

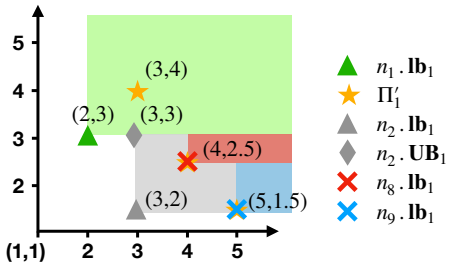

Figure 3: An example of disjoint cost splitting. Each marker corresponds to a vector of length two. The x- and y-axes correspond to the first and the second component of each vector, respectively.

a root CT node n with path π_i^j for agent a_i, $n.\mathbf{UB}_i$ is initialized to $\mathbf{ND}(\{\mathbf{comax}(\mathbf{c}(\pi_i^j), \mathbf{c}(\pi_i^k)) \mid k = 1, 2 \dots j-1\})$. For example, consider the cost upper bounds of CT nodes n_1 and n_2 in Figure 1b. If we sort the paths of Π_1 in lexicographic order with respect to their path costs (that is, path $\pi_1^1 = [A, C, D]$ comes before path $\pi_1^2 = [A, B, D]$), then $n_1.\mathbf{UB}_1$ is set to $\emptyset$, and $n_2.\mathbf{UB}_1$ is set to $\mathbf{ND}(\{\mathbf{comax}(\mathbf{c}(\pi_1^2), \mathbf{c}(\pi_1^1))\}) = \{(3, 3)\}$. Both $n_1.\mathbf{UB}_2$ and $n_2.\mathbf{UB}_2$ are set to $\emptyset$ because the Pareto-optimal frontier for agent a_2 contains only one path. The grey triangle and diamond in Figure 3 represent $n_2.\mathbf{lb}_1$ and $n_2.\mathbf{UB}_1$, respectively. The grey region in Figure 3 represents the set of vectors that are weakly dominated by $(3, 2)$ ($n_2.\mathbf{lb}_1$) but not weakly dominated by $(3, 3)$ (the only vector in $n_2.\mathbf{UB}_1$). Conceptually, CT node n_2 only "considers" solutions whose path costs for agent a_1 are inside this grey region. It does not need to consider any solution with a path cost for a_1 weakly dominated by $(3, 3)$ because such a solution has already been considered by CT node n_1.

Property 5. *Any solution is UB-compatible with exactly one root CT node.*

Proof. Given a solution Π, for $i = 1, 2 \dots m$, we use π_i, to denote the path for agent a_i in Π and $\pi_i^{j_i}$ the first path in the sorted sequence of Π_i with $\mathbf{c}(\pi_i^{j_i}) \preceq \mathbf{c}(\pi_i)$ during the initialization. We prove that Π is only UB-compatible with the root CT node n generated for combination $\pi_1^{j_1}, \pi_2^{j_2} \dots \pi_m^{j_m}$.

We first prove that Π is UB-compatible with n. We have $n.\mathbf{lb}_i = \mathbf{c}(\pi_i^{j_i}) \preceq \mathbf{c}(\pi_i)$ for every agent a_i (the first condition of Definition 1 holds). Since n, as a root CT node, has no constraint, the second condition of Definition 1 and hence the first condition of Definition 2 hold. For all $k = 1, 2 \dots j_i - 1$, we have $\mathbf{c}(\pi_i^k) \not\preceq \mathbf{c}(\pi_i)$ and hence $\mathbf{comax}(\mathbf{c}(\pi_i^{j_i}), \mathbf{c}(\pi_i^k)) \not\preceq \mathbf{c}(\pi_i)$. Therefore, no vector in $n.\mathbf{UB}_i$ weakly dominates $\mathbf{c}(\pi_i)$, and the second condition of Definition 2 holds. Thus, Π is UB-compatible with n.

We then prove that there does not exist another root CT node n' that Π is UB-compatible with. If n' exists, then there must exist some agent a_i for which CT nodes n and n' have different paths. Let π'_i denote the path for agent a_i in CT node n'. Since π_i is UB-compatible with both root CT nodes n and n' and the cost lower bounds are equal to the path costs in root CT nodes, $\mathbf{c}(\pi) \preceq \mathbf{c}(\pi_i)$ holds for both $\pi = \pi_i^{j_i}$ and $\pi = \pi'_i$, and thus $\mathbf{c}' = \mathbf{comax}(\mathbf{c}(\pi'_i), \mathbf{c}(\pi_i^{j_i})) \prec \mathbf{c}(\pi_i)$ holds. Furthermore, since $\pi_i^{j_i}$ is the first path in the sorted sequence of Π_i with $\mathbf{c}(\pi_i^{j_i}) \preceq \mathbf{c}(\pi_i)$, π'_i comes after $\pi_i^{j_i}$ in this sorted sequence. As a result, $\mathbf{c}'$ is either in $n'.\mathbf{UB}_i$ or dominated by a vector in $n'.\mathbf{UB}_i$, π_i and thus solution Π are not UB-compatible with n'; a contradiction is found. □

Algorithm 3: Disjoint cost splitting

```
1  Function SPLIT(n, conf):
2      children ← ∅
3      foreach a_i involved in conf do
4          cons ← the constraint imposed on a_i
5          C'_i ← n.constraints ∪ {cons}
6          Π'_i ← LOWLEVELSEARCH(a_i, C'_i)
7          LB_i ← ND({comax(n.lb_i, c(π)) | π ∈ Π'_i})
*8         UB ← n.UB_i
9          foreach lb ∈ LB_i do
10             n' ← n
11             n'.constraints ← C'_i
12             n'.lb_i ← lb
13             n'.paths[i] ← a path π ∈ Π'_i such that
                 comax(n.lb_i, c(π)) = lb
*14            n'.UB_i ← ND({comax(n'.lb_i, v) | v ∈
                 UB})
*15            if n'.lb_i ∈ n'.UB_i then continue
16             add n' to children
*17            add lb to UB
18     return children
```

Algorithm 3 shows the SPLIT function for disjoint cost splitting. Algorithm 3 is similar to Algorithm 2 but with a few changes. We highlight these changes by using "*" before line numbers. For each agent a_i involved in the conflict, disjoint cost splitting generates a set of cost lower bounds $\mathbf{LB}_i$ and creates a child CT node for each vector in $\mathbf{LB}_i$ (Lines 9-17). Disjoint cost splitting uses variable $\mathbf{UB}$ to store $n.\mathbf{UB}_i$ (Line 8) and all cost lower bounds of the CT nodes that have been generated (Line 17). When generating a CT node n', MO-CBS sets $n'.\mathbf{UB}_i$ to $\mathbf{ND}(\{\mathbf{comax}(n'.\mathbf{lb}_i, \mathbf{v}) \mid \mathbf{v} \in \mathbf{UB}\})$ (Line 14). If $n'.\mathbf{UB}_i$ contains $n'.\mathbf{lb}_i$, then there does not exist a path for agent a_i (and hence a solution) that is UB-compatible with CT node n'. In such case, CT node n' is pruned (Line 15).

Property 6. *Any solution that is UB-compatible with a CT node is UB-compatible with either exactly one of its child nodes or two of its child nodes that have different constraints.*

Proof. Similar to the proof of Property 3, we only need to prove that any solution Π that is UB-compatible with CT node n and satisfies the new constraint imposed on agent a_i is UB-compatible with exactly one child node in $nodes_i$, where $nodes_i$ is the set of child CT nodes generated after replanning agent a_i. We use π_i to denote the path for agent a_i in solution Π. From the proof for Property 3, there exists at least one vector in $\mathbf{LB}_i$ that weakly dominates $\mathbf{c}(\pi_i)$. Let n' be the CT node generated for the first such vector in the in-

ner iteration of SPLIT (Lines 9-17). When node n' is created, every vector in $\mathbf{UB}$ is some vector in $\mathbf{LB}_i$ that comes before $n'.\mathbf{lb}_i$ and hence does not weakly dominates $\mathbf{c}(\pi_i)$. We have that $\mathbf{ub} \not\preceq \mathbf{c}(\pi_i)$ for all $\mathbf{ub} \in n'\mathbf{UB}_i$ because every $\mathbf{ub}$ is the comax between $n'.\mathbf{lb}_i$ and some vector in $\mathbf{UB}$.

Therefore, path π_i is UB-compatible with CT node n'. Since CT node n' has the same constraints, cost lower bound, and cost upper bound as its parent CT node for all agents except a_i, solution Π is UB-compatible with CT node n', too.

Then, we prove that there exists exactly one such child CT node. Because CT node n' is the first child CT node of n that Π is UB-compatible with, we only need to consider those nodes n'' generated after n' with $n''.\mathbf{lb}_i \preceq \mathbf{c}(\pi_i)$. Together with $n'.\mathbf{lb}_i \preceq \mathbf{c}(\pi_i)$, we have $\mathbf{c}' = \mathbf{comax}(n'.\mathbf{lb}_i, n''.\mathbf{lb}_i) \preceq \mathbf{c}(\pi_i)$. Since $\mathbf{c}'$ is either in $n''.\mathbf{UB}_i$ or dominated by a vector in $n''.\mathbf{UB}_i$, π_i and thus solution Π are not UB-compatible with n''. □

Example 4. *Consider CT node n_2 in Figure 1b. When replanning for agent a_1 with new constraint $\langle a_1, D, 2\rangle$, disjoint cost splitting generates three child CT nodes that correspond to CT nodes n_7, n_8, and n_9 in Figure 1b. From Example 3, we have $n_7.\mathbf{lb}_1 = (3, 4)$. When generating CT node n_7, $\mathbf{UB}$ consists of $(3, 3)$ from $n_2.\mathbf{UB}_1$. Therefore, $n_7.\mathbf{UB}_1$ is set to $\mathbf{ND}(\{\mathbf{comax}(n_7.\mathbf{lb}_1, (3, 3))\}) = \{(3, 4)\}$. CT node n_7 is then pruned because $n_7.\mathbf{lb}_1 \in n_7.\mathbf{UB}_1$. The red and blue crosses in Figure 3 represent $n_8.\mathbf{lb}_1$ and $n_9.\mathbf{lb}_1$, respectively. The red and blue regions in Figure 3 are two disjoint regions corresponding to the sets of path costs for agent a_1 that CT nodes n_8 and n_9 consider, respectively. CT node n_7 can be safely pruned because $n_7.\mathbf{lb}_1$ is outside the grey region. Overall, MO-CBS with disjoint cost splitting generates two fewer CT nodes (namely, CT nodes n_5 and n_7) than MO-CBS with standard splitting.*

Theoretical Results

We now prove the completeness and optimality of MO-CBS with either of our splitting strategies. Our proof follows the proof by Ren, Rathinam, and Choset (2022). Let PO be a cost-unique Pareto-optimal frontier for a given MO-MAPF instance, S be the current solution set, $\mathbf{S} = \{\mathbf{c}(\Pi) \mid \Pi \in S\}$ be their costs, and $PO \mid \mathbf{S} = \{\Pi \mid \Pi \in PO, \mathbf{c}(\Pi) \notin \mathbf{S}\}$ be the solutions in PO whose costs have not been found so far. Note that a solution is compatible with a CT node if it is UB-compatible with the CT node.

Lemma 1. *Any CT node that a solution in $PO \mid \mathbf{S}$ is compatible with is not pruned on Lines 5 or 12 in Algorithm 1.*

Proof. Suppose that a solution $\Pi \in PO \mid \mathbf{S}$ is compatible with a CT node n. For each agent a_i and its corresponding path π_i in Π, we have $n.\mathbf{lb}_i \preceq \mathbf{c}(\pi_i)$ (from Definition 1) and $\mathbf{c}(n.paths[i]) \preceq n.\mathbf{lb}_i$ (because $n.\mathbf{lb}_i$ is calculated by taking the $\mathbf{comax}$ of $\mathbf{c}(n.paths[i])$ and another cost vector). We thus have $\mathbf{c}(n.paths[i]) \preceq \mathbf{c}(\pi_i)$ and hence $\mathbf{c}(n.paths) \preceq \mathbf{c}(\Pi)$. Since Π is Pareto-optimal with $\mathbf{c}(\Pi) \notin \mathbf{S}$, there is no solution in S whose cost weakly dominates $\mathbf{c}(\Pi)$ and thus $\mathbf{c}(n.paths)$. Hence, CT node n is not pruned. □

Lemma 2. *Every solution in $PO \mid \mathbf{S}$ is compatible with at least one CT node in $Open$ in the beginning of every iteration (that is, before Line 4 gets executed) of Algorithm 1.*

Proof. We prove this lemma by induction. Properties 2 and 5 show that the lemma holds after INITIALIZATION (Line 1 of Algorithm 1). We assume that this lemma holds before the k-th CT node is extracted from $Open$. When the k-th CT node n is extracted, for every solution $\Pi \in PO \mid \mathbf{S}$, there are three cases: (1) If Π is not compatible with CT node n, then, based on the assumption, it is compatible with one CT node in $Open$; (2) If Π is compatible with CT node n and $n.paths$ is conflict-free, then $\mathbf{c}(n.paths) = \mathbf{c}(\Pi)$ because $\mathbf{c}(n.paths) \preceq \mathbf{c}(\Pi)$ and Π is Pareto-optimal. After MO-CBS adds $n.paths$ to S (Line 7), Π is not in $PO \mid \mathbf{S}$ anymore; or (3) If Π is compatible with CT node n and $n.paths$ is not conflict-free, n is not pruned and hence expanded according to Lemma 1, and Π is still compatible with at least one child CT node that is added to $Open$ according to Properties 3 and 6. Therefore, the lemma holds. □

Our Lemma 2 corresponds to Lemma 3 by Ren, Rathinam, and Choset (2022). The following theorem combines Theorems 1 and 2 by Ren, Rathinam, and Choset (2022) and shows that (disjoint) cost splitting maintains the optimality and completeness of MO-CBS. The proofs for Theorems 1 and 2 by Ren, Rathinam, and Choset (2022) apply here if combined with our Lemma 2 and hence are omitted.

Theorem 1. *MO-CBS with (disjoint) cost splitting finds a cost-unique Pareto-optimal frontier for a given MO-MAPF instance in finite time, if it exists.*

Experimental Results

In this section, we evaluate cost splitting and disjoint cost splitting. The variants of MO-CBS are MO-CBS, MO-CBS-c, and MO-CBS-dc, where c adds cost splitting and dc adds disjoint cost splitting. We implemented all MO-MAPF algorithms in Python. They share the same code base as much as possible.[1] We ran experiments on a server with an AMD EPYC 7742 CPU. We limited the memory to 16 GB and set the time limit for solving each MO-MAPF instance to 1,000 seconds.

We use three different cost metrics:

1. (*random-bi-obj*) Every edge is assigned a 2-dimensional cost vector with each component being an integer randomly sampled from $\{1, 2\}$.
2. (*random-tri-obj*) Similar to *random-bi-obj* except that every edge is assigned a 3-dimensional cost vector.
3. (*time-energy*) Every vertex is assigned an integer indicating its height. *Travel time* and its *energy consumption*, where moving upward from a vertex of height i to a vertex of height j costs $j - i$ units of energy, and the edge treaverses otherwise costs one unit of energy. The height

[1] Source code and detailed usage is available on https://github.com/mepear/MOMAPF

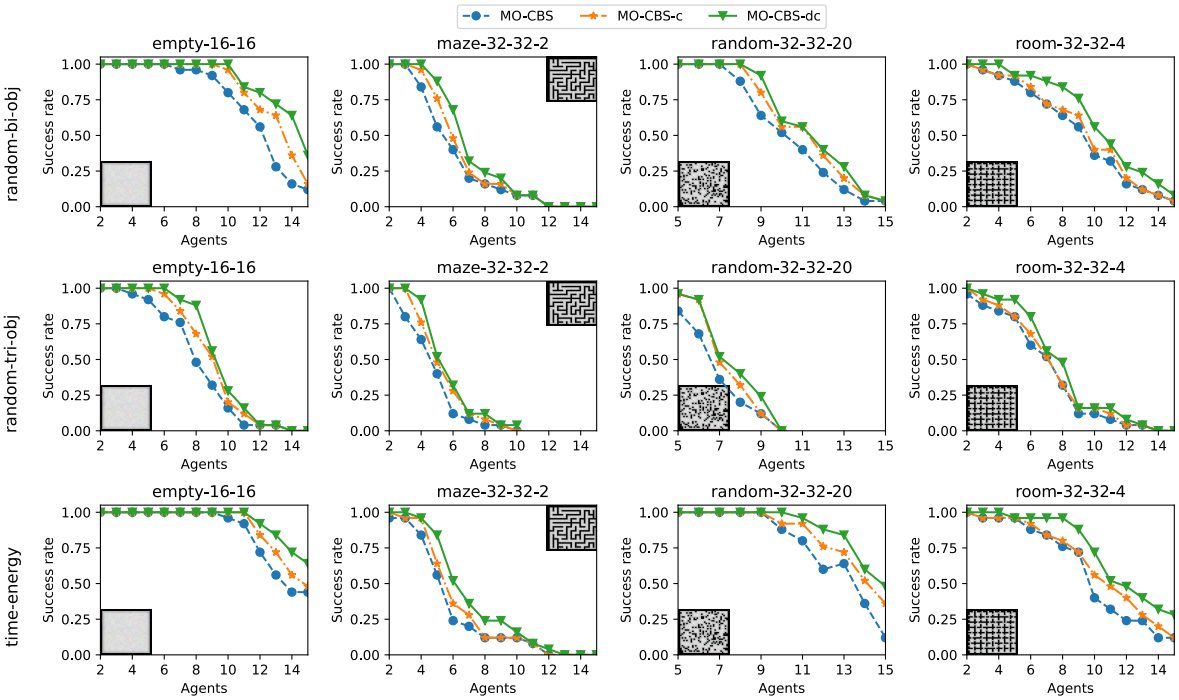

Figure 4: Success rate results for MO-CBS variants on different grid maps with different combinations of objectives.

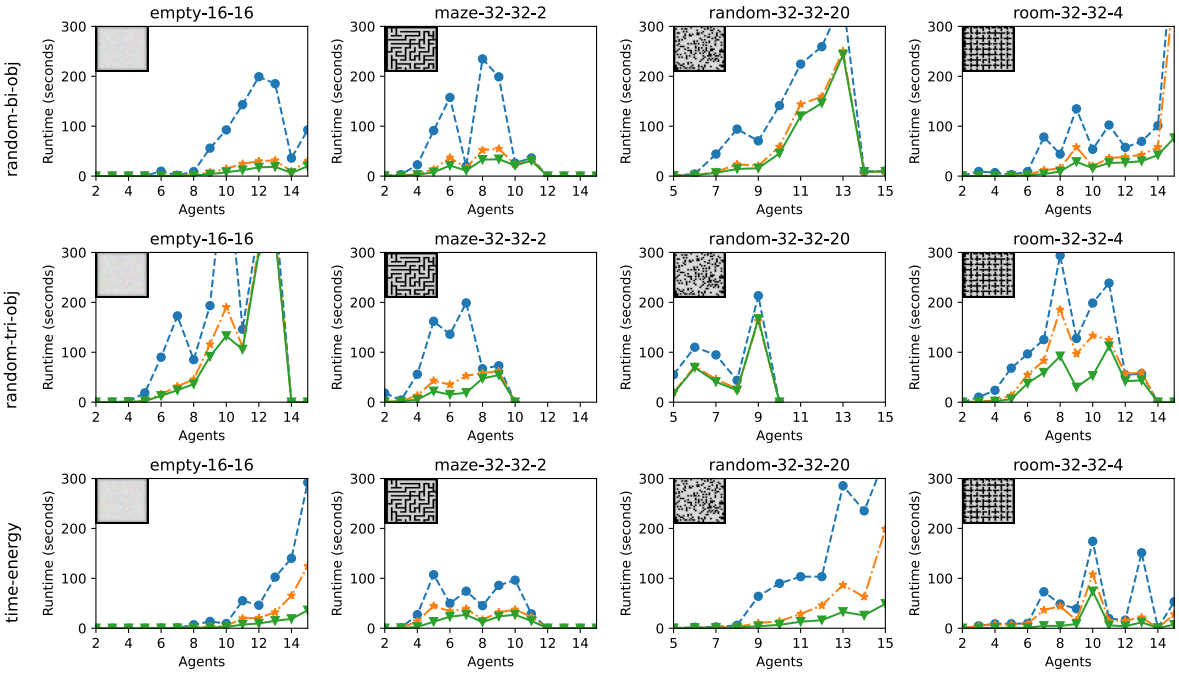

Figure 5: Average runtimes (in seconds) over MO-MAPF instances that are solved by all MO-CBS variants on different grid maps with different combinations of objectives.

of a vertex is proportional to vertex's distance to its distance from the center point of the map, which creates a hill-like height map.

We use four grid maps from the MAPF benchmark (Stern et al. 2019),[2] namely *empty-16-16*, *maze-32-32-2*, *random-*

[2] https://movingai.com/benchmarks/mapf.html

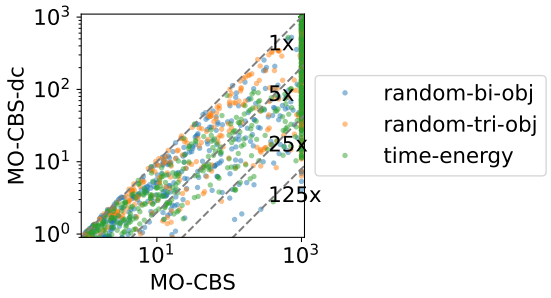

Figure 6: Runtimes (in seconds) of MO-CBS and MO-CBS-dc on all MO-MAPF instances. The x- and y-coordinate of a dot corresponds to the runtime (in seconds) of MO-CBS and MO-CBS-dc on an MO-MAPF instance, respectively.

	MO-CBS	MO-CBS-c	MO-CBS-dc
random-bi-obj	5.28	3.42	**2.72**
random-tri-obj	16.31	6.34	**3.60**
time-energy	4.06	2.92	**2.52**

Table 1: Average branching factors for each MO-MAPF algorithm and each cost metric.

32-32-20, and *room-32-32-4*. For each cost metric and grid map, we vary the number of agents from 1 to 15 and, for each number of agents, average over 25 "random scenarios" from the benchmark.

Figure 4 shows the *success rates*, that is, the percentages of MO-MAPF instances that an algorithm solves within the time limit, for different cost metrics, maps, and numbers of agents. In most cases, MO-CBS-dc has higher success rates than MO-CBS-c, and they both have higher success rates than MO-CBS. Figure 5 shows the average runtimes. For each combination of cost metrics and number of agents the average runtime of an algorithm is taken over all MO-MAPF instances solved by all three algorithms. In some cases, MO-CBS-c and MO-CBS-dc have much smaller runtimes than MO-CBS. Figure 6 shows the runtimes (in seconds) of MO-CBS and MO-CBS-dc for all instances individually. MO-CBS-dc has smaller runtimes than MO-CBS for almost all instances with a maximum speed-up of more than 125 times. In general, the speed-ups achieved by new splitting strategies increase with the number of agents, although this is not always the case.

Table 1 shows the average branching factors, that is, the average numbers of child CT nodes that are returned by the SPLIT function when expanding a CT node, for different combinations of cost metrics and splitting strategies. In all cases, MO-CBS-dc has smaller branching factors than MO-CBS-c, and they both have smaller branching factors than MO-CBS. The comparison of *random-bi-obj* and *random-tri-obj* shows that the branching factors achieved by cost splitting and disjoint cost splitting increase with the number of metrics. This is because, with more cost metrics, the cost-unique Pareto-optimal frontiers for each agent are larger, and cost splitting or disjoint cost splitting becomes more important for reducing numbers of generated child CT nodes.

Conclusions

We proposed two splitting strategies for MO-CBS, namely cost splitting and disjoint cost splitting, both of which speed up MO-CBS without losing its optimality or completeness guarantees. Our experimental results showed that disjoint cost splitting, our best splitting strategy, speeds up MO-CBS by up to two orders of magnitude and substantially improves its success rates in various settings. Future work includes improving MO-CBS with CBS enhancements, such as adding heuristics (Felner et al. 2018) and symmetry breaking (Li et al. 2019).

Acknowledgements

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779, 1935712, and 2112533. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or any government.

References

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2018. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 83–87.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019. Symmetry-Breaking Constraints for Grid-Based Multi-Agent Path Finding. In *AAAI Conference on Artificial Intelligence (AAAI)*, 6087–6095.

Ma, H.; Tovey, C.; Sharon, G.; Kumar, T. K. S.; and Koenig, S. 2016. Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing problem. In *AAAI Conference on Artificial Intelligence (AAAI)*, 3166–3173.

Morris, R.; Pasareanu, C. S.; Luckow, K.; Malik, W.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2016. Planning, Scheduling and Monitoring for Airport Surface Operations. In *AAAI-16 Workshop on Planning for Hybrid Systems*.

Pulido, F.-J.; Mandow, L.; and Pérez-de-la Cruz, J.-L. 2015. Dimensionality Reduction in Multiobjective Shortest Path Search. *Computers & Operations Research*, 64: 60–70.

Ren, Z.; Rathinam, S.; and Choset, H. 2021. Subdimensional Expansion for Multi-Objective Multi-Agent Path Finding. *IEEE Robotics and Automation Letters*, 6(4): 7153–7160.

Ren, Z.; Rathinam, S.; and Choset, H. 2022. A Conflict-Based Search Framework for Multi-Objective Multi-Agent Path Finding. https://arxiv.org/abs/2108.00745.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219: 40–66.

Stern, R.; Sturtevant, N. R.; Atzmon, D.; Walker, T.; Li, J.; Cohen, L.; Ma, H.; Kumar, T. K. S.; Felner, A.; and Koenig, S. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Symposium on Combinatorial Search (SOCS)*, 151–158.

Ulloa, C. H.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazo, L.; and Koenig, S. 2020. A Simple and Fast Bi-Objective Search Algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 143–151.

Wagner, G.; and Choset, H. 2015. Subdimensional Expansion for Multirobot Path Planning. *Artificial intelligence*, 219: 1–24.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine*, 29(1): 9–20.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *AAAI Conference on Artificial Intelligence (AAAI)*, 1443–1449.