Multi-objective Conflict-based Search for Multi-agent Path Finding

Zhongqiang Ren[1], Sivakumar Rathinam[2] and Howie Choset[1]

***Abstract*— Conventional multi-agent path planners typically compute an ensemble of paths while optimizing a single objective, such as path length. However, many applications may require multiple objectives, say fuel consumption and completion time, to be simultaneously optimized during planning and these criteria may not be readily compared and sometimes lie in competition with each other. Naively applying existing multi-objective search algorithms to multi-agent path finding may prove to be inefficient as the size of the space of possible solutions, i.e., the Pareto-optimal set, can grow exponentially with the number of agents (the dimension of the search space). This article presents an approach named Multi-objective Conflict-based Search (MO-CBS) that bypasses this so-called curse of dimensionality by leveraging prior Conflict-based Search (CBS), a well-known algorithm for single-objective multi-agent path finding, and principles of dominance from multi-objective optimization literature. We prove that MO-CBS is able to compute the entire Pareto-optimal set. Our results show that MO-CBS can solve problem instances with hundreds of Pareto-optimal solutions which the standard multi-objective A* algorithms could not find within a bounded time.**

I. Introduction

Multi-agent Path Finding (MAPF) computes a set of collision-free paths for multiple agents connecting their respective start and goal locations while optimizing a scalar measure of paths. Variants of MAPF have been widely studied in the robotics community over the last few years [18]. In this article, we investigate a natural generalization of the MAPF to include multiple objectives for multiple agents and hence the name *multi-objective multi-agent path finding* (MOMAPF). In MOMAPF, agents have to trade-off multiple objectives such as completion time, travel risk and other domain-specific measures (Fig. 1). MOMAPF is a generalization of MAPF and is therefore NP-Hard.

In the presence of multiple objectives, the goal of MOMAPF is to find the set of all Pareto-optimal[1] solutions rather than a single optimal solution as in MAPF. Finding this set of solutions while ensuring collision-free paths for agents in each solution is quite challenging: even though there are many multi-objective single-agent search algorithms [10], [19], [22] that can compute all Pareto-optimal solutions, a naive application of such algorithms to the MOMAPF problem may prove to be inefficient as the size of the Pareto-optimal set grows exponentially with respect to the number of agents and the dimension of the search space [15], [26].

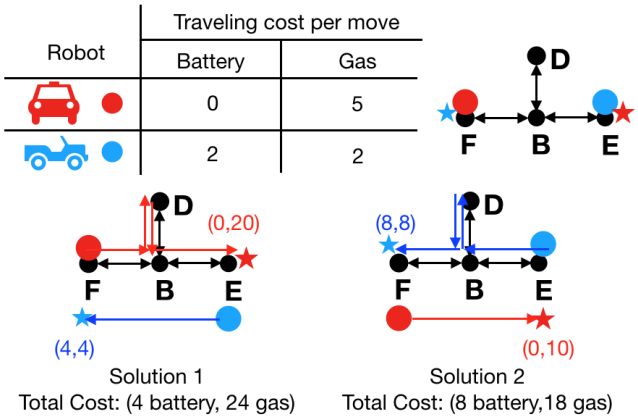

Fig. 1: A toy example of the MOMAPF problem with two objectives: minimizing total battery usage and gas costs.

Among the algorithms that optimally solve the single-objective MAPF problems, conflict-based search (CBS) [16] has received significant attention due to its computational efficiency on average. This method has also been extended to solve several other variants of MAPF as noted in [3], [8]. However, we are not currently aware of any CBS or other algorithms that can directly solve a MOMAPF to optimality. This article takes a first step in addressing this gap. By leveraging multi-objective dominance techniques [5], we develop a new algorithm named multi-objective conflict-based search (MO-CBS) that is able to compute the entire Pareto-optimal set of collision-free paths with respect to multiple objectives.

To bypass the curse of dimensionality, MO-CBS takes a similar strategy as CBS to resolve conflicts along paths of agents while extending CBS to handle multiple objectives. MO-CBS begins by computing individual Pareto-optimal paths for each agent ignoring agent-agent conflicts and letting agents follow those paths. When a conflict between agents is found along their paths, MO-CBS splits the conflict by adding constraints to either agent's individual search space and calls a multi-objective single-agent planner to compute new individual Pareto-optimal paths subject to those added constraints. In addition, MO-CBS uses dominance rules to select candidate solutions for conflict-checking and compare them until all the candidates are either pruned or identified as Pareto-optimal. To speed up the identification of the first solution, we also developed a variant of MO-CBS named MO-CBS-t, which uses a different candidate selection strategy. Compared with existing approaches that are guaranteed to find all Pareto-optimal solutions, such as A New Approach to

[1] Zhongqiang Ren and Howie Choset are with Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA.

[2]Sivakumar Rathinam is with Texas A&M University, College Station, TX 77843-3123.

[1]A solution is Pareto-optimal if there exists no other solution that will yield an improvement in one objective without causing a deterioration in at least one of the other objectives.

Multi-Objective A* (NAMOA*) [10], the numerical results show that the proposed MO-CBS outperforms NAMOA* in terms of success rates under bounded time as well as average run time for MOMAPF in general.

Related Work. MAPF methods [7], [17], [20], [24], [26] range from centralized to decentralized, trading off completeness and optimality for scalability. Conflict-based search (CBS) [16], a leading method in the middle of the spectrum, have been improved [1], [2] and extended to many different problems [3], [8], to name a few.

To optimize multiple objectives, dominance principles from multi-objective optimization (MOO) [5] have been applied to single agent path finding [10], [19], [22], motion planning [21], reinforcement learning [13] and others [4], [11], to name a few. With respect to MOO for multiple agents, the only work we are aware of is on evolutionary algorithms [25] for a related variant of MOMAPF considered in this work. Specifically, in [25], the conflicts between the agent's paths is modeled in one of the objectives and not as a constraint; as a result, a solution in the Pareto-optimal set may have agent's paths that conflict with each other. In addition, the work in [25] enforces a pre-defined number of waypoints to be visited in the middle of each agent's path and assumes no wait times for agents at any vertex in the graph, which greatly reduces the size of the Pareto-optimal set. In this work, we allow for agents to wait at any vertex in the graph and seek conflict-free paths for agents in any feasible solution as in the standard MAPF [18].

One of the basic challenges in multi-objective search is to compute the entire set of Pareto-optimal solutions [5], [15]. To overcome this challenge, search algorithms like A* [6], as a best-first search method, are extended to multi-objective A* (MOA*) [19] and later improved in NAMOA* [10]. Even though MOA*-based approaches can be applied to the product of the configuration spaces of agents, numerical results in Sec. VI show that it is significantly less efficient compared to the proposed MO-CBS.

II. Problem Formulation

Let index set $I = \{1, 2, \dots, N\}$ denote a set of N agents. All agents move in a workspace represented as a finite graph $G = (V, E)$, where the vertex set V represents all possible locations of agents and the edge set $E = V \times V$ denotes the set of all the possible actions that can move an agent between any two vertices in V. An edge between two vertices $u, v \in V$ is denoted as $(u, v) \in E$ and the cost of an edge $e \in E$ is a M-dimensional non-negative vector $\text{cost}(e) \in (\mathbb{R}^+)^M \backslash \{0\}$ with M being a positive integer.

In this work, we use a superscript $i, j \in I$ over a variable to represent the specific agent that the variable belongs to (e.g. $v^i \in V$ means a vertex with respect to agent i). Let $\pi^i(v_1^i, v_\ell^i)$ be a path that connects vertices v_1^i and v_ℓ^i via a sequence of vertices $(v_1^i, v_2^i, \dots, v_\ell^i)$ in the graph G. Let $g^i(\pi^i(v_1^i, v_\ell^i))$ denote the M-dimensional cost vector associated with the path, which is the sum of the cost vectors of all the edges present in the path, *i.e.*, $g^i(\pi^i(v_1^i, v_\ell^i)) = \Sigma_{j=1,2,\dots,\ell-1}\text{cost}(v_j^i, v_{j+1}^i)$.

All agents share a global clock and all the agents start their paths at time $t = 0$. Each action, either wait or move, for any agent requires one unit of time. Any two agents $i, j \in I$ are said to be in conflict if one of the following two cases happens. The first case is a "vertex conflict" where two agents occupy the same location at the same time. The second case is an "edge conflict" where two agents move through the same edge from opposite directions at times t and $t + 1$ for some t.

Let $v_o^i, v_f^i \in V$ respectively denote the initial location and the destination of agent i. Without loss of generality, to simplify the notations, we also refer to a path $\pi^i(v_o^i, v_f^i)$ for agent i between its initial and final locations as simply π^i. Let $\pi = (\pi^1, \pi^2, \dots, \pi^N)$ represent a joint path for all the agents, which is also called a solution. The cost vector of this solution is defined as the vector sum of the individual path costs over all the agents, *i.e.*, $g(\pi) = \Sigma_i g^i(\pi^i)$.

To compare any two solutions, we compare the cost vectors corresponding to them. Given two vectors a and b, a is said to *dominate* b if every component in a is no larger than the corresponding component in b and there exists at least one component in a that is strictly less than the corresponding component in b. Formally, it is defined as:

Definition 1 (Dominance [10]): Given two vectors a and b of length M, a dominates b, notationally $a \succeq b$, if and only if $a(m) \leq b(m)$, $\forall m \in \{1, 2, \dots, M\}$ and $a(m) < b(m)$, $\exists m \in \{1, 2, \dots, M\}$.

If a does not dominate b, this non-dominance is denoted as $a \not\succeq b$. Any two solutions are non-dominated if the corresponding cost vectors do not dominate each other. The set of all non-dominated conflict-free solutions is called the *Pareto-optimal* set. In this work, we aim to find any maximal subset of the Pareto-optimal set, where any two solutions in this subset do not have the same cost vector.

III. A Brief Review of Conflict-based Search

A. Conflicts and Constraints

Let (i, j, v^i, v^j, t) denote a *conflict* between agent $i, j \in I$, with $v^i, v^j \in V$ representing the vertex of agent i, j at time t. In addition, to represent a vertex conflict, v^i is required to be the same as v^j and they both represent the location where vertex conflict happens. To represent an edge conflict, v^i, v^j denote the adjacent vertices that agent i, j swap at time t and $t + 1$. Given a pair of individual paths π^i, π^j of agent $i, j \in I$, to detect a conflict, let $\Psi(\pi^i, \pi^j)$ represent a conflict checking function that returns either an empty set if there is no conflict, or the first conflict detected along π^i, π^j.

A conflict (i, j, v^i, v^j, t) can be avoided by adding a corresponding constraint to the path of either agent i or agent j. Specifically, let $\omega^i = (i, u_a^i, u_b^i, t), u_a^i, u_b^i \in V$ denote a *constraint* generated from conflict (i, j, v^i, v^j, t) that belongs to agent i with $u_a^i = v^i, u_b^i = v^j$ and the following specifications.

- If $u_a^i = u_b^i$, ω^i forbids agent i from entering u_a^i at time t and is named as a *vertex constraint* as it corresponds to a vertex conflict.

- If $u_a^i \neq u_b^i$, ω^i forbids agent i from moving from u_a^i to u_b^i between time t and $t+1$ and is named as an *edge constraint* as it corresponds to an edge conflict.

Given a set of constraints Ω, let $\Omega^i \subseteq \Omega$ represent the subset of all constraints in Ω that belong to agent i (*i.e.*, $\Omega = \bigcup_{i \in I} \Omega^i$). A path π^i is *consistent* with respect to Ω if π^i satisfies every constraint in Ω^i. A joint path π is consistent with respect to Ω if every individual path $\pi^i \in \pi$ is consistent.

B. Two Level Search

CBS is a two level search algorithm. The low level search in CBS is a single-agent optimal path planner that plans an optimal and consistent path for an agent i with respect to a set of constraints Ω^i. If there is no consistent path for agent i given Ω^i, low level search reports failure.

For the high level search, CBS constructs a search tree $\mathcal{T}$ with each tree node P containing:

- $\pi = (\pi^1, \pi^2, \ldots, \pi^N)$, a joint path that connect start and goal vertices of agents respectively,
- g, a scalar cost value associated with π and
- a set of constraints Ω.

The root node P_o of $\mathcal{T}$ has an empty set of constraints $\Omega_o = \emptyset$ and the corresponding joint path π_o is constructed by the low level search for every agent respectively with $\Omega_o = \emptyset$.

To process a high level search node $P_k = (\pi_k, g_k, \Omega_k)$, where subscript k identifies a specific node, conflict checking $\Psi(\pi_k^i, \pi_k^j)$ is computed for any pair of individual paths in π_k with $i, j \in I, i \neq j$. If there is no conflict detected, a solution is found and the algorithm terminates. If there is a conflict (i, j, v^i, v^j, t) detected, to resolve the detected conflict, CBS conducts the following procedures. First, CBS *splits* the detected conflict to generate two constraints $\omega^i = (i, u_a^i = v^i, u_b^i = v^j, t)$ and $\omega^j = (j, u_a^j = v^j, u_b^j = v^i, t)$. Secondly, CBS generates two corresponding nodes $P_{l^i} = (\pi_{l^i}, g_{l^i}, \Omega_{l^i}), P_{l^j} = (\pi_{l^j}, g_{l^j}, \Omega_{l^j})$, where $\Omega_{l^i} = \Omega_k \cup \{w^i\}$ and $\Omega_{l^j} = \Omega_k \cup \{w^j\}$. Finally, CBS updates individual path π^i in joint path π_{l^i} (and π^j in π_{l^j}) by calling the low level search for agent i (and j) with a set of constraints Ω_{l^i} (and Ω_{l^j} respectively). If low level search fails to find a consistent path for i (or j), node P_{l^i} (or P_{l^j}) is discarded.

After conflict resolving, CBS inserts generated nodes into OPEN, which is a priority queue containing all candidate high level nodes. CBS optimally solves a (single-objective) MAPF problem by iteratively selecting candidate node from OPEN with the smallest g cost, detect conflicts, and then either claims success (if not conflict detected) or resolves the detected conflict which generates new candidate nodes.

Intuitively, from the perspective of the search tree $\mathcal{T}$ constructed by CBS, OPEN contains all leaf nodes in $\mathcal{T}$. In each round of the high-level search, a leaf node P_k is selected and checked for conflict. CBS either claims success if paths in P_k are conflict-free or generates new leaf nodes.

IV. Multi-objective Conflict-based Search

Multi-objective conflict-based search (MO-CBS), as described in Alg. 1, follows a similar workflow as CBS. MO-CBS generalizes CBS to handle multiple objectives with the following several key differences.

Algorithm 1 Pseudocode for MO-CBS, MO-CBS-t

```
1:  Initialization()
2:  S ← ∅
3:  while OPEN not empty do
4:      P_k = (π_k, g⃗_k, Ω_k) ← OPEN.pop()
5:      // P_k = (π_k, g⃗_k, Ω_k) ← OPEN.pop-tree-by-tree()
6:      if no conflict detected in π_k then
7:          FilterSolution(P_k)
8:          add P_k to S
9:          FilterOpen(P_k)
10:         continue
11:     Ω ← Split detected conflict
12:     for all ω^i ∈ Ω do
13:         Ω_l = Ω_k ∪ {ω^i}
14:         Π*_i ← LowLevelSearch(i, Ω_l)
15:         for all π*_i ∈ Π*_i do
16:             π_l ← π_k
17:             Replace π^i_l (in π_l) with π*_i
18:             g_l ← compute path cost π_l
19:             if NonDom(g_l) then
20:                 P_l = (π_l, g⃗_l, Ω_l)
21:                 add P_l to OPEN
22: return S
```

A. Initialization

In MO-CBS, to initialize OPEN (line 1 in Alg. 1), a single-agent multi-objective planner (e.g. NAMOA* [10]) is used for each agent $i \in I$ separately to compute all cost-unique Pareto-optimal paths, Π_o^i, for agent i. A set of joint paths Π_o is generated by taking the combination of $\Pi_o^i, \forall i \in I$, *i.e.* $\Pi_o = \{\pi_o | \pi_o = (\pi_o^1, \pi_o^2, \ldots, \pi_o^N), \pi_o^i \in \Pi_o^i, \forall i \in I\}$. Clearly, the size of Π_o is $|\Pi_o| = |\Pi_o^1| \times |\Pi_o^2| \times \cdots \times |\Pi_o^N|$. For each $\pi_o \in \Pi_o$, a high level node containing π_o, a cost vector associated with π_o and an empty constraint set is generated and added into OPEN.Intuitively, while the original CBS initializes a single root node and a single search tree $\mathcal{T}$, MO-CBS initializes a number of $R = |\Pi_o|$ root nodes and a "search forest" $\mathcal{T}_r, r = \{1, 2, \ldots, R\}$ where each tree $\mathcal{T}_r$ corresponds to a root node.

B. Finding a Solution

For every iteration in MO-CBS, a high level node $P_k = (\pi_k, \vec{g}_k, \Omega_k)$ with a non-dominated cost vector $\vec{g}_k$ in OPEN is popped and processed. If π_k is conflict-free, a solution node is identified, and P_k is inserted into $\mathcal{S}$, a set of high level nodes where each node contains conflict-free paths for the agents. Besides, two additional procedures need to be conducted when finding a solution.

Procedure FilterSolution(P_k) uses the cost vector in P_k to filter all previously found solution node in $\mathcal{S}$. For every node $P_l = (\pi_l, \vec{g}_l, \Omega_l) \in \mathcal{S}$, $\vec{g}_l$ is compared with $\vec{g}_k$ in P_k and if $\vec{g}_k \succeq \vec{g}_l$ (or $\vec{g}_k = \vec{g}_l$), then P_l is removed from $\mathcal{S}$. The necessity of this procedure is rooted at the difference that

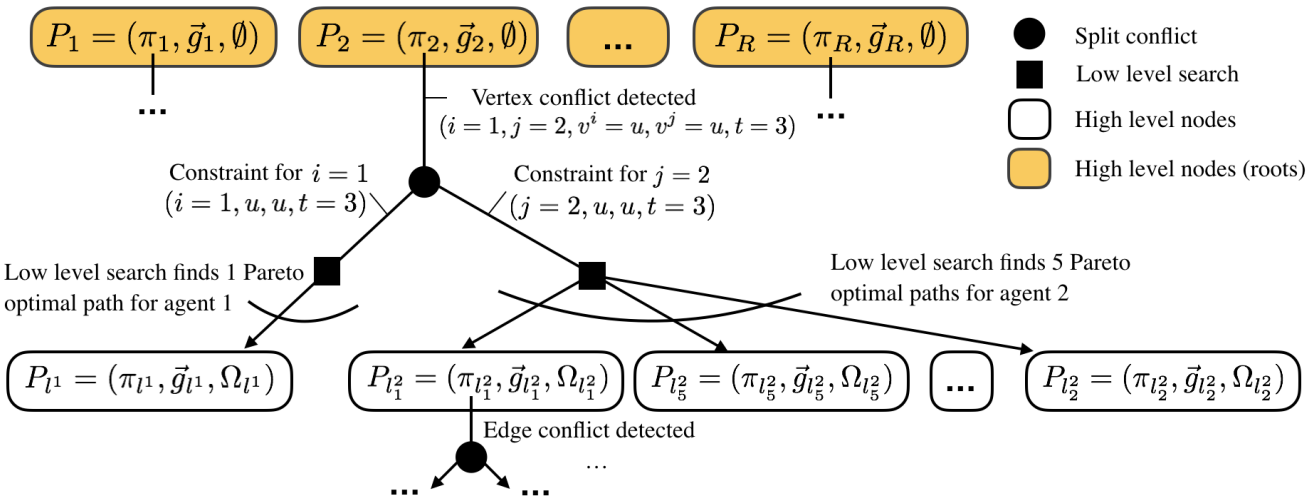

Fig. 2: An illustration of the search process of MO-CBS.

a search forest $\mathcal{T}_r, r = \{1, 2, \ldots, R\}$, rather than a single search tree, is constructed by MO-CBS. While each node P_l in $\mathcal{S}$ is guaranteed to be non-dominated with respect to the search tree that contains it (see Sec. V for a proof), P_l is not guaranteed to be non-dominated with respect to other search trees. When a solution node P_k is found in a tree $\mathcal{T}_k$, FilterSolution(P_k) removes solution nodes with dominated cost vectors that are found in other trees $\mathcal{T}_l, l \neq k$. As a result, $\mathcal{S}$ is guaranteed to be a set, where each node contains a Pareto-optimal solution when the algorithm terminates.

In addition, to filter nodes with dominated cost vectors in OPEN, procedure FilterOpen is called when a solution node is found. As $P_k = (\pi_k, \vec{g}_k, \Omega_k)$ contains a conflict-free joint path, any candidates node $P_l = (\pi_l, \vec{g}_l, \Omega_l)$ in OPEN with a dominated cost vector $\vec{g}_l$ (*i.e.* $\vec{g}_k \succeq \vec{g}_l$) can not lead to a Pareto-optimal solution and is thus removed from OPEN.

While CBS terminates when the first solution is identified, MO-CBS continues to search when a solution is identified and terminates only when OPEN is empty in order to identify all cost-unique Pareto-optimal solutions.

C. Conflict Resolution

When a node $P_k = (\pi_k, \vec{g}_k, \Omega_k)$ is popped from OPEN, if π_k contains a conflict, just as in CBS, the detected conflict is split into two constraints and a new set of constraints Ω_l is generated correspondingly. Given an agent i and a constraint set Ω_l, the low level search (which is explained next) is called to compute consistent paths with respect to Ω_l for agent i.

Given Ω_l and an agent i, while in CBS, only one individual optimal path for agent i (that is consistent with Ω_l) is computed, in MO-CBS, there can be multiple consistent Pareto-optimal individual paths for agent i. To find all of them, the low level search uses NAMOA* [10] to search over a time-augmented graph $G^t = (V^t, E^t) = G \times \{0, 1, \ldots, T\}$, where each vertex in $v \in V^t$ is defined as $v = (u, t), u \in V, t \in \{0, 1, \ldots, T\}$ and T is a pre-defined time horizon (a large positive integer). Edges within G^t is represented as $E^t = V^t \times V^t$ where $(u_1, t_1), (u_2, t_2)$ is connected in G^t if $(u_1, u_2) \in E$ and $t_2 = t_1 + 1$. Wait in place is also allowed in G^t, *i.e.* $(u, t_1), (u, t_1 + 1), u \in V$ is connected in G^t. In addition, all vertices and edges in G^t that correspond to vertex constraints and edge constraints in $\Omega_l^i \subseteq \Omega_l$ are removed from G^t.

This NAMOA*-based low level search guarantees to return a set of consistent, Pareto-optimal individual paths Π_*^i for agent i subject to the given constraint set. For each path $\pi_*^i \in \Pi_*^i$ computed by the low level search, a corresponding joint path π_l is generated by first copy from π_k and then update the individual path π_l^i in π_l with π_*^i (line 16-17 in Alg. 1). If the cost vector of π_l is neither dominated nor equalized by any solutions found so far, a new node $P_l = (\pi_l, \vec{g}_l, \Omega_l)$ is generated and inserted into OPEN.

D. Tree-by-tree Expansion

In MO-CBS, a node with a non-dominated cost vector is selected from OPEN and expanded (conflict checking and splitting). This expansion strategy has two drawbacks. First of all, all root nodes need to be generated so that a non-dominated one can be selected. Considering an example with ten agents and each agent has ten individual non-dominated paths, in this case, MO-CBS needs to generate 10^{10} root nodes, which is computationally prohibitive. Another drawback is that nodes are selected in a "breadth-first" manner in the sense that selected nodes can belong to different trees. As the number of agents (or objectives) increases, this expansion strategy may lead to a large number of nodes expanded before finding the first solution.

Here we propose a new expansion strategy. Let candidates in OPEN be sorted by the tree $\mathcal{T}_r$ they belong to, and let OPEN_r denote the open list that contains only candidate in tree $\mathcal{T}_r, r \in \{1, 2, \ldots, R\}$. Clearly, OPEN$= \bigcup_{r \in 1,2,\ldots,R} \text{OPEN}_r$. Instead of selecting an arbitrary non-dominated node in OPEN as MO-CBS does, initially, only nodes in OPEN_1 are selected for expansion. The selected

node is then expanded in the same manner as MO-CBS. As any newly generated nodes belong to $\mathcal{T}_1$, those nodes must be inserted into OPEN_1. Only when OPEN_1 depletes, the algorithm then selects candidates from OPEN_2 (and then OPEN_3, and so on) for expansion. The algorithm terminates when OPEN_R is depleted. We denote MO-CBS with such a "tree-by-tree" node selection strategy as MO-CBS-t, where "-t" stands for tree-by-tree.

MO-CBS-t enables *on-demand* generation of roots and performs a "depth-first" like search by exhaustively examining one tree after another, which makes MO-CBS-t find the first feasible solution potentially earlier than MO-CBS. Note that those computed feasible solutions are only non-dominated within the tree it belongs to and is not guaranteed to be non-dominated among all trees. $\mathcal{S}$ is guaranteed to be Pareto-optimal only when the algorithm terminates. However, numerical results in Sec. VI show that the cost vector of the first solution found by MO-CBS-t is very "close" to be Pareto-optimal.

E. Relationship to CBS

With only one objective, $i.e.$ $M = 1$, MO-CBS is equivalent to CBS in the following sense. Dominance between vectors becomes the "less than" comparison between scalars and the candidate with the minimum g cost in OPEN is popped in each iteration. When the first solution with the minimum g cost is found, all other nodes in OPEN are filtered by the FilterSolution procedure and MO-CBS terminates. The low level search returns one optimal solution at any time of the search. Only one root is generated and there is one corresponding search tree built during the search.

V. Analysis

Let $\mathcal{T}_r, r \in \{1, 2, \dots, R\}$ represent a set of search trees where $\mathcal{T}_r$ corresponds to the r-th root node created in the initialization step. Let $\mathcal{T}(P_k)$ denote the search tree that node P_k belongs to.

Definition 2 (CV set): Given a high level node $P_k = (\pi_k, \vec{g}_k, \Omega_k)$, let $CV(P_k)$ be a set of high level nodes in $\mathcal{T}(P_k)$ where each node $P_l = (\pi_l, \vec{g}_l, \Omega_l) \in CV(P_k)$ satisfies the following requirements: π_l is consistent with Ω_k, and π_l is conflict-free ($i.e.$ valid).

In addition, if $P_l = (\pi_l, \vec{g}_l, \Omega_l) \in CV(P_k)$, we say P_k permits π_l.

Lemma 1: Given a node $P_k = (\pi_k, \vec{g}_k, \Omega_k)$, none of the nodes in $CV(P_k)$ has a cost vector that dominates $\vec{g}_k$, $i.e.$ $\vec{g}_l \not\succeq \vec{g}_k, \forall P_l = (\pi_l, \vec{g}_l, \Omega_l) \in CV(P_k)$.

Proof: Every individual path in $\pi_k^i \in \pi_k, \forall i \in I$ is computed by the low level planner, which guarantees π_k^i is a non-dominated individual path that is consistent with $\Omega_k^i \subseteq \Omega_k$, the subset of constraints that belongs to agent i. Since $\Omega_k = \bigcup_{i \in I} \Omega_k^i$, π_k is a non-dominated joint path among all joint paths consistent with Ω_k. Assume that there exists a node $P_l = (\pi_l, \vec{g}_l, \Omega_l) \in CV(P_k)$ with $\vec{g}_l \succeq \vec{g}_k$. Since for any node $P_l = (\pi_l, \vec{g}_l, \Omega_l) \in CV(P_k)$, we have $\Omega_k \subseteq \Omega_l$. Thus, every individual path $\pi_l^i \in \pi_l$ is also a consistent path for $\Omega_k^i, \forall i \in I$. This implies that π_l is also consistent with Ω_k with g_l dominating g_k which contradicts. ■

Lemma 2: Let Π_* denote the set of cost-unique Pareto-optimal joint paths for a MOMAPF problem. For each $\pi_* \in \Pi_*$, there is always a candidate node in OPEN that permits π_*.

Proof: This lemma can be shown by induction. Initially, every root node contains an empty set of constraints and permits all conflict-free Pareto-optimal paths. Assume this lemma holds for the k-th iteration and let P_k be a node in OPEN that permits some $\pi_* \in \Pi_*$. Then in the $(k+1)$-th iteration, P_k is expanded and two new nodes P_{l^i}, P_{l^j} are generated. π_* must be contained in a node that belongs to either $CV(P_{l^i})$ or $CV(P_{l^j})$ because any conflict-free joint path must satisfy either of the constraints. Therefore, for $(k+1)$-th iteration, either P_{l^i} or P_{l^j} permits π_*. ■

Theorem 1: MO-CBS finds all $\pi_* \in \Pi_*$.

Proof: From Lemma 1, every identified solution $P_k = (\pi_k, \vec{g}_k, \Omega_k)$ must be Pareto-optimal within $\mathcal{T}(P_k)$ since none of the nodes in $CV(P_k)$ can have a cost vector that dominates $\vec{g}_k$. With FilterSolution procedure, $\mathcal{S}$ is guaranteed to contain only Pareto-optimal solutions over all search trees $\mathcal{T}_r, r = 1, 2, \dots, R$. From Lemma 2, nodes in OPEN permits all $\pi_* \in \Pi_*$ and MO-CBS terminates when OPEN depletes. Therefore, MO-CBS finds all $\pi_* \in \Pi_*$ at termination. ■

Algorithm MO-CBS-t, as a variant of MO-CBS, is also able to find all cost-unique Pareto-optimal solutions and the above proof also applies to MO-CBS-t.

VI. Numerical Results

A. Test Settings and Implementation

We implemented MO-CBS, MO-CBS-t and NAMOA* [10] in Python. NAMOA* serves as a baseline approach, which is applied to the the joint graph corresponding to agents to search for all cost-unique Pareto-optimal solutions. All algorithms are compared on a computer with an Intel Core i7 CPU and 16GB RAM. We selected four maps (grids) from different categories in [18] and generated an un-directed graph G by making each grid four-connected. To assign cost vectors to edges in G, we first assigned every agent a cost vector $a^i, \forall i \in I$ of length M (the number of objectives) and assigned every edge e in G a scaling vector $b(e)$ of length M, where each element in both a^i and $b(e)$ were randomly sampled from integers in $[1, 10]$. The range $[1, 10]$ follows the convention used in [9]. The cost vector for agent i to go through e is the component-wise product of a^i and $b(e)$. If agent i wait in place, the cost incurred is a^i. We tested the algorithms with different number of objectives M and different number of agents N within a run time limit of *five* minutes.

In the implementation of MO-CBS and MO-CBS-t, for the low level search, the heuristic vector for a node is implemented as a unit vector scaled by the Manhattan distance to the destination, which is same to the heuristics used in [9]. For the high level search, all nodes with non-dominated cost vectors in OPEN are lexicographically sorted and the minimum one is popped from OPEN.

B. MO-CBS vs NAMOA* with Different M

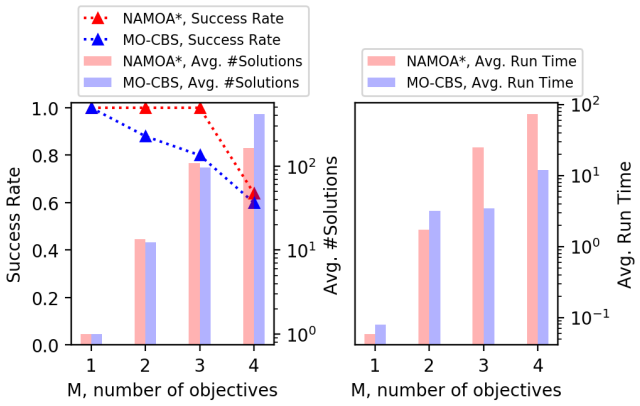

Fig. 3: Comparison of NAMOA* and MO-CBS with $N = 2$ against different metrics as functions of M.

We first limited our focus to the grid "empty-16-16" in [18], which is a 16 by 16 grids without obstacles. Fig. 3 and Fig. 4 shows the results with $N = 2$ and $N = 4$ (fixed N) respectively, against the metrics of (1) success rates of finding all cost-unique Pareto-optimal solutions, (2) average number of solutions found, (3) average number of states (NAMOA*) or high-level nodes (for MO-CBS) expanded and (4) average run time, over all test instances. All the averages are taken over instances where all cost-unique Pareto-optimal solutions are found and these instances are called solved instances in the subsequent sections. From Fig. 3, we find that NAMOA* has a higher success rates $M = 2, 3, 4$. However, from Fig. 4, with four agents ($N = 4$), MO-CBS achieves higher success rates and shorter run time than NAMOA* with all $M = 1, 2, 3, 4$.

In addition, from both figures, the number of solutions increases as M increases, which indicates the difficulty of MOMAPF. Note that, when $M = 1$, the MOMAPF problem with a single objective is the same as the conventional MAPF problem and MO-CBS is equivalent to CBS.

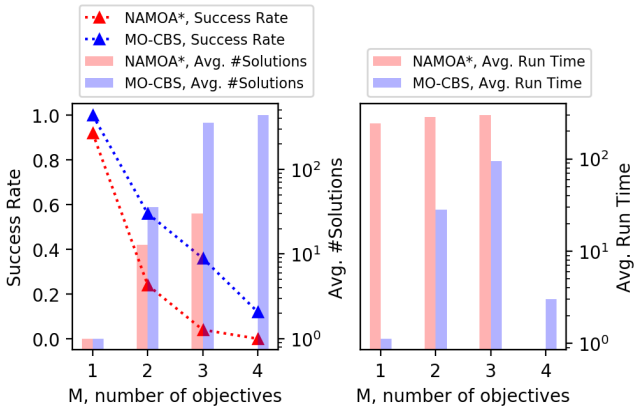

Fig. 4: Comparison of NAMOA* and MO-CBS with $N = 4$ against different metrics as functions of M.

C. MO-CBS with Different N

Fig. 5 reports the performance of MO-CBS and MO-CBS-t in terms of (1) success rates of finding *all* cost-unique Pareto-optimal solutions, (2) success rates of finding at least *one* feasible solution, (3) average numbers of high-level nodes expanded and (4) average run times, in three different types of grids, with a fixed $M = 2$ and varying N. All the averages are taken over solved instances. MO-CBS-t outperforms MO-CBS in terms of identifying the first solution due to its tree-by-tree search strategy. In grid such as den312d of size 65x81, the number of individual Pareto-optimal solutions for each agent can exceed a hundred. As a result, the number of roots to be searched can grow tremendously when N increases. In this case, MO-CBS cannot finish initialization as generating all those roots is computationally burdensome, while MO-CBS-t can start the search without generating all roots and return a feasible solution as soon as possible.

In addition, from all the plots in Fig. 5, both MO-CBS and MO-CBS-t do not scale well with an increasing N, which is an interesting topic for our future work.

D. Sub-optimality of the First Solution Found by MO-CBS-t

To estimate the sub-optimality of the first solution found by MO-CBS-t before termination, we recorded the corresponding cost vector x. For the instances where MO-CBS-t successfully finds all cost-unique Pareto-optimal solutions, let X^* denote the set of Pareto-optimal cost vectors. We first computed $d(x, X^*) = \inf_{x^* \in X^*} ||x - x^*||_2$ [14], the distance of x to X^* in Euclidean norm. We then computed a measure of sub-optimality in percentage by dividing $d(x, X^*)$ with the averaged Euclidean norm of all vectors in X^*. The results show that for all three grids shown in Fig. 5, this percentage is on average less than 1%, maximally less than 3% and minimally zero. It indicates, empirically, the first solution found by MO-CBS-t is exactly Pareto-optimal or very "close" to be Pareto-optimal.

VII. Conclusion

New algorithms, MO-CBS and its variant MO-CBS-t, were presented for a multi-objective multi-agent path finding problem. We showed that the algorithm is able to find the entire Pareto-optimal set. Numerical results showed the proposed approach outperforms the baseline against different metrics. There are several directions for future work. One can focus on approximating the Pareto-optimal sets for better scability. One can also develop other types of multi-objective MAPF algorithms such as the ones based on the subdimensional expansion framework [23]. We have preliminary results in this research direction as shown in [12].

References

[1] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

[2] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Eyal Shimony. Icbs: improved conflict-based search algorithm for multi-agent pathfinding. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

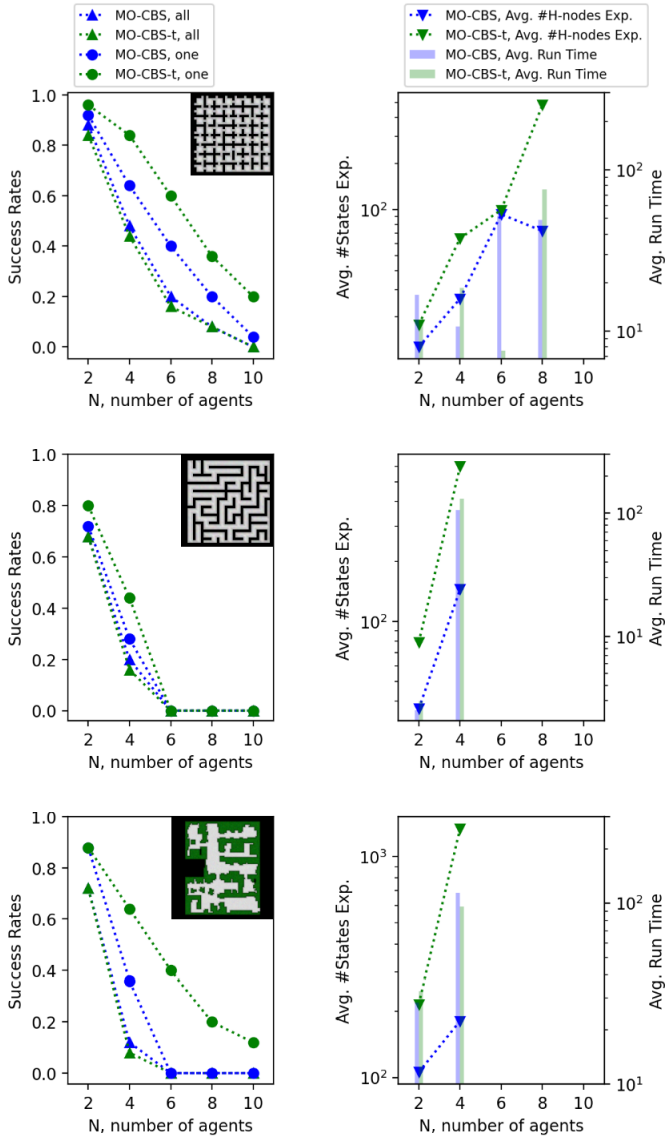

Fig. 5: Comparison between MO-CBS and MO-CBS-t with varying N in different grids (room 32x32, maze 32x32, den312d 65x81). All the averages in the right sub-plots are taken over solved instances.

[3] Liron Cohen, Tansel Uras, TK Satish Kumar, and Sven Koenig. Optimal and bounded-suboptimal multi-agent motion planning. In *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[4] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[5] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[7] Ryan J Luna and Kostas E Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[8] Hang Ma, Jiaoyang Li, T K Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Conference on Autonomous Agents & Multiagent Systems*, 2017.

[9] Lawrence Mandow, JL Pérez De la Cruz, et al. A new approach to multiobjective aˆ* search. In *IJCAI*, volume 8. Citeseer, 2005.

[10] Lawrence Mandow and José Luis Pérez De La Cruz. Multiobjective a* search with consistent heuristics. *Journal of the ACM (JACM)*, 57(5):1–25, 2008.

[11] Justin Montoya, Sivakumar Rathinam, and Zachary Wood. Multi-objective departure runway scheduling using dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):399–413, 2013.

[12] Zhongqiang Ren, Sivakumar Rathinam, and Howie Choset. Subdimensional expansion for multi-objective multi-agent path finding. *arXiv preprint arXiv:2102.01353*, 2021.

[13] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[14] Oliver Schutze, Xavier Esquivel, Adriana Lara, and Carlos A Coello Coello. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522, 2012.

[15] Paolo Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In *Recent advances and historical development of vector optimization*, pages 222–232. Springer, 1987.

[16] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[17] David Silver. Cooperative pathfinding. pages 117–122, 01 2005.

[18] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. *arXiv preprint arXiv:1906.08291*, 2019.

[19] Bradley S. Stewart and Chelsea C. White. Multiobjective a*. *J. ACM*, 38(4):775–814, October 1991.

[20] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Modifying optimal sat-based approach to multi-agent path-finding problem to suboptimal variants. 07 2017.

[21] Matthew Tesch, Jeff Schneider, and Howie Choset. Expensive multiobjective optimization for robotics. In *2013 IEEE International Conference on Robotics and Automation*, pages 973–980. IEEE, 2013.

[22] Carlos Hernández Ulloa, William Yeoh, Jorge A Baier, Han Zhang, Luis Suazo, and Sven Koenig. A simple and fast bi-objective search algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 143–151, 2020.

[23] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[24] Ko-Hsin Cindy Wang, Adi Botea, et al. Fast and memory-efficient multi-agent pathfinding. In *ICAPS*, pages 380–387, 2008.

[25] J. Weise, S. Mai, H. Zille, and S. Mostaghim. On the scalable multi-objective multi-agent pathfinding problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[26] Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.