Bounded Sub-optimal Algorithms for Teamwise Cooperative Multi-Agent Path Finding

Zhenlong Fang[1,2], Yilin Cai[3] and Zhongqiang Ren[1†]

***Abstract*— Multi-Agent Teamwise Cooperative Path Finding (TC-MAPF) seeks collision-free paths for the agents from their start to goal locations. In addition, agents are grouped into multiple teams, and each team has its own objective function to optimize. TC-MAPF arises in scenarios such as the coordination of multiple autonomous vehicles at a signal-free traffic intersection, where, for example, the vehicles from each direction naturally forms a team. TC-MAPF was recently studied and optimal planners such as TC-CBS has been developed. While being able to find all Pareto-optimal solutions for TC-MAPF, these optimal planners usually suffer from limited scalability as the number of agents grows. This paper develops a bounded sub-optimal planner TC-CBS-TF for TC-MAPF, trading off solution quality for scalability, by leveraging and extending several bounded sub-optimal search techniques for MAPF to handle multiple teams as in TC-MAPF. We test TC-CBS-TF and baselines on various maps with up to 50 agents, and TC-CBS-TF achieves up to 68% higher success rates.**

I. Introduction

Multi-Agent Path Finding (MAPF) aims to plan collision-free paths for multiple agents from start to goal locations, which has been widely studied over the last decade [1]. Traditional MAPF planner typically considers the scenario where all agents are cooperative in a sense that all agents seek to optimize a common objective function such as the sum of agents' arrival times [1], [2]. However, there are scenarios where agents are not fully cooperative. A motivating example is the traffic intersection (Fig. 1), where agents from one direction only pay attention to their traversal time through the intersection, without worrying about the traversal time of the agents from other directions. Consider a four-way intersection, the agents naturally form four teams, and each team seeks to minimize its own traversal time. Since the intersection is shared by all the teams and agent-agent collision must be avoided, the motion coordination has to trade the traversal time of one team for the other.

This motivates the Multi-Agent Teamwise Cooperative Path Finding (TC-MAPF) problem [3], [4], a generalization of MAPF where agents are grouped into teams, and each team has its own objective function. Specifically, each agent has its own start and goal locations and belongs to at least one team. Every team has its own objective function to minimize such as min-sum, the sum of arrival times of the agents within the team. As there are more than one teams in general, TC-MAPF seeks to minimize an objective vector, where each component of the vector corresponds to the objective of a team. In the presence of multiple objectives, in general, there is no single solution that minimizes all objectives simultaneously. TC-MAPF thus aims to find a set of Pareto-optimal solutions. A solution is Pareto-optimal if one cannot improve over one objective without deteriorating another objective. TC-MAPF differs from the existing Multi-Agent Multi-Objective Path Finding (MOMAPF) [5], [6], self-interested MAPF [7] and adversarial MAPF [8].

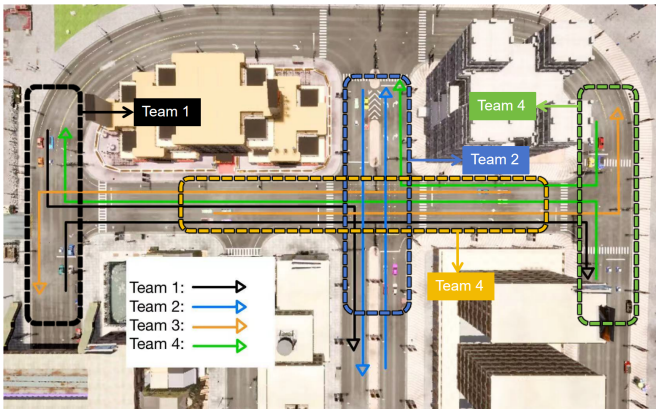

Fig. 1: An example with 32 agents divided into 4 teams traversing several intersections. TC-MAPF seeks Pareto-optimal solutions with different trade-off among the teams.

TC-MAPF generalizes MAPF from one team to multiple teams. MAPF is NP-hard [9] and so is TC-MAPF. Our prior work developed exact algorithms for TC-MAPF that can find all Pareto-optimal solutions, such as TC-CBS and TC-M* [3], [4], which extends the popular conflict-based search (CBS) [2] and M* [10] algorithms for MAPF to handle TC-MAPF, and discussed the conditions under which TC-CBS is complete and can find all Pareto-optimal solutions. TC-CBS was then improved by TC-CBS-T [4] with a transformation method to mitigate its incompleteness. While being able to find Pareto-optimal solutions, these exact algorithms often scale poorly as the number of agents or teams increases.

To address the issue of scalability, this paper develops a bounded sub-optimal planner for TC-MAPF, trading off solution quality for scalability. We propose TC-CBS-TF (F for Focal), which leverages and integrates several existing bounded sub-optimal search techniques such as focal search [11] into TC-CBS-TF. It also leverages the idea of flex distribution [12] in MAPF and proposes a technique to

[1]Zhenlong Fang conducted this work during his internship at Shanghai Jiao Tong University. Zhongqiang Ren is with Shanghai Jiao Tong University, Shanghai, China.

[2]Zhenlong Fang is with University of Minnesota, Twin Cities, MN, USA.

[2]Yilin Cai is with Georgia Institute of Technology, Atlanta, GA, USA.

† Corresponding author. zhongqiang.ren@sjtu.edu.cn

This work was supported by the Natural Science Foundation of Shanghai under Grant 24ZR1435900, and the Natural Science Foundation of China under Grant 62403313.

smartly allocate the sub-optimality bound factor among the teams, based on the paths of all agents within the same teams, to improve the planning efficiency.

We evaluated our approach and baselines in both the conventional grid maps for MAPF and traffic intersection grid maps. The results show that TC-CBS-TF doubles the amount of agents that the original TC-CBS-T can handle. The results also verify that the proposed flex distribution for TC-MAPF can enhance the success rates for more than 20% especially when the bound is tight.

A. Related Work

MAPF often minimizes a single-objective. When there is only one team including all agents, TC-MAPF becomes MAPF. To solve MAPF to optimality, various methods were developed, which focus on either min-sum [2], [10] or min-max [13], [14] problems. Recent work generalizes MAPF to Multi-Objective MAPF [5], [6], [15], [16] by associating a vector-cost (rather than a scalar-cost) to the action of an agent, where each component of the cost vector represents an objective to be minimized. MOMAPF requires minimizing the sum of accumulated cost vectors over all agents along their paths. The TC-MAPF differs from MOMAPF, since the action costs are scalars, and there are multiple teams, each with its own objective. Currently, there are only exact approaches for TC-MAPF [3], [4], which have limited scalability with respect to the number of agents.

Bounded sub-optimal MAPF algorithms have been extensively studied in the literature. Variants of CBS typically use focal search to ensure sub-optimality bound [11], [17], [18]. Recent work further studies how to smartly distribute the bound among the agents to improve the runtime efficiency [12]. However, these algorithms cannot directly handle the teams in TC-MAPF.

II. Problem Statement

Let $I = \{1, 2, \dots, N\}$ denote a set of N agents. All agents share a workspace represented as a finite graph $G = (V, E)$, where V is the set of possible locations (vertices) and $E \subseteq V \times V$ is the set of possible actions (edges) that move an agent between adjacent vertices. Each edge $e \in E$ has a positive cost, $cost(e) \in \mathbb{R}^+$. Let $v_o^i \in V$ and $v_d^i \in V$ denote the start and goal locations for agent $i \in I$, respectively.

We use a superscript $i \in I$ to associate variables with a specific agent (e.g., $v^i \in V$). A path for agent i from v_1^i to v_ℓ^i is a sequence of vertices $\pi^i(v_1^i, v_\ell^i) = (v_1^i, v_2^i, \dots, v_\ell^i)$ in G. The cost of this path is the sum of the costs of the edges traversed: $g(\pi^i(v_1^i, v_\ell^i)) = \sum_{j=1}^{\ell-1} cost(v_j^i, v_{j+1}^i)$. We simply use π^i for the path from v_o^i to v_d^i, and $g^i = g(\pi^i)$ for its cost, when unambiguous.

Agents run synchronously based on a common global clock starting at $t = 0$. Each action (moving to an adjacent vertex or waiting at the current vertex) takes one unit of time. A conflict occurs if two different agents $i, j \in I$ occupy the same vertex at the same time (vertex conflict) or traverse the same edge in opposite directions between times t and $t + 1$ (edge conflict).

Let $\mathcal{T} = \{T_j, j = 1, 2, \dots, M\}$ be a given set of M *teams*, where each team $T_j \subseteq I$ is a subset of agents. An agent can belong to multiple teams, and teams need not be disjoint. For a team T_j, let $\pi^{T_j} = \{\pi^i | \forall i \in T_j\}$ denote the set of individual paths for agents in that team. Each team T_j has an associated *objective function* g^{T_j} whose value depends on the paths π^{T_j}. This function g^{T_j} is non-decreasing with respect to the individual path costs g^i for $i \in T_j$. Common examples include the sum of costs ($g^{T_j} := \sum_{i \in T_j} g^i$) or the maximum cost ($g^{T_j} := \max_{i \in T_j} g^i$).

A *solution* π consists of conflict-free path π^i for every agent $i \in I$. The quality of a solution π is represented by an *objective vector* $\vec{g}(\pi) := \{g^{T_j}(\pi) \mid j = 1, 2, \dots, M\}$, where each component $g^{T_j}(\pi)$ is the objective value for team T_j based on the paths in π. We also refer to objective vectors as cost vectors. We compare solutions by comparing their objective vectors using the concept of dominance [19].

Definition 1 (Dominance)**.** *Given two objective vectors $\vec{a}$ and $\vec{b}$ of length M, $\vec{a}$ dominates $\vec{b}$, denoted $\vec{a} \preceq \vec{b}$, if and only if $a_m \leq b_m$ for all $m \in \{1, \dots, M\}$, and $a_k < b_k$ for at least one $k \in \{1, \dots, M\}$. In addition, if $a_m \leq b_m$ for all m, we write $\vec{a} \leq \vec{b}$.*

A solution π is *Pareto-optimal* if no other solution π' exists such that $\vec{g}(\pi') \preceq \vec{g}(\pi)$. The set of all Pareto-optimal solutions is denoted by Π_*, and the corresponding set of objective vectors $C_* = \{\vec{g}(\pi) | \pi \in \Pi_*\}$ forms the *Pareto-optimal front*. The goal of TC-MAPF is to find Π_* and C_*.

Definition 2 (α-approximation Set and Front)**.** *A set of solutions Π_α is an α-approximation of the Pareto-optimal set Π_* if for every Pareto-optimal solution $\pi^* \in \Pi_*$, there exists at least one solution $\pi \in \Pi_\alpha$ such that $\vec{g}(\pi) \leq \alpha \vec{g}(\pi^*)$, $\alpha \geq 1$. Here, $\alpha \vec{g}(\pi^*) = (\alpha g^{T_1}(\pi^*), \dots, \alpha g^{T_M}(\pi^*))$ denotes element-wise product. The corresponding cost vectors of Π_α is called an α-approximation of the Pareto-optimal front.*

Problem 1 (Bounded Sub-optimal TC-MAPF)**.** *Given a TC-MAPF instance $(G, \{v_o^i\}, \{v_d^i\}, \{T_j\}, \{g^{T_j}\})$ and a sub-optimality factor $\alpha \geq 0$, the goal is to find a α-approximation of the Pareto-optimal set Π_* and the corresponding cost vectors. Additionally, Π_α should be minimal in a sense that it contains only cost-unique and mutually non-dominated solutions in Π_α.*

III. Method

Our TC-CBS-TF builds upon TC-CBS [3], [4], bounded sub-optimal search techniques from Enhanced CBS (ECBS) [11] and Flex Distribution [12]. The main idea is to distribute the sub-optimality bound among the agents in an intelligent way to speed up the conflict resolution. We leverage this idea and take the team settings in TC-MAPF into consideration.

A. Preliminaries

1) Conflict-Based Search (CBS): CBS [2] is a two-level algorithm for (single-objective) MAPF, including a high-level search over a constraint tree (CT), and a low-level search that computes individual agent paths under given

constraints. In CBS, a vertex conflict is represented by a tuple (i, j, v, t) indicating that two agents i and j occupy the same vertex v at the same time t. To resolve it, the high-level search generates two child nodes in the CT, each introducing a constraint: (i, v, t) or (j, v, t), which forbids the corresponding agent from occupying vertex v at time t. Similarly, an edge conflict $(i, j, u, v, t, t+1)$ indicates agents i and j traverse the same edge (u, v) in opposite directions between times t and $t+1$. To resolve it, each of the two generated CT nodes adds an edge constraint: (i, u, v, t) or (j, v, u, t), which forbids the corresponding agent from traversing the edge at time t. Each of the generated CT node then invokes the low-level of CBS, which employs a single-agent planner (like A*) to find optimal paths for agents under a given set of constraints.

The high-level of CBS performs a best-first search on a CT, where each CT node P contains a set of constraints Ω, a *joint path* π, which consists of paths satisfying Ω of all agents $i \in I$, and the total cost $g(\pi)$. If π contains a conflict (i, j, v, t), the node P is split into two children, each inheriting Ω and adding a new constraint (e.g., (i, v, t) for one child, (j, v, t) for the other). The low-level planner is invoked for the newly constrained agent to update its path, and then, a new CT node is created. All leaf nodes of the CT forms the OPEN list, which is a priority queue that ranks nodes based on their $g(\pi)$ value from the minimum to the maximum. The high-level iteratively expands the CT node with the minimum cost $g(\pi)$ from OPEN until a solution is found, which is guaranteed to be an optimal solution for any solvable MAPF instance.

2) Teamwise Cooperative CBS with Transformation (TC-CBS-T): TC-CBS extends CBS to the TC-MAPF setting [3], which is shown in Alg. 1. The key differences in TC-CBS from CBS can be summarized as follows. First, TC-CBS stores team objective vectors $\vec{g}$, as opposed to the scalar cost g in CBS, in CT nodes, and uses lexicographic order based on $\vec{g}$ of CT nodes in the OPEN list. Second, TC-CBS maintains a set of solution cost vectors C, each of the cost vector there corresponds to a Pareto-optimal solution that were found so far during planning. TC-CBS filters CT nodes that are already dominated by solution cost vectors in C. It has been shown that TC-CBS is incomplete [4], in a sense that even if a given problem instance is solvable, TC-CBS may still never terminate in finite time.

TC-CBS-T [4] addresses this incompleteness by applying a transformation to the objective vectors during the search. For each team T_j, the transformed objective is:

$$g_f^{T_j}(\pi) := g^{T_j}(\pi) + \delta \Sigma_{i \notin T_j} g^i(\pi^i) \quad (1)$$

where δ is a small positive constant. Such a transformation in the objective vector allows TC-CBS-T to terminate in finite time for all solvable instances, however, at the cost of potentially missing some Pareto-optimal solutions. The conditions on δ under which TC-CBS-T finds all Pareto-optimal solutions is detailed in [4].

As shown by the black text in Alg. 1, TC-CBS-T prioritizes CT nodes based on their transformed objective vectors

Algorithm 1 Pseudocode for TC-CBS-T and TC-CBS-TF

1: Compute $P_o = (\pi_o, (\vec{g}_f(\pi_o)), \emptyset)$ and add P_o to OPEN
2: Add P_o to FOCAL
3: $C \leftarrow \emptyset$
4: **while** OPEN $\neq \emptyset$ **do**
5: $P_k = (\pi_k, \vec{g}_k, \vec{g}_{lb,k}, \Omega_k) \leftarrow \arg\min_{n \in \text{FOCAL}} h_c(n)$
6: Remove P_k from OPEN and FOCAL
7: // $P_k = (\pi_k, \vec{g}_k, \Omega_k) \leftarrow$ OPEN.pop()
8: **if** ϵ-Filter (P_k) **then continue**
9: // **if** Filter (P_k) **then continue**
10: $cft \leftarrow$ DetectConflict(π_k)
11: **if** $cft = \emptyset$ **then**
12: $C \leftarrow$ UpdateSolutionSet(C, P_k)
13: **continue**
14: $\Omega \leftarrow$ GenConstr(cft)
15: **for all** $\omega^i \in \Omega$ **do**
16: $\Omega_l = \Omega_k \cup \{\omega^i\}$
17: $K \leftarrow$ GetFlexFactor(i, P_k, w)
18: $\pi_*^i \leftarrow$ FocalSearch(i, Ω_l)
19: // $\pi_*^i \leftarrow$ LowLevelOptimalSearch(i, Ω_l)
20: **if** $\pi_*^i = \emptyset$ **then continue**
21: $\pi_l \leftarrow \pi_k$, replace π_k^i in π_l with π_*^i
22: $P_l \leftarrow (\pi_l, \vec{g}_l, \vec{g}_{lb,l}, \Omega_l)$
23: // $P_l \leftarrow (\pi_l, \vec{g}_l, \Omega_l)$
24: $P_l \leftarrow$ Transform(P_l)
25: **if** ϵ-Filter (P_l) **then continue**
26: // **if** Filter (P_l) **then continue**
27: Add P_l to OPEN
28: Update FOCAL
29: **return** Untransform(C)

using lexicographic order (Line 7 in Alg. 1). When a new CT node P_l is generated, the transformed objective vector is calculated for P_l using Eq. (1) before P_l is added to OPEN (Line 27 in Alg. 1). When a node P_k representing a solution is popped, its transformed vector is added to the solution set C for filtering. Solution filtering (Lines 9, 26 in Alg. 1) compares the transformed cost vector $\vec{g}_f(P)$ of the current node P against any solution vector in C that was already found during planning. If any vector in C dominates or is equal to $\vec{g}_f(P)$, then P is discarded. When TC-CBS-T terminates, all solution cost vectors are untransformed (Line 29) and then returned.

Intuitively, TC-CBS-T plans in a transformed objective space, and untransforms the solution cost vectors found right before termination. By finding the Pareto-optimal front in a transformed objective space, TC-CBS-T guarantees completeness and ensures that each solution it returns is Pareto-optimal to the original problem. However, TC-CBS-T may miss some Pareto-optimal solution to the original problem if the parameter δ in Eq. (1) is not chosen properly [4].

3) Enhanced CBS (ECBS): ECBS [11] is a bounded sub-optimal variant of CBS for MAPF which returns a solution with cost g that is at most w times the true optimal cost g^*, where $w \geq 1$ is a given sub-optimality factor. The low-level in ECBS employs focal search, a bounded sub-optimal variant of A*. This low-level focal search also returns $f_{\min}(i)$, the minimum f-value of all nodes in the (low-level) open list when the search terminates. This $f_{\min}(i)$ is a lower bound on the optimal path cost for agent i.

At the high-level, each CT node P also stores the lower

bounds $f_{min}(i), i \in I$ returned by the low-level, and computes $g_{lb}(P) = \sum_{i \in I} f^i_{\min}$, a lower bound on the cost of any solution that can be found from node P. The high-level OPEN prioritizes CT nodes based on their g_{lb} values. The high-level also maintains a global lower bound $g_{lb} = \min\{g_{lb}(P) \mid P \in \text{OPEN}\}$ of all nodes in the high-level OPEN list, which is a lower bound on the optimal solution cost g^*. In addition to OPEN, ECBS maintains another priority queue FOCAL at its high-level to contain nodes $\{P \in \text{OPEN} \mid g_{lb}(P) \leq w \cdot g_{lb}\}$. The node for expansion is selected from FOCAL based on h_c, the number of vertex and edge conflicts among all agents' paths in that CT node. Since $g_{lb} \leq g^*$, any node P in FOCAL satisfies $g(P) \leq w g_{lb} \leq w g^*$, and ECBS is thus guaranteed to return a solution cost no more than wg^*.

B. High-Level of TC-CBS-TF

TC-CBS-TF introduces three new techniques into TC-CBS-T: (i) the focal search from ECBS, (ii) ϵ-dominance (defined later) for solution filtering, and (iii) a new method to determine a flexible sub-optimality factor for the focal search, Intuitively, with (i), TC-CBS-TF tends to select CT nodes with fewer conflicts for expansion so as to quickly identify bounded sub-optimal solutions, which can then be used to filter nodes. The use of (ii) allows early pruning of more nodes than the regular dominance, while maintaining bounded sub-optimality of the final solution set returned. Finally, the flexible factor method allows focal search to intelligently distribute the bounds to the agents based on the team information and speeds up the search.

The algorithm is detailed in Alg. 1 with the differences marked in brown, supported by the helper function in Alg. 2. TC-CBS-TF employs two key parameters: the factor w for focal search and the ϵ-dominance. We discuss their relationship to α in the next section. For a CT node P, let $\vec{g}_{lb}(P)$ denote the lower bound cost vector computed based on the f^i_{min} of each agent $i \in I$ returned by the low-level search. Nodes in OPEN are prioritized based on their transformed lower bound cost vector $\vec{g}_{f,lb}$, and a node $P \in$ OPEN is eligible to enter FOCAL if there exists at least a node $Q \in$ OPEN such that $\vec{g}_{f,lb}(P) \leq (1+w) \cdot \vec{g}_{f,lb}(Q)$. In FOCAL, nodes are prioritized according to $h_c(P_k)$, the number of conflicts in the path stored in P_k.

TC-CBS-TF also introduces *ϵ-dominance* (Lines 8, 25 in Alg. 1) for solution filtering. For two vectors $\vec{g}_1$ and $\vec{g}_2$, $\vec{g}_1$ ϵ-dominates $\vec{g}_2$ if $g_{1,m} \leq (1+\epsilon) \cdot g_{2,m}$, $\epsilon \geq 0$, for every component $m = 1, 2 \cdots, M$ in the vectors. The ϵ-Filter prunes nodes whose cost vectors are ϵ-dominated by any existing solution cost vector stored in C.

C. Low-Level of TC-CBS-TF

TC-CBS-TF uses focal search for the low-level planning (Line 18 in Alg. 1) in a similar way as ECBS.[1] While the factor w in focal search in ECBS is fixed, TC-CBS-TF seeks to set w adaptively for each low-level call using the GetFlexFactor (Alg. 2), which is invoked before calling FocalSearch. *For the rest of this section, we present the method by assuming all teams' objective functions are min-sum, and leave min-max as our future work.*

Algorithm 2 GetFlexFactor (i,P,w)

```
w^i ← ∞
for all teams T_k ∈ T^i do
    g^others ← Σ_{j∈T_k, j≠i} g(π^j)
    g^others_lb ← Σ_{j∈T_k, j≠i} f_min(j)
    g^i_lb ← f_min(i)
    if g^i_lb = 0 then g^i_lb ← 1e-6          ▷ Avoid division by zero
    w^i_k ← Eq. (2)
    w^i ← min(w^i, w^i_k)
return w^i
```

Specifically, GetFlexFactor takes as input an agent i for which a factor w^i is to be determined, a CT node including agents' paths and lower bounds $f_{min}(i), i \in I$, and the sub-optimality factor w for focal search. Let $\mathcal{T}^i \subseteq \mathcal{T}$ denote the subset of teams that include agent i. For each team $T_k \in \mathcal{T}^i$ that includes agent i, a new sub-optimality factor w^i_k is computed based on the team's current cumulative path cost $g^{others} = \sum_{j \in T_k, j \neq i} g(\pi^j)$ (all other agents in team T_k except i) and the lower bounds of other agents $g^{others}_{lb} = \sum_{j \in T_k, j \neq i} f_{\min}(j)$, as well as the lower bound $g^i_{lb} = f_{\min}(i)$ of agent i itself. The team-based factor w^i_k is computed as

$$w^i_k = \frac{w \cdot (g^i_{lb} + g^{others}_{lb}) - g^{others}}{g^i_{lb}}. \tag{2}$$

Intuitively, w^i_k shows the amount of sub-optimality that agent i can leverage given the current paths of other agents in team T_k, without exceeding the overall sub-optimality bound w.

After calculating w^i_k for each relevant team $T_k \in \mathcal{T}^i$, the final factor w^i for agent i is the minimum across all teams $w^i = \min_{T_k \in \mathcal{T}} w^i_k$, which is used in the subsequent focal search for agent i. To summarize, this flexible factor method ensures that each team's cost remains bounded by w, while allowing an agent i with the largest possible sub-optimality bound w^i given other agents' paths in the related teams.

D. Properties of TC-CBS-TF

For a TC-MAPF problem instance with Pareto-optimal set Π_* and Pareto-optimal front C_*, let C^f_* denote the *transformed Pareto-optimal front*, which transforms each component of each $\vec{g} \in C_*$ using Eq. (1). Let $\Pi'_* \subseteq \Pi_*$ (and $C'_* \subseteq C^f_*$) denote the subset of Pareto-optimal set (and Pareto-optimal front) that can be found by TC-CBS-T (respectively) for a specific δ.

Theorem 1. *TC-CBS-TF finds an α-approximation of Π'_* and C'_* (as opposed to Π_* and C_*). When δ is properly chosen [4] such that $C'_* = C_*$, then as a result, TC-CBS-TF finds an α-approximation of Π_*. The sub-optimality factor α for the TC-MAPF problem is related to ϵ-dominance and focal search by $\alpha = w(1+\epsilon)$, where w and ϵ are the factors for focal search and ϵ-dominance respectively.*

[1] Note that, the low-level of TC-CBS-TF is also single-agent single-objective planning, which is same as the low-level of ECBS, and is different from the low-level of MOMAPF [20], [21], which requires solving a multi-objective single-agent problem [22], [23].

Team	1	2	3	4	5	6
Team 1 (Makespan)	18	19	20	21	22	24
Team 2 (Sum)	220	213	206	198	191	182

TABLE I: Solutions in Scenario I (Motorcade).

The detailed proof relies on the analysis of TC-CBS-T [4] and CBS [2]. Due to space limit, we only present the main ideas. First, to understand the bound $\alpha = w(1+\epsilon)$, consider two CT nodes P_1, P_2 with lower bounds $\vec{g}_{lb,1}, \vec{g}_{lb,2}$, cost vector $\vec{g}_1, \vec{g}_2$ respectively. Consider that P_1 is a solution node found by TC-CBS-TF with $\vec{g}_1$ added to the solution set C, while P_2 will lead to a Pareto-optimal solution with cost $\vec{g}_* \in C'_*$ but is filtered by P_1 using ϵ-dominance. Due to focal search, we know $\vec{g}_2 \leq w\vec{g}_{lb,2}$ and $\vec{g}_{lb,2} \leq \vec{g}_*$. Since $\vec{g}_* \in C'_*$ is filtered by $\vec{g}_1$, we know $\vec{g}_1 \leq (1+\epsilon)\vec{g}_2 \leq (1+\epsilon)(w\vec{g}_{lb,2}) \leq (1+\epsilon)w\vec{g}_*$. As a result, we have the following lemma.

Lemma 1. *If any $\vec{g}_* \in C'_*$ is filtered, TC-CBS-TF must have found a solution with cost $\vec{g}$ such that $\vec{g} \leq (1+\epsilon)w\vec{g}_* = \alpha\vec{g}_*$ with $\alpha = w(1+\epsilon)$.*

Second, during the search, TC-CBS-TF terminates only when OPEN depletes, which means a CT node will either be filtered by ϵ-dominance or lead to a solution eventually. Due to Lemma 1, for any filtered node P, TC-CBS-TF must have already found a solution that "approximates" P with bound α. Therefore, at termination, the solution set C found by TC-CBS-TF is an α-approximation of C'_*.

IV. Experimental Results

We first evaluate TC-CBS-TF in two scenarios based on the CARLA simulator [24]. The **first** scenario (Fig. 2) simulates an intersection in which a motorcade traverses the intersection along with other vehicles. Team 1 represents the motorcade, consisting of agents 1-6. Team 2 includes agents 7-14, representing the other vehicles. The **second** scenario (Fig. 3) considers 32 agents for high-density coordination. The agents are evenly divided into four teams, one from each direction. In both scenarios, TC-CBS-TF is applied with $w = 1.5$ and $\epsilon = 0.1$. Then, we test in grid maps that are commonly used for MAPF in various settings. The runtime limit is 300 seconds per instance.

A. Coordination in Traffic Intersections

Compared to the exact approach TC-CBS-T [4] which was evaluated with at most 8 agents in traffic intersection maps, our TC-CBS-TF can readily handle up to 32 agents in similar maps, which shows better scalability. Besides, our TC-CBS-TF finds relatively fewer solutions in comparison with TC-CBS-T as reported in [4], which is expected since TC-CBS-TF only approximate the Pareto-optimal front.

Solution Id	1	2	3	4	5
Team 1	209	207	212	208	210
Team 2	206	210	202	208	204
Team 3	208	204	210	207	206
Team 4	207	209	206	209	211

TABLE II: Solutions in Scenario 3

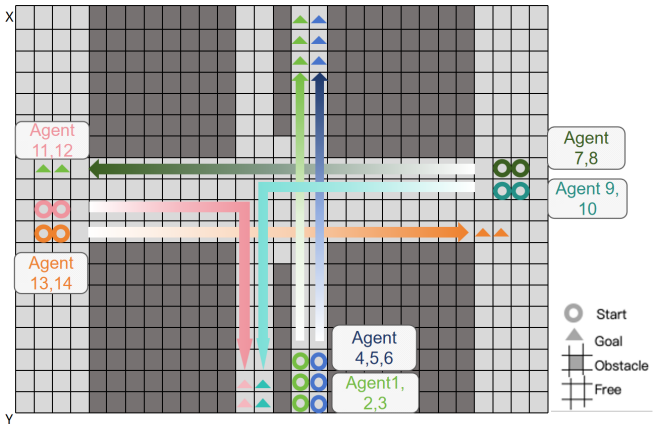

Fig. 2: Scenario I with a motorcade (Team 1, agents 1–6) and other vehicles (Team 2, agents 7–14).

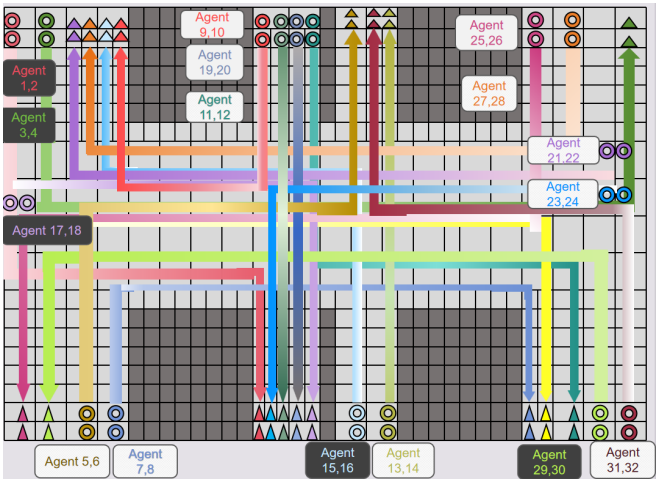

Fig. 3: Scenario 2 with 32 agents evenly divided in 4 teams.

B. Flexible Factors

We then test the flexible factor method in TC-CBS-TF, and compare it against TC-CBS-T and TC-CBS-TF without the flexible factors ("w/o flex" in short) in the traffic intersection maps. Fig. 4 shows the success rates for varying numbers of agents under different factors ($w = 1.1, 1.2, 1.3, 1.4$), with ϵ fixed at 0.1. The TC-CBS-TF outperforms the other two methods, especially for small $w = 1.1, 1.2$. For large w, TC-CBS-TF has similar performance with and w/o flex. The main reason is that, for a large w, FOCAL often has a large size and the algorithm already has many CT nodes to choose from FOCAL. Then, adjusting the sub-optimality based on the teams has little benefit.

C. Performance on Different Maps

We then test TC-CBS-TF in the maps "empty-16-16" and "random-32-32-20" from a dataset for MAPF [1].

1) Success Rate vs. Number of Agents: We first evaluate success rates with increasing numbers of agents under fixed factors $w = 1.1$ and $\epsilon = 0.1$. As shown in Fig. 5, TC-CBS-TF consistently outperforms both TC-CBS-T and TC-CBS-TF (w/o flex) on both maps. Notably, on the empty-16-16 map, TC-CBS-TF maintains high success rates even with 32

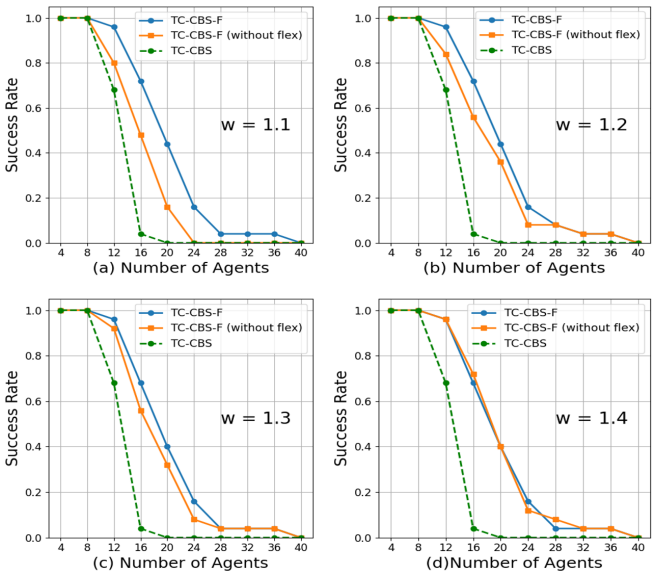

Fig. 4: Success rates for varying number of agents with different w from 1.1 to 1.4 in the traffic intersection map. The flexible factor mechanism obviously improves success rates when w is small.

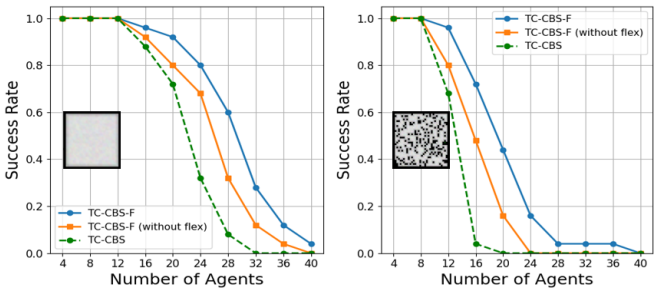

Fig. 5: Success rates comparison in different maps with varying agent numbers ($w = 1.1$).

agents, whereas others degrade rapidly.

2) Runtime vs. Sub-optimality Factors: We evaluate runtime under varying sub-optimality factors with 12 agents. As shown in Fig. 6, the runtime decreases as w increases for both versions of TC-CBS-TF. The runtime gap between TC-CBS-TF and its variant w/o flex is obvious when w is small, indicating that the flexible factor is more effective when the sub-optimality bounds are narrow.

D. Different Team Setups

Finally, we evaluate TC-CBS-TF under different team setups on a random 32x32 map. We consider two team setups: (a) Two teams where all agents belong to both teams where one team minimizes the sum of individual costs (min-sum) and the other team minimizes the maximum arrival time (min-max).[2] This setup is to solve MAPF while simultaneously optimizing min-sum and min-max objectives. (b) Each agent forms a team and optimizes only its own arrival time.

[2]Our flexible factor method is only tested for the min-sum objective, and is not applied to the min-max objective in this test.

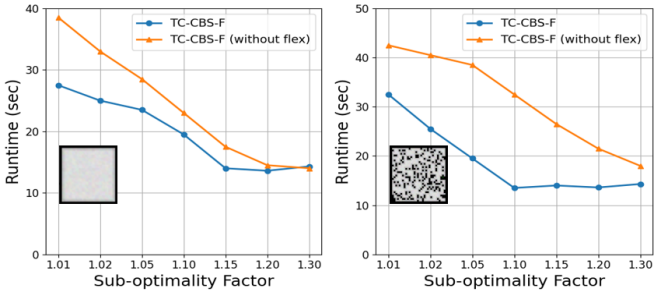

Fig. 6: Average runtime of all solved instances in different maps with varying sub-optimality factors.

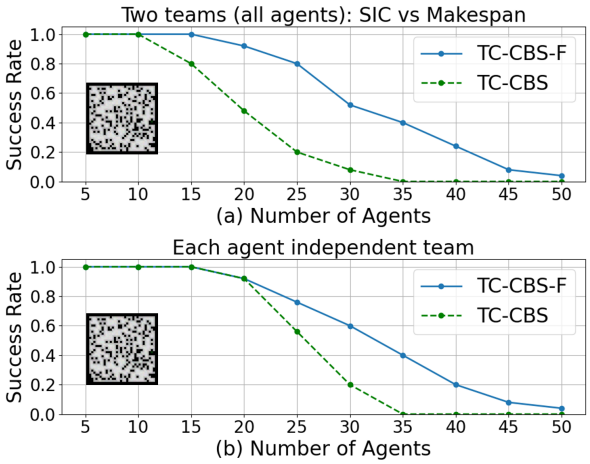

Fig. 7: Success rates of TC-CBS-TF and TC-CBS-T on a random-32-32 map under different team setups. (a) Two teams with min-sum (SIC) and min-max (Makespan) objectives. (b) Each agent as a team.

Here, we test agents N ranging from 5 to 50. The factors are $w = 1.5$ and $\epsilon = 0.1$. As shown in Fig. 7, TC-CBS-TF achieves obviously better scalability than the existing TC-CBS-T: The success rates of TC-CBS-T drop below 0.2 around 25-30 agents while the success rates of our TC-CBS-TF drop below 0.2 around 40-45 agents.

V. Conclusion and Future Work

This paper develops bounded sub-optimal algorithms for TC-MAPF, which scale better than the existing exact approach, at the cost of finding a set of bounded sub-optimal Pareto-optimal solutions that approximate the true Pareto-optimal set. For future work, one can investigates the flexible factor methods for min-max objectives or consider TC-MAPF for agents with non-unit-time actions, which may better model the agents in traffic intersections [25]–[28].

References

[1] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[2] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," in *Artificial Intelligence*, vol. 219. Elsevier, 2015, pp. 40–66.

[3] Z. Ren, C. Zhang, S. Rathinam, and H. Choset, "Search algorithms for multi-agent teamwise cooperative path finding," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1407–1413.

[4] Z. Ren, Y. Cai, and H. Wang, "Multi-agent teamwise cooperative path finding and traffic intersection coordination," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 7990–7995.

[5] Z. Ren, S. Rathinam, and H. Choset, "A conflict-based search framework for multiobjective multiagent path finding," *IEEE Transactions on Automation Science and Engineering*, pp. 1–13, 2022.

[6] 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)*, 2020, pp. 1–8.

[7] Z. Bnaya, R. Stern, A. Felner, R. Zivan, and S. Okamoto, "Multi-agent path finding for self interested agents," in *International Symposium on Combinatorial Search*, vol. 4, no. 1, 2013.

[8] M. Ivanová and P. Surynek, "Adversarial multi-agent path finding is intractable," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 481–486.

[9] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[10] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[11] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[12] S.-H. Chan, J. Li, G. Gange, D. Harabor, P. J. Stuckey, and S. Koenig, "Flex distribution for bounded-suboptimal multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 9313–9322.

[13] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[14] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Modifying optimal sat-based approach to multi-agent path-finding problem to suboptimal variants," 07 2017.

[15] Z. Ren, S. Rathinam, and H. Choset, "Subdimensional expansion for multi-objective multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7153–7160, 2021.

[16] Z. Ren, J. Li, H. Zhang, S. Koenig, S. Rathinam, and H. Choset, "Binary branching multi-objective conflict-based search for multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, Jul. 2023, pp. 361–369.

[17] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," *arXiv preprint arXiv:2010.01367*, 2020.

[18] J. Lim and P. Tsiotras, "Cbs-budget (cbsb): A complete and bounded suboptimal search for multi-agent path finding," *Artificial Intelligence*, vol. 346, p. 104349, 2025.

[19] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2005, vol. 491.

[20] Z. Ren, S. Rathinam, and H. Choset, "A conflict-based search framework for multiobjective multiagent path finding," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1262–1274, 2022.

[21] F. Wang, H. Zhang, S. Koenig, and J. Li, "Efficient approximate search for multi-objective multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, 2024, pp. 613–622.

[22] Z. Ren, R. Zhan, S. Rathinam, M. Likhachev, and H. Choset, "Enhanced multi-objective A* using balanced binary search trees," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 15, no. 1, 2022, pp. 162–170.

[23] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, 2022.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[25] A. Andreychuk, K. Yakovlev, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 39–45.

[26] S. Zhou, S. Zhao, and Z. Ren, "Loosely synchronized rule-based planning for multi-agent path finding with asynchronous actions," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 39, Apr. 2025, pp. 14 763–14 770.

[27] Z. Ren, S. Rathinam, and H. Choset, "Loosely synchronized search for multi-agent path finding with asynchronous actions," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2021.

[28] S. Zhou, S. Zhao, and Z. Ren, "Lsrp*: Scalable and anytime planning for multi-agent path finding with asynchronous actions (extended abstract)," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 18, no. 1, Jul. 2025, pp. 275–276.