2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Multi-Agent Teamwise Cooperative Path Finding and Traffic Intersection Coordination

Zhongqiang Ren, Yilin Cai, Hesheng Wang

***Abstract*— When coordinating the motion of connected autonomous vehicles at a signal-free intersection, the vehicles from each direction naturally forms a team and each team seeks to minimize their own traversal time through the intersection, without concerning the traversal times of other teams. Since the intersection is shared by all teams and agent-agent collision must be avoided, the coordination has to trade the traversal time of one team for the other. This paper thus investigates a problem called Multi-Agent Teamwise Cooperative Path Finding (TCPF), which seeks a set of collision-free paths for the agents from their respective start to goal locations, and agents are grouped into multiple teams with each team having its own objective function to optimize. In general, there are more than one teams and hence multiple objectives. TCPF thus seeks the Pareto-optimal front that represents possible trade-offs among the teams. We develop a centralized planner for TCPF by leveraging the Multi-Agent Path Finding techniques to resolve agent-agent collision, and Multi-Objective Optimization to find Pareto-optimal solutions. We analyze the completeness and optimality of the planner, which is then tested in various settings with up to 40 agents to verify the runtime efficiency and showcase the usage in intersection coordination.**

I. Introduction

Multi-Agent Path Finding (MAPF) seeks a set of collision-free paths for multiple agents from their respective start to goal locations, which was widely studied over the last decade. This problem often requires optimizing a single objective (function), such as min-sum, i.e., minimizing the sum of individual path costs [1] or min-max, i.e., minimizing the maximum individual path cost of the agents [2]. The objective is typically defined over all agents, where all agents plan their paths to optimize this common objective, and hence the name cooperative path finding [3].

This paper considers a generalization of MAPF, called Multi-Agent Teamwise Cooperative Path Finding (TCPF), where agents are grouped into multiple teams, and each team optimizes its own objective. Specifically, each agent has its own start and goal and belongs to at least one team. Every team has its own objective such as min-sum or min-max. In TCPF, the team setting (which agent belongs to which team) and the team objectives are given as part of a problem instance. As there are more than one teams in general, TCPF optimizes an objective vector, where each element of the vector corresponds to the objective of a team. For vector optimization, in general, there is no single solution that simultaneously optimizes all objectives. The problem thus seeks a set of Pareto-optimal solutions, whose objective vectors form the Pareto-optimal front. A solution is Pareto-optimal if one objective cannot be improved without deteriorating another objective. TCPF generalizes MAPF and is thus NP-hard to find Pareto-optimal solutions.

Zhongqiang Ren and Hesheng Wang are with Shanghai Jiao Tong University in China. (email: zhongqiang.ren@sjtu.edu.cn, wanghesheng@sjtu.edu.cn). Yilin Cai is with Georgia Institute of Technology in USA. (email: yilincai@gatech.edu).

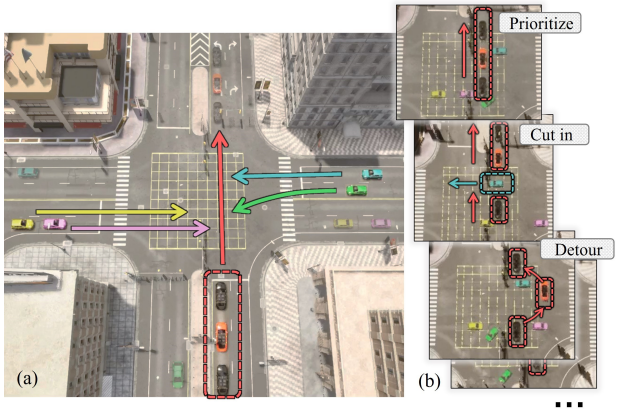

Fig. 1: An example of TCPF in intersection coordination of connected autonomous vehicles. (a) A motorcade (marked in red box, team 1) traverse the intersection shared with other vehicles (team 2). Both teams have the min-sum objective. (b) Pareto-optimal solutions (collision-free paths), where the motorcade can have priority, get cut in by others, or make a detour.

Our prior work [4] formulated the TCPF problem, developed two planners to solve it, and showed that the Teamwise Cooperative Conflict-Based Search (TC-CBS) often enjoys better scalability, but suffers from incompleteness in theory, i.e., TC-CBS may not terminate in finite time even for a solvable TCPF instance. This paper handles this incompleteness by proposing a new planner TC-CBS-t ("t" for transformation) by imposing a transformation on the objective vectors during planning. We prove TC-CBS-t is complete (i.e., terminates in finite time for any solvable instance), and all solutions found by TC-CBS-t are guaranteed to be Pareto-optimal. Furthermore, we analyze the condition under which TC-CBS-t can find the entire Pareto-optimal front.

Additionally, this paper connects TCPF with traffic management in scenarios where all vehicles are fully autonomous, conencted and can be coordinated in a centralized fashion. See Fig. 1 for an example, where a motorcade goes through an intersection that is shared with vehicles from other directions. The motorcade and all other vehicles naturally form two teams, and each team seeks to minimize their own traversal time through the intersection. The motion coordination has to trade the traversal time of one team for the other, and the set of Pareto-optimal solutions unveils trade-offs between the teams, which can potentially support the intelligent decision making in traffic management.

To evaluate the planner, we set up a intersection simulation. Our approach finds Pareto-optimal solutions and shows how teams interact with each other. We also consider a special case of TCPF, a bi-objective MAPF problem that seeks to simultaneously minimize both the min-sum and the min-max objective of all agents. Here, min-sum and min-max are two widely used objective functions in MAPF, each with its own set of specialized algorithms. Our approach can solve it to optimize both objectives simultaneously.

A. Related Work

MAPF often minimizes a single-objective, such as min-sum (also called min-flowtime) or min-max (also called min-makespan). When there is only one team including all agents, TCPF becomes MAPF. To solve MAPF to optimality, various methods were developed, which focus on either min-sum [1], [5] or min-max [2], [6] problems.

MO-MAPF [7]–[10] associates 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. MO-MAPF requires minimizing the sum of accumulated cost vectors over all agents along their paths. The TCPF differs from MO-MAPF, since the action costs are scalars, and there are multiple teams, each with its own objective.

Other related variants of MAPF include self-interested MAPF [11], where each agent aims to find its individually min-cost path and the goal is to design a taxation scheme so that all agents become cooperative after adding an additional tax-cost to the agents' paths. Adversarial MAPF [12] divides agents into teams, and seeks a policy for a selected team so that the agents in the selected team can navigate to their goals subject to any actions other teams can take.

Traffic Management of autonomous vehicles at intersections has been researched a lot [13]–[15]. This paper investigates the connection between traffic management and MAPF with the focus on the notion of teamwise cooperativeness.

II. Problem Statement

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 $E \subseteq V \times V$ denotes the set of all actions that can move an agent between a pair of vertices in V. An edge between $u, v \in V$ is denoted as $(u, v) \in E$ and the cost of $e \in E$ is a finite positive real number $cost(e) \in \mathbb{R}^+$. Let $v_o^i, v_d^i \in V$ respectively denote the start and goal location of agent i.

We use a superscript $i \in I$ over a variable to show the agent that the variable belongs to (e.g. $v^i \in V$ is a vertex related to agent i). Let $\pi^i(v_1^i, v_\ell^i)$ be a path from v_1^i to v_ℓ^i via a sequence of vertices $(v_1^i, v_2^i, \dots, v_\ell^i)$ in G. Let $g(\pi^i(v_1^i, v_\ell^i))$ denote the cost value of the path, which is the sum of the cost of all the edges present in the path, i.e., $g(\pi^i(v_1^i, v_\ell^i)) = \Sigma_{j=1,2,\dots,\ell-1} cost(v_j^i, v_{j+1}^i)$. We denote $\pi^i(v_1^i, v_\ell^i)$ simply as π^i and $g^i = g(\pi^i)$ when there is no confusion.

All agents share a global clock and they start the paths at time $t = 0$. Each action of an agent, either wait or move, requires one unit of time. Any two agents 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 between times t and $t + 1$ for some t.

Let $\{T_j, j = 1, 2, \dots, M\}$ denote a set of M *teams*, where each team $T_j \subseteq I$ includes a subset of agents. Each agent belongs to at least one team and teams are not required to be mutually disjoint to each other. Let π^{T_j} denote a joint path, which is a set of individual paths $\{\pi^i, \forall i \in T_j\}$. Let g^{T_j} denote the *objective* function value of team T_j that is to be minimized, which is a non-decreasing function of the individual path cost of all the agents in the team T_j. In other words, if the path cost g^i of any agent $i \in T_j$ increases, then g^{T_j} either stays the same or increases. Two widely used objective functions are the sum (and the maximum) of individual path costs, i.e., $g^{T_j} := \sum_{i \in T_j} g(\pi^i)$, (and $g^{T_j} := \max_{i \in T_j} g(\pi^i)$). When a team T_j includes only one agent i, then $g^{T_j} = g^i = g(\pi^i)$, which is the path cost of agent i. Let π (no superscript) denote a joint path of all agents, which is also called a *solution*. Let $\vec{g}(\pi) := \{g(\pi^{T_j}), j = 1, 2, \dots, M\}$ denote an *objective vector* of length M, where each component corresponds to the objective of a team.

To compare two solutions, we compare the objective 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 [16].

*Definition 1 (***Dominance):** 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 $a(m) < b(m)$, $\exists m \in \{1, 2, \dots, M\}$.

Any two solutions are non-dominated with respect to each other if the corresponding objective vectors do not dominate each other. A solution π is non-dominated with respect to a set of solutions Π, if π is not dominated by any $\pi' \in \Pi$. Among all conflict-free (i.e., feasible) solutions, the set of all non-dominated solutions is called the *Pareto-optimal* set Π_*, and the corresponding set of objective vectors C^* is called the Pareto-optimal front.

*Problem 1 (***General Problem):** TCPF seeks all *cost-unique* Pareto-optimal solutions, i.e., any maximal subset of the Pareto-optimal set, where any two solutions in this subset do not have the same objective vector.

*Problem 2 (***Fully Cooperative Problem):** A TCPF is *fully cooperative* if each team $T_j, j = 1, 2 \dots, M$ contains all agents (i.e., $T_j = I$). Otherwise (i.e., there exists at least one team that does not include all agents), the TCPF is not fully cooperative.

III. Method

A. Review of CBS and TC-CBS

Conflict-Based Search (CBS) [1] is a two-level search algorithm that computes a conflict-free solution to a MAPF problem. On the high-level, every search node P is defined

as a tuple of (π, g, Ω), where: $\pi = (\pi^1, \pi^2, \ldots, \pi^N)$ is a joint path that connects starts and goals of agents respectively; g is the scalar cost value of π (i.e., $g = g(\pi) = \Sigma_{i \in I} g^i(\pi^i)$); Ω is a set of constraints, and each constraint is of form (i, v, t) (or i, e, t), which indicates agent i is forbidden to enter node v (or edge e) at time t. CBS constructs a search tree with the root node $P_{root} = (\pi_o, g(\pi_o), \emptyset)$, where the joint path π_o is constructed by running the low-level (single-agent) planner, such as A*, for every agent respectively with an empty set of constraints while ignoring any other agents. P_{root} is added to OPEN, a queue that prioritizes nodes based on their g-values.

In a search iteration, a node $P = (\pi, g, \Omega)$ with the minimum g-value is popped from OPEN for expansion. To expand P, every pair of individual paths in π is checked for vertex conflict (i, j, v, t) (and edge conflict (i, j, e, t)). If no conflict is detected, π is conflict-free and is returned as an optimal solution. Otherwise, the detected conflict (i, j, v, t) is *split* into two constraints (i, v, t) and (j, v, t) respectively and two new constraint sets $\Omega \bigcup \{i, v, t\}$ and $\Omega \bigcup \{j, v, t\}$ are generated. Edge conflicts are handled in a similar way and is thus omitted. Then, for the agent i in each split constraint (i, v, t) and the corresponding newly generated constraint set $\Omega' = \Omega \bigcup \{i, v, t\}$, the low-level planner is invoked to plan an individual minimum cost path π'^i of agent i subject to all constraints related to agent i in Ω'. The low-level planner typically runs A*-like search in a time-augmented graph with constraints marked as obstacles. A new joint path π' is then formed by first copying π and then updating agent i's individual path π^i with π'^i. Finally, for each of the two split constraints, a corresponding node is generated and added to OPEN for future expansion. CBS terminates when the first conflict-free joint path is found which is a min-cost solution.

Teamwise Cooperative CBS (TC-CBS) follows a similar workflow as CBS. We use Alg. 1 to show both TC-CBS and the new TC-CBS-t, where the blue color highlights the difference and will be explained later.

TC-CBS differs from CBS as follows. **First**, given a node P_k and its corresponding joint path π_k, TC-CBS computes an objective vector $\vec{g}(\pi_k)$ based on the teams, rather than a scalar cost value g as in CBS. This arises on Lines 1 and 14, when generating the root node and a new node respectively. Then, nodes are organized in lexicographic (abbreviated as lex.) order in OPEN based on their $\vec{g}$, and in each iteration, a lex. min node is popped from OPEN for processing (Line 4). **Second**, since there are multiple Pareto-optimal solutions in general, TC-CBS stores all Pareto-optimal solutions found during the search in a set C (Line 7). We denote C as a set of objective vectors. Each vector in C identifies a unique node and thus a unique solution. **Third**, to find all cost-unique Pareto-optimal solutions, TC-CBS terminates when OPEN depletes, while CBS terminates when the first conflict-free solution is found. Additionally, when a node P_k is popped from OPEN (Line 4) or newly generated (Line 15), P_k is tested for filtering, i.e., P_k is discarded if its objective vector is dominated by or equal to any existing vectors in C.

Algorithm 1 Pseudocode for TC-CBS and TC-CBS-t

1: Compute P_{root} and insert into OPEN.
2: $C \leftarrow \emptyset$
3: **while** OPEN not empty **do**
4: $P_k = (\pi_k, \vec{g}_k, \Omega_k) \leftarrow$ OPEN.pop()
5: **if** *Filter*(P_k) **then continue** ▷ End of iteration
6: **if** no conflict detected in π_k **then**
7: add *Untransform*$(\vec{g}_k)$ to C
8: **continue** ▷ End of iteration
9: $\Omega \leftarrow$ split detected conflict
10: **for all** $\omega^i \in \Omega$ **do**
11: $\Omega_l = \Omega_k \cup \{\omega^i\}$
12: $\pi^i_* \leftarrow$ *LowLevelSearch*(i, Ω_l)
13: $\pi_l \leftarrow \pi_k$, replace π^i_l (in π_l) with π^i_*
14: $\vec{g}_l \leftarrow$ *Transform*$(\vec{g}(\pi_l))$ ▷ Computed based on teams.
15: $P_l = (\pi_l, \vec{g}_l, \Omega_l)$
16: **if** not *Filter*(P_l) **then**
17: add P_l to OPEN
18: **return** C

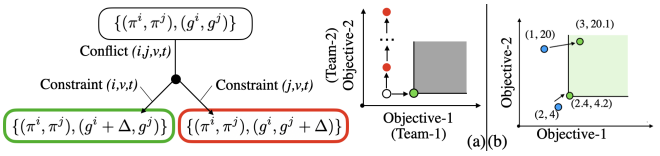

Fig. 2: (a) An illustrative case where TC-CBS is incomplete. The grey area show the set of objective vectors dominated by the green solution. (b) An illustrative case where TC-CBS-t fails to find all Pareto-optimal solutions. After the transformation, $(3, 20.1)$ is dominated by $(2.4, 4.2)$ and is not found by TC-CBS-t.

B. Properties and Incompleteness of TC-CBS

A problem instance is *feasible* if there exists a feasible solution. Given a feasible instance, TC-CBS is said to be *complete* if it terminates in finite time.[1] For fully cooperative TCPF, TC-CBS is guaranteed to be complete, and is guaranteed to find all cost-unique Pareto-optimal solutions. The analysis in MO-CBS [7] can be applied to TC-CBS for fully cooperative TCPF problems, since TC-CBS has the same high-level search as MO-CBS.[2] We summarize this property with the following theorem.

Theorem 1: TC-CBS is complete and can find all Pareto-optimal solutions for Fully Cooperative TCPF (Problem 2).

However, for TCPF that are not fully cooperative (which is often the case for intersection coordination), TC-CBS is incomplete: TC-CBS fails to terminate in finite time even if the problem instance is feasible. The condition for TC-CBS being complete is [4]: there is a finite number of joint paths whose objective vectors are non-dominated by the Pareto-optimal front. This condition may not hold for TCPF that is not fully cooperative. See the example in Fig. 2(a): there are two agents $I = \{i, j\}$ and two teams $T_1 = \{i\}, T_2 = \{j\}$; the objective vector is $(g^{T_1}, g^{T_2}) = (g^i, g^j)$. Consider a conflict $(i = 1, j = 2, v, t)$, which leads to constraints (i, v, t) and

[1] It is known that for unsolvable instances, CBS-based algorithms do not terminate in finite time, and this can be handled by running a feasibility checking in advance [1]. We thus only focus on solvable instances.

[2] The high-level search in TC-CBS is the same as MO-CBS [7], while the low-level search is different. In MO-CBS, each low-level search requires solving a multi-objective shortest path problem subject to constraints, while the low-level search in TC-CBS is single-objective.

(j, v, t) and two new nodes (red and green). For either of the two nodes, one agent's path cost may increase (as a constraint is added), while the other agent's path cost remains the same. Consider that the green node leads to the only Pareto-optimal solution, and the red node still contains conflicts and leads to further expansion. There can be infinitely many joint paths whose objective vectors are non-dominated by the Pareto-optimal front, and TC-CBS never terminates since OPEN never depletes. An example is agent i reaches its destination which blocks the only path for agent j to reach its destination. An infinite number of nodes will be generated.

C. TC-CBS-t Algorithm

TC-CBS-t addresses the aforementioned incompleteness issue of TC-CBS, at the cost of finding a subset (as opposed to the full set) of Pareto-optimal solutions. TC-CBS-t modifies Lines 1, 7, 14 when computing the objective vector of a node using a transformation (or untransformation). For a team T_j, define its transformed objective value g_f as

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

Here, ϵ is a small positive value and $\sum_{i \notin T_j} g(\pi^i)$ is the sum of individual path costs of all agents that are not part of the team T_j. In other words, $g_f(\pi^{T_j})$ differs from $g(\pi^{T_j})$ by adding the path cost of the agents outside T_j weighted by a small factor ϵ. By doing so, the objective of every team $T_j, j = 1, 2, \ldots, M$ involves all agents. The resulting TCPF problem, which is hereafter referred to as the "transformed problem", minimizes $\vec{g}_f(\pi) = \{g_f(\pi^{T_j}), j = 1, 2, \ldots, M\}$, and is a fully cooperative problem. With Theorem 1, TC-CBS is complete and can find the Pareto-optimal front (denoted as C_f^*) for this transformed problem.

The remaining question is whether the Pareto-optimal front C^* of the original TCPF problem can be recovered from C_f^*, the Pareto-optimal front of the transformed problem. To answer this question, we need to discuss the following three sub-questions: (i) how to untransform the vectors in C_f^*, (ii) is the untransformed vector Pareto-optimal, i.e., is part of C^*, and (iii) can we obtain all vectors in C^* with this untransformation.

To answer sub-question (i), with Equation 1, the vectors in C_f^* can be untransformed by calculting $g(\pi_j^T) = g_f(\pi_j^T) - \epsilon \sum_{i \in T_j} g(\pi^i)$ for each $j = 1, 2, \ldots, M$. In other words, for a vector $\vec{g}_f \in C_f^*$, a corresponding vector $\vec{g}$ (referred to as the untransformed vector) can be computed by extracting $\epsilon \sum_{i \in T_j} g(\pi^i)$ for each element $j = 1, 2, \ldots, M$ of $\vec{g}_f$. We analyze the properties of TC-CBS-t in the next sub-section to answer sub-question (ii) and (iii).

D. Properties of TC-CBS-t

The following theorem answers the sub-question (ii).

Theorem 2: Every untransformed vector $\vec{g}$ belongs to the Pareto-optimal front C^* of the original problem.

Proof: We prove this theorem by contradiction. Assume the transformed vector $\vec{g}$ is not Pareto-optimal. Then, there is another vector $\vec{g}' \in C^*$ that dominates $\vec{g}$, which means $\vec{g}'$ is component-wise no larger than $\vec{g}$ and there exists a component (say the k-th component) g'_k that is strictly less than g_k (i.e., $g'_k < g_k$). Let $\vec{g}'_f$ denote the transformed vector of $\vec{g}'$ with Equation 1. Then $\vec{g}'_f$ must dominate $\vec{g}_f$ due to the existence of $g'_k < g_k$. Therefore, $\vec{g}_f$ cannot be Pareto-optimal for the transformed problem and cannot be part of C_f^*, which leads to contradiction. ■

For sub-question (iii), the answer is negative: given C_f^*, one may fail to obtain all vectors in C^* using this untransformation. We explain it with a toy example (Fig. 2(b)). Consider the case with two teams, where each team has only one unique agent, and the Pareto-optimal front of this problem has two objective vectors $C^* = \{(2, 4), (1, 20)\}$. After the transformation with $\epsilon = 0.1$, the resulting transformed vectors are $\{(2.4, 4.2,), (3, 20.1)\}$, where $(3, 20.1)$ is dominated by $(2.4, 4.2)$ and cannot be found by TC-CBS-t. In other words, TC-CBS-t finds a Pareto-optimal front $C_f^* = \{(2.4, 4.2)\}$ with only one vector, and after the untransformation, TC-CBS-t only returns $(2, 4)$ and fails to find the whole Pareto-optimal front. Even with a smaller ϵ, one can still construct other examples where TC-CBS-t fails to find the whole Pareto-optimal front.

When the following conditions hold, TC-CBS-t is guaranteed to find the whole Pareto-optimal front of the original TCPF problem. The intuition is to bound the maximum increment of any component in the objective vector after the transformation, so that TC-CBS-t finds the whole Pareto-optimal front. Specifically, let C^* denote the Pareto-optimal front of a TCPF problem. Let $\vec{a}, \vec{b} \in C^*$ denote two different objective vectors in C^*. Let m_{C^*} denote the smallest difference between the corresponding component of any two vectors in C^*, i.e., $m_{C^*} := \min\{|a_j - b_j|, \vec{a}, \vec{b} \in C^*, j = 1, 2, \ldots, M\}$. Let M_{C^*} denote the maximum value of any component of any vector in C^*, i.e., $m_{C^*} := \max\{g_j | \vec{g} \in C^*, j = 1, 2, \ldots, M\}$.

Theorem 3: If the ϵ in TC-CBS-t satisfies $M_{C^*} N \epsilon < m_{C^*}$, then TC-CBS-t finds the whole Pareto-optimal front.

Proof: For any $\vec{g}^* \in C^*$ and its transformed vector $\vec{g}_f^*$, the maximum increment in any element of $\vec{g}_f^*$ is no greater than $M_{C^*} N \epsilon$, where N is the number of agents, and M_{C^*} is an upper bound of the path cost of any agent. Let $\vec{a}, \vec{b} \in C^*$ denote two different vectors, and $\vec{a}_f, \vec{b}_f$ denote the corresponding transformed vectors. Consider the j-the element of them, and the case where $a_j^* \geq b_j^*$. By definition, $a_j^* - b_j^* \geq m_{C^*}$, and $a_{f,j}^* - b_{f,j}^* \geq (a_j^* + 0) - (b_j^* + M_{C^*} N \epsilon) = (a_j^* - b_j^*) + M_{C^*} N \epsilon \geq m_{C^*} - M_{C^*} N \epsilon > 0$. After the transformation, any element of the two vectors remain in the same order as before the transformation. $\vec{a}_f, \vec{b}_f$ are thus non-dominated by each other, and TC-CBS-t can find a set C_f^* where each transformed vector corresponds to a Pareto-optimal vector in C^*. ■

E. Planning for Intersection Coordination

We limit our focus to only one intersection where the vehicles have known starts and goals and all vehicles are autonomous and connected. We simplify the dynamics of the vehicles by considering the following discrete representation. The state of a vehicle is (x, y, θ, t), where x, y are the

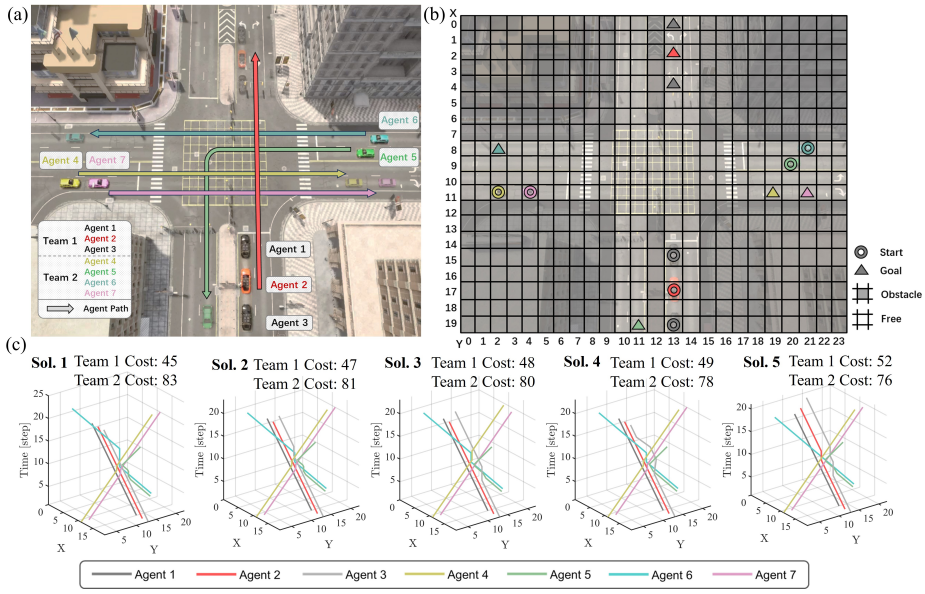

Fig. 3: Simulation of an intersection. (a) A motorcade (agent 1-3, team 1) goes through the intersection shared by other vehicles (agent 4-7, team 2). (b) A graph representation of the workspace showing the start and goal of each vehicle. (c) Visualization of the Pareto-optimal solutions (collision-free paths). The motorcade (team 1) has the objective value ranging between 45 and 52, while the public (team 2) has the objective value ranging from 76 and 83.

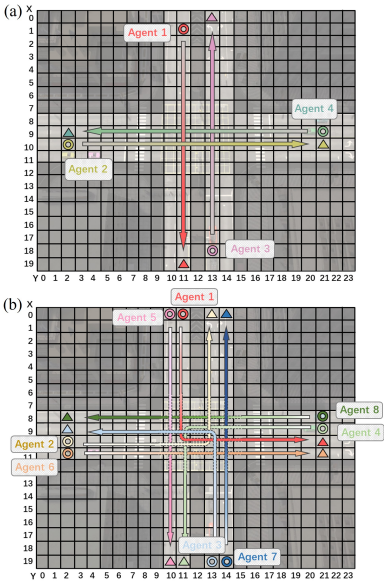

Fig. 4: (a) Four agents driving straight are divided into 2 or 4 teams. (b) Eight agents come from four directions, and for each direction, one agent drives straight and one turns left.

TABLE I: Solution costs of the four agents driving straight.

Team Id	1	2	3	4	5	6	7
Team 1 (Agent 1)	18	18	20	20	20	22	22
Team 2 (Agent 2)	22	22	20	20	21	20	21
Team 3 (Agent 3)	18	20	18	22	20	22	20
Team 4 (Agent 4)	22	20	22	21	21	20	20

coordinates of the geometric center of the vehicle footprint, θ is the orientation, and t is the time step. The state space of a vehicle is represented by a state lattice, where each of x, y, θ are divided into a finite number of values that represent the possible states around the intersection. The edge set in the lattice is a pre-computed set of motion primitives (i.e., small trajectories that connects two states), that are kinematically feasible for the vehicles. We can then use a state lattice planner [17] as the low-level planner in TC-CBS-t. To ensure collision avoidance between the vehicles, we abandon the edge conflicts in Sec. II and only consider vertex conflicts. The vehicle footprints are all approximated by rectangles, and the vertex conflicts are computed given the motion primitives of the vehicles to check if a (x, y, θ) is occupied by the footprints of two vehicles at the same time.

IV. Experimental Results

A. Intersection Coordination

We set up a intersection simulator using CARLA with a map based on Town11 [18]. As shown in Figure 1 (a), the map includes a four-lane, bidirectional intersection. The intersection was described as a 20-by-24 grid map as shown in Fig. 3b. The lane width are one unit size of the grid map and each lane can only accommodate one vehicle.

1) Scenario I with a Motorcade: We first consider a motorcade and public vehicles. Team 1 includes agents 1,2,3 and represents a motorcade (Fig. 3), while the remaining agents forms Team 2. Fig. 3c shows five different solutions, showing the trade-off between the motorcade and the public. For example, if the motorcade is of absolute priority (e.g. ambulances), then solution 1 will be picked for execution.

2) Scenario II with Four Agents: We then examine a four-agent scenario where all agents drive straight forwards in their respective lane to cross the intersection. As Fig. 4(a) shows, agent 1 will collide into agent 2, and agent 3 will collide into agent 4 if there is not coordination. Each agent forms a unique team. Based on the Pareto-optimal solutions, we can argue that solution 5 in Table I is the most "fair" solution since the maximum delay of any agent is at most 2 time units away from their minimum possible arrival time.

3) Scenario III with Eight Agents: We then scale up the number of agents to eight, with four driving straight and four taking a left turn. As shown in Table II, there are more Pareto-optimal solutions since the interaction between agents (or teams) become complicated.

B. MAPF with Different Team Settings

We use a random grid map (Fig. 5), and set a runtime limit of five minutes for each instance (i.e., a set of starts and goals of all agents). We test with N ranging from 5 to 40 with a step size of 5. We set the parameter ϵ to values of 0.05, 0.15, and 0.25 for TC-CBS-t. The implementation is in Python and executed on a laptop with a Core i9-13900K 3.00GHz CPU and 128 GB RAM.

1) MAPF with Both Min-Sum and Min-Max Objectives: The first team setting has two teams and each team includes all agents. One team has the min-sum objective while the other team has the min-max objective. This type of problem is a bi-objective MAPF (BO-MAPF) problem that simultaneously minimizes both min-sum and min-max objectives.

TABLE II: Pareto-optimal solution costs of the eight agents divided into 4 teams.

Solution Id	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Team 1 (Agent 1&5)	38	38	39	39	39	39	40	40	41	41	41	41	41	43	44	44
Team 2 (Agent 2&6)	40	42	39	40	41	42	39	40	39	39	40	40	41	39	39	39
Team 3 (Agent 3&7)	41	41	41	39	44	43	44	41	40	41	38	39	38	39	39	40
Team 4 (Agent 4&8)	41	40	40	40	39	39	39	39	40	39	42	39	40	42	41	39
Sum of Cost	160	161	159	158	163	163	162	160	160	160	161	159	160	163	163	162
Max Cost	41	42	41	40	44	43	44	41	41	41	42	41	41	43	44	44

TABLE III: Average/max solution numbers in Random map with (1) min-sum and min-max objectives, (2) each agent as a team.

	Agent Number		**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
Min-Sum and Min-Max Objectives	TC-CBS		1.04 / 2	1.04 / 2	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	NaN / NaN
	TC-CBS-t	$\epsilon = 0.05$	1.04 / 2	1.04 / 2	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	NaN / NaN
		$\epsilon = 0.15$	1.04 / 2	1.04 / 2	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	NaN / NaN
		$\epsilon = 0.25$	1.04 / 2	1.04 / 2	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	1.00 / 1	NaN / NaN
Each Agent as a Team	TC-CBS		1.12 / 2	1.68 / 6	3.24 / 14	3.67 / 14	3.00 / 8	4.50 / 8	NaN / NaN	NaN / NaN
	TC-CBS-t	$\epsilon = 0.05$	1.12 / 2	1.68 / 6	3.33 / 14	3.92 / 14	2.75 / 8	4.50 / 8	NaN / NaN	NaN / NaN
		$\epsilon = 0.15$	1.12 / 2	1.60 / 6	2.74 / 8	3.50 / 8	3.00 / 8	4.50 / 8	NaN / NaN	NaN / NaN
		$\epsilon = 0.25$	1.08 / 2	1.56 / 6	2.52 / 7	3.24 / 7	3.33 / 8	2.33 / 4	4.00 / 4	NaN / NaN

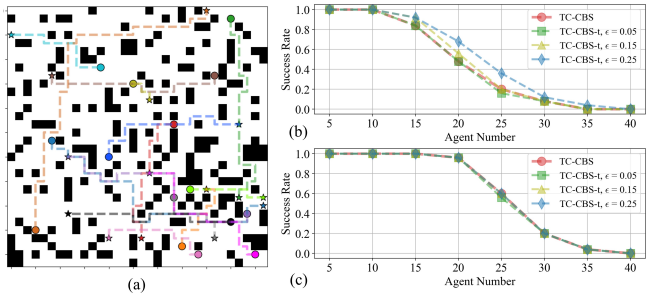

Fig. 5: Numerical results in a grid map with random obstacles. (a) An instance with 15 agents, where "o" shows the starts, and "*" shows the goals. (b) and (c) show the success rate of TC-CBS and TC-CBS-t with different ϵs. (b) MAPF with both Min-Sum and Min-Max objectives, and (c) TCPF with each agent as a team.

As shown in Fig. 5b, both the existing TC-CBS and our new TC-CBS-t with various ϵ achieve similar success rates. Table III shows the average and maximum number of Pareto-optimal solutions, and for most of the instances, there is only one solution that simultaneously optimizes both the min-sum and the min-max objectives.

2) MAPF with Each Agent as a Team: The second team setting has each agent as a team. Fig. 5(c) shows, TC-CBS-t with a large ϵ ($\epsilon = 0.25$) has higher success rates than the others. The reason is, larger ϵ makes TC-CBS-t prune more branches and find fewer Pareto-optimal solutions, which is supported by Table III. In addition, TC-CBS sometimes find fewer solutions than TC-CBS-t, since they have different number of succeeded instances. A large ϵ may speed up the computation while leading to a smaller subset of Pareto-optimal solutions. Finally, for the same number of agents N, this team setting often leads to more Pareto-optimal solutions than the former BO-MAPF setting.

REFERENCES

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

[2] 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.

[3] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[4] 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.

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

[6] 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.

[7] 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.

[8] ——, "Subdimensional expansion for multi-objective multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7153–7160, 2021.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE transactions on intelligent transportation systems*, vol. 17, no. 2, pp. 570–586, 2015.

[14] M. Khayatian, M. Mehrabian, E. Andert, R. Dedinsky, S. Choudhary, Y. Lou, and A. Shirvastava, "A survey on intersection management of connected autonomous vehicles," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 4, pp. 1–27, 2020.

[15] R. Katole and A. Sinha, "Autonomous intersection management for non-communicative autonomous vehicles," *arXiv preprint arXiv:2311.17681*, 2023.

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

[17] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[18] 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.