Team Coordination on Graphs: Problem, Analysis, and Algorithms

Manshi Limbu, Yanlin Zhou, Gregory Stein, Xuan Wang, Daigo Shishika, and Xuesu Xiao

***Abstract*— Team Coordination on Graphs with Risky Edges (TCGRE) is a recently emerged problem, in which a robot team collectively reduces graph traversal cost through support from one robot to another when the latter traverses a risky edge. Resembling the traditional Multi-Agent Path Finding (MAPF) problem, both classical and learning-based methods have been proposed to solve TCGRE, however, they lacked either computation efficiency or optimality assurance. In this paper, we reformulate TCGRE as a constrained optimization and perform rigorous mathematical analysis. Our theoretical analysis shows the NP-hardness of TCGRE by reduction from the Maximum 3D Matching problem and that efficient decomposition is a key to tackle this combinatorial optimization problem. Further more, we design three classes of algorithms to solve TCGRE, *i.e.*, Joint State Graph (JSG) based, coordination based, and receding-horizon sub-team based solutions. Each of these proposed algorithms enjoy different provable optimality and efficiency characteristics that are demonstrated in our extensive experiments.**

I. Introduction

Multi-Agent Path Finding (MAPF) is a trending problem in robotics [1]–[6], as it lies in the core of many robotic applications, e.g., drone swarm control [2], autonomous delivery [7], and public transportation scheduling [1]. Adding the possibility of team coordination [8] between robots to MAPF makes the problem more difficult. The need of coordination behaviors on large-scale multi-robot planning problems may exceed the computation capability of a centralized planner [9], giving rise to decentralized planning that distributes the computation into each robot [10]. However, despite the efficiency, dexterity, and responsiveness, the distribution itself may induce certain performance degradation and lead to suboptimal team coordination behaviors. Therefore, some centralized pre-planning is still crucial for multi-robot planning, especially for large-scale problems.

Team Coordination on Graphs with Risky Edges (TCGRE) [11] is such a centralized planning problem in an environment represented as a graph—multiple robots travel from their start to goal nodes with possible support from some nodes to reduce the cost of traversing certain risky edges, requiring team coordination behaviors to reduce the total cost of team graph traversal. By converting the environment graph to a Joint State Graph (JSG), optimal coordination can be solved using Dijkstra's search algorithm on the JSG [11]. However, the conversion to JSG does not scale well with large environment graphs and number of robots. To address the curse of dimensionality, a Critical Joint State Graph (CJSG) approach has been proposed for large graphs with a small amount of support with up to two robots, still assuring solution optimality. Reinforcement Learning (RL) has been utilized [12] to reduce the time complexity and scale the solution to a large group of robots and the size of the graph, but at the cost of sacrificing optimality.

George Mason University {klimbu2, yzhou30, gjstein, xwang64, dshishik, xiao}@gmu.edu.

To acquire theoretical insights into this problem, we reformulate TCGRE in a constrained optimization framework and present rigorous mathematical analysis of this reformulated problem. We prove the NP-hardness of TCGRE by reduction from the Maximum 3D Matching problem. We further show that such a difficult combinatorial optimization problem can be effectively addressed by efficient decomposition. In addition to providing a theoretical explanation for previous algorithms, we further introduce three distinct classes of methods to solve the TCGRE problem: (1) Based on the idea of JSG, we introduce new search algorithms that do not need to fully construct the JSG in advance and can be guided by a new admissible heuristic, while guaranteeing optimal solutions; (2) Inspired by the Conflict-Based Search [13], the second class of algorithms is based on coordination and we design a Coordination-Exhaustive Search (CES) algorithm. CES starts with individual optimal paths and finds the lowest cost among every possible coordination for every robot to achieve the optimal solution within polynomial time with respect to the number of robots, under the assumption that the coordination between every pair of support node and risky edge is only necessary for a limited number of times; (3) Motivated by CES's assumption, we also propose a class of receding-horizon sub-team solutions that further decomposes the order of coordination by only looking at sub-team coordinations in a local region. We design a Receding-Horizon Optimistic Cooperative A* (RHOC-A*) search, in order to reduce the time complexity without much performance loss. Extensive experiment results are presented and discussed to inform best ways to solve different TCGRE problems.

II. Related Work

We first review related work on the classical MAPF problem and common classes of algorithms. We then review previous approaches to solve the TCGRE problem.

A. MAPF *and Classes of Algorithms*

MAPF is a specific type of multi-agent planning problem with a key constraint that no agents can collide with one another [14]. A feasible solution to the problem is a joint plan that allows all agents to reach their goals from their starts. Two common objectives are makespan and total cost. Classical MAPF problem may include extra assumptions, such as no vertex conflict, no edge conflict, no cycle conflict, and no swapping conflict [15], [16].

Algorithms to solve MAPF include A*-based search with exponential space and time compelexity [15], [17], conflict-based search [13] by decomposing into many constrained single-agent problems, reduction-based approaches to SAT [18], [19], ILP [20], ASP [21], or CSP [22], [23], rule-based algorithms based on Kornhauser's algorithm [24], Push-and-Rotate [25], or BIBOX [26], and suboptimal solutions [27], [28] to sacrifice optimality for efficiency.

MAPF is NP-hard [29], and no optimal solutions can be found in polynomial time. The time complexity of all above optimal algorithms [13], [15], [17]–[23] is exponential to the number of agents. Similarly, we prove in this paper that our TCGRE problem that utilizes, instead of avoiding, interactions between agents in the form of support is also NP-hard.

B. Team Coordination on Graphs with Risky Edges (TCGRE)

TCGRE [11] is a recently proposed problem, in which a team of robots traverse a graph from their starts to goals and also support each other while traversing certain risky (high-cost) edges to reduce overall cost. Instead of focusing on collision-free paths in the traditional MAPF, the TCGRE problem pursues team coordination. To solve TCGRE, Limbu et al. [11] have proposed JSG and CJSG, both of which construct a single-agent joint-state graph. After the construction, the original team coordination problem can be solved using Dijkstra's algorithm to solve a shortest path problem with optimality guarantee. The CJSG construction deals with the team coordination problem more efficiently, although it can only solve problems with two agents. To scale up TCGRE, RL [11] has been utilized to handle many nodes and robots, but at the cost of optimality.

In this work, we reformulate the TCGRE in a constrained optimization framework and conduct mathematical analysis of this problem. We prove its NP-hardness and point out the necessity of efficient decomposition to effectively solve this problem. We further present three classes of algorithms to solve TCGRE from different perspectives.

III. Problem Formulation

Assuming a team of N homogeneous robots traverse an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of nodes the robots can traverse to and $\mathbb{E}$ is the set of edges connecting the nodes, i.e., $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$. The team of robots traverse in the graph from their start nodes $\mathbb{V}_0 \subset \mathbb{V}$ to goal nodes $\mathbb{V}_g \subset \mathbb{V}$ via edges in $\mathbb{E}$. Each edge $e_{ij} = (V_i, V_j) \in \mathbb{E}$ is associated with a cost c_{ij}, depending on its length, condition, traffic, obstacles, etc. Specially, some edges with high costs are difficult to traverse through, denoted as risky edges $\mathbb{E}' \subset \mathbb{E}$, but with the support from a teammate from a supporting node, their costs can be significantly reduced to $\tilde{c}_{ij}$. In this problem, we only consider such coordination behaviors between two robots. In one coordination behavior, one receiving robot receives support while traversing a risky edge, and another supporting robot offers support from some (nearby) location, called support node. Note that each risky edge $e_{ij} \in \mathbb{E}'$ corresponds to certain support node(s) $\mathbb{S}_{e_{ij}} \subset \mathbb{V}$ ($\mathbb{S}_{e_{ij}} = \emptyset$ if $e_{ij} \notin \mathbb{E}'$). Additionally, the coordination also induces some cost for the supporter, denoted by c'. A central planner needs to schedule the paths of all agents and coordination on their ways.

A. Action & Cost Model

Without coordination, at each time step t, a robot n can choose to stay where it is, or move to its neighbor (V_i is the neighbor of V_j if $e_{ij} \in \mathbb{E}$). Its movement can be denoted by $M_n^t = (l_n^t, l_n^{t+1}) \in \mathbb{E}$, where $l_n^t, l_n^{t+1} \in \mathbb{V}$ indicate its current and next location and l_n^{t+1} is a neighbor of l_n^t. Specially, the robot stays at its current location if $l_n^{t+1} = l_n^t$ with zero cost, i.e., $c_{ii} = 0, \forall i$. The movement set can thus be denoted by $\mathcal{M} = \{M_n^t | \forall n, \forall t\}$. Moreover, the movement decision M_n^t can be rewritten as an 0/1 variable M_{ij}^{nt}, where $M_{ij}^{nt} = 1$ represents edge e_{ij} is selected by robot n at time t, and 0 otherwise. A robot can only move once at each time step to a neighbor node or not at all, i.e., $\sum_{\forall e_{ij} \in \mathcal{N}_{l_n^t}} M_{ij}^{nt} = 1$ & $\sum_{\forall e_{ij} \notin \mathcal{N}_{l_n^t}} M_{ij}^{nt} = 0$, where $\mathcal{N}_{l_n^t} = \{(l_n^t, l_n^{t+1}) | \forall (l_n^t, l_n^{t+1}) \in \mathbb{E}\}$. The movement set can thus be denoted by $\mathcal{M} = \{M_{ij}^{nt} | \forall i, j, \forall n, \forall t\}$.

When a coordination behavior is available—when robot n is going to traverse a risky edge, another robot m happens to be at one of the support nodes of the risky edge or vice versa, i.e., $M_n^t \in \mathbb{E}'$ and $l_m^t \in \mathbb{S}_{M_n^t}$, or $M_m^t \in \mathbb{E}'$ and $l_n^t \in \mathbb{S}_{M_m^t}$—the robot pair needs to decide whether to provide/receive support. Denote the coordination decision of agent n at time t as s_{nm}^t. It is clear that agent n's coordination decision is dependent on its movement decision, so the cost is twofold: (1) When agent n has no coordination opportunity (the above coordination behavior is not available for any other robot m), i.e., $\forall m$, $l_m^t \notin \mathbb{S}_{M_n^t}$ and $l_n^t \notin \mathbb{S}_{M_m^t}$, its cost C_n^t is only decided by its movement, i.e., $C_n^t = c_{ij}$, where $M_{ij}^{nt} = 1$. (2) When coordination is possible for agent n, i.e., $\exists m$, $l_m^t \in \mathbb{S}_{M_n^t}$ or $l_n^t \in \mathbb{S}_{M_m^t}$, the cost C_n^t can be represented as

$$C_n^t = \begin{cases} c_{ij}, & \textit{if} \;\; s_{nm}^t = \;\;\, 0; \\ \tilde{c}_{ij} & \textit{if} \;\; s_{nm}^t = \;\;\, 1; \\ c', & \textit{if} \;\; s_{nm}^t = -1. \end{cases} \tag{1}$$

where $s_{nm}^t = 1$ means agent n decides to receive support from m, $s_{nm}^t = -1$ indicates agent n decides to offer support to m, and $s_{nm}^t = 0$ stands for no coordination between the robot pair n and m at t. Specially, no coordination happens for one single robot, i.e., $s_{nn}^t = 0, \forall n, \forall t$, or when n and m cannot support each other, i.e., $s_{nm}^t = 0$ if $\forall m$, $l_m^t \notin \mathbb{S}_{M_n^t}$ and $l_n^t \notin \mathbb{S}_{M_m^t}$. The coordination decision set can be written as $\mathcal{S} = \{s_{nm}^t | \forall n, m, \forall t\}$. In addition, a coordination decision is made for a pair, so $s_{nm}^t + s_{mn}^t = 0$ for every pair of robots. Furthermore, the robots can wait now (no movement) for future coordination, but there is no point for all robots to stay still at the same time, i.e., $\sum_{\forall n} \sum_{\forall i \neq j} M_{ij}^{nt} \neq 0$.

B. Problem Definition

Given the node set $\mathbb{V}$, the edge set $\mathbb{E}$, support nodes for each edge $\mathbb{S}_{e_{ij}}$, cost of each edge without and with coordination $c_{ij}, \tilde{c}_{ij}$, N robots with their starts $\mathbb{V}_0$ and goals $\mathbb{V}_g$, optimize the movement and coordination decisions $\mathcal{M}$ and $\mathcal{S}$, in order to minimize the total cost for each agent

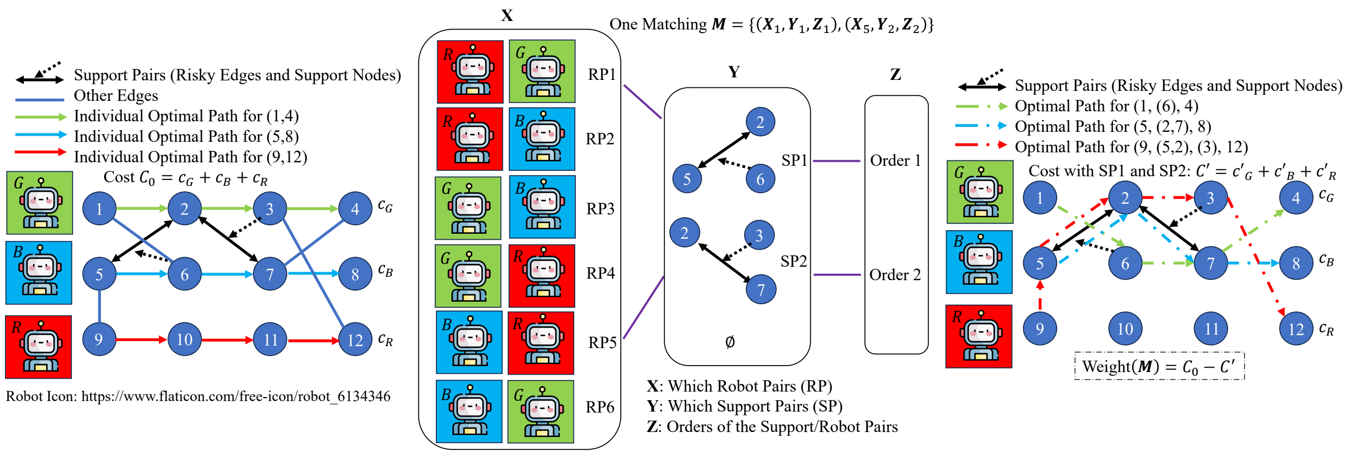

Fig. 1: Reduction from Maximum 3D Matching (Middle) to TCGRE (Left and Right) and Inspiration for CES.

to traverse from its start to its goal within a time limit T. Formally, the problem can be represented as

$$\min_{\mathcal{M},\mathcal{S}} \sum_{t=0}^{T-1} \sum_{n=1}^{N} C_n^t. \tag{2}$$

$$s.t. \sum_{\forall m \in \{1,2,...,N\}} |s_{nm}^t| \leq 1,$$
$$\forall n \in \{1,2,...,N\}, \forall t \in \{0,1,...,T-1\}. \tag{3}$$

$$s_{nm}^t, s_{mn}^t \in \{-1,0,1\}, s_{nm}^t + s_{mn}^t = 0,$$
$$\forall n,m \in \{1,2,...,N\}, \forall t \in \{0,1,...,T-1\}. \tag{4}$$

$$l_n^0 = \mathbb{V}_0(n), \forall n \in \{1,2,...,N\}. \tag{5}$$

$$l_n^T = \mathbb{V}_g(n), \forall n \in \{1,2,...,N\}. \tag{6}$$

$$\sum_{\forall e_{ij} \in \mathcal{N}_{l_n^t}} M_{ij}^{nt} = 1, \sum_{\forall e_{ij} \notin \mathcal{N}_{l_n^t}} M_{ij}^{nt} = 0,$$
$$\forall n \in \{1,2,...,N\}, \forall t \in \{0,1,...,T-1\}. \tag{7}$$

$$\sum_{\forall n} \sum_{\forall i \neq j} M_{ij}^{nt} \neq 0,$$
$$\forall t \in \{0,1,...,T-1\}. \tag{8}$$

Eqn. (2) suggests the goal of the problem is to minimize the total cost of all agents across all time steps with two decision variables, movement set $\mathcal{M}$ and coordination set $\mathcal{S}$. Eqn. (3) indicates, at each time step, each robot can participate in at most one coordination behavior. Eqn. (4) regulates that, at each time step, one coordination behavior only occurs between one robot pair. Eqn. (5) and Eqn. (6) set the start and the goal for each robot. Eqn. (7) guarantees, at each time step, a robot can only move to a neighbor node or stay still. Eqn. (8) assures no unnecessary stagnation.

IV. Mathematical Analysis

In this section, we prove our TCGRE problem reduces from the Maximum 3D Matching problem, a NP-hard problem. Then, we start a rigorous analysis on the mathematical problem, which suggests decomposition is a promising solution to this combinatorial optimization problem.

A. NP-Hardness

Definition 1. *Maximum 3D Matching:* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *are 3 finite sets.* $\mathbf{T}$ *is the subset of* $\mathbf{X} * \mathbf{Y} * \mathbf{Z}$*, with triples* (x, y, z)*, where* $x \in \mathbf{X}, y \in \mathbf{Y}, z \in \mathbf{Z}$. $\mathbf{M} \subset \mathbf{T}$ *is a 3D matching if for any two distinct triples* (x_1, y_1, z_1) *and* $(x_2, y_2, z_2) \in \mathbf{M}$*, we have* $x_1 \neq x_2$*,* $y_1 \neq y_2$*, and* $z_1 \neq z_2$*; each triple has a weight* $w(x_i, y_j, z_k)$*. Maximum 3D Matching problem is to find a 3D matching with maximum total weight.*

Theorem 1. *Maximum 3D Matching reduces to* TCGRE*.*

Proof sketch: Without coordination, we can generate individual optimal paths for all robots; the cost is C_0 (Fig. 1 left). With coordination, we let some robot pairs take detours to the support pairs (i.e., one risky edge and one support node); the new cost is C', where the first robot in the pair traverses the risky edge while the second visits and supports from the support node (Fig. 1 right). Minimizing total cost is equivalent to maximizing the cost reduction $C_0 - C'$.

Consider $\mathbf{X}$ contains all robot pairs; $\mathbf{Y}$ is the set of all support pairs plus an empty set element; $\mathbf{Z}$ is a list of time orders of events (time steps are not necessarily needed, because robots can stay still and wait, Fig. 1 middle). Consider the weight $w(x_i, y_j, z_k)$ as the sum of cost reduction through coordination for robot pair x_i to detour to support pair y_j with time order z_k. The weight of a 3D matching $\mathbf{M}$ is the total cost reduction of all robots, $C_0 - C'$. Maximum 3D Matching is to find the paths for all robot pairs to achieve maximum total cost reduction from their original without-coordination costs, which is a specific case of the general TCGRE problem that only needs to use each robot and support pair once with a smaller solution space than TCGRE.

Since TCGRE reduces from Maximum 3D Matching, a classical NP-hard problem [30], TCGRE is also NP-hard. We cannot find an optimal solution in polynomial time. □

B. Problem Analysis

Because we only care about the total cost of all robots, and every coordination requires a robot pair, we can reassign the coordination cost c' from the supporter to the receiver in addition to the reduced edge cost $\tilde{c}_{ij}$, without changing the problem. So, the cost with coordination in Eqn. (1) becomes

$$C_n^t = \begin{cases} c_{ij}, & \text{if } s_{nm}^t = \ \ 0; \\ \hat{c}_{ij}, & \text{if } s_{nm}^t = \ \ 1; \\ 0, & \text{if } s_{nm}^t = -1. \end{cases} \tag{9}$$

where $\hat{c}_{ij} = \tilde{c}_{ij} + c'$. Therefore, the original objective function (Eqn. (2)) can be rewritten as

$$\min_{\mathcal{M},\mathcal{S}} \sum_{t=0}^{T-1} \sum_{n=1}^{N} \sum_{\forall i \neq j} M_{ij}^{nt}[(1 - s_{nm}^t)c_{ij} + s_{nm}^t \hat{c}_{ij}], \tag{10}$$
$$s.t. \quad (3), (4), (5), (6), (7), (8).$$

Notice that because when $s_{nm}^t = -1$, $M_{ij}^{nt} = 0, \forall i \neq j$, the last *if* condition in Eqn. (9) does not need to be considered in Eqn. (10). To solve the combinatorial optimization problem, a typical approach is dynamic programming [31], by decoupling the interdependency among the decision variables. The ideal case is to break down the problem into two sub-problems: one optimizing the movement decisions $\mathcal{M}$ and the other optimizing the coordination decisions $\mathcal{S}$.

Based on such a motivation, if we can find a way to eliminate n from M_{ij}^{nt}, Eqn. (10) can be rewritten as

$$\underbrace{\min_{\mathcal{M}} \sum_{t=0}^{T-1} \sum_{\forall i \neq j} M_{ij}^{nt}}_{\text{Sub-Problem 2}} \underbrace{\min_{\mathcal{S}} \sum_{n=1}^{N} [(1 - s_{nm}^t)c_{ij} + s_{nm}^t \hat{c}_{ij}]}_{\text{Sub-Problem 1}}. \tag{11}$$

The first half of the function contains only the movement decisions $\mathcal{M}$, while the second half only has the coordination decisions $\mathcal{S}$. Then, decomposition is possible and the NP-hard TCGRE problem can be solved with significantly reduced complexity.

V. SOLUTIONS

Based on the mathematical analysis, we propose three classes of algorithms to solve TCGRE from different perspectives with different optimality and efficiency characteristics. We first propose a class of JSG-based solutions that utilizes the decomposition of the original problem and provides optimal solutions to the two sub-problems (Eqn. 11). Second, to reduce the time complexity, we focus on coordination (i.e., $\mathcal{S}$) and propose a class of coordination-based solutions that decomposes the problem differently. To be specific, we introduce Coordination-Exhaustive Search (CES), which can achieve optimal solutions under a reasonable assumption that each coordination behavior (i.e., support pair composed of support node and risky edge) is only needed for a constant time in the optimal solution. Finally, when the same support may need to be repeated many times and the global optimality does not need to be guaranteed, a class of algorithms that focuses only on local sub-team coordination behaviors are introduced, for which we develop Receding-Horizon Optimistic Cooperative A* (RHOC-A*).

A. JSG-*Based Solutions*

JSG-based solutions perform the decomposition into the two sub-problems by 1) implicitly solving $\mathcal{S}$ by calculating the minimum edge cost for each edge in the JSG; and 2) explicitly solving $\mathcal{M}$ by solving a single-robot shortest path problem with $\mathcal{S}$ implicitly encoded. As mentioned in Sec. IV-B, n is effectively eliminated from M_{ij}^{nt} by building the JSG.

1) JSG *Construction:* In the action model (Sec. III-A), we use $l_n^t \in \mathbb{V}$ to represent robot n's location at time t. In a joint state graph, however, one state is the set of all robots' locations $\mathbf{L}^t = \{l_1^t, l_2^t, ..., l_N^t\}$. The new node set is $\mathbb{L} = \mathbb{V}^N$ and each node $\mathbf{L}_p \in \mathbb{L}$ correlates to N nodes $(V_{i_1}, V_{i_2}, ..., V_{i_n})$ in $\mathbb{V}$. By checking the constraints in Eqns. (3), (4), (7), and (8), for each pair of joint-states, we can form the new edge set $\mathbb{M} \subset \mathbb{L}^2$. Specially, Eqn. (8) assures no self-loops in the JSG. An action is the move from current state to next state $\mathbf{M}^t = (\mathbf{L}^t, \mathbf{L}^{t+1})$, which can also be written as a 0/1 variable, i.e., $\mathbf{M}_{pq}^t = \prod_{n=1}^N M_{ij}^{nt}$, where $\mathbf{e}_{pq}$ is any edge in the new edge set, and the movement decision set becomes $\mathcal{M} = \{\mathbf{M}^t | \forall t\}$.

2) Sub-Problem 1: After the construction of JSG, the second half of Eqn. (11) is calculating the minimum edge cost for each $\mathbf{e}_{pq} \in \mathbb{M}$ by optimizing $\mathcal{S}$, which is a 0/1 Integer Linear Programming (ILP) problem and can be solved by classical methods, such as Branch and Bound [32].

3) Sub-Problem 2: After solving sub-problem 1, we have the cost C_{pq} for each edge $\mathbf{e}_{pq} \in \mathbb{M}$. Sup-problem 2 is to optimize movement decisions $\mathcal{M}$ to minimize the total cost:

$$\min_{\mathcal{M}} \sum_{t=0}^{T-1} \sum_{\forall \mathbf{e}_{pq} \in \mathbb{M}} \mathbf{M}_{pq}^t C_{pq}^t, \tag{12}$$

Now it is a single-robot shortest path problem with non-negative costs and no self-loops solvable by any shortest path algorithms. Instead of using Dijkstra's algorithm on a fully constructed JSG beforehand [11], we present results with Uniform Cost Search (UCS) and A* (guided by an admissible heuristics assuming all future risky edges will be supported by a teammate) while constructing JSG on the fly, i.e., interleaving partial solutions of the two sub-problems.

B. Coordination-Based Solutions

Some JSG edge costs are simply the sum of individual edge costs of all robots, suggesting possible total cost separation into the costs with and without coordination. Thus, while minimizing costs without coordination can be simply solved for individual robots, the second class of algorithms focuses on coordination. Specifically, we present a Coordination-Exhaustive Search (CES) algorithm based on a slightly different and interleaving decomposition of Eqn. (10):

$$\min_{\mathcal{M},\mathcal{S}} \overbrace{\sum_{t=0}^{T-1} \sum_{n=1}^{N} \sum_{\forall e_{ij} \in \mathbb{E}} M_{ij}^{nt} c_{ij}}^{\text{Cost without Coordination}} - \tag{13}$$
$$\underbrace{\sum_{t=0}^{T-1} \sum_{n=1}^{N} \sum_{\substack{l_n^t = l_n^{t+1} \in \\ \bigcup_{m=1}^N \mathbb{S}_{M_m^t} \\ \forall e_{ij} \in \mathbb{E}' or}} \frac{1}{2} M_{ij}^{nt}[(1 - s_{nm}^t)c_{ij} + (1 + s_{nm}^t)\Delta c_{ij}]}_{\text{Cost Reduction due to Coordination}}.$$

where $\Delta c_{ij} = c_{ij} - \hat{c}_{ij}$. When $s_{nm}^t = 1$ (receiving support) the second part reduces the cost in the first part by Δc_{ij}; When $s_{nm}^t = -1$ (providing support), the cost is reduced to zero. Inspired by Conflict-Based Search, we can start with finding the individual shortest path for each robot without coordination. Then, we find the coordination behaviors (with some detours) that can cause the maximum cost reduction.

While the first half can be solved individually for each robot (Fig. 1 left), CES uses an exhaustive search for the second half. If an optimal solution requires a coordination behavior between a robot pair, it is equivalent to make the robot pair detour to the support pair (risky edge and support node) from their original individual shortest paths, while other robots remain on their individual shortest paths (Fig. 1 right). This robot pair's shortest paths are a combination of two path segments—their shortest paths from their starts to the support pair, and their shortest paths from the support pair to their goals. Given certain coordination behaviors, we can solve the shortest path and minimum total cost of all robots for each path segment (Fig. 1 right), as shown in Alg. 1.

CES is a coordination-based method through an exhaustive search. Because one support pair may be assigned to the same/different robot pair(s) for infinite times, to conduct an exhaustive search, we assume each support pair can only occur for a fixed number of times. In our implementation, we assume a support pair can only happen once, but it can be easily expanded to a more general case, by repetitively adding the same support pairs to the coordination set.

The CES algorithm is shown in Alg. 2: In lines 1-2, using any shortest path algorithm, it generates an individual optimal path and cost for each robot, with original edge costs $\mathbf{c} = \{c_{ij} | \forall e_{ij} \in \mathbb{E}\}$ and reduced edge costs $\hat{\mathbf{c}} = \{\hat{c}_{ij} | \forall e_{ij} \in \mathbb{E}\}$, called pessimistic/optimistic paths $\mathbb{P}_1/\mathbb{P}_2$ and total cost C_1/C_2. If $C_1 = C_2$, which means no coordination is needed in the optimal solution, then simply return $\mathbb{P}_1$ and C_1 (lines 3-4). Else, it starts the scheduling process. In line 5, it generates a coordination set, $\mathbb{CS}$ that contains all coordination behaviors, and $\mathbb{SCS}$ that contains all subsets of $\mathbb{CS}$, to decide which coordination behaviors/support pairs are needed. In line 6, it generates a set of all robot pairs $\mathbb{RP}$ (order in the pair matters since we need to decide which robot moves to the support node/risky edge) to determine which robot pair should be assigned to each support pair. Now, it looks like a Maximum 3D Matching problem as stated in Definition 1, except that the matching problem has one more assumption that one robot pair can be only assigned once as shown in Fig. 1 middle. For all possible sets of support pairs (line 8), it generates all possible time orders for this support pair set using permutation (line 9). Notice that robots can wait for one another, so the order of each coordination behavior, not necessarily the exact time step, is sufficient. Then, it iterates through every possible support pair order (line 10). There could be $N(N-1)$ possible robot pairs assigned to each support pair, so a N-Fold Cartesian Product is applied to generate all possible sets of support robot pairs (line 11). Then, it explores every set (line 12), where each robot pair in $\mathbf{SRP}$ is assigned to each support pair in $\mathbf{PSCS}$ with the same index. Thus, in lines 13-18, it adds the risky edge of each support pair to the individual coordination set of the first robot of the robot pair, and the support node of each support pair to the individual coordination set of the second robot of the robot pair. $\mathbf{ICS}$ then contains individual coordination set of all robots. With $\mathbf{ICS}$, we use Alg. 1 to calculate the shortest paths and minimum total cost of this assignment in line 19, which is one solution. Last, lines 20-21 records the best solution with minimum total cost. To sum up, the loop in line 8 decides a subset of support pairs we need for cost reduction. The loop in line 10 decides an order for the subset. The loop in line 12 selects a robot pair for each support pair in the subset. Iterating through the three loops explores every possible solution under the assumption that each support pair can be applied for a constant number of times, making CES optimal.

Algorithm 1: $CostCalculation$ $(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g, \mathbf{ICS})$

$\mathbb{P} = [\emptyset] * N$;
$totalcost = 0$;
for $n = 1$ *to* N **do**
    if $\mathbf{ICS}[n] = \emptyset$ **then**
        $\mathbf{P}, C = ShortestPath(\mathbb{G}, \mathbb{V}_0[n], \mathbb{V}_g[n], \mathbf{c})$;
        $\mathbb{P}[n] = \mathbf{P}$;
        $totalcost = totalcost + C$;
    else
        for $item \in \mathbf{ICS}[n]$ **do**
            $start = \mathbb{V}_0[n]$;
            if *item is a risky edge* **then**
                $\mathbf{P}, C = ShortestPath(\mathbb{G}, start, item[0], \mathbf{c})$;
                $\mathbb{P}[n].extend(\mathbf{P} \bigcup item)$;
                $totalcost = totalcost + C + \hat{c}_{item}$;
                $start = item[1]$;
            else if *item is a support node* **then**
                $\mathbf{P}, C = ShortestPath(\mathbb{G}, start, item, \mathbf{c})$;
                $\mathbb{P}[n].extend(\mathbf{P})$;
                $totalcost = totalcost + C$;
                $start = item$;
        $\mathbf{P}, C = ShortestPath(\mathbb{G}, start, \mathbb{V}_g[n], \mathbf{c})$;
        $\mathbb{P}[n].extend(\mathbf{P})$;
        $totalcost = totalcost + C$;
return $\mathbb{P}, totalcost$

There are total $O(2^{|\mathbb{CS}|})$ subsets in $\mathbb{SCS}$. For each subset, there are $O(|\mathbb{CS}|!)$ permutations. For every permutation, there are $O(N^2)$ possible robot pairs for each support pair, totally $O(N^{2|\mathbb{CS}|})$ assignments. Therefore, the number of possible solutions is $O(2^{|\mathbb{CS}|} \cdot N^{2|\mathbb{CS}|} \cdot |\mathbb{CS}|!) = O((2N^2)^{|\mathbb{CS}|} \cdot |\mathbb{CS}|!)$. For each solution, we run the shortest path algorithm for $O(|\mathbb{CS}|)$ times, each run costs $O(|\mathbb{E}| log(|\mathbb{V}|))$. Therefore, the time complexity of CES is $O((2N^2)^{|\mathbb{CS}|} \cdot |\mathbb{CS}|! \cdot |\mathbb{E}| log(|\mathbb{V}|))$, which is not exponential to the number of robots N anymore. Note that the above algorithm is for directed graphs. For an undirected graph, each edge is actually two directed edges in a directed graph, so after line

Algorithm 2: CES $(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g)$

1 $\mathbb{P}_1, C_1 = MultipleShortestPath(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g, \mathbf{c})$;
2 $\mathbb{P}_2, C_2 = MultipleShortestPath(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g, \hat{\mathbf{c}})$;
3 **if** $C_1 \leq C_2$ **then**
4   **return** $\mathbb{P}_1, C_1$;
5 Generate subsets $\mathbb{SCS} = \{\mathbf{SCS} | \forall \mathbf{SCS} \subset \mathbb{CS}\}$ of the coordination set $\mathbb{CS}$;
6 Generate a set of all robot pairs $\mathbb{RP}$;
7 $\mathbb{P}_{min}, C_{min} \leftarrow \mathbb{P}_1, C_1$;
8 **for** $\mathbf{SCS} \in \mathbb{SCS}$ **do**
9     $\mathbb{PSCS} = AllPermutations(\mathbf{SCS}, len(\mathbf{SCS}))$;
10   **for** $\mathbf{PSCS} \in \mathbb{PSCS}$ **do**
11         $\mathbb{SRP} = CartesianProduct(\mathbb{RP}, len(\mathbf{PSCS}))$;
12   **for** $\mathbf{SRP} \in \mathbb{SRP}$ **do**
13             $\mathbf{ICS} = [\emptyset] * N$;
14   **for** $n = 1$ *to* $len(\mathbf{PSCS})$ **do**
15                 $SP = \mathbf{PSCS}[n]$;
16                 $RP = \mathbf{SRP}[n]$;
17                 $\mathbf{ICS}[RP[0]].append(SP[0])$;
18                 $\mathbf{ICS}[RP[1]].append(SP[1])$;
19             $\mathbb{P}, C = CostCalculation(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g, \mathbf{ICS})$;
20   **if** $C < C_{min}$ **then**
21                 $\mathbb{P}_{min}, C_{min} \leftarrow \mathbb{P}, C$;
22 **return** $\mathbb{P}_{min}, C_{min}$.

10, there should be an additional loop that iterates through all possible directions of selected risky edges, which won't change the time complexity class.

C. Receding-Horizon Sub-Team Solutions

TCGRE's computation complexity arises from two fronts, the large size of the graph and the team. Therefore, the third class of algorithms reduces the complexity from both fronts by planning with a limited horizon and for a sub-team of all robots at a time, efficiently facilitating local coordination while compromising global optimality. Such sub-team local coordination within the receding horizon prioritizes actions that yield the best short-term outcomes and potentially allows dynamic adaptation to changing circumstances, e.g., updated graph structure from robot perception. One specific algorithm is Receding-Horizon Optimistic Cooperative A* (RHOC-A*), which provides flexible and efficient solution by assuring optimal robot pair coordination within the horizon while assuming optimistic cooperation beyond the horizon (Fig. 2). Alg. 3 presents RHOC-A*. All robots are initially not at their goals and therefore on duty (lines 1-2). We compute the heuristic for all nodes assuming always-available support (line 3). RHOC-A* iterates until all robots arrive at their goals (lines 4). If there is still at least one pair of robots on duty (line 5), RHOC-A* sequentially plans for each robot pair (lines 6-7). The computation efficiency is enabled by only looking at a small JSG with only two robots, n and m, within K steps (lines 8-12). Notice that optimality is assured

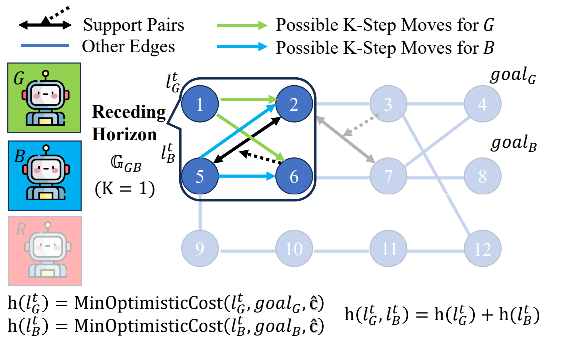

Fig. 2: Receding-Horizon Optimistic Cooperative A*.

on this small JSG while the cost-to-go on the horizon K is estimated by the always-available support heuristic. Lines 13-18 address the situation where only one robot is on duty, who has to traverse to its goal alone.

For N robots, generating all possible pairs of robots can be done in $O(N^2)$ time. RHOC-A*'s time complexity for each pair's search of K steps can be approximated as $O(b^K)$, where $b = O(\frac{|\mathbb{E}|}{|\mathbb{V}|})$ is the effective branching factor in the joint action space. Thus, in K steps, running RHOC-A* for all pairs results in complexity of $O(N^2(\frac{|\mathbb{E}|}{|\mathbb{V}|})^K)$. There will be no cycles for any robot, due to graph search. As a result, there will be at most $O(|V|)$ steps for each robot, so we need to run the K-step A* search for $\frac{|V|}{K}$ times for one robot pair. There will be $\frac{N}{2}$ runs for N robots. Therefore, the time complexity of the algorithm is $O(N^3 \cdot \frac{|V|}{K} \cdot (\frac{|\mathbb{E}|}{|\mathbb{V}|})^K)$.

Algorithm 3: RHOC-A*$(\mathbb{G}, \mathbb{V}_0, \mathbb{V}_g, K)$

1 Initialize $atGoal_n \leftarrow False, \forall n \in \{1, 2, ..., N\}$
2 $\mathbf{OnDuty} = \{n | \forall n, atGoal_n = False\}$
3 Compute optimistic heuristic $h(\cdot)$ for each node in $\mathbb{G}$
4 **while** $len(\mathbf{OnDuty}) \neq 0$ **do**
5   **if** $len(\mathbf{OnDuty}) \geq 2$ **then**
6         $\mathbb{RP} = \{\{n, m\} | \forall n, m \in \mathbf{OnDuty}\}$
7   **for** *each pair* $\{n, m\} \in \mathbb{RP}$ **do**
8   Initialize a K-step JSG $\mathbb{G}_{nm}$ for the pair $\{n, m\}$ with start $\{l_n^t, l_m^t\}$;
9   **if** *not* $atGoal_{n,m}$ **then**
10   A* on $\mathbb{G}_{nm}$ for K steps using $h(\cdot)$;
11   Update l_n^t, l_m^t, $atGoal_{n,m}$, and $\mathbf{OnDuty}$;
12   Update individual and total costs;
13   **else**
14         $n = \mathbf{OnDuty}.pop()$;
15   Initialize K-step graph $\mathbb{G}_n$ with start l_n^t;
16   A* on $\mathbb{G}_n$ for K steps using $h(\cdot)$;
17   Update l_n^t, $atGoal_n$, and $\mathbf{OnDuty}$;
18   Update individual and total costs;
19 **return** Paths and costs for all robots.

VI. Results

We conduct experiments on a variety of graphs to evaluate the optimality and efficiency of the three classes of proposed

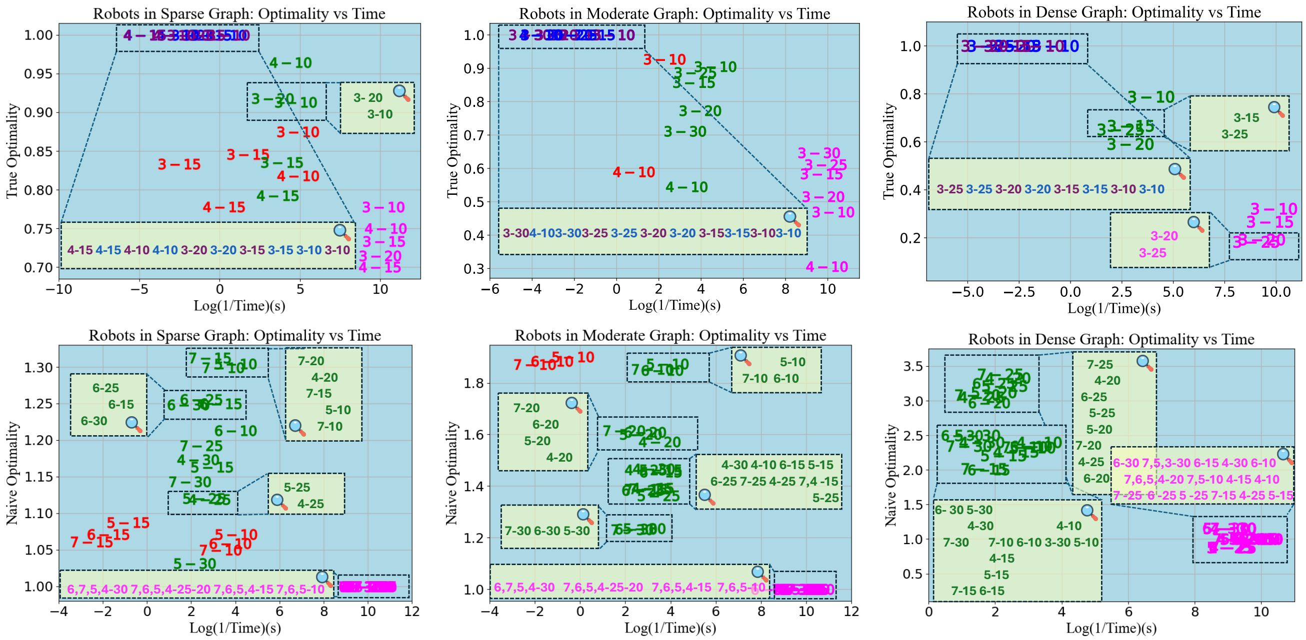

Fig. 3: True and Naive Optimality vs. Time with JSG-UCS, JSG-A*, CES, RHOC-A*, and Naive. Each data point denotes the result for the experiment with # of Robots–# of Nodes. For visibility, cluttered areas are magnified in the dashed boxes.

algorithms. To be specific, we implement UCS and A* for JSG-based solutions, CES for coordination-based solutions, and RHOC-A* for receding-horizon sub-team solutions. We first conduct a set of large-scale, method-agnostic experiments on a variety of randomly generated graphs and then present focused experiments to study the pros and cons of specific methods.

A. *Large-Scale Method-Agnostic Experiments*

To evaluate each method in an objective manner, we generate a set of graphs with randomly generated support pairs, including sparse, moderate, and dense connectivities and five different numbers of nodes ($|\mathbb{V}| \in \{10, 15, 20, 25, 30\}$), three graphs each type, i.e., a total of 45 distinct graphs. A total of 900 trials are conducted with five different team sizes ($N \in \{3, 4, 5, 6, 7\}$) and four methods.

We evaluate the optimality and runtime of all methods along with a naive approach, in which each robot executes its individual optimal path without coordination. While the True Optimality value is defined as the optimal cost divided by the actual cost, for scenarios where the optimal cost cannot be found due to excessive computation, we define Naive Optimality to be the naive cost divided by actual cost. If a data point does not exist in Fig. 3, the corresponding method cannot produce a solution for the robot and node number. As shown in Fig. 3, the JSG-based solutions achieve optimal solutions, but require significant runtime even in small graphs with only a few robots and fail to produce a solution when the problem becomes larger; CES has better runtime but loses some performance because we assume each support pair can be applied only once; with a fine-tuned K, surprisingly, RHOC-A* in many cases achieves better results than CES with less runtime.

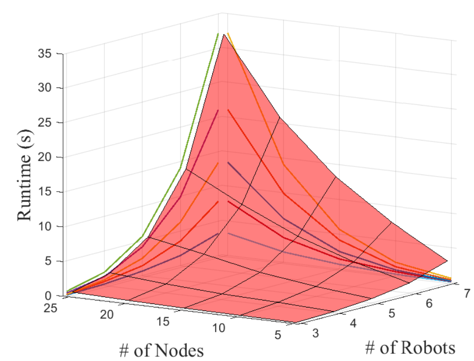

Fig. 4: CES Planning Time on Graphs with Two Support Pairs

B. *Focused Experiments*

1) CES's Insensitivity to Robot and Node Numbers: Fig. 4 showcases that, when there are not many support pairs, CES works well with different numbers of robots and different sizes of graphs (polynomial time to both N and $|\mathbb{V}|$). However, its runtime increases drastically with the number of support pairs, as shown in our method-agnostic experiments, which verifies our time complexity analysis (Sec. V-B).

2) RHOC-A's Sensitivity to Planning Horizon:* Fig. 5 showcases how RHOC-A*'s computation time scales with different planning horizons. A large horizon K comes closer to solving the original TCGRE problem with multiple robot pairs, which significantly increases the solution time. While the total cost can be reduced with a longer horizon, it is necessary to strike a balance between horizon and efficiency.

VII. Conclusions and Discussions

We present a systematic problem formulation and mathematical analysis of TCGRE, which proves its NP-hardness and shows efficient decomposition is the key to solving this

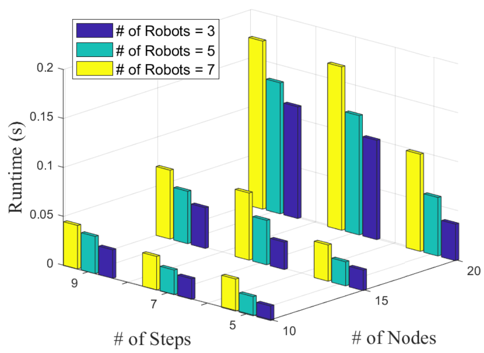

Fig. 5: RHOC-A* Planning Time.

problem. We propose three classes of solutions with a set of implementations and present their experiment results.

As given by the analysis in Sec. IV-B, all of the proposed solutions are trying to solve a form of a decomposed problem. For example, JSG-based solutions solve a 0/1 ILP problem [20] and a single-agent shortest path problem, after constructing a JSG; coordination-based solutions, like CES, deal with a 3D matching problem embedded with multiple single-agent shortest path problems. By applying some approximation methods to the subproblems—for the former, forming only a few edges instead of all feasible edges and calculating approximate edge costs; for the latter, omitting unpromising matchings—we can significantly reduce runtime without sacrificing too much performance. RHOC-A*, though efficient, does not consider the order of the robot pair selection, with which its performance can improve while still maintaining coordination efficiency.

References

[1] J. L. Adler and V. J. Blue, "A cooperative multi-agent transportation management and route guidance system," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5-6, pp. 433–454, 2002.

[2] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3177–3192, 2021.

[3] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," 2022.

[4] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 24-1, pp. 1261–1263, 2010.

[5] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *International conference on machine learning*, pp. 1146–1155, PMLR, 2017.

[6] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pp. 66–83, Springer, 2017.

[7] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[8] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[9] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3268–3275, IEEE, 2011.

[10] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.

[11] M. Limbu, Z. Hu, S. Oughourli, X. Wang, X. Xiao, and D. Shishika, "Team coordination on graphs with state-dependent edge costs," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 679–684, IEEE, 2023.

[12] M. Limbu, Z. Hu, X. Wang, D. Shishika, and X. Xiao, "Team coordination on graphs with reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[13] 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.

[14] R. Stern, "Multi-agent path finding–an overview," *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pp. 96–115, 2019.

[15] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24-1, pp. 173–178, 2010.

[16] A. Felner, R. Stern, S. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 8-1, pp. 29–37, 2017.

[17] M. R. Ryan, "Exploiting subgraph structure in multi-robot path planning," *Journal of Artificial Intelligence Research*, vol. 31, pp. 497–542, 2008.

[18] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *Pacific Rim international conference on artificial intelligence*, pp. 564–576, Springer, 2012.

[19] P. Surynek, "Makespan optimal solving of cooperative path-finding via reductions to propositional satisfiability," *arXiv preprint arXiv:1610.05452*, 2016.

[20] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3612–3617, IEEE, 2013.

[21] E. Erdem, D. Kisa, U. Oztok, and P. Schüller, "A general formal framework for pathfinding problems with multiple agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27-1, pp. 290–296, 2013.

[22] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient sat approach to multi-agent path finding under the sum of costs objective," in *Proceedings of the twenty-second european conference on artificial intelligence*, pp. 810–818, 2016.

[23] R. Barták, N.-F. Zhou, R. Stern, E. Boyarski, and P. Surynek, "Modeling and solving the multi-agent pathfinding problem in picat," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 959–966, IEEE, 2017.

[24] D. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," in *25th Annual Symposium onFoundations of Computer Science, 1984.*, pp. 241–250, 1984.

[25] B. De Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: a complete multi-agent pathfinding algorithm," *Journal of Artificial Intelligence Research*, vol. 51, pp. 443–492, 2014.

[26] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3613–3619, IEEE, 2009.

[27] R. C. Holte, M. B. Perez, R. M. Zimmer, and A. J. MacDonald, "Hierarchical a*: Searching abstraction hierarchies efficiently," in *AAAI/IAAI, Vol. 1*, pp. 530–535, 1996.

[28] R. E. Korf, "Real-time heuristic search," *Artificial intelligence*, vol. 42, no. 2-3, pp. 189–211, 1990.

[29] O. Goldreich, *Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard*. Springer, 2011.

[30] V. Kann, "Maximum bounded 3-dimensional matching is max snp-complete," *Information Processing Letters*, vol. 37, no. 1, pp. 27–35, 1991.

[31] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[32] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.