Human-Robot Teaming on Graphs with Risky Edges

Sara Oughourli, Manshi Limbu, Zechen Hu, Xuan Wang, Xuesu Xiao, and Daigo Shishika

I. INTRODUCTION

We are interested in designing coordinated group motion, where the safety or cost for one agent to move from one location to another may depend on the support provided by its teammate. For example, consider a scenario where a team of human and robot must traverse an environment with some "risk" edges as shown in Fig. 1. Those risks might represent actions such as going up a ladder, crossing a "shaky" bridge, or walking through a dark tunnel. In these situations, a human (or robot) teammate can support the other by holding the ladder, stabilizing the bridge, or lighting up the tunnel. We capture the feasibility of these "supporting" actions in the green dashed arrows in Fig. 1, extending from the nodes from which the support can be provided. The core questions we seek to answer are: (i) when such support/coordination is beneficial, and (ii) how to best coordinate the actions as a team to minimize the overall cost.

We formulate a problem that incorporates support actions to a minimum-cost graph traversal problem. We then propose a solution approach based on the notion of joint state graph (JSG) formulation, converting the problem into single-agent path planning. To address the curse of dimensionality, a hierarchical decomposition method based on Critical Joint State Graph (CJSG) is introduced for two-level planning. Complexity and statistical analyses demonstrate the efficacy of our algorithm.

II. PROBLEM FORMULATION

The base graph is denoted by $\mathbb{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes, and $\mathcal{E}$ is a set of edges. Each edge $e_{i,j} \in \mathcal{E}$ is associated with a set of support nodes, $\mathcal{Z}_{i,j} \subseteq \mathcal{V}$. The *environment graph* incorporates the notion of risk and support. The action set for an agent $n \in \{A, B\}$ at node i is given as $\mathcal{A}_i^n = \{\{a_{i,j}\}_{j\in\mathcal{N}_i}, a_s\}$, where the first set is for moving, and a_s is for supporting its teammate. To capture the effect of supporting actions, the state and action dependent cost (for agent A) is given by

$$c_A^t(\cdot) = \begin{cases} c_{i,j}, & \text{if } a_A = a_{i,j} \text{ and } p_B \notin \mathcal{Z}_{i,j} \text{ or } a_B \neq a_s, \\ \tilde{c}_{i,j}, & \text{if } a_A = a_{i,j},\ p_B \in \mathcal{Z}_{i,j}, \text{ and } a_B = a_s, \\ \tilde{c}, & \text{if } a_A = a_s, \end{cases}$$

where $(\cdot)$ represents joint state and actions (p^t, a^t). The reduced cost due to support is captured by $\tilde{c}_{i,j}$.

Given a sequence of actions that takes the agents from start to goal node, we can compute the sum of total costs to obtain the overall cost of the path. The goal is to find a pair of sequences (one for each agent) that minimizes the overall cost. While this problem can be solved as an instance of MDP, we will introduce a simplification using the concept of Joint State Graph (JSG).

The authors are with George Mason University. Corresponding authors: {xwang64,xiao,dshishik}@gmu.edu

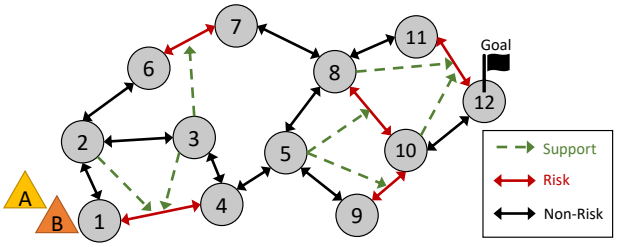

Fig. 1. Example of an environment graph with risk edges and supporting nodes.

III. METHOD

A. Joint State Graph

Let the JSG be a graph where its nodes represent the joint states, and its edges represent possible transitions between those joint states. The cost associated with each edge is the sum of costs for each agent's actions. This entails that JSG subsumes the action selection in the original problem. Now the joint action-selection problem is converted into a single-agent path-planning in JSG, which can be solved with any standard shortest-path alrogithm. Although planning on JSG is conceptually simple, it can become computationally expensive with greater graph sizes. The next section addresses this issue.

B. Critical Joint State Graph

For computational efficiency, we propose to categorize the agents' movements into coupled and decoupled modes, where only the coupled movements need to be planned in JSG, and the decoupled movements can be independently planned by each agent on base graph. To do this, depending on whether the edges in the environment graph have at least one support node, we can represent the possible supporting behaviors between two agents based on their locations. Then by collecting all possible supporting behaviors, we construct a Critical Joint State Graph (CJSG) that only captures two agents' coupled movements. The node of CJSG is any joint state that the two agents (i) can initiate or complete supporting behaviors, (ii) at their start or goal position of the planning task. We let CJSG be fully connected. The edge costs are associated with two agents moving over the base graph or a possible lower cost when they perform a support

behavior. We can theoretically validate that the path planned on CJSG has the same optimal cost as the one on JSG.

IV. RESULTS

Animations of coordinated team behavior are available at `https://youtu.be/oDqfBxmjx_E`. We present results on computational complexity in the following.

A. Quantification of Computational Complexity.

We quantify the computational complexity of the search algorithm applied to the JSG and the CJSG methods. Consider using Dijkstra's Algorithm. The complexity of optimal path search on JSG is

$$\mathcal{O}_{\text{JSG}} = \mathcal{O}(|\mathcal{V}|^4).$$

where the number of nodes in JSG is $|\mathcal{V}|^2$. In contrast, the complexity of the CJSG method is

$$\mathcal{O}_{\text{CJSG}} = \mathcal{O}(|\mathcal{V}|^2 \log(|\mathcal{V}|)) + \mathcal{O}(|\mathcal{M}|^2).$$

where the first term is the graph construction complexity of CJSG. It requires computing the shortest path between any pair of nodes in the base graph, assuming the use of Johnson's algorithm. The second term is the search complexity, where $|\mathcal{M}|$ is the number of nodes in the constructed CJSG.

To compare $\mathcal{O}_{\text{CJSG}}$ and $\mathcal{O}_{\text{JSG}}$, we only need to compare $\mathcal{O}(|\mathcal{M}|^2)$ and $\mathcal{O}(|\mathcal{V}|^4)$. If we assume the number of support edges in the environment graph is small, then $|\mathcal{M}| \ll |\mathcal{V}|^2$. The worst boundary scenario happens when support edges widely exist in environment graph. In this case, $|\mathcal{M}| \to |\mathcal{V}|^2$, but $|\mathcal{M}|$ is still upper bounded by $|\mathcal{V}|^2$ because critical joint states are subsets of joint states. Thus, $\mathcal{O}_{\text{CJSG}}$ is always no worse than $\mathcal{O}_{\text{JSG}}$.

B. Graph Construction Analysis

Our analysis of graph construction times for JSG and CJSG demonstrates CJSG's consistent efficiency advantage. This holds true across different conditions, such as varying node counts (from 10 to 30) and increasing risk edges ratios (from 1/5 to 1/2). Despite the advantage, when the risk edge ratio approaches 1/2 nearly all joint states become critical joint states i.e., $|\mathcal{M}| \to |\mathcal{V}|^2$. Thus, the construction times of both methods converge and are close to each other.

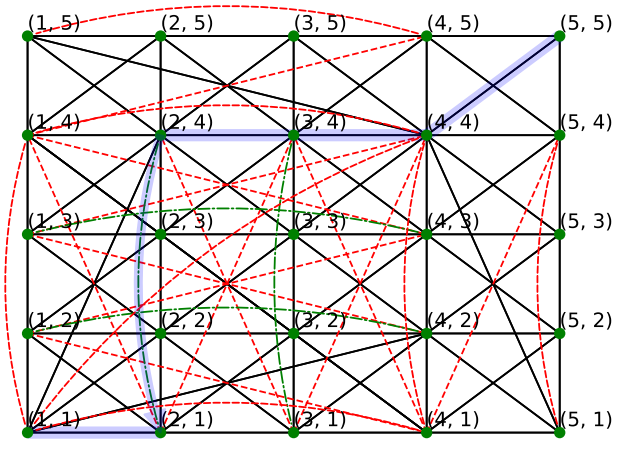

Fig. 2. Joint State Graph from the 5-node environment graph. Red (resp. green) edges represent traversing risk edge without (resp. with) support.

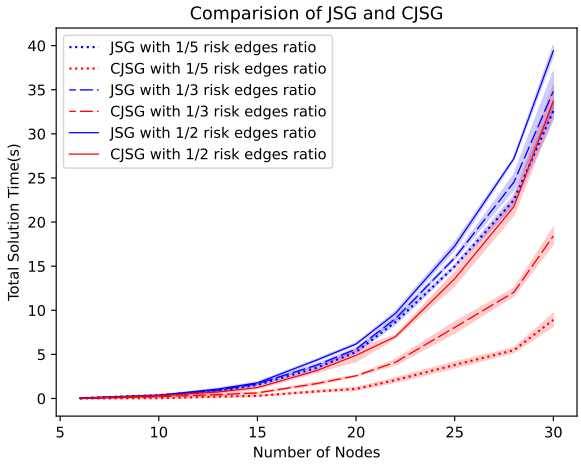

Fig. 3. Comparison of time taken by JSG and CJSG to generate total solution with respect to increasing number of nodes and risk edges ratio.

C. Path Planning Analysis

Our evaluation of path planning time for JSG and CJSG reveals CJSG's efficiency advantage across varying nodes and risk edges ratios. This advantage persists as the nodes increase from 10 to 30 or when the risk edges ratio is increased within a fixed node size. This pattern is consistent for nodes 10, 20, and 30. These results emphasize CJSG's superior efficiency in shortest path planning as the ratio of risk edges to nodes increases.

Based on experimental results, we calculate the total time taken by both JSG and CJSG to determine the final solution. This total time involves the time spent on graph construction and shortest path planning. In Fig. 3, as the number of nodes increases, the total solution generation time for JSG rises more significantly than for CJSG. Similarly, when the risk edges ratio increases, JSG's solution generation time grows more rapidly compared to CJSG. These findings indicate that CJSG is more efficient than JSG in overall solution generation.

V. CONCLUSION

We propose a novel formulation to study human-robot teaming in graph environments, where an agent can leverage "support" from its teammate to effectively reduce traversal costs on specific edges. Our method transforms multi-agent path planning problems into single-agent planning by incorporating agent actions into the edges in the joint state space. To address scalability, we introduced a hierarchical decomposition for path planning which is significantly more efficient in path planning. Future research directions include integrating advanced risk concepts from game theory and stochastic costs, as well as exploring scalability in terms of the number of agents in complex graph environments.