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

Toward Universal and Scalable Road Graph Partitioning for Efficient Multi-Robot Path Planning

Xingyao Han, Bo Cao, Zhe Liu, Shunbo Zhou, Heng Zhang and Hesheng Wang

***Abstract*— To date, multi-robot path planning has primarily been addressed by centralized solvers, typically aiming to maintain optimality. However, given its NP-hard nature, directly applying existing solvers in large and complex scenarios proves inefficient. A promising alternative lies in adopting a divide-and-conquer strategy to break down the problem into manageable sub-problems. In this work, we propose a systematic, universal, and scalable graph partitioning method, aiming to automatically divide any real-world environment into multiple regions. Building upon this, we convert the path planning on the entire graph into distributed sub-region path planning and devise corresponding inter-regional strategies. Our work can be easily implementable in practical systems and effectively enhances the scalability of existing solvers. Experimentally, our approach contributes to a tenfold improvement in computational efficiency while only sacrificing about 10% of optimality.**

I. Introduction

The Multi-Robot Path Planning (MRPP) problem refers to the coordination of multiple robots to plan collision-free paths to pre-assigned goals while minimizing the sum of path costs [1]. Researchers have demonstrated its wide applicability in various domains such as warehouse logistics, formation control, and video game development [2]–[4]. However, the scale of multi-robot systems in real-world scenarios has grown significantly, posing challenges for existing research in practical deployment. On the one hand, direct deployment of centralized optimal and sub-optimal MRPP solvers [5]–[7] relies on high-performance computing resources, which hinders their widespread adoption in large-scale systems. On the other hand, these solvers require substantial computational time when running on hardware devices, resulting in poor real-time responsiveness. In turn, congestion and deadlocks among robots are more likely to occur, ultimately reducing the overall operational efficiency.

In response to the above dilemma, a promising approach is to employ a divide-and-conquer strategy, decomposing the large-scale MRPP problem into a series of sub-problems [8]. However, existing research [8]–[12] rely on manual or partially automated map partitioning, lacking consideration for the generality. Furthermore, most of the above methods are confined to grid-like maps and have not been validated on more practical and general graph structures (also referred to as ***roadmaps*** [13]). This further exacerbates the gap between research and practical deployments.

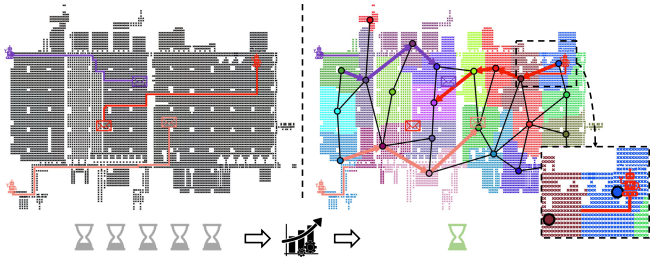


Fig. 1: We propose an automatic graph partitioning approach to divide any roadmaps into multiple regions and then transfer traditional centralized path planning (as shown in the left) into distributed path planning among and within sub-regions (as shown in the right), thus greatly reducing the planning time while maintaining the optimality.

In this work, as shown in Figure 1, we integrate graph theory to achieve universal and scalable partitioning of roadmaps. Then we employ a divide-and-conquer strategy to tackle large-scale MRPP problems, enhancing the scalability and practicality of existing MRPP solvers in real-world systems. Our main contributions are as follows:

i) We propose a systematic, universal and scalable method for partitioning road graphs in any real-world environment.

ii) Leveraging the partitioned graph, we introduce a divide-and-conquer-based deployment strategy to distribute the solving of MRPP into sub-problems within each subgraph.

iii) Simulation and real-world application results demonstrate a tenfold improvement in planning efficiency while only sacrificing 10% of optimality.

II. Related Works

A. Scalability of MRPP Solvers

Existing MRPP solvers such as Integer Linear Programming (ILP) [5], Hierarchical Cooperative A* (HCA*) [6], and Conflict-Based Search (CBS) [7] have demonstrated remarkable performance in simulated environments. Subsequently, numerous variants [14] of CBS have been proposed to enhance scalability to both the scale of robots and the size of the operational maps. Nonetheless, achieving a balance between minimizing computational time and maintaining optimality remains a significant challenge. Some studies [15]–[17] have significantly reduced the computational time by temporally decomposing the global plan into sub-plans. In contrast, we focus on spatial decomposition, i.e., *Aiming to improve the scalability of the MRPP solvers through road graph partitioning and divide-and-conquer strategies.*

This work was supported by the Natural Science Foundation of China under Grant 62303307 and the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102. Additionally, it was also supported by Huawei Cloud Computing Technologies Co., Ltd. under the Project 9450529. This work was done during Xingyao Han's internship at Huawei's Edge Cloud Innovation Lab.

X. Han, B. Cao, and Z. Liu are with the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China. S. Zhou and H. Zhang are with the Edge Cloud Innovation Lab, Huawei Cloud Computing Technologies Co., Ltd. Hesheng Wang is with the Department of Automation, Shanghai Jiao Tong University, China.

Corresponding authors: Zhe Liu and Hesheng Wang.

B. Spatial Partition-Enhanced MRPP Solvers

Spatial partition-enhanced MRPP solvers typically divide the workspace into regions and perform distributed planning within each region. An arbitrary partitioning strategy [9] is proposed and each partition is managed based on answer set programming (ASP). Another approach [10] automatically identifies high-contention regions (narrow corridors) and divides the workspace into numerous regions connected by these corridors. Then the corridors are assigned independent controllers, while the remaining regions are each allocated an HCA* [6] solver. Subsequent research [8], [11] divides the grid-based workspace into rectangular regions of similar sizes and incorporates a high-level planner for inter-region path planning. The above preliminary validations suggest the potential of spatial partitioning to enhance scalability [8]–[11]. However, the aforementioned methods are either based on specific assumptions about the map [10] or require manual partitioning to adapt to particular maps [8], resulting in a lack of adaptability to diverse operational environments. The lack of general partitioning poses a challenge in addressing the complexities inherent in various real-world scenarios. *Automating universal and scalable graph partitioning for real-world environments remains an open challenge*.

C. Graph Partitioning in Graph Theory

In graph theory, ***Graph Partitioning*** involves dividing the nodes of a graph into multiple subgraphs for reduction [18], which is NP-hard [19]. Typically, algorithms condense the input graph iteratively until convergence to generate partitions [20]. Additionally, multi-level partitioning methods start by condensing nodes and edges to reduce graph size, then partition the smaller graph, and finally expand to obtain the original graph's partitioning [21]. *In this work, we make the first step to demonstrate the power of automatic graph partitioning to efficient MRPP.*

III. Problem Formulation

A. Preliminaries on Graph Theory

In this section, we summarize the main notions of graph theory throughout the paper. Further details can be found in [18]. We denote graph as $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ represent the sets of nodes and edges, respectively. Additionally, $|\mathscr{V}|$ and $|\mathscr{E}|$ are used to denote the cardinality of nodes and edges. Meanwhile, $\forall u \in \mathscr{V}, e = (u,v) \in \mathscr{E}$, $w(u)$ and $w(u,v)$ denote the weight of nodes and edges, respectively.

Definition 1: We define $\mathscr{H} = (\mathscr{V}', \mathscr{E}')$ to be a ***subgraph*** of $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ if and only if:

$$i)\mathscr{V}' \subset \mathscr{V};\ ii)\mathscr{E}' \subset \mathscr{E};\ iii)\forall e' = (u,v) \in \mathscr{E}',\ u,v \in \mathscr{V}'. \quad (1)$$

Definition 2: We define subgraph $\mathscr{M}$ to be a ***matching*** of $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where each node has a degree of at most 1. A matching consists of edges that do not share nodes. $\mathscr{M}$ is a ***maximal matching*** if no other edge in $\mathscr{G}$ can be added.

Definition 3: We define $\mathscr{C} = (\mathscr{G}_1, \mathscr{G}_2)$ to be a ***cut*** of $\mathscr{G}$, splitting $\mathscr{V}$ into two node subsets $\mathscr{V}_1$ and $\mathscr{V}_2$. The ***cut-set*** $\mathscr{E}_c$ related to $\mathscr{C}$ is defined as $\mathscr{E}_c = \{(u,v) \in \mathscr{E} | u \in \mathscr{V}_1, v \in \mathscr{V}_2\}$, denoted as $\mathscr{E}_c = \text{cut}(\mathscr{G}_1, \mathscr{G}_2)$. The weight of the ***cut-set*** $\mathscr{E}_c$ is defined as $w(\mathscr{E}_c) = \sum_{e \in \mathscr{E}_c} w(e)$.

Definition 4: We define $(\mathscr{V}_1, \mathscr{V}_2, \cdots, \mathscr{V}_k)$ to be a ***k-partition*** of $\mathscr{V}$ (denoted as $\mathscr{P}$) if and only if:

$$i)\bigcup_{i=1}^{k} \mathscr{V}_i = \mathscr{V};\quad ii)\mathscr{V}_i \cap \mathscr{V}_j = \emptyset. \quad (2)$$

The ***partition matrix*** P corresponding to $\mathscr{P}$ satisfies $1 \leq P[u] \leq k, P[u] \in \mathbb{Z}$, indicating u belongs to $\mathscr{V}_{P[u]}$.

B. Multi-Robot Path Planning

In a transportation system, the environment is commonly modeled as a road graph, often referred to as a ***roadmap*** $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. The roadmap represents all possible locations that robots can access, which can be obtained through automatic generation [13] or manual design. Each v_i on $\mathscr{G}$ represents a location in the environment, which can be categorized as robot stations, pickup stations, work stations, and traversable free areas, among others [12]. Meanwhile, $e = (v_i, v_j)$ defines the adjacency relationship between nodes. At timestep t, the position of r_i is distributed over $\mathscr{V}$, denoted as $q_i^t = v_i \in \mathscr{V}$. Each r_i is assigned a start node $v_{s,i} = q_i^0$ and a goal node $v_{g,i} = q_i^T$. Thus, for $0 \leq t \leq T$, r_i has a path from $v_{s,i}$ to $v_{g,i}$, denoted as $p_i = [q_i^0, q_i^1, q_i^2, \ldots, q_i^T]$. Let the set of robots denoted as $\mathscr{R} = \{r_i | i = 1,2,\ldots,N_r\}$, and let the sets of start nodes and goal nodes denoted as $\mathscr{V}_s = \{v_{s,i} | i = 1,2,\ldots,N_r\}$ and $\mathscr{V}_g = \{v_{g,i} | i = 1,2,\ldots,N_r\}$, respectively. The MRPP problem can thus be described as a triple $Q = (\mathscr{G}, \mathscr{V}_s, \mathscr{V}_g)$. The objective is to output the cooperative paths $\mathscr{D} = \{p_i = [q_i^0, q_i^1, q_i^2, \ldots, q_i^T] | i = 1,2,\ldots,N_r\}$, such that:

$$i)\ q_i^t \neq q_j^t;\quad ii)\ (q_i^{t-1}, q_i^t) \neq (q_j^t, q_j^{t-1}). \quad (3)$$

where *i)* and *ii)* respectively ensure the absence of node conflicts and edge conflicts in the paths.

C. Graph Partitioning

In this work, we aim to construct a graph partitioner $\mathbb{P}$ to automatically partition the roadmap $\mathscr{G}$ into N_s subgraphs $\mathscr{S} = \{\mathscr{G}_{x,j} | j = 1,2,\cdots,N_s\}$. Then the large-scale MRPP problem $(\mathscr{G}, \mathscr{V}_s, \mathscr{V}_g)$ is transformed into a series of smaller MRPP problems $\mathscr{Q} = \{(\mathscr{G}_{x,j}, \mathscr{V}_{s,j}, \mathscr{V}_{g,j}) | j = 1,2,\ldots,N_s\}$. How the subgraphs are partitioned significantly impacts the efficiency and performance of solving MRPP sub-problems. Therefore, we focus on the following aspects:

a) Load Balancing entails minimizing the variance of $|\mathscr{V}_{x,j}|$ within each $\mathscr{G}_{x,j}$, expressed as:

$$\min_{\mathscr{S}} f_{\text{load}} = \text{Var}_{j=1}^{N_x}(|\mathscr{V}_{x,j}|). \quad (4)$$

Note: By balancing node distribution among subgraphs and leveraging CPU parallel-solving capabilities, computational resource utilization is maximized, thereby enhancing the overall problem-solving efficiency.

b) Minimizing Cut-Set Weight is a commonly used object in graph partitioning. It can be expressed as:

$$\min_{\mathscr{S}} f_{\text{cut}} = \sum_{i=1}^{k} \sum_{j=1}^{k} w(\text{cut}(\mathscr{G}_{x,i}, \mathscr{G}_{x,j})), i \neq j. \quad (5)$$

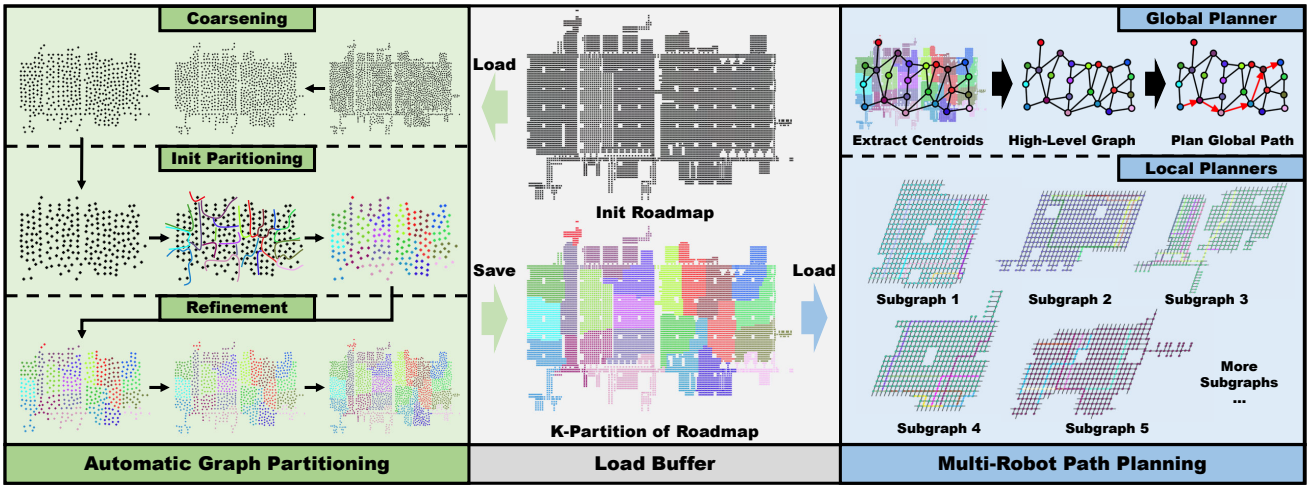


Fig. 2: Our architecture is primarily divided into offline graph partitioning (left) and online multi-robot path planning (right). In graph partitioning, we load the init roadmap $\mathscr{G}$, partition it, and ultimately save the partitioning results $\mathscr{S}$. In multi-robot path planning, we extract the subgraph centroids and construct a high-level topological graph $\mathscr{G}_s$. During system execution, the high-level planner $\mathbb{S}_g$ performs global planning while each subgraph planner $\mathbb{S}_{x,j}$ executes local cooperative planning.

Note: The smaller the cut-set weight, the fewer costs cross the subgraphs, thereby decreasing the communication and computational overhead across subgraphs[1]. This enhances partitioning quality and solving efficiency on sub-problems.

IV. Main Approach

A. System Framework

As in Figure 2, our approach mainly consists of two parts:

In the **graph partitioning stage**, the graph partitioner $\mathbb{P}$ automatically identifies structural features of $\mathscr{G}$ and divides it into several subgraphs $\mathscr{S} = \{\mathscr{G}_{x,j} | j = 1,2,\ldots,N_x\}$, i.e., $\mathscr{S} \leftarrow \mathbb{P}(\mathscr{G})$. Then, with each $\mathscr{G}_{x,j}$ considered as a node, we can construct a high-level graph $\mathscr{G}_s$ satisfying $|\mathscr{V}_s| = |\mathscr{S}|$. In this way, inter-subgraph search is described by $\mathscr{G}_s$, while intra-subgraph search is described by $\mathscr{G}_{x,j}$.

In the **multi-robot path planning stage**, $\mathscr{G}_{x,j}$ is assigned an MRPP solver $\mathbb{S}_{x,j}$ to plan intra-subgraph collaborative paths. Specifically, robots within $\mathscr{G}_{x,j}$ (denoted as $\mathscr{R}_j = \{r_1, r_2, \cdots, r_k\}$) form the MRPP sub-problem $(\mathscr{G}_{x,j}, \mathscr{V}_{s,j}, \mathscr{V}_{g,j})$ by the set of start and goal nodes within $\mathscr{G}_{x,j}$ (denoted as $\mathscr{V}_{s,j}, \mathscr{V}_{g,j}$). Then, $\mathbb{S}_{x,j}$ solves the sub-problem and generates collision-free collaborative paths $\mathscr{D}_j \leftarrow \mathbb{S}_{x,j}(\mathscr{G}_{x,j}, \mathscr{V}_{s,j}, \mathscr{V}_{g,j})$ within $\mathscr{G}_{x,j}$. During system execution, each r_i dynamically updates its associated subgraph $\mathscr{G}_{x,j}$ to ensure clear objectives for the sub-problem by $\mathscr{G}_s$. Meanwhile, inter-subgraph planning hinges on communication interactions between subgraphs. With the edges connecting subgraphs denoted as e_{con}, it is evident that $e_{con} \in \mathscr{E}_{\text{cut}}$. The objective generation within subgraphs and the communication interaction between subgraphs are both elaborated in Section IV-C.

B. Automatic Graph Partitioning

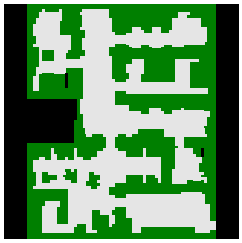
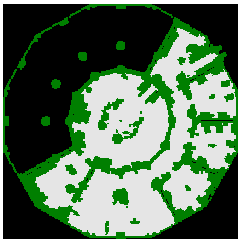
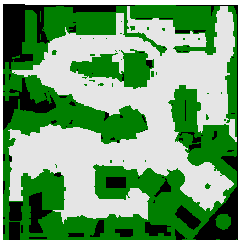
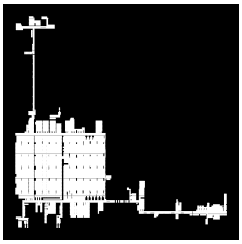
In this section, we partition the roadmap into a series of subgraphs based on the graph partitioner $\mathbb{P}$. Inspired by the METIS framework [19], we develop $\mathbb{P}$ with three phases: coarsening, initial partitioning, and refinement.

In the ***coarsening phase***, we obtain a set of smaller graphs $\mathscr{G}_i = (\mathscr{V}_i, \mathscr{E}_i)$ from $\mathscr{G}_0 = \mathscr{G} = (\mathscr{V}_0, \mathscr{E}_0)$ by merging vertices, where $|\mathscr{V}_{i+1}| < |\mathscr{V}_i|$. Specifically, a group of nodes $\mathscr{V}_i^u$ in $\mathscr{G}_i$ is merged into a single node u in $\mathscr{G}_{i+1}$, with the weight of u set as the sum of weights of nodes in $\mathscr{V}_i^u$. In MRPP, nodes in $\mathscr{G}$ have no weights, thus each node's weight is set to 1, i.e., $w(u) = |\mathscr{V}_i^u|$. To preserve connectivity, the edges associated with u are the edges in $\mathscr{V}_i^u$. If multiple nodes in $\mathscr{V}_i^u$ have edges pointing to another node v in $\mathscr{G}_{i+1}$, the edge weight from u to v is the sum of weights of these edges. We employ ***matching*** to merge nodes and generate $\mathscr{G}_{i+1}$. The essence of coarsening lies in obtaining maximal matching. However, finding maximal matching is highly inefficient, with a time complexity of $O(|\mathscr{V}||\mathscr{E}|^2)$ [22]. Hence, our heuristic approach is based on random traversal and heavy-edge matching [23] to construct a sub-optimal matching. Specifically, we randomly traverse unmatched node u and find v among unmatched neighbors of u such that $w(u,v)$ is maximized. The above greedy algorithm has a time complexity of $O(|\mathscr{E}|)$, albeit at the cost of obtaining a sub-optimal matching. The coarsening phase has the following properties: 1) the weight of the cut-set in the coarsened graph equals that of the original graph, and 2) the balancing partition in the coarsened graph leads to a similarly balanced partition in the original graph.

In the ***initial partitioning phase***, we perform a k-partition on the final coarsened graph $\mathscr{G}_m$ (where m is the number of coarsening iterations). As the weights of nodes and edges in $\mathscr{G}_m$ reflect those in $\mathscr{G}$, we only need to compute $\mathscr{P}_m$ such that the sum of node weights in each partitioned subgraph is close to $\frac{|\mathscr{V}_0|}{k}$. Specifically, we arbitrarily select u in $\mathscr{G}_m$ and then perform breadth-first search (BFS) traversal. When the sum of node weights exceeds $\frac{|\mathscr{V}_0|}{k}$, we obtain a subgraph and repeat the above process until obtaining $\mathscr{P}_m$.

[1]In this configuration, the system can implement a simple queuing strategy [12], enabling robots to cross regions sequentially. This approach enhances the overall order and efficiency of the planning system.

TABLE I. EVALUATION ON VARIOUS ENVIRONMENTS

Map		**Benchmark DAO (den312d)**		**Benchmark DAO (ost003d)**		**Benchmark DAO (den520d)**		**Actual Factory**	
Layout									
Map Size		**Grid** with 65×81 shape; 2,445 states.		**Grid** with 194×194 shape; 13,214 states.		**Grid** with 256×257 shape; 28,178 states.		**Graph** with 10,155 nodes; 17,906 edges.	
Metrics $\downarrow$		**Runtime(s)**	**Optimality Ratio**	**Runtime(s)**	**Optimality Ratio**	**Runtime(s)**	**Optimality Ratio**	**Runtime(s)**	**Optimality Ratio**
Subgraph Number k	**1**	3.099	1.000	41.395	1.000	161.160	1.000	34.249	1.000
	5	1.680	1.173	28.548	1.225	53.269	1.215	10.433	1.268
	10	0.965	1.143	12.114	1.121	16.487	1.100	9.429	1.163
	15	0.695	1.142	6.772	1.096	8.513	1.083	5.886	1.150
	20	0.545	1.139	4.008	1.079	5.615	1.080	3.383	1.152
	25	0.348	1.153	3.513	1.089	4.470	1.102	2.731	1.126
	30	0.266	1.123	2.733	1.074	3.641	1.056	2.160	1.096

In the ***refinement phase***, we reconstruct $\mathscr{P}$ of the original graph $\mathscr{G}$ by backtracking the coarsened partition $\mathscr{P}_m$. During the process, each level of coarsening $\mathscr{G}_{m-1}, \mathscr{G}_{m-2}, \ldots, \mathscr{G}_1$ is backtracked to $\mathscr{G}$. Since each u in $\mathscr{G}_{i+1}$ corresponds to a set of nodes $\mathscr{V}_i^u$ in $\mathscr{G}_i$, restoring $\mathscr{P}_i$ from $\mathscr{P}_{i+1}$ involves assigning the nodes in $\mathscr{V}_i^u$ to the subgraph of u in $\mathscr{P}_{i+1}$:

$$\forall v \in \mathscr{V}_i^u, \quad P_i[v] = P_{i+1}[u]. \tag{6}$$

It is worth mentioning that $\mathscr{P}$ is prone to erosion[2], which can lead to severely degraded results. To address the challenge, heuristic approaches for local refinement can be employed to optimize $\mathscr{P}_i$ while restoring from $\mathscr{P}_{i+1}$. In each refinement, the cut-set weight is further minimized by node permutation [19], thus mitigating the impact of coarsening and refinement. Through a series of refinements $\mathscr{P}_m, \mathscr{P}_{m-1}, \ldots, \mathscr{P}_0$, we obtain the partition $\mathscr{P}_0$ corresponding to $\mathscr{G}$. To further overcome the erosion issue and achieve uniform partitioning, we can iteratively apply the bisection mode of the entire algorithm. Specifically, we set $k = 2$ each time the partitioning is executed and obtain k subgraphs by iterative partitioning.

Finally, we obtain $\mathscr{P}_0$ and construct a series of subgraphs from $\mathscr{G}$. For each $\mathscr{G}_{x,j}$, we take the centroids as a node u_j. Furthermore, the cut-sets between two subgraphs $\mathscr{G}_{x,j}$ and $\mathscr{G}_{x,k}$ form edges between the nodes u_j and u_k. In this way, we construct a high-level graph $\mathscr{G}_s$, where nodes represent subgraphs and edges represent cut-sets between subgraphs. The edges in $\mathscr{G}_s$ facilitate robot movement between subgraphs, while the edges in $\mathscr{G}_{x,j}$ facilitate movement within subgraphs. As a result, we decompose the entire MRPP problem into intra-subgraph and inter-subgraph sub-problems.

[2]Erosion refers to a phenomenon where the shape of a subgraph becomes highly irregular, with many nodes extending outward to form protrusions. In such cases, the cut-set weight becomes significantly large.

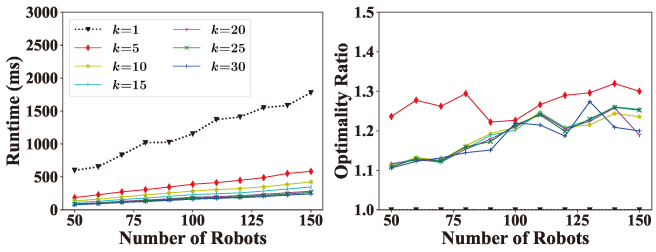
C. Graph Partitioning Enhanced Multi-Robot Path Planning

During system execution, the planning module loads $\mathscr{P}_0$ and converts the original roadmap $\mathscr{G}$ into an high-level graph $\mathscr{G}_s$ and a set of low-level subgraphs $\{\mathscr{G}_{x,j} | j = 1, 2, \cdots, N_s\}$. Then, the MRPP solver on $\mathscr{G}$ consists of global planner $\mathbb{S}_g$ on $\mathscr{G}_s$ and local planners $\mathbb{S}_{x,j}$ on $\mathscr{G}_{x,j}$, used respectively for path planning between and within subgraphs.

Global Planner $\mathbb{S}_g$ serves as the top-level path planner, responsible for generating global paths. Since $\mathscr{G}_s$ does not involve actual robot motion, collision avoidance between them is not a concern. Typically, $\mathbb{S}_g$ is configured with a single-robot planner. Meanwhile, traffic flow can also be introduced to potentially avoid conflicts, as detailed in our previous work [8]. $\mathbb{S}_g$ plans subgraph-level path $[\mathscr{G}_{x,i_1}, \mathscr{G}_{x,i_2}, \cdots, \mathscr{G}_{x,i_k}]$ for each r_i. Consequently, r_i can obtain the communication edge $e_{\text{con}} = (v_{i_1}, v_{i_2})$ between the current subgraph $\mathscr{G}_{x,i_1}$ and the next subgraph $\mathscr{G}_{x,i_2}$, where the start node v_{i_1} of the edge e_{con} corresponds to the temporary goal node $v_{g,i}$ within $\mathscr{G}_{x,i_1}$. By iterating through $\mathscr{R}_j$, we obtain the sets of start nodes $\mathscr{V}_{s,j}$ and goal nodes $\mathscr{V}_{g,j}$ within the subgraph. Subsequently, we construct the MRPP sub-problem $(\mathscr{G}_{x,j}, \mathscr{V}_{s,j}, \mathscr{V}_{g,j})$. This sub-problem is further solved by the local planner $\mathbb{S}_{x,j}$.

Local Planner $\mathbb{S}_{x,j}$ serves as the low-level path planner, generating cooperative paths for robots. Each $\mathscr{G}_{x,j}$ is assigned a local planner $\mathbb{S}_{x,j}$ to solve the MRPP sub-problem, i.e., $\mathscr{P}_j \leftarrow \mathbb{S}_{x,j}(\mathscr{G}_{x,j}, \mathscr{V}_{s,j}, \mathscr{V}_{g,j})$. $\mathbb{S}_{x,j}$ can be configured as any MRPP solver. In this work, we choose HCA* [6]:

1) For initialization, we directly use HCA* to solve the MRPP sub-problem. If two robots $r_{1,2}$ have the same goal node u, we only need to execute two-sided space-time A*. The last states of $r_{1,2}$ within $\mathscr{G}_{x,j}$ are (u, t_1) and (u, t_2), respectively, ensuring collision-free paths. The one-shot planning problem is also solved using a similar approach.



(a) Runtime (b) Optimality Ratio

Fig. 3: Evaluations with different robot numbers.

2) For lifelong planning, robots may enter or exit the subgraph. Exiting robots have their related reservations removed, while entering robots undergo independent space-time A* execution to find collision-free paths with running robots. Failures and motion uncertainties prompt the introduction of wait nodes as temporary intermediate nodes [24]. In addition, we also introduce the memory mechanism to reuse the previous local planning and motion coordination experiences to enable the lifelong learning ability of the local planner.

Remark: Our approach enables dynamic and flexible MRPP sub-problem solving on subgraphs instead of entire graphs, thus saving time and allowing scalability to massive maps.

V. Evaluations on Benchmarks and Actual Factory Roadmaps

In this section, we evaluate our efficiency and performance using existing benchmark maps[3], in which we select three *DAO* maps (den312d, ost003d, den520d) for testing, with details on their parameters available in [25]. Furthermore, we include a map of an actual factory, which has a more general graph structure, comprising 10,155 nodes and 17,906 edges, with different weights assigned to each edge. The main layout of these maps can be seen in Table I. In each scenario, we repeat 25 independent tests and take the average as the experimental result. For each test, we set the same random seed to ensure consistency across test cases under different graph partitions. We evaluate ***runtime*** and ***optimization ratio***, where *runtime* refers to the average planning time per frame during the system's actual operation and *optimization ratio* is calculated based on the *Sum of Cost* (*SoC*) [25], generated as follows: the result obtained by solving directly on the entire map serves as the baseline SoC_0, and the *SoC* obtained under different partitioning is divided by SoC_0 to obtain the *optimization ratio*. We respectively choose A* and HCA* [6] as the high-level and low-level planners. Moreover, our graph partitioning is implemented based on the *nxmetis*[4] library. The partitioning results are saved offline. All experiments are executed at Intel(R) Core(TM) i5-13600k 3.50GHz.

A. Generalizability to Various Environments

To evaluate the impact of graph partitioning on the MRPP solver and demonstrate our generalizability, we conducted experiments on four maps without partitioning ($k = 1$) and with partitioning ($k = 5, 10, 15, 20, 25, 30$). The results shown in Table I lead to the following conclusions: 1) With an increasing number of subgraphs, the runtime significantly decreases. When $k = 30$, runtime on the four maps is only 8.58%, 6.60%, 2.26%, and 6.31% compared to planning on the origin map; 2) Graph partitioning leads to a decrease in the optimality of the solution, especially noticeable under a smaller k such as $k = 5$. However, as the number of subgraphs increases, the degradation in optimality is effectively mitigated. When $k = 30$, the decay in optimality ratio is only about $7\% \sim 12\%$; 3) Both runtime and optimality ratio show signs of maturing as subgraph number continues to rise; 4) The benchmark maps comprise grid maps, while the actual factory map is a graph map. In addition, they have very different roadmap sizes. Nonetheless, the efficiency and performance trends are similar across the four maps. This indicates that our partitioning method can adapt to different map formats, providing more generalized results.

Disscussion: As k increases, considering the limiting case where $k = |\mathcal{V}|$, each subgraph contains only a single node, resulting in $\mathcal{G}_s$ being identical to $\mathcal{G}$, with each $\mathcal{G}_{x,j}$ corresponding to an individual node. Then, the hierarchical planner degenerates into the upper-level single-robot path planner, rendering it incapable of accomplishing the task. Therefore, the selection of k is crucial and should not be arbitrarily large. Our preliminary results indicate that the optimal selection of k is influenced by factors such as map topology, robot configuration, and task distribution. A future improvement is to enable the automatic selection of the optimal or a bounded suboptimal k for a given configuration.

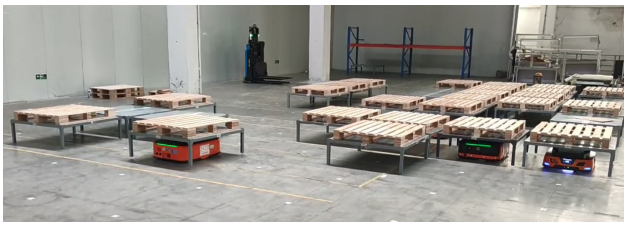
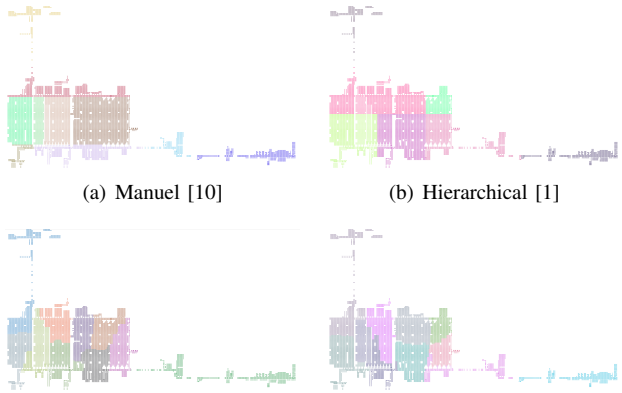


Fig. 4: Real-world applications on an actual factory at Huawei.



(a) Manuel [10] (b) Hierarchical [1]

(c) METIS [19] (d) Ours

Fig. 5: Partitioning results of different approaches.

B. Scalability to Various Robot Number Scales

In this section, we evaluate our efficiency and optimality under various subgraph numbers k and robot numbers. The

[3]https://movingai.com/benchmarks/mapf/index.html

[4]https://github.com/networkx/networkx-metis/tree/main

TABLE II. COMPARISONS IN REAL-WORLD APPLICATIONS

Test ID	Metrics	[1]	[10]	[19]	Ours
1	Runtime(s)	1.195	0.923	0.941	**0.827**
	Optimality Ratio	1.406	1.223	1.197	**1.161**
2	Runtime(s)	1.158	0.918	0.916	**0.835**
	Optimality Ratio	1.370	1.269	1.184	**1.065**
3	Runtime(s)	1.096	0.847	0.869	**0.836**
	Optimality Ratio	1.501	1.206	1.175	**1.073**

experimental results, depicted in Figure 3, lead to the following conclusions: 1) Regarding efficiency, as the number of subgraphs increases, the runtime decreases for any robot scales. Meanwhile, runtime grows linearly with the increase in the number of robots, and the growth rate is much lower than that without partitioning. Furthermore, with a larger number of subgraphs, the slope of the runtime to the robot number becomes smaller. 2) Regarding optimality, when $k = 5$, the optimality of planning is poorest. For $k > 10$, the optimality shows similar performance (the same as the saturation effect shown in the previous section). In addition, the optimality ratio also shows signs of maturing as robot numbers continue to rise. 3) Different from runtime, there is significant fluctuation in optimality ratio with the increase of robot number, due to the randomness and system instability under large robot numbers (more potential motion conflicts).

VI. REAL-WORLD APPLICATIONS

As depicted in Figure 4, we implement real-world applications comprising 100 robots through three independent tests. We compare our approach with existing graph partitioning methods (Manuel[5], Hierarchical [1], and native METIS [19]) alongside our algorithm in iterative bisection mode. The graph partitioning results of different approaches are shown in Figure 5 and the experimental results are presented in Table II. Compared to existing approaches, our method has the smallest runtime while keeping the optimality ratio best. Unlike manual operations in existing works, our method exhibits better flexibility (allowing arbitrary numbers of partitions) and automatically achieves partitioning, thus reducing implementation costs in practical deployment.

VII. CONCLUSIONS

In this paper, we propose a systematic, universal, and scalable approach to road graph partitioning within the MRPP domain, demonstrating its potential to reduce planning time while preserving optimal performance. Future work will focus on automating the selection of k and exploring time-varying graph partitioning using Graph Neural Networks to better adapt to complex and dynamic scenarios.

REFERENCES

[1] Z. Liu, H. Wang, H. Wei, M. Liu, and Y.-H. Liu, "Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties," *IEEE Transactions on Automation Science and Engineering*, pp. 1705–1717, 2020.

[5]Manual refers to graph partition designed by professional engineers similar to [10], where corridors are extracted and areas are divided.

[2] X. Xiong, X. Han, Z. Liu, and H. Wang, "Exhaustiveness does not necessarily mean better: Selective task planning for multi-robot systems," in *2023 IEEE International Conference on Robotics and Biomimetics*, pp. 1–6, IEEE, 2023.

[3] Z. Liu, W. Chen, J. Lu, H. Wang, and J. Wang, "Formation control of mobile robots using distributed controller with sampled-data and communication delays," *IEEE Transactions on Control Systems Technology*, pp. 2125–2132, 2016.

[4] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, pp. 6932–6939, 2020.

[5] J. Yu and S. LaValle, "Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics,IEEE Transactions on Robotics*, Jul 2015.

[6] D. Silver, "Cooperative pathfinding," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 117–122, 2005.

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

[8] X. Han, S. Chen, X. Xiong, Q. Liu, S. Zhou, H. Zhang, and Z. Liu, "Traffic flow learning enhanced large-scale multi-robot cooperative path planning under uncertainties," in *2024 IEEE International Conference on Robotics and Automation*, pp. 16581–16587, IEEE, 2024.

[9] P. Pianpak, T. C. Son, P. O. Toups Dugas, and W. Yeoh, "A distributed solver for multi-agent path finding problems," in *First International Conference on Distributed Artificial Intelligence*, Oct 2019.

[10] C. Wilt and A. Botea, "Spatially distributed multiagent path planning," *International Conference on Automated Planning and Scheduling*, p. 332–340, Sep 2022.

[11] H. Zhang, M. Yao, Z. Liu, J. Li, L. Terr, S.-H. Chan, T. K. S. Kumar, and S. Koenig, "A hierarchical approach to multi-agent path finding," *International Symposium on Combinatorial Search*, p. 209–211, 2022.

[12] Z. Liu, H. Wei, H. Wang, H. Li, and H. Wang, "Integrated task allocation and path coordination for large-scale robot networks with uncertainties," *IEEE Transactions on Automation Science and Engineering*, pp. 2750–2761, 2021.

[13] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "An automatic approach for the generation of the roadmap for multi-agv systems in an industrial environment," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1736–1741, IEEE, 2014.

[14] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *International Symposium on Combinatorial Search*, pp. 19–27, 2014.

[15] E. Ephrati and J. S. Rosenschein, "Divide and conquer in multi-agent planning," in *AAAI*, p. 80, 1994.

[16] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, pp. 1163–1177, 2016.

[17] T. Guo, S. D. Han, and J. Yu, "Spatial and temporal splitting heuristics for multi-robot motion planning," in *2021 IEEE International Conference on Robotics and Automation*, May 2021.

[18] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2001.

[19] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, pp. 96–129, 1998.

[20] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science*, Jan 2006.

[21] G. Karypis and V. Kumar, "Multilevel k -way hypergraph partitioning," in *36th ACM/IEEE Design Automation Conference*, Jun 1999.

[22] N. Zadeh, "Theoretical efficiency of the edmonds-karp algorithm for computing maximal flows," *Journal of the ACM*, pp. 184–192, 1972.

[23] G. Karypis and V. Kumar, "Analysis of multilevel graph partitioning," in *1995 ACM/IEEE conference on Supercomputing*, pp. 29–es, 1995.

[24] C. Leet, J. Li, and S. Koenig, "Shard systems: Scalable, robust and persistent multi-agent path finding with performance guarantees," *AAAI Conference on Artificial Intelligence*, p. 9386–9395, Jul 2022.

[25] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *International Symposium on Combinatorial Search*, pp. 151–158, 2019.