Efficient Multi-Robot Task and Path Planning in Large-Scale Cluttered Environments

Gang Xu, Yuchen Wu, Sheng Tao, Yifan Yang, Tao Liu, Tao Huang, Huifeng Wu, and Yong Liu

***Abstract*—As the potential of multi-robot systems continues to be explored and validated across various real-world applications, such as package delivery, search and rescue, and autonomous exploration, the need to improve the efficiency and quality of task and path planning has become increasingly urgent, particularly in large-scale, obstacle-rich environments. To this end, this letter investigates the problem of multi-robot task and path planning (MRTPP) in large-scale cluttered scenarios. Specifically, we first propose an obstacle-vertex search (OVS) path planner that quickly constructs the cost matrix of collision-free paths for multi-robot task planning, ensuring the rationality of task planning in obstacle-rich environments. Furthermore, we introduce an efficient auction-based method for solving the MRTPP problem by incorporating a novel memory-aware strategy, aiming to minimize the maximum travel cost among robots for task visits. The proposed method effectively improves computational efficiency while maintaining solution quality in the multi-robot task planning problem. Finally, we demonstrated the effectiveness and practicality of the proposed method through extensive benchmark comparisons.**

***Index Terms*—Multi-robot systems, task planning, path planning, auction mechanism.**

I. Introduction

MULTI-ROBOT task and path planning (MRTPP) problem aims to enhance the collaborative capabilities of multi-robot systems, enabling a fleet of robots to perform a series of tasks in the shortest possible travel distance or time in real-world applications, such as package delivery [1], target reconnaissance [2], autonomous exploration [3], search and rescue [4], and power inspection [5]. In these applications, the environment is often cluttered with numerous obstacles. Decoupling task planning and path planning—by first performing task planning without collision constraints, then generating collision-free paths for the assigned tasks—can lead to higher task execution costs for robots. To ensure high-quality solutions for the MRTPP problem, it is crucial to consider the coupling between task planning and path planning [6]. However, as the number of robots and tasks increases, constructing the cost matrix for collision-free paths becomes computationally expensive, particularly in large-scale environments with dense obstacles. Thus, achieving a trade-off between solution quality and computational efficiency remains a significant challenge in the MRTPP problem.

Many methods have been proposed for the MRTPP problem. For example, Camisa et al. [7] solved a combined task and path planning problem using a distributed primal decomposition approach for package delivery applications. Xu et al. [8] addressed the MRTPP problem in dense environments by separately solving both the task and motion planning problems. However, the above methods decouple task planning and path planning, which may increase the task execution costs for robots in environments with complex obstacles. In contrast, many existing approaches consider their coupling. For instance, Liu et al. [9] designed an integrated optimization method to minimize the total traveling cost and potential path conflicts. Jin et al. [10] proposed a method based on differential dynamic programming to solve the integrated task allocation and trajectory optimization problem. In addition, some methods address the MRTPP problem by modeling it as a capacitated vehicle routing problem (CVRP) [11] or a multiple traveling salesman problem (mTSP) [12]. However, constructing a collision-free cost matrix using these methods is computationally expensive. In practical applications, ensuring real-time performance often requires reducing the number of robots and tasks. As the environment grows or the number of agents increases, the computational burden escalates, making real-time matrix construction increasingly challenging. Moreover, to simplify the MRTPP problem, most methods formulate the objective to minimize the total travel cost of all robots rather than the maximum individual cost, which better aligns with real-world needs. However, this simplification inevitably increases individual travel costs.

To address these issues, we first propose an **O**bstacle-**V**ertex **S**earch (OVS) path planner that efficiently constructs the cost matrix of collision-free paths between tasks and robots in large-scale, cluttered environments. Moreover, we formulate the objective function to minimize the maximum travel cost for robots in executing tasks and introduce an efficient auction-based method for solving the MRTPP problem by incorporating a novel memory-aware strategy. We evaluate the proposed method using extensive MRTPP instances and make detailed comparisons with

Received 22 April 2025; accepted 11 July 2025. Date of publication 23 July 2025; date of current version 31 July 2025. This article was recommended for publication by Associate Editor G. Pereira and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported by theNational Natural Science Foundation of China under Grant U21A20484. *(Corresponding author: Yong Liu.)*

Gang Xu, Sheng Tao, Tao Huang, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: wuuya@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Yuchen Wu and Yifan Yang are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China, and also with the Polytechnic Institute of Zhejiang University, Hangzhou 310015, China.

Tao Liu is with the Ocean College, Zhejiang University, Zhoushan 316021, China.

Huifeng Wu is with the Institute of Intelligent and Software Technology, Hangzhou Dianzi University, Hangzhou 310018, China.

Digital Object Identifier 10.1109/LRA.2025.3592146

current state-of-the-art (SOTA) solvers. The experimental results show that our method outperforms others in terms of solution quality and computational efficiency.

The main contributions are summarized as follows:

1) A fast path planner suitable for large-scale and cluttered workspaces that efficiently constructs the cost matrix of collision-free paths between tasks and robots for solving the MRTPP problem.
2) An efficient auction-based method for solving the MRTPP problem by incorporating a novel memory-aware strategy, aiming to minimize the maximum travel cost for robots to visit tasks.
3) The source code of our method, along with the compared SOTA solvers, is released to benefit the community.[1]

II. Related Works

One common method for solving the MRTPP problem is to decouple the task and path planning and solve them separately. Among the approaches, heuristic methods [13], [14], [15] are the most popular ones. They improve task planning efficiency and solutions quality by designing appropriate heuristic functions. Additionally, many methods [16], [17] employ the mTSP or CVRP formulations to solve the problem. Furthermore, auction-based methods [18], [19] have also attracted significant attention from researchers due to their computation efficiency. However, these methods plan tasks using a centralized server communicating with all robots. As the number of robots and tasks increases, computational complexity grows exponentially, making deployment in real-world applications difficult. In contrast, decentralized methods can significantly reduce computational complexity. Among them, the most representative is auction-based methods, such as [20], [21], [22], [23], which dynamically iterate the allocation results based on rewards, significantly saving computation time. In addition, due to the high computational efficiency, deep reinforcement learning (DRL) methods [24], [25], [26] have also been widely studied for solving the MRTPP problem. After task planning, classic path planners like A* [27], jump point search (JPS) [28], and RRT* [29] are often used to find safe paths for robots to execute tasks. At the same time, some methods [8], [30] also improve the computational efficiency of path planning. However, decoupled methods tend to increase the cost of task execution in environments with cluttered obstacles [7], [8].

Compared to decoupled methods, approaches that consider the coupling between task planning and path planning—i.e., those that account for the impact of collision-free paths on task planning—can achieve higher solution quality in dense obstacle environments. Among them, many methods [9], [11], [31] use a path planner to construct the cost matrix of collision-free paths before task planning and then use this matrix in the task planning solver to obtain high-quality solutions for the MRTPP problem. However, since these methods aim to minimize the total cost of the robots, this may increase the time required for task completion, affecting the efficiency of the multi-robot system. To address this, some methods [32], [33], [34], [35] set the objective function of the MRTPP problem to minimize the maximum cost among the robots, thereby ensuring the collaboration efficiency of the multi-robot system in terms of task completion time. Additionally,some methods [10], [11] transform the MRTPP problem into classical problems and resort to current advanced solvers, such as LKH [36], Gurobi [37], and OR-Tools [38]. However, as robots' workspace expands, existing path planners struggle to efficiently find collision-free paths. Therefore, it remains difficult to construct the cost matrix for task planning efficiently in large-scale, cluttered scenarios. To this end, the only way to ensure real-time computation for the multi-robot system is to reduce the scale of the MRTPP problem and find local collision-free paths instead of global ones at the cost of sacrificing solution quality.

Unlike the abovementioned methods, we propose a fast path planner to construct the cost matrix efficiently. In addition, to enhance the collaboration ability of the multi-robot system, we formulate the objective function to minimize the maximum travel cost among robots and introduce an efficient auction-based approach to solve the MRTPP problem. In conclusion, the proposed method aims to strike a better trade-off between solution quality and computational efficiency in MRTPP.

III. Problem Formulation

This letter addresses the MRTPP problem in large-scale cluttered environments, aiming to minimize the maximum travel distance for a fleet of robots to visit all tasks, starting and ending at their respective depots. Without loss of generality, we assume that there are m robots and n tasks, where the robot set and task set are represented as $\mathcal{R} = \{1, 2, \ldots, m\}$ and $\mathcal{T} = \{1, 2, \ldots, n\}$, respectively. Each robot $\mathcal{R}_i$, where $\mathcal{R}_i$ denotes the i-th robot in the robot set $\mathcal{R}$, is required to complete the assigned tasks with the shortest possible travel distance, with a maximum of $L^i_{\max}$ tasks and a maximum allowable travel distance of $D^i_{\max}$ due to its limited battery life. In addition, we define λ_α as the distance discount factor, where $0 < \lambda_\alpha < 1$. Thus, the reward function for robot $\mathcal{R}_i$ executing the assigned tasks can be defined as

$$f_i(\mathcal{T}_i) = \sum_{j \in \mathcal{T}_i} \lambda_\alpha^{(d(\mathcal{P}_{i,j}))} \tag{1}$$

where $\mathcal{T}_i \subseteq \mathcal{T}$ is the ordered set of tasks allocated to robot $\mathcal{R}_i$, sorted in their execution order, $\mathcal{P}_i$ is the collision-free path that robot $\mathcal{R}_i$ follows to visit all tasks in $\mathcal{T}_i$, $\mathcal{P}_{i,j}$ is the part of $\mathcal{P}_i$ from the position of $\mathcal{R}_i$ to the position of task j, and $d(\mathcal{P}_{i,j})$ is the length of the path segment $\mathcal{P}_{i,j}$.

Accordingly, the MRTPP objective to minimize the maximum travel cost among robots can be formulated as

$$\begin{aligned} &\min_{\mathcal{T}_i \in \wp(\mathcal{T})} \max_{i \in \mathcal{R}} \left(-f_i(\mathcal{T}_i)\right) \\ &\text{s.t.} \quad |\mathcal{T}_i| \le L^i_{\max}, \quad \forall i \in \mathcal{R} \\ &\qquad \sum_{i=1}^{|\mathcal{R}|} x_{i,j} = 1, \quad \forall j \in \mathcal{T} \\ &\qquad d(\mathcal{P}_i) \le D^i_{\max}, \quad \forall i \in \mathcal{R}, \end{aligned} \tag{2}$$

[1] https://github.com/wuuya1/MRTPP

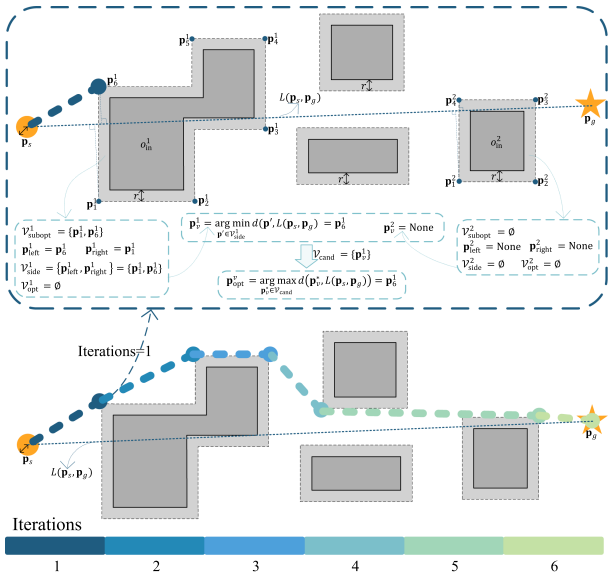

Fig. 1. Illustration of the proposed OVS planner. The light gray area enclosed by the dashed box represents the inflated obstacles in $\mathcal{O}$, and a vertex being None indicates that the corresponding vertex does not exist.

where $\wp(\mathcal{T})$ denotes the power set of $\mathcal{T}$, $x_{i,j}$ is a binary variable equal to 1 if task j is assigned to robot $\mathcal{R}_i$, and 0 otherwise. The constraints are as follows: the number of tasks allocated to $\mathcal{R}_i$ must not exceed its task capacity $L^i_{\max}$; each task j can be assigned to only one robot, though a robot may be assigned multiple tasks; and the distance traveled by $\mathcal{R}_i$ must not exceed its allowable travel distance $D^i_{\max}$. In each iteration, we introduce the marginal reward $\omega_{i,j}$, which measures the cost change from assigning an unallocated task j to robot $\mathcal{R}_i$, to iteratively optimize the objective function. The marginal reward $\omega_{i,j}$ is derived as

$$\omega_{i,j} = f_i(\mathcal{T}_i \cup \{j\}) - f_i(\mathcal{T}_i) = \lambda_\alpha^{(d(\mathcal{P}_{i,j}))}, \quad j \in \mathcal{T}. \tag{3}$$

As shown in (3), a higher marginal reward corresponds to a lower objective value, making $\omega_{i,j}$ an effective bidding metric during the auction process. In this way, the iterative bidding effectively drives the algorithm toward minimizing the maximum travel cost across robots. Moreover, the marginal reward $\omega_{i,j}$ naturally decreases as more tasks are assigned to robot $\mathcal{R}_i$, exhibiting a property known as diminishing marginal gain. This property guarantees the convergence of the algorithm, as demonstrated in [20]. Meanwhile, we assume perfect communication between all robots.

IV. Methodology

A. Obstacle-Vertex Search Planner

This subsection presents our OVS planner, illustrated geometrically in Fig. 1.

We assume that all obstacles are non-intersecting polygons, which is reasonable as polygons can accurately approximate diverse shapes in a two-dimensional workspace. Next, we define the notations used subsequently. We regard the robot as a disc with radius r, and let $\mathcal{O}$ denote the set of obstacles in the environment. Note that all obstacles in $\mathcal{O}$ are pre-inflated by the robot's radius r. Let the robot's starting and goal positions be $\mathbf{p}_s$ and $\mathbf{p}_g$, respectively. Let $L(\mathbf{p}_1, \mathbf{p}_2)$ represent a directed segment in Euclidean space, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are its starting and ending positions, respectively, directed from $\mathbf{p}_1$ to $\mathbf{p}_2$. In each iteration, let $\mathbf{p}$ and $\mathbf{p}_v$ denote the current path node and the goal path node, respectively. Let $\mathbf{p}^v_{\text{opt}}$ be the optimal convex vertex from obstacles in $\mathcal{O}$ that minimizes the cost from $\mathbf{p}$ to $\mathbf{p}_v$. Let $\mathcal{O}_{\text{in}} \subseteq \mathcal{O}$ denote the set of obstacles that geometrically intersect the segment $L(\mathbf{p}, \mathbf{p}_v)$. Initially, $\mathbf{p} = \mathbf{p}_s$ and $\mathbf{p}_v = \mathbf{p}_g$.

Next, we describe how to find the optimal vertex $\mathbf{p}^v_{\text{opt}}$ in each iteration. According to the principles of computational geometry [39], an optimal collision-free path in an environment with obstacles must consist of nodes formed by the vertices of the obstacles, excluding the starting and goal positions. Based on this principle, our key idea is to find suitable vertices from the obstacles' vertices to serve as path nodes for forming the collision-free path. Suppose there are H obstacles intersecting with segment $L(\mathbf{p}, \mathbf{p}_v)$ in an iteration. Thus, we have $\mathcal{O}_{\text{in}} = \{o^1_{\text{in}}, o^2_{\text{in}}, \ldots, o^H_{\text{in}}\}$. Then, let $\mathcal{V}_h = \{\mathbf{p}^h_1, \mathbf{p}^h_2, \ldots, \mathbf{p}^h_k\}$ be the set of convex vertices for the intersecting obstacle o^h_{in}, where $o^h_{\text{in}} \in \mathcal{O}_{\text{in}}$, with h indexing the intersecting obstacles in $\mathcal{O}_{\text{in}}$, and k indexing the convex vertices in $\mathcal{V}_h$. Meanwhile, we define the set $\mathcal{V}^h_{\text{subopt}} \subseteq \mathcal{V}_h$ of the suboptimal convex vertices for o^h_{in} as

$$\mathcal{V}^h_{\text{subopt}} = \{\mathbf{p}^h_k \mid L(\mathbf{p}, \mathbf{p}^h_k) \cap o_{\text{ob}} = \emptyset\}, \forall \mathbf{p}^h_k \in \mathcal{V}_h, \forall o_{\text{ob}} \in \mathcal{O}, \tag{4}$$

where the segment $L(\mathbf{p}, \mathbf{p}^h_k)$, formed by any vertex $\mathbf{p}^h_k$ in $\mathcal{V}^h_{\text{subopt}}$ and the current path node $\mathbf{p}$, does not intersect with any obstacle o_{ob} in $\mathcal{O}$. Further, let $\mathcal{V}^h_{\text{side}} = \{\mathbf{p}^h_{\text{left}}, \mathbf{p}^h_{\text{right}}\}$ represent the set of the leftmost and rightmost vertices in $\mathcal{V}^h_{\text{subopt}}$ along the direction of segment $L(\mathbf{p}, \mathbf{p}_v)$. We then define the set $\mathcal{V}^h_{\text{opt}} \subseteq \mathcal{V}^h_{\text{side}}$ of optimal convex vertices of o^h_{in} as

$$\mathcal{V}^h_{\text{opt}} = \{\mathbf{p}'_v \mid L(\mathbf{p}, \mathbf{p}'_v) \cap o_{\text{ob}} = \emptyset, L(\mathbf{p}'_v, \mathbf{p}_v) \cap o_{\text{ob}} = \emptyset\}, \tag{5}$$

where $\mathbf{p}'_v \in \mathcal{V}^h_{\text{side}}$, and both the segments $L(\mathbf{p}, \mathbf{p}'_v)$ and $L(\mathbf{p}'_v, \mathbf{p}_v)$ do not intersect with any obstacle o_{ob} in $\mathcal{O}$. As illustrated in Fig. 1, two obstacles intersect with segment $L(\mathbf{p}, \mathbf{p}_v)$ in the first iteration, where $H = 2$, $\mathbf{p} = \mathbf{p}_s$, and $\mathbf{p}_v = \mathbf{p}_g$. Taking the intersecting obstacle o^1_{in} as an example (i.e., $h = 1$), its convex vertices are $\mathcal{V}_1 = \{\mathbf{p}^1_1, \mathbf{p}^1_2, \ldots, \mathbf{p}^1_6\}$, and the suboptimal convex vertices are $\mathcal{V}^1_{\text{subopt}} = \{\mathbf{p}^1_1, \mathbf{p}^1_6\}$. Additionally, the leftmost and rightmost vertices in $\mathcal{V}^1_{\text{subopt}}$ along the direction of the segment $L(\mathbf{p}, \mathbf{p}_v)$ are $\mathbf{p}^1_{\text{left}} = \mathbf{p}^1_6$ and $\mathbf{p}^1_{\text{right}} = \mathbf{p}^1_1$, respectively. Thus, the leftmost and rightmost vertices form the set $\mathcal{V}^1_{\text{side}} = \{\mathbf{p}^1_{\text{left}}, \mathbf{p}^1_{\text{right}}\}$, and the set of optimal convex vertices is $\mathcal{V}^1_{\text{opt}} = \emptyset$.

Thus, we can derive the formulation for the candidate vertex $\mathbf{p}^h_v$ related to the intersecting obstacle o^h_{in} as

$$\mathbf{p}^h_v = \begin{cases} \arg\min\limits_{\mathbf{p}' \in \mathcal{V}^h_{\text{opt}}} d(\mathbf{p}', L(\mathbf{p}, \mathbf{p}_v)), & \text{if } \mathcal{V}^h_{\text{opt}} \neq \emptyset, \\ \arg\min\limits_{\mathbf{p}' \in \mathcal{V}^h_{\text{side}}} d(\mathbf{p}', L(\mathbf{p}, \mathbf{p}_v)), & \text{if } \mathcal{V}^h_{\text{side}} \neq \emptyset, \end{cases} \tag{6}$$

where $d(\mathbf{p}', L(\mathbf{p}, \mathbf{p}_v))$ represents the distance from vertex $\mathbf{p}'$ to segment $L(\mathbf{p}, \mathbf{p}_v)$. Then, let $\mathcal{V}_{\text{cand}}$ be the set of potential optimal vertices for all intersecting obstacles, i.e., $\mathcal{V}_{\text{cand}} = \{\mathbf{p}^h_v \mid \forall h \in \{1, 2, \ldots, H\}, \mathcal{V}^h_{\text{side}} \neq \emptyset\}$. Note that as long as a collision-free path exists, there must be at least one vertex in the intersecting

obstacles that satisfies the above conditions [39], so $\mathcal{V}_{\text{cand}}$ is guaranteed to be a non-empty set. Therefore, we can deduce the formulation for the optimal vertex $\mathbf{p}_{\text{opt}}^{v}$ as

$$\mathbf{p}_{\text{opt}}^{v} \in \arg \max_{\mathbf{p}_v^* \in \mathcal{V}_{\text{cand}}^h} d(\mathbf{p}_v^*, L(\mathbf{p}, \mathbf{p}_v)) \tag{7}$$

where $d(\mathbf{p}_v^*, L(\mathbf{p}, \mathbf{p}_v))$ denotes the distance from vertex $\mathbf{p}_v^*$ to segment $L(\mathbf{p}, \mathbf{p}_v)$. Note that if there are multiple maximizers, we select the optimal vertex corresponding to the first one encountered during the search. As shown in Fig. 1, during the first iteration, the set $\mathcal{V}_{\text{cand}} = \{\mathbf{p}_v^1\} = \{\mathbf{p}_6^1\}$ for the intersecting obstacles o_{in}^1 and o_{in}^2 is obtained using (6). At the same time, based on (7), the optimal vertex $\mathbf{p}_{\text{opt}}^{v}$ equals $\mathbf{p}_6^1$.

We summarize the OVS planner in Algorithm 1. Initially, the current path node $\mathbf{p}$ and goal path node $\mathbf{p}_v$ are set to $\mathbf{p}_s$ and $\mathbf{p}_g$, respectively (Line 1). Then, $\mathbf{p}_s$ is added to the robot's global path $\mathcal{P}_{\text{path}}$, and all obstacles in $\mathcal{O}$ that intersect with the segment $L(\mathbf{p}, \mathbf{p}_v)$ are identified to form the set $\mathcal{O}_{\text{in}}$ (Lines 2-3). Next, we obtain the optimal vertex $\mathbf{p}_{\text{opt}}^{v}$ in each iteration using the proposed method (Line 5). We then add $\mathbf{p}_{\text{opt}}^{v}$ to $\mathcal{P}_{\text{path}}$ and update $\mathbf{p}$ and $\mathbf{p}_v$ if segment $L(\mathbf{p}, \mathbf{p}_{\text{opt}}^{v})$ does not intersect with any obstacles in $\mathcal{O}$ (Lines 8-12). Otherwise, $\mathbf{p}_v$ is updated to $\mathbf{p}_{\text{opt}}^{v}$ (Lines 13-14). This process is repeated until $\mathbf{p}_{\text{opt}}^{v}$ equals $\mathbf{p}_g$ or $\mathbf{p}_{\text{opt}}^{v}$ does not exist.

1) Complexity Analysis: As summarized in Algorithm 1, the OVS planner's computation complexity primarily stems from searching for the optimal vertex $\mathbf{p}_{\text{opt}}^{v}$ and performing intersection checks. Specifically, the time complexity of searching for $\mathbf{p}_{\text{opt}}^{v}$ is approximately $O(N)$ in the worst case, where N is the total number of vertices of all polygonal obstacles. Intersection checks are performed by determining whether the segment $L(\mathbf{p}, \mathbf{p}_v)$ intersects with the edges of any obstacle in $\mathcal{O}$, so the time complexity is $O(N)$ in the worst case. Thus, the computational complexity per iteration of the OVS planner is approximately $O(N)$. Assuming K iterations, the total computational complexity for the planner to find a collision-free path is $O(K \times N)$. Notably, since the OVS planner does not require constructing a visibility graph [39], it is much more efficient than visibility graph-based methods. In fact, its complexity depends on the number of convex vertices of polygonal obstacles, making it significantly more efficient in large-scale environments compared to traditional search-based [28] and sampling-based methods [29]. In particular, our method supports more efficient recomputation of collision-free paths when the positions of the robot and tasks are updated.

2) Completeness and Optimality Analysis: Based on the details of the OVS planner, it is known that as long as there exists a collision-free path between the current path node $\mathbf{p}$ and the vertex $\mathbf{p}_v$, the OVS planner will always find a optimal vertex $\mathbf{p}_{\text{opt}}^{v}$ such that the segment $L(\mathbf{p}, \mathbf{p}_{\text{opt}}^{v})$ does not intersect with any obstacles in $\mathcal{O}$. Then, the OVS planner updates $\mathbf{p}$ to $\mathbf{p}_{\text{opt}}^{v}$ and continues to find a new non-intersecting segment between the updated $\mathbf{p}$ and the vertex $\mathbf{p}_v$. By repeating this process, the OVS planner will always find a series of non-intersecting segments connecting the initial $\mathbf{p}$ to $\mathbf{p}_v$. Since each segment does not intersect with any obstacles, the resulting path formed by these segments will also be collision-free. Therefore, it can be concluded that the OVS planner is complete in path planning. In this letter, to improve the computational efficiency of constructing the cost matrix for collision-free paths in task planning, the OVS planner focuses on finding a path that is close to optimal with lower computational complexity. Therefore, the optimality of the OVS planner is approximate.

Algorithm 1: Obstacle-Vertex Search Planner.

Input: $\mathbf{p}_s$, $\mathbf{p}_g$, $\mathcal{O}$
Output: the global path $\mathcal{P}_{\text{path}}$ of robot

1 $\mathbf{p} \leftarrow \mathbf{p}_s$, $\mathbf{p}_v \leftarrow \mathbf{p}_g$
2 $\mathcal{P}_{\text{path}} \leftarrow \{\mathbf{p}_s\}$
3 $\mathcal{O}_{\text{in}} \leftarrow$ **CheckIntersect**$(L(\mathbf{p}, \mathbf{p}_v), \mathcal{O})$
4 **while do**
5 $\quad \mathbf{p}_{\text{opt}}^{v} \leftarrow$ **SearchOptVertex**$(\mathbf{p}, \mathbf{p}_v, \mathcal{O}_{\text{in}})$
6 $\quad$ **if** $\mathbf{p}_{\text{opt}}^{v}$ does not exist **then**
7 $\quad\quad$ **return** $\emptyset$
8 $\quad$ **if CheckIntersect**$(L(\mathbf{p}, \mathbf{p}_{\text{opt}}^{v}), \mathcal{O})$ is $\emptyset$ **then**
9 $\quad\quad \mathcal{P}_{\text{path}} \leftarrow \mathcal{P}_{\text{path}} \cup \{\mathbf{p}_{\text{opt}}^{v}\}$
10 $\quad\quad$ **if** $\mathbf{p}_{\text{opt}}^{v} == \mathbf{p}_g$ **then**
11 $\quad\quad\quad$ **return** $\mathcal{P}_{\text{path}}$
12 $\quad\quad \mathbf{p} \leftarrow \mathbf{p}_{\text{opt}}^{v}$, $\mathbf{p}_v \leftarrow \mathbf{p}_g$
13 $\quad$ **else**
14 $\quad\quad \mathbf{p}_v \leftarrow \mathbf{p}_{\text{opt}}^{v}$
15 $\quad \mathcal{O}_{\text{in}} \leftarrow$ **CheckIntersect**$(L(\mathbf{p}, \mathbf{p}_v), \mathcal{O})$

B. Auction-Based Task Planning

In this subsection, we first employ the OVS planner to construct cost matrices for collision-free paths—between robots and tasks, as well as among tasks. We then propose a memory-aware strategy for efficient task planning.

The notations used subsequently are consistent with those in Section III. Furthermore, we denote $\mathbf{p}_i$ as the position of robot $\mathcal{R}_i$ and $\mathbf{p}_j$ as the position of task j, where $i \in \mathcal{R}$ and $j \in \mathcal{T}$. Then, let $\mathbf{C}_t \in \mathbb{R}^{m \times m}$ be the cost matrix that represents the connection costs among all tasks, where m is the number of tasks. Specifically, its element $\mathbf{C}_t(j_1, j_2)$ can be computed by

$$\mathbf{C}_t(j_1, j_2) = \mathbf{Len}[\mathcal{P}_{\text{path}}(\mathbf{p}_{j_1}, \mathbf{p}_{j_2})], j_1, j_2 \in \mathcal{T}, \tag{8}$$

where $\mathbf{Len}[\mathcal{P}_{\text{path}}(\mathbf{p}_{j_1}, \mathbf{p}_{j_2})]$ is the length of the collision-free path $\mathcal{P}_{\text{path}}(\mathbf{p}_{j_1}, \mathbf{p}_{j_2})$, which is obtained using our OVS planner. Similarly, the costs between robot $\mathcal{R}_i$ and tasks are accounted for by $\mathbf{C}_{\mathcal{R}_i,t} \in \mathbb{R}^{1 \times m}$, and its element $\mathbf{C}_{\mathcal{R}_i,t}(j)$ can be computed by

$$\mathbf{C}_{\mathcal{R}_i,t}(j) = \mathbf{Len}[\mathcal{P}_{\text{path}}(\mathbf{p}_i, \mathbf{p}_j)], i \in \mathcal{R}, j \in \mathcal{T}. \tag{9}$$

We next present how to implement the proposed memory-aware strategy. By incorporating our strategy into the auction-based task planning method, we aim to improve computational efficiency while ensuring that the solution quality is maintained. Let ℓ_{cur}^{i} be the current total travel distance required for $\mathcal{R}_i$ to visit all tasks in $\mathcal{T}_i$ in order, and ℓ_{temp}^{i} be the robot's total travel distance after adding unallocated task j. Additionally, it is important to note that the marginal reward $\omega_{i,j}$ is also the bid

Algorithm 2: Auctions with Memory-Aware Strategy.

Input: $j, \ell^i_{\text{cur}}, \mathcal{T}_i, D^i_{\max}, \mathbf{C}_t, \mathbf{C}_{\mathcal{R}_i,t}$
Output: the value of $\omega_{i,j}$ for assigning task j to $\mathcal{R}_i$
1 **if** a new task is added to $\mathcal{T}_i$ **then**
2 **if** $|\mathcal{T}_i| > 1$ **then**
3 $\ell^i_{\text{cur}} \leftarrow \ell^i_{\text{cur}} + \mathbf{C}_t(\mathcal{T}_i[-2], \mathcal{T}_i[-1])$
4 **else**
5 $\ell^i_{\text{cur}} \leftarrow \ell^i_{\text{cur}} + \mathbf{C}_{\mathcal{R}_i,t}(\mathcal{T}_i[-1])$
6 $\ell_{\text{temp}} \leftarrow \ell^i_{\text{cur}}$
7 **if** $\mathcal{T}_i$ is $\emptyset$ **then**
8 $\ell_{\text{temp}} \leftarrow \mathbf{C}_{\mathcal{R}_i,t}(j)$
9 **else**
10 $\ell_{\text{temp}} \leftarrow \ell_{\text{temp}} + \mathbf{C}_t(\mathcal{T}_i[-1], j)$
11 $\omega_{i,j} \leftarrow \lambda_\alpha^{(\ell_{\text{temp}})}$
12 **if** $\ell_{\text{temp}} + \mathbf{C}_{\mathcal{R}_i,t}(j) > D^i_{\max}$ **then**
13 $\omega_{i,j} \leftarrow 0.1 \cdot \omega_{i,j}$
14 **return** $\omega_{i,j}$

that robot $\mathcal{R}_i$ places for each unallocated task j, which can be obtained by (3). The core idea of the memory-aware strategy is that, during each iteration, robot $\mathcal{R}_i$ records the bid for each unallocated task from the previous iteration. Therefore, in the latest iteration, if $\mathcal{R}_i$ does not receive any new tasks, its bid (i.e., $\omega_{i,j}$) for each unallocated task j will remain unchanged, and $\mathcal{R}_i$ only needs to use the bid from the previous iteration. According to (3), if we know the robot's total travel distance after adding task j, the marginal reward $\omega_{i,j}$ can be calculated with $O(1)$ computational complexity. Thus, by remembering the current total travel distance ℓ^i_{cur} in each iteration, we can quickly compute the total travel distance ℓ^i_{temp} after adding task j, and then use (3) to compute $\omega_{i,j}$ with $O(1)$ complexity.

We summarize the implementation of memory-aware strategy in Algorithm 2. Specifically, before task planning begins, we first calculate the cost matrix $\mathbf{C}_t$ for collision-free paths between tasks, as well as the cost matrix $\mathbf{C}_{\mathcal{R}_i,t}$ for collision-free paths between robot $\mathcal{R}_i$ and the tasks. When calculating $\omega_{i,j}$ for each unallocated task j, if robot $\mathcal{R}_i$ has been assigned a new task (i.e., $\mathcal{T}_i$ is updated), the current total distance ℓ^i_{cur} is updated; otherwise, it remains unchanged (Lines 1-5). Note that $\mathcal{T}_i[-1]$ and $\mathcal{T}_i[-2]$ denote the last and second-to-last elements of $\mathcal{T}_i$, respectively. Next, we use the matrices $\mathbf{C}_t$ and $\mathbf{C}_{\mathcal{R}_i,t}$, along with ℓ^i_{cur}, to compute the total distance ℓ^i_{temp} after adding task j (Lines 6-10). Finally, by substituting ℓ^i_{temp} into (3), we can obtain $\omega_{i,j}$ with $O(1)$ complexity (Line 11). Furthermore, due to the maximum travel distance $D^i_{\max}$ limitation, if the total travel distance after completing task j and returning to robot $\mathcal{R}_i$'s depot exceeds $D^i_{\max}$, the bid $\omega_{i,j}$ will be discounted by a factor (empirically 0.1) to ensure that task j is assigned to a more suitable robot (Lines 12-13). Compared with traditional auction-based methods, such as [8] [20], [21], [22], [23], each iteration requires the robot to first compute its current total reward using (1), then calculate the total reward after adding task j, and finally determine the bid $\omega_{i,j}$ by taking their difference. Although [8], [22] adopt a lazy strategy to reduce redundant computation, they overlook repeated calculations of marginal reward $\omega_{i,j}$. In contrast, the memory-aware strategy effectively resolves this issue.

In the consistency conflict phase of task planning, we resolve task conflicts using the minimum variance consistency approach to minimize the maximum travel distance among robots. Specifically, if multiple robots place the highest bids for the same task compared to other tasks, we prioritize assigning the task to the robot that results in the least variance in travel distances. Finally, we implement the remaining parts of the task planning using the lazy-based review consensus algorithm (LRCA) proposed in [8]. In brief, each robot broadcasts the obtained $\omega_{i,j}$ and the corresponding task j from the current iteration to its neighbors, and receives similar information from them. Then, the best task-robot pair is determined locally using the LRCA algorithm. Due to page limitations, readers are referred to [8] for details.

V. Experiment Results

In this section, we evaluate the effectiveness of the proposed method through extensive and challenging experiments. All code is implemented in Python and executed on an ASUS desktop equipped with an Intel(R) Core(TM) i7-8700 CPU and 32 GB of memory.

A. Evaluation of Path Planners

We compare the proposed OVS planner with classic state-of-the-art planners, including the search-based A* [27] and JPS [28], the sampling-based RRT* [29], and the learning-based Neural A* [40], to validate its effectiveness in constructing the cost matrix for task planning. We conduct 1,000 Monte Carlo tests in three cluttered environments of small, medium, and large sizes, with areas of $32 \times 32\,\text{m}^2$, $256 \times 256\,\text{m}^2$, and $6000 \times 4000\,\text{m}^2$, respectively, as shown in Fig. 2. In each test, the robot's start and goal positions are randomly generated, and path planning is performed using the five methods. The average planning time and average path length from the 1,000 tests are recorded as evaluation metrics. Note that if a single planning exceeds 1499.99 ms, the path length is recorded as a penalty of 3499.99 m. Additionally, the grid map resolutions used by A*, JPS, and Neural A* in these three environments are 0.25 m, 0.5 m, and 5 m, respectively. To improve efficiency, RRT* ends the iteration once a collision-free path is found. All methods apply the same smoothing process to the initial collision-free paths to remove redundant path nodes.

The results are shown in Figs. 2 and 3. Specifically, Fig. 2 shows the collision-free paths of five planners in a pathfinding test across three scenarios. In Fig. 3, the **Time** and **Length** represent the average planning time and average path length in 1,000 tests, with the best results highlighted in bold. By observing Fig. 3(a), it is evident that OVS achieves the shortest average planning time, remaining under 10 ms in all three scenarios. In terms of average planning time, OVS outperforms the other four planners by up to 3× in the worst case and 7× in the best case. Additionally, as shown in Figs. 2 and 3(b), A* achieves the best performance in terms of average path length in both

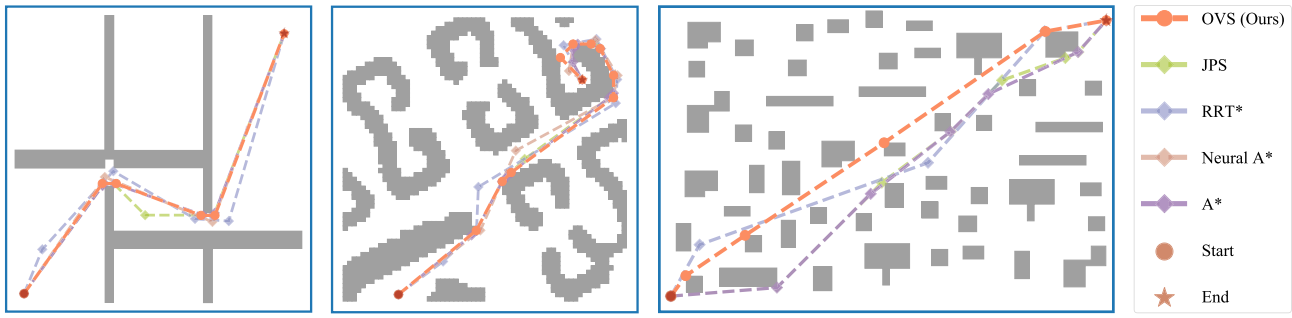

Fig. 2. (a) and (b) show the collision-free paths generated by five methods in the small and medium scenarios, respectively, with A* yielding the shortest path in both cases. (c) presents the pathfinding results of the four remaining methods (excluding Neural A*) in the large-scale scenario. Neural A* failed to generate a collision-free path due to a planning timeout. Among the remaining methods, the OVS planner yielded the shortest path. (d) The same colors denote the same methods, and the light red circle and star represent the start and goal positions, respectively.

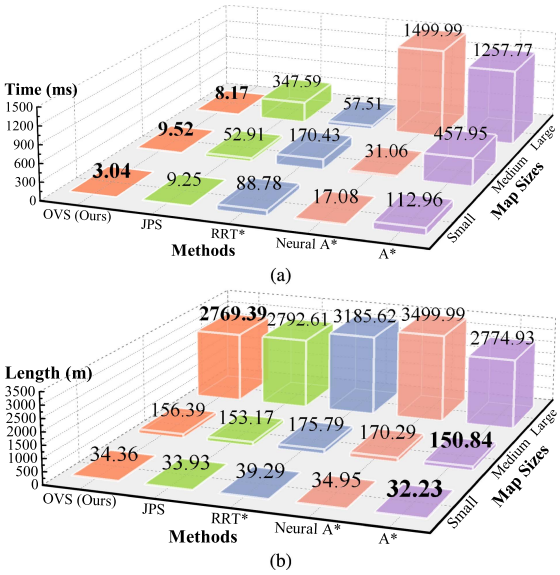

Fig. 3. (a) The average planning time over 1,000 tests across the three environments. (b) The average path length across the same 1,000 tests in the three environments.

small and medium scenarios. However, the average path length of OVS is also on par with that of A*. In the large scenario, OVS achieves the shortest average path length, although the result of A* is also close. Notably, A* does not perform as well because the coarse resolution (set to 5 m for computational efficiency) leads to longer collision-free paths. In contrast, Neural A* fails to generate collision-free paths within the 1499.99 ms limit in all trials, resulting in a penalty path length of 3499.99 m.

In summary, the OVS planner can efficiently find collision-free paths while staying close to the optimal ones generated by planners like A* and JPS, ensuring negligible impact on task planning quality in most application scenarios.

B. Comparisons in Large-Scale Cluttered Environments

To thoroughly evaluate the proposed MRTPP method, we conduct benchmark comparisons in a large-scale, cluttered scenario against the following state-of-the-art methods:

- OR-Tools [38]: A state-of-the-art heuristic solver widely applied for vehicle routing problems (VRP). In solving the MRTPP problem, we incorporate capacity constraints to minimize maximum costs among robots.
- LKH3 [36]: A state-of-the-art solver for constrained TSP and VRP. We formulate the MRTPP problem as an asymmetric distance-constrained vehicle routing problem.
- Gurobi [37] + LKH3: Gurobi, a well-known commercial solver, is used to solve a mixed-integer linear programming (MILP) problem to minimize the maximum travel distance for the robots to reach a task, followed by LKH3 to solve a TSP to determine the task execution order.
- CBBA [20]: A classical auction-based method for task planning.
- LRCA [8]: A state-of-the-art auction-based method for solving the MRTPP problem.

In the large-scale cluttered scenario shown in Fig. 4, we conducted two sets of benchmark comparisons: one with a fixed number of tasks and the other with a fixed number of robots. Each MRTPP instance is defined by a constant number of robots and tasks; for example, T200R20 represents an instance with 200 tasks and 20 robots. Each instance is tested 10 times, with task and robot positions randomly generated in the task area and robot depot area, respectively, for each test. Since the advantages of the OVS planner have already been validated in the previous evaluation, we use it to construct the cost matrix for collision-free paths required for task planning across all solvers, ensuring fair comparisons. Finally, we record the average task planning time (**Time**) and the average maximum travel distance (**Dist.**) of the robot team for each instance, along with their respective standard deviations, as shown in Table I, with the best results highlighted in bold. Note that the task planning time does not include the time for constructing the cost matrix. Additionally, if the solving time exceeds 600 s, it is considered a failure, with a penalty of 30 km for the maximum travel distance in the robot team.

From Table I, it can be observed that, in the majority of instances, the proposed method achieves the shortest average solving time. Specifically, the proposed method achieves the shortest solving time when the number of robots does not exceed 80 in instances with a fixed number of tasks. Similarly, it

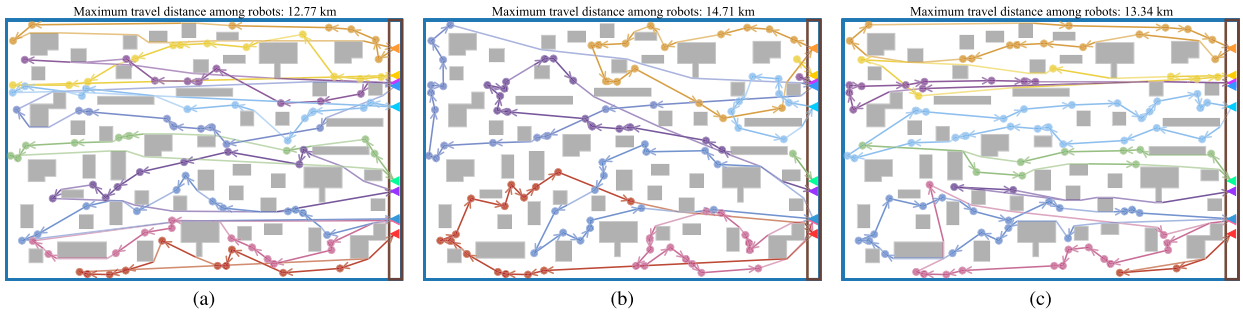

Fig. 4. (a), (b), and (c) The task planning results obtained using our method, LKH3, and the LKH3+Gurobi solvers for an instance with 10 robots and 100 tasks, respectively. The strong blue box indicates the task area, the dark brown box indicates the robot depot area, circles represent tasks, triangles represent robot depots, and the robot's color is the same as that of the tasks it visits.

TABLE I
TASK PLANNING STATISTICS FOR LARGE-SCALE CLUTTERED SCENARIOS

Group	Instance	OR-Tools		LKH3		Gurobi+LKH3		CBBA		LRCA		Proposed	
		Time [s]	Dist. [km]	Time [s]	Dist. [km]	Time [s]	Dist. [km]	Time [s]	Dist. [km]	Time [s]	Dist. [km]	Time [s]	Dist. [km]
Fixed Number of Tasks	T200R20	51.83±7.95	17.16±0.93	14.46±13.23	14.55±0.56	0.22±0.01	13.43±0.24	601.7±1.4	30.00±0.00	0.28±0.01	**12.78±0.12**	**0.14±0.01**	**12.78±0.12**
	T200R30	53.07±10.92	15.29±0.57	20.50±22.88	14.57±0.42	0.30±0.01	13.48±0.35	602.8±1.4	30.00±0.00	0.30±0.01	**12.50±0.09**	**0.19±0.01**	**12.50±0.09**
	T200R40	57.80±11.48	15.11±0.46	30.93±25.13	14.59±0.46	0.37±0.01	13.63±0.22	604.3±1.9	27.67±6.99	0.35±0.01	**12.45±0.09**	**0.25±0.01**	**12.45±0.09**
	T200R50	60.77±9.95	14.69±1.02	37.57±33.73	14.47±0.38	0.42±0.05	13.53±0.37	606.0±3.0	23.14±10.48	0.41±0.02	**12.33±0.06**	**0.31±0.01**	**12.33±0.06**
	T200R60	65.73±10.66	14.34±0.80	41.50±27.95	14.58±0.31	0.45±0.01	13.47±0.34	607.9±4.5	23.49±9.96	0.49±0.02	**12.33±0.05**	**0.40±0.01**	**12.33±0.05**
	T200R70	68.76±11.33	13.57±0.43	53.28±32.05	14.45±0.88	0.57±0.08	13.14±0.18	609.7±5.0	21.10±10.91	0.59±0.02	**12.33±0.05**	**0.50±0.02**	**12.33±0.05**
	T200R80	73.75±18.46	13.88±0.52	74.90±47.57	14.57±0.71	**0.58±0.02**	13.08±0.19	613.7±8.5	23.21±10.38	0.69±0.02	**12.33±0.08**	0.64±0.07	**12.33±0.08**
	T200R90	76.19±18.09	13.64±0.40	83.86±57.78	14.60±0.59	**0.64±0.02**	13.01±0.25	613.9±7.8	25.44±9.14	0.82±0.02	**12.28±0.06**	0.72±0.02	**12.28±0.06**
	T200R100	66.83±13.85	13.99±0.89	143.8±132.4	14.32±0.49	**0.70±0.03**	12.96±0.19	612.6±8.6	17.08±10.60	0.95±0.04	**12.28±0.04**	0.89±0.06	**12.28±0.04**
Fixed Number of Robots	T100R50	17.86±2.81	13.69±0.58	8.78±5.60	15.17±0.79	0.28±0.01	12.77±0.39	602.7±1.6	20.24±1.33	0.16±0.01	**12.25±0.12**	**0.14±0.01**	**12.25±0.12**
	T150R50	37.45±6.61	14.21±0.49	23.69±15.49	14.75±0.50	0.33±0.02	13.06±0.35	607.0±1.6	25.66±8.70	0.27±0.01	**12.37±0.08**	**0.22±0.01**	**12.37±0.08**
	T200R50	70.35±9.62	14.85±0.80	38.90±30.51	14.69±0.56	0.42±0.05	13.60±0.46	603.2±3.0	27.87±6.38	0.41±0.01	**12.36±0.08**	**0.31±0.01**	**12.36±0.08**
	T250R50	87.15±16.25	15.04±0.54	41.17±22.39	14.52±0.74	0.50±0.05	13.95±0.20	605.2±4.1	30.00±0.00	0.58±0.01	**12.41±0.06**	**0.42±0.01**	**12.41±0.06**
	T300R50	116.4±33.8	15.72±0.65	67.79±28.81	14.42±0.56	0.60±0.07	14.63±0.26	608.5±4.7	30.00±0.00	0.79±0.04	**12.41±0.04**	**0.54±0.01**	**12.41±0.04**
	T350R50	153.2±21.2	16.20±0.34	150.8±65.6	14.09±0.36	0.66±0.06	14.69±0.32	609.5±4.1	30.00±0.00	1.04±0.07	**12.54±0.20**	**0.66±0.02**	**12.54±0.20**
	T400R50	194.3±22.7	16.17±0.55	181.4±62.8	13.90±0.34	**0.79±0.07**	15.02±0.26	612.8±5.9	30.00±0.00	1.35±0.08	**12.69±0.21**	0.81±0.02	**12.69±0.21**
	T450R50	287.2±65.8	17.39±0.81	283.2±168.7	15.45±4.86	**0.95±0.09**	15.70±0.27	616.6±6.7	30.00±0.00	1.75±0.06	**13.15±0.55**	1.00±0.05	**13.15±0.55**
	T500R50	277.8±62.4	17.66±0.41	516.4±255.0	18.66±7.43	**1.00±0.05**	15.72±0.28	613.1±7.2	30.00±0.00	2.13±0.12	**14.52±1.34**	1.18±0.11	**14.52±1.34**

performs best in terms of solving time when the number of tasks does not exceed 400 in instances with a fixed number of robots. In other instances, LKH3+Gurobi yields the shortest solving time, but the proposed method also achieves comparable results. Additionally, as the MRTPP problem scale increases, the solving times of OR-Tools and LKH3 solvers exceed 100 s, while the CBBA algorithm exceeds 600 s. Among all the methods, the proposed method and the LRCA algorithm achieve the best results in all cases, followed by the LKH3+Gurobi, LKH3, OR-Tools, and CBBA solvers in most cases. It is worth noting that, in most cases, the CBBA algorithm fails to complete the task planning within 600 s, leading to the worst results. Furthermore, Fig. 4 shows the task planning results solved by the proposed method, LKH3, and LKH3+Gurobi, with an instance of 10 robots and 100 tasks. It can be observed that the proposed method better balances travel distance and task capacity, resulting in the shortest maximum travel distance. In contrast, while LKH3 and LKH3+Gurobi provide relatively good results in travel routes, they struggle to balance the travel distances between robots, which leads to a longer maximum travel distance. However, determining the constraints on task capacity and travel distance for robots while ensuring the solver achieves shorter maximum travel distances remains challenging for the LKH3 solver. Next, we analyze the impact of the proposed memory-aware strategy on computational efficiency by comparing the proposed method with the LRCA algorithm. As shown in the statistical results in Table I, for instances with a fixed number of tasks, as the number of robots increases, the proposed method gradually approaches the LRCA in terms of solving time while maintaining an advantage. For instances with a fixed number of robots, we observe that as the number of tasks increases, the computational efficiency advantage of the proposed method over LRCA becomes more pronounced, with efficiency improving by nearly twofold.

In conclusion, the proposed method achieves a superior trade-off between solution quality and computational efficiency compared to other state-of-the-art solvers, making it highly applicable to tasks such as preloaded package delivery and autonomous exploration in large-scale, cluttered environments. It should be noted that dynamic collision avoidance between robots is not considered in this work. We plan to incorporate such capabilities in future work to facilitate the real-world application of the proposed method.

VI. CONCLUSION

This letter presents an efficient multi-robot task and path planning method for a fleet of robots operating in large-scale, cluttered environments. We first introduce an obstacle-vertex

search (OVS) path planner to rapidly construct the cost matrix for task planning. The evaluation of path planners demonstrates that our OVS planner can generate near-optimal paths while improving computational efficiency by up to $7\times$, significantly reducing the time required for cost matrix construction. Additionally, we propose a memory-aware strategy, which is integrated into an auction-based task planning framework. This strategy improves the computational efficiency of task planning by up to $2\times$ without compromising solution quality. Benchmark comparisons with five existing state-of-the-art task planning solvers demonstrate that our proposed method achieves the best overall performance in computation time and solution quality for solving the MRTPP problem. Our method does not consider dynamic collision avoidance, but this can be addressed by integrating existing approaches. Moreover, the solution quality may degrade when a large number of tasks are handled by few robots. In future work, we aim to address these limitations and further promote the practical deployment of our approach.

REFERENCES

[1] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5816–5823, Jul. 2021.

[2] X. Li, X. Lu, W. Chen, D. Ge, and J. Zhu, "Research on UAVs reconnaissance task allocation method based on communication preservation," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 684–695, Feb. 2024.

[3] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Sci. Robot.*, vol. 8, no. 80, 2023, Art. no. eadf0970.

[4] Y. Du, "Multi-UAV search and rescue with enhanced A* algorithm path planning in 3D environment," *Int. J. Aerosp. Eng.*, vol. 2023, pp. 1–18, Feb. 2023.

[5] A. B. Alhassan, X. Zhang, H. Shen, and H. Xu, "Power transmission line inspection robots: A review, trends and challenges for future research," *Int. J. Elect. Power Energy Syst.*, vol. 118, Jun. 2020, Art. no. 105862.

[6] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 779–786, Apr. 2021.

[7] A. Camisa, A. Testa, and G. Notarstefano, "Multi-robot pickup and delivery via distributed resource allocation," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 1106–1118, Apr. 2023.

[8] G. Xu et al., "Distributed multi-vehicle task assignment and motion planning in dense environments," *IEEE Trans. Automat. Sci. Eng.*, vol. 21, no. 4, pp. 7027–7039, Oct. 2024.

[9] 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 Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 2750–2761, Oct. 2022.

[10] T. Jin, H. -S. Shin, A. Tsourdos, and S. He, "Integrated target assignment and trajectory optimization for many-to-many midcourse guidance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 61, no. 1, pp. 853–867, Feb. 2025.

[11] B. Zhou, H. Xu, and S. Shen, "RACER: Rapid collaborative exploration with a decentralized multi-UAV system," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1816–1835, Jun. 2023.

[12] J. Son, M. Kim, S. Choi, H. Kim, and J. Park, "Solving NP-hard min-max routing problems as sequential generation with equity context," in *Proc. ICML 2023 Workshop: Sampling Optim. Discrete Space*, 2023, pp. 1–19.

[13] K. -M. Lo, W. -Y. Yi, P. -K. Wong, K. -S. Leung, Y. Leung, and S. -T. Mak, "A genetic algorithm with new local operators for multiple traveling salesman problems," *Int. J. Comput. Intell. Syst.*, vol. 11, no. 1, pp. 692–705, 2018.

[14] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: A multi-objective approach," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2530–2537, Apr. 2020.

[15] J. Li, Y. Xiong, and J. She, "UAV path planning for target coverage task in dynamic environment," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17734–17745, Oct. 2023.

[16] S. Dong et al., "Multi-robot collaborative dense scene reconstruction," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–16, 2019.

[17] T. Kusnur et al., "A planning framework for persistent, multi-UAV coverage with global deconfliction," in *Proc. Field Serv. Robot.: Results 12th Int. Conf.*, 2021, pp. 459–474.

[18] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *Int. J. Robot. Res.*, vol. 25, no. 1, pp. 73–101, Jan. 2006.

[19] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2015, vol. 29, no. 1, pp. 2110–2116.

[20] H. -L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[21] H. -S. Shin, T. Li, H. -I. Lee, and A. Tsourdos, "Sample greedy based task allocation for multiple robot systems," *Swarm Intell.*, vol. 16, no. 3, pp. 233–260, Sep. 2022.

[22] T. Li, H. -S. Shin, and A. Tsourdos, "Efficient decentralized task allocation for UAV swarms in multi-target surveillance missions," in *Proc. 2019 Int. Conf. Unmanned Aircr. Syst.*, Jun. 2019, pp. 61–68.

[23] O. Shorinwa, R. N. Haksar, P. Washington, and M. Schwager, "Distributed multirobot task assignment via consensus ADMM," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1781–1800, Jun. 2023.

[24] P. Sankaran, K. McConky, M. Sudit, and H. Ortiz-Pena, "GAMMA: Graph attention model for multiple agents to solve team orienteering problem with multiple depots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9412–9423, Nov. 2023.

[25] H. Gao, X. Zhou, X. Xu, Y. Lan, and Y. Xiao, "AMARL: An attention-based multiagent reinforcement learning approach to the min-max multiple traveling salesmen problem," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9758–9772, Jul. 2024.

[26] M. Fan, H. Liu, G. Wu, A. Gunawan, and G. Sartoretti, "Multi-UAV reconnaissance mission planning via deep reinforcement learning with simulated annealing," *Swarm Evol. Computation*, vol. 93, 2025, Art. no. 101858.

[27] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. TSSC-4, no. 2, pp. 100–107, Jul. 1968.

[28] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Proc. AAAI Conf. Artif. Intell.*, Aug. 2011, vol. 25, no. 1, pp. 1114–1119.

[29] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[30] F. Nawaz and M. Ornik, "Multiagent, multitarget path planning in Markov decision processes," *IEEE Trans. Autom. Control*, vol. 68, no. 12, pp. 7560–7574, Dec. 2023.

[31] W. Hönig, S. Kiesel, A. Tinka, J. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 757–765.

[32] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Auton. Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[33] K. Okumura and X. Défago, "Solving simultaneous target assignment and path planning efficiently with time-independent execution," *Artif. Intell.*, vol. 321, Aug. 2023, Art. no. 103946.

[34] I. Saha et al., "Optimal makespan in a minute timespan! A scalable multi-robot goal assignment algorithm for minimizing mission time," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 9, pp. 10280–10287.

[35] A. Aryan, M. Modi, I. Saha, R. Majumdar, and S. Mohalik, "Optimal integrated task and path planning and its application to multi-robot pickup and delivery," 2024, *arXiv:2403.01277*.

[36] K. Helsgaun, "An extension of the lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde Univ.*, vol. 12, pp. 966–980, 2017.

[37] L. Gurobi, *Optimization, Gurobi Optimizer Reference Manual*, Beaverton, OR, USA, 2023. [Online]. Available:https://www.gurobi.com

[38] P. L. and F. V., "Or-tools,", Mountain View, CA, USA, 2023. [Online]. Available: https://developers.google.com/optimization/

[39] M. De Berg, *Computational Geometry: Algorithms and Applications*. Berlin, Germany: Springer, 2000.

[40] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural A* search," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12029–12039.