

Priority-Aware Multi-Robot Coverage Path Planning

Kanghoon Lee, Hyeonjun Kim, Jiachen Li, Jinkyoo Park

Abstract—Multi-robot systems are widely used for coverage tasks that require efficient coordination across large environments. In Multi-Robot Coverage Path Planning (MCP), the objective is typically to minimize the makespan by generating non-overlapping paths for full-area coverage. However, most existing methods assume uniform importance across regions, limiting their effectiveness in scenarios where some zones require faster attention. We introduce the Priority-Aware MCP (PA-MCP) problem, where a subset of the environment is designated as prioritized zones with associated weights. The goal is to minimize, in lexicographic order, the total priority-weighted latency of zone coverage and the overall makespan. To address this, we propose a scalable two-phase framework combining (1) greedy zone assignment with local search, spanning-tree-based path planning, and (2) Steiner-tree-guided residual coverage. Experiments across diverse scenarios demonstrate that our method significantly reduces priority-weighted latency compared to standard MCP baselines, while maintaining competitive makespan. Sensitivity analyses further show that the method scales well with the number of robots and that zone coverage behavior can be effectively controlled by adjusting priority weights.

I. INTRODUCTION

Multi-robot systems are increasingly used to tackle complex spatial and temporal tasks that exceed the capabilities of a single robot [1], [2], [3], [4]. Operating in parallel enables faster task completion and scalability to large and complex environments. Such systems are applied in diverse domains, including search and rescue [5], [6], surveillance [7], [8], monitoring [9], warehouse logistics [10], motion/occupancy prediction [11], [12], [13], and object transport [14]. However, realizing the full potential of multi-robot systems requires solving challenging coordination problems, particularly when task objectives involve spatial coverage [15].

A prominent class of such structured tasks involves coverage, where robots are required to collectively visit or observe every region of a known environment [16]. In MCP, the main objective is to minimize the overall makespan by generating coordinated, non-overlapping paths that enable the robots to cover the entire region efficiently [17], [18], [19]. In some settings, reducing the number of turns are also considered to improve motion smoothness or energy efficiency [20], [21]. While MCP has been widely explored, most approaches assume uniform importance across all regions. This assumption becomes inadequate when certain zones demand faster attention, such as hazardous areas, critical assets, or time-sensitive inspection sites. Delayed coverage in these regions

K. Lee is with the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. leehoon@kaist.ac.kr.

H. Kim is with the Korea Military Academy (KMA), Seoul, South Korea. hyunjoon0605@kma.ac.kr.

J. Li is with the University of California, Riverside (UCR), CA, USA. jiachen.li@ucr.edu.

J. Park is with the Korea Advanced Institute of Science and Technology (KAIST) and Omelet, South Korea. jinkyoo.park@kaist.ac.kr.

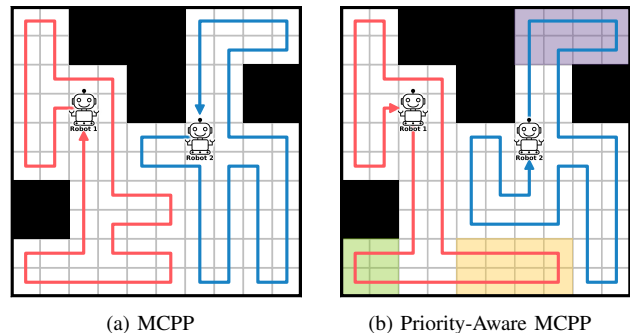


Fig. 1. Comparison of MCP and Priority-Aware MCP. While standard MCP minimizes overall makespan by distributing coverage among robots, Priority-Aware MCP additionally accounts for spatial priority by encouraging early completion of high-priority zones. Prioritized zones are indicated by green, yellow, and purple shaded areas in the right figure.

can degrade task performance or safety, underscoring the need to incorporate spatial priorities in path planning.

To address the limitations of uniform coverage, we define the Priority-Aware MCP (PA-MCP) problem, where certain regions are assigned priority weights reflecting their urgency or value. The objective is to plan robot paths that cover the entire environment while ensuring that high-priority zones are visited as early as possible, reflecting safety-critical missions where strict priority compliance is essential [22], [23]. Specifically, we minimize two criteria in lexicographic order: (1) the priority-weighted latency, defined as the time it takes to fully cover each zone multiplied by its priority weight, and (2) the makespan, defined as the time when the last robot completes its coverage task as shown in Figure 1. This lexicographic objective makes a direct application of existing MCP methods challenging, as they lack the mechanism to first minimize latency and then re-balance the remaining coverage. It is motivated by the practical need to prioritize critical zones, even at the cost of a slightly longer makespan. Unlike weighted-sum approaches, the lexicographic objective enforces this strict priority structure, ensuring covered first.

To solve the PA-MCP problem, we propose a two-phase framework aligned with the lexicographic objective. In the first phase, prioritized zones are assigned to robots using a greedy heuristic minimizing priority-weighted latency, followed by local search refinement. Each robot then generates its trajectory by sequentially covering assigned zones via zone-wise spanning tree construction and traversal. In the second phase, full coverage is ensured by building a Steiner tree over the remaining areas and distributing it among robots using a workload-balancing scheme based on prior effort. The final paths integrate results from both phases, achieving both early coverage of critical zones and efficient overall coverage.

In summary, our main contributions are as follows:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

- We formalize the PA-MCPP problem with lexicographic objectives that prioritize early coverage of critical regions while ensuring complete environment coverage.
- We develop a scalable two-phase framework that integrates greedy zone assignment with local search refinement for prioritized zones, and Steiner-tree-guided residual coverage with workload balancing.
- We validate our method on diverse layouts, achieving significant gains in priority-weighted latency while maintaining competitive makespan against MCPP baselines.

II. RELATED WORKS

A. Multi-Robot Coverage Path Planning (MCPP)

MCPP aims to generate coordinated paths for multiple robots to fully cover an environment while minimizing makespan and avoiding redundancy. Divide-and-conquer strategies, from early decentralized partitioning methods [24] to more recent approaches such as DARP [25] and GM-VPC [26], partition the environment into balanced, non-overlapping regions before computing coverage paths. Spanning tree coverage (STC) [27], a single-robot method, generates continuous coverage paths based on a spanning tree. Its multi-robot extension, including MSTC [16], MFC [28], AWSTC [17], MSTC* [18], and MIP-SRH [19] construct spanning trees or forests to generate coverage paths with reduced overlap. To handle large-scale numbers of robots and practical conflict resolution, LS-MCPP [15] incorporates local search and path deconfliction to avoid robot-robot collisions. Turn-minimization methods such as TMC [20] and TMSTC* [21] reduce turns for smoother motion, and a refinement based on local search for a single robot is explored [29]. For partially known or dynamic environments, online methods like ConCPP [30] generate paths incrementally during execution. While prior works focus on balanced workload, collision avoidance, and motion efficiency, they assume uniform importance across all regions. In contrast, our PA-MCPP performs local search at a higher level for zone assignment, integrating lexicographic objectives to prioritize early coverage of high-weight zones.

B. Task Planning with Regional Priorities

Task planning with regional priorities addresses scenarios where some areas require earlier attention than others, a situation not handled by uniform coverage approaches. Classical formulations such as the Traveling Repairman Problem (TRP, [31]) and its weighted and multi-depot variants explicitly minimize the weighted sum of arrival times to targets, ensuring high-importance locations are serviced promptly [32], [33]. Beyond these spatial formulations, persistent monitoring studies also model time-varying priorities through information decay [34], [35]. Extensions of these ideas appear in coverage and inspection tasks, where priority maps or region weights guide agents to valuable areas earlier in the mission [22], [23], [36]. Multi-robot informative path planning similarly emphasizes revisiting outdated areas to preserve sensing quality over time [37]. Similarly, in maritime Search and Rescue (SAR), the task often involves locating targets in high-probability regions

based on drift predictions and environmental constraints, aiming to reduce path overlap and travel time [5], [6]. While these works improve responsiveness to prioritized regions, they do not ensure complete coverage. Our approach enforces both full coverage and prioritization through a lexicographic objective.

III. PROBLEM FORMULATION

We consider a PA-MCPP problem defined over a two-dimensional grid environment. The environment is modeled as an undirected graph $G = (V, E)$, where each node $v \in V$ represents a discrete cell in the grid. Each vertex v is assigned a traversal cost $c(v)$. An edge $e \in E$ exists between two nodes if they are horizontally or vertically adjacent. To facilitate STC-based path generation, we introduce a hypergraph representation $H = (V_h, E_h)$ derived from G . Specifically, each 2×2 block of grid cells in V is contracted into a single hypervertex in V_h , and a hyperedge in E_h connects two hypervertices if at least one pair of their constituent grid cells is adjacent in G . The environment contains a set of n zones, $Z = \{Z_j\}_{j=1}^n$, where each zone $Z_j \subseteq V$ is associated with a positive priority weight $w_j > 0$ for all $j \in \{1, \dots, n\}$. All nodes within a zone are assumed to be connected. Let $R = \{r_i\}_{i=1}^k \subseteq V$ denote the initial positions of k robots, indexed by the robot set $I = \{1, 2, \dots, k\}$. Then, a PA-MCPP instance is represented by the tuple (G, Z, R, I) .

For each robot $i \in I$, a path is defined as a sequence of nodes $\pi_i = (v_1, v_2, \dots, v_{|\pi_i|})$ such that the path starts and ends at the initial position of robot, i.e., $v_1 = v_{|\pi_i|} = r_i$, and each consecutive node pair (v_{j-1}, v_j) is an edge in E for all $j = 2, 3, \dots, |\pi_i|$. Let $\pi_i^{(t)} = \{v_1, v_2, \dots, v_t\}$ denote the set of nodes visited by robot i up to timestep $t \leq |\pi_i|$. The coverage time of zone Z_j , denoted T_j , is the earliest timestep when all its nodes are visited by at least one robot:

$$T_j = \min \left\{ t \mid Z_j \subseteq \bigcup_{i \in I} \pi_i^{(t)} \right\}, \quad \forall j \in \{1, \dots, n\}. \quad (1)$$

The goal of PA-MCPP is to find a set of robot paths that optimizes a lexicographic objective, where multiple criteria are prioritized and one solution dominates another if it performs better on a higher-priority objective:

$$\text{lex min}_{\{\pi_i\}_{i \in I}} \sum_{j=1}^n w_j T_j, \max_{i \in I} |\pi_i| \quad (2)$$

$$\text{s.t.} \quad \bigcup_{i \in I} \pi_i = V. \quad (3)$$

The first objective is to minimize the total priority-weighted latency of zone coverage. The second objective minimizes the makespan, defined as the maximum path length among all robots. While a weighted-sum formulation can express similar trade-offs, its weights are difficult to set consistently across different environments. If no prioritized zones exist, the PA-MCPP problem reduces to the standard MCPP [15], whose notation and structure we adopt for better consistency.

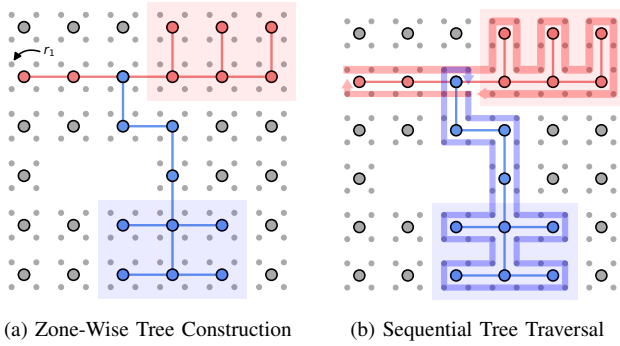


Fig. 2. **Overview of the single-robot path planning strategy.** Red and blue shaded areas indicate the first and second assigned zones (Z_1, Z_2), respectively, along with their corresponding spanning trees (T_1, T_2).

IV. METHODS

We propose a two-phase method to solve the PA-MCPP problem, explicitly grounded in the STC paradigm. The overall approach aligns with the lexicographic objective: the first phase minimizes the total priority-weighted latency of zone coverage, and the second ensures full coverage of the environment, as shown in Figure 3. Constructing multiple coverage trees is related to the min-max tree cover problem [38], but we build them to prioritize zones and then generate trees for full coverage. We first describe the single-robot path generation for multiple assigned zones in a given order in Section IV-A, then extend it to the multi-robot case with greedy and local search zone assignment in Section IV-B. Finally, residual areas are covered by constructing and concatenating residual coverage trees with prioritized trajectories in Section IV-C. For clarity of presentation, we use the symbol u to denote a vertex in the underlying graph used for path planning, which in our case is a hypervertex $u \in V_h$ of the hypergraph $H = (V_h, E_h)$. The cost of a hypervertex is the average cost of its four constituent grid cells, and the cost of a hyperedge is the average of the costs of the two hypervertices it connects [19].

A. Single Robot Path Planning with Multiple Zones

We describe the path planning process for a single robot covering multiple zones in a predefined order, as shown in Figure 2. Starting from its initial position, the robot visits each zone sequentially. To cover a zone, it first connects from its current position to the closest point in the target zone using a shortest path, then builds a spanning tree covering the entire zone. This path and the spanning tree are combined into a single tree for each zone. After constructing these trees, the robot generates a complete path by traversing them in order. When a zone is fully covered, it directly moves to the next one, ensuring ordered and continuous coverage across zones.

Zone-Wise Tree Construction: Algorithm 1 outlines the procedure for constructing a sequence of spanning trees that enables a robot to cover a set of assigned zones $\mathbf{Z} = [Z_j]_{j=1}^m$ in order, starting from its initial position r . [Lines 1-2] The algorithm begins by initializing an empty list to store the resulting trees and setting the anchor node to the starting position. [Line 4] For each zone in the sequence, the robot

Algorithm 1: Zone-Wise Tree Construction

Input: Starting node u^{start} ,
List of assigned zones $\mathbf{Z} = [Z_j]_{j=1}^m$
Output: List of spanning trees $\mathbf{T} = [T_j]_{j=1}^m$

```

1  $\mathbf{T} \leftarrow []$ 
2  $u^{\text{anchor}} \leftarrow u^{\text{start}}$ 
3 foreach  $Z_j \in \mathbf{Z}$  do
4    $u^{\text{entry}} \leftarrow \operatorname{argmin}_{u \in Z_j} \operatorname{DIST}(u, u^{\text{anchor}})$ 
5    $\tilde{T} \leftarrow$  shortest path from  $u^{\text{anchor}}$  to  $u^{\text{entry}}$ 
6    $\hat{T} \leftarrow$  find MST of  $Z_j$ 
7    $\mathbf{T} \leftarrow \mathbf{T} \parallel [\tilde{T} \cup \hat{T}]$ 
8    $u^{\text{anchor}} \leftarrow \operatorname{argmin}_{u \in \tilde{T}} \operatorname{DIST}(u, u^{\text{entry}})$ 
9 return  $\mathbf{T}$ 
    
```

Algorithm 2: Sequential Tree Traversal (STT)

Input: List of assigned zones $\mathbf{Z} = [Z_j]_{j=1}^m$,
List of spanning trees $\mathbf{T} = [T_j]_{j=1}^m$
Output: Full coverage path π

```

1  $\hat{\pi} \leftarrow$  Tree-Traversal( $\mathbf{T}[0]$ )
2 if  $|\mathbf{Z}| = 1$  then
3   return  $\hat{\pi}$ 
4 else
5    $\pi \leftarrow []$ 
6    $C \leftarrow \emptyset$ 
7   foreach  $v \in \hat{\pi}$  do
8     if  $C = \mathbf{Z}[0]$  then
9        $\hat{\pi} \leftarrow$  STT( $\mathbf{Z}[1:]$ ,  $\mathbf{T}[1:]$ )
10       $\pi \leftarrow \pi \parallel \hat{\pi}$ 
11     else
12        $\pi \leftarrow \pi \parallel [v]$ 
13     if  $v \in \mathbf{Z}[0]$  then
14        $C \leftarrow C \cup \{v\}$ 
15 return  $\pi$ 
    
```

identifies the closest node in the zone Z_j to the current anchor u^{anchor} by computing the shortest distances from the anchor to all nodes using Dijkstra's algorithm [39], and selects this node as the entry point v^{entry} . [Line 5] The resulting shortest path from the anchor to the entry point is then stored as \tilde{T}_i . [Line 6] Then, an internal coverage tree \hat{T}_i is constructed over Z_j using Kruskal's algorithm [40], yielding a minimum-cost spanning tree. [Line 7] \tilde{T}_i and \hat{T}_i are merged into a single tree, which is then appended to the list \mathbf{T} . [Line 8] The anchor node is updated to the node within the connecting path \tilde{T}_j that is nearest to the entry point v^{entry} , allowing the robot to efficiently initiate traversal toward the next zone. [Line 9] The final output is an ordered list of spanning trees, each corresponding to one of the assigned zones which is illustrated in Figure 2a.

Sequential Tree Traversal: Once a list of zone-wise spanning trees \mathbf{T} is constructed, the robot generates a complete coverage path by traversing these trees, as outlined in Algorithm 2. [Line 1] The traversal begins with the first tree using a depth-first strategy, following the standard procedure used in CPP problems based on STC. [Lines 2-3] If there is only one zone, the traversal path $\hat{\pi}$ is directly returned. [Lines 5-6] Otherwise, the algorithm initializes a coverage buffer C and iteratively appends visited nodes to the final

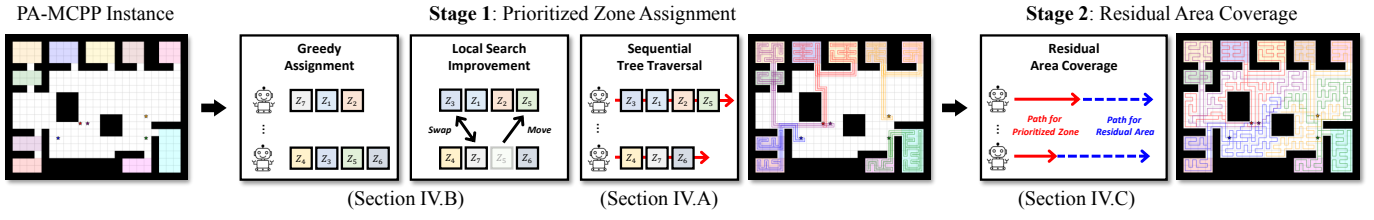


Fig. 3. **Overview of the PA-MCPP algorithm.** The instance consists of maps with prioritized zones and associated weights. Stage 1 assigns prioritized zones to robots using a greedy allocation followed by local search to refine assignments, then plans traversal sequences within each zone. Stage 2 covers remaining areas to ensure complete coverage, balancing workloads based on previous assignments to minimize makespan.

path π . [Lines 8–10] Once all nodes in the current zone are visited, the algorithm recursively calls itself with the remaining zones and their trees. The returned subpath $\tilde{\pi}$ already contains the connection to the next zone within the pre-constructed trees, and by skipping the current node v , the algorithm avoids multiple visits and ensures a continuous transition. [Line 12] Otherwise, the node is appended to the path as usual. [Lines 13–14] Throughout the process, the buffer C is continuously updated to monitor coverage progress within the current zone. [Line 15] The final output is a full coverage path π for the spanning trees \mathbf{T} , which is illustrated in Figure 2b.

B. Prioritized Zone Assignment for Multiple Robots

The goal of this step is to assign zones, together with their visiting order, to multiple robots in a way that minimizes the priority-weighted latency objective defined in Equation (2). We formulate this as a variant of the multi-depot k -TRP, where each robot's initial position acts as a depot and each zone serves as a target to be visited [33]. The objective is to minimize the total priority-weighted arrival time of all zones across robots. However, obtaining an exact optimal solution through integer programming (IP) is computationally intractable for large-scale scenarios, as noted in prior work [33]. To address this, we adopt a practical approach combining a greedy assignment with a local search refinement.

Greedy Zone Assignment: Algorithm 3 outlines the procedure for assigning ordered zones to multiple robots, minimizing the total priority-weighted latency. [Lines 1-3] The algorithm begins by initializing an empty sequence \mathbf{Z}_i and traversal time T_i for each robot, with all zones marked as unassigned. [Lines 4-5] For each zone, the internal cost of traveling within the zone, $c(Z_j)$, is computed using Kruskal's algorithm. [Lines 6-10] Depot-to-zone traversal costs $c(i, Z_j)$ are computed via Dijkstra's algorithm for each robot-zone pair, while inter-zone traversal costs $c(Z_j, Z_{j'})$ are optimistically set as the minimum distance between any node pairs from the two zones. [Lines 11-28] Then, a greedy assignment loop iteratively selects the robot-zone pair (i, Z_j) that minimizes the incremental priority-weighted latency Δ , considering both travel and internal zone costs. If a robot has no prior assignment, the cost from its depot is used; otherwise, the cost from the last assigned zone is considered. The chosen zone Z_{j^*} is appended to robot i^* 's sequence \mathbf{Z}_i , and its traversal time T_{i^*} is updated. The selected zone is removed from the unassigned set \mathcal{U} . This process continues until all zones are assigned. [Line 29] The final output is the ordered zone assignments \mathbf{Z}_i for all

Algorithm 3: Greedy Zone Assignment

Input: Robot set I with initial positions $\{r_i\}_{i \in I}$, Zones $Z = \{Z_j\}_{j=1}^n$, Priority weights $\{w_j\}_{j=1}^n$
Output: Ordered zone assignments $\{\mathbf{Z}_i\}_{i \in I}$

```

1  $\mathcal{U} \leftarrow Z$  // unassigned zones
2 foreach  $i \in I$  do
3    $\mathbf{Z}_i \leftarrow ()$ ;  $T_i \leftarrow 0$ 
   // Compute traversal costs
4 foreach  $Z_j \in Z$  do
5    $c(Z_j) \leftarrow$  MST cost of  $Z_j$ 
6 foreach  $i \in I, Z_j \in Z$  do
7    $c(i, Z_j) \leftarrow \min_{v \in Z_j} \text{DIST}(r_i, v)$ 
8 foreach  $Z_j, Z_{j'} \in Z$  with  $j < j'$  do
9    $c(Z_j, Z_{j'}) \leftarrow \min_{v \in Z_j, v' \in Z_{j'}} \text{DIST}(v, v')$ 
10   $c(Z_{j'}, Z_j) \leftarrow c(Z_j, Z_{j'})$ 
   // Greedy assignment loop
11 while  $\mathcal{U} \neq \emptyset$  do
12    $\Delta_{\min} \leftarrow +\infty$ 
13   foreach  $Z_j \in \mathcal{U}$  do
14     foreach  $i \in I$  do
15       if  $\mathbf{Z}_i = \emptyset$  then
16          $\Delta \leftarrow w_j \cdot (T_i + c(i, Z_j) + c(Z_j))$ 
17       else
18          $Z_{\text{end}} \leftarrow$  last zone in  $\mathbf{Z}_i$ 
19          $\Delta \leftarrow w_j \cdot (T_i + c(Z_{\text{end}}, Z_j) + c(Z_j))$ 
20       if  $\Delta < \Delta_{\min}$  then
21          $\Delta_{\min} \leftarrow \Delta$ ;  $(i^*, Z_{j^*}) \leftarrow (i, Z_j)$ 
22    $\mathbf{Z}_{i^*} \leftarrow \mathbf{Z}_{i^*} \parallel [Z_{j^*}]$ 
23   if  $|\mathbf{Z}_{i^*}| = 1$  then
24      $T_{i^*} \leftarrow T_{i^*} + c(i^*, Z_{j^*}) + c(Z_{j^*})$ 
25   else
26      $Z_{\text{prev}} \leftarrow$  previous last zone in  $\mathbf{Z}_{i^*}$ 
27      $T_{i^*} \leftarrow T_{i^*} + c(Z_{\text{prev}}, Z_{j^*}) + c(Z_{j^*})$ 
28    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{Z_{j^*}\}$ 
29 return  $\{\mathbf{Z}_i\}_{i \in I}$ 

```

robots, which serve as input for subsequent single-robot path planning described in Section IV-A.

Local Search Refinement: After obtaining an initial assignment of ordered zone sequences $\{\mathbf{Z}_i\}_{i \in I}$, we refine assignments using local search to further reduce the total priority-weighted latency. At each iteration, we define a local neighborhood by applying one of two types of perturbations:

- 1) **Move operator:** Randomly select a zone Z_j from a robot i 's sequence and move it to a random position within a random robot i' 's sequence.
- 2) **Swap operator:** Randomly select two zones Z_j and $Z_{j'}$

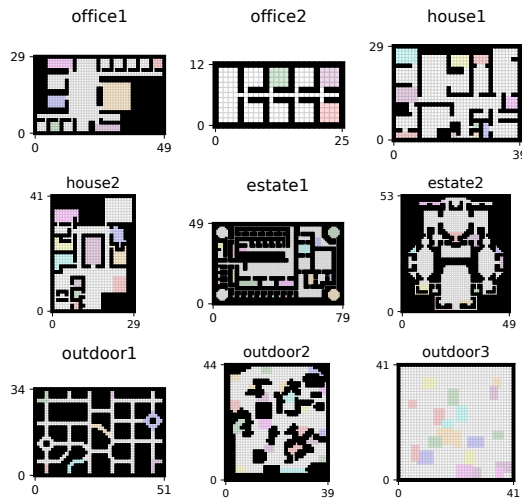


Fig. 4. PA-MCPP map instances. Black, white, and colored squares represent obstacles, normal terrain, and prioritized zones, respectively. Different colors indicate different zones.

assigned to possibly different robots, and directly swap their assignments and respective positions.

After each modification, we recompute the total objective for the new assignment. If the new configuration results in a reduced cost, it is accepted; otherwise, it is discarded. This process iteratively explores the neighborhood around the current solution, gradually improving it.

C. Residual Area Coverage with a Steiner Tree

After each robot completes its prioritized zone coverage by traversing the constructed trees $\{\mathbf{T}_i\}_{i \in I}$ in the hypergraph $H = (V_h, E_h)$, we identify the set of already covered hypervertices. This set is denoted as $\mathcal{V}_{\text{covered}} = \bigcup_{i \in I} V_h(\mathbf{T}_i)$, where $V_h(\mathbf{T}_i)$ represents the hypervertices included in robot i 's coverage tree. The remaining uncovered region is $\mathcal{V}_{\text{res}} = V_h \setminus \mathcal{V}_{\text{covered}}$. To efficiently cover \mathcal{V}_{res} , we construct a Steiner tree T_S on the hypergraph H , where the terminal set is the set of residual hypervertices \mathcal{V}_{res} . A Steiner tree is the MST over the terminal set, allowing the inclusion of additional non-terminal vertices if this reduces the total connection cost.

We then generate a single traversal path on T_S and partition it among robots using an MSTC* [18]. The partitioning aims to minimize $\max_{i \in I} (C_i^{(1)} + C_i^{(2)})$, where $C_i^{(1)}$ is the cost from phase one and $C_i^{(2)}$ is the residual cost after partitioning. This min-max allocation ensures a balanced workload and minimal makespan by directly accounting for each robot's prior effort. The resulting residual path is concatenated after the prioritized zone path to form the final path for each robot.

V. EXPERIMENTS

In the following experiments, we aim to validate four key hypotheses. (1) Can our proposed method effectively solve the PA-MCPP problem? (2) Does our method scale well with the number of robots? (3) Can we control the behavior of the proposed method by adjusting the priority weights assigned to zones? (4) Can our greedy zone assignment combined with local search produce sufficiently high-quality assignments? Note that in all experiments, we conducted 10 trials using

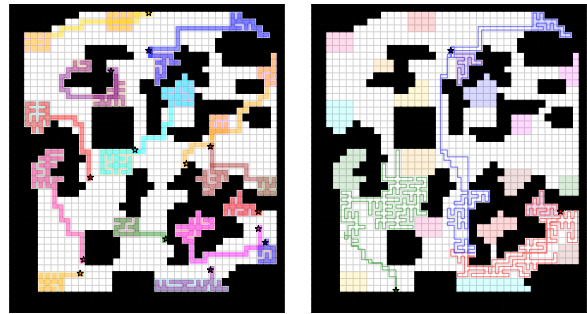


Fig. 5. Visualization of zone assignments and robot coverage paths. Left: Matched priority zones with corresponding spanning trees and assigned paths. Right: Full coverage paths for Robot 1 (red), Robot 3 (green), and Robot 15 (blue), showing prioritized zone coverage followed by residual area coverage.

random initial robot positions on non-overlapping normal terrain to ensure robustness and fairness of the results, following the experimental protocol in the previous work [15]. All experiments were performed on a Linux workstation equipped with an AMD Ryzen Threadripper 3970X 32-core processor.

A. Performance Comparison

This subsection evaluates whether the proposed PA-MCPP lowers priority-weighted zone coverage latency while keeping the makespan competitive. For comparison, we consider several established multi-robot coverage algorithms as baselines: MFC [28] and MSTC* [18]. Although these methods are not explicitly designed for priority objectives, they serve as useful makespan references. We computed the zone coverage latency by post-processing their generated paths. Including these baselines allows us to verify that our method does not significantly compromise the makespan while optimizing the zone coverage latency. All baseline implementations are based on publicly available code to ensure fairness and reproducibility¹.

We evaluate the proposed method on nine different map layouts designed to represent a wide range of scenarios, including estate, house, office, and outdoor environments, as illustrated in Figure 4. Each map is configured with varying numbers of robots and zones, as noted in Table I. All zones are assigned uniform priority weights of 1. In the third row of map instances, vertex weights are randomly sampled from a uniform distribution $\mathcal{U}(0.8, 1.2)$ to introduce variability and better reflect realistic, non-uniform environments. Table I shows the performance comparison with baselines on different map instances. Across all instances, PA-MCPP reduces latency by an average of $62.5 \pm 14.4\%$ relative to MSTC*, while maintaining a comparable makespan with only $9.7 \pm 9.4\%$ overhead. Overall, PA-MCPP achieves the desired trade-off, substantially reducing the delay time to complete high-priority zones while introducing only a modest increase in makespan.

Figure 5 illustrates the priority-aware coverage assignments and the final paths of certain robots. In Figure 5a, each priority zone is properly assigned to the robot to minimize travel delay, after which coverage paths are generated by

¹https://github.com/reso1/MSTC_Star

TABLE I
PERFORMANCE COMPARISON WITH BASELINES ON DIFFERENT MAP INSTANCES

Instance	Map Size	# of Zone	# of Robot	Weighted	Zone Coverage Latency (↓)			Makespan (↓)		
					MFC	MSTC*	PA-MCPP	MFC	MSTC*	PA-MCPP
office1	49×29	6	10	✗	431.1 ± 26.0	443.9 ± 50.5	283.7 ± 5.5	121.7 ± 10.2	96.7 ± 3.9	127.9 ± 5.2
office2	25×12	3	3	✗	104.2 ± 13.1	109.5 ± 17.5	65.7 ± 4.8	64.8 ± 8.8	47.7 ± 1.9	54.1 ± 3.2
house1	39×29	10	5	✗	1182.3 ± 152.9	1138.3 ± 226.8	339.9 ± 15.0	226.5 ± 19.8	183.1 ± 3.6	193.2 ± 3.9
house2	29×41	10	10	✗	1065.8 ± 145.8	1066.2 ± 85.4	455.5 ± 17.1	213.6 ± 13.6	157.8 ± 3.3	168.5 ± 5.4
estate1	79×49	30	5	✗	7623.7 ± 834.9	7480.0 ± 772.8	1964.3 ± 85.6	566.5 ± 57.1	433.1 ± 14.1	508.4 ± 13.5
estate2	49×53	10	10	✗	1026.4 ± 157.0	995.1 ± 102.3	248.6 ± 17.6	200.6 ± 27.1	150.8 ± 5.5	153.7 ± 6.9
outdoor1	51×34	7	10	✓	443.9 ± 52.8	446.7 ± 46.0	158.9 ± 6.3	125.3 ± 9.7	88.9 ± 3.5	89.6 ± 3.7
outdoor2	39×44	20	15	✓	1793.9 ± 75.0	1698.7 ± 111.6	558.2 ± 22.6	171.5 ± 10.8	128.2 ± 3.5	134.4 ± 4.6
outdoor3	41×41	20	20	✓	7982.4 ± 1002.8	7690.2 ± 391.4	1616.2 ± 35.3	661.3 ± 79.7	545.1 ± 4.4	570.1 ± 5.5
Avg. Improvement vs. MSTC* (%)					-1.1 ± 3.3	-	62.5 ± 14.4	-31.2 ± 6.0	-	-9.7 ± 9.4

traversing the corresponding spanning trees sequentially. Figure 5b depicts the complete coverage trajectories for three representative robots: Robot 1 (red), Robot 3 (green), and Robot 15 (blue). Robot 3 is not assigned any priority zone and thus immediately proceeds to its residual coverage region. Robot 1 has a single priority zone located directly adjacent to its starting position, which it covers first before continuing to residual areas. Robot 15 is assigned two priority zones, which it completes sequentially before transitioning to its remaining coverage region. These results indicate that our zone assignment strategy not only considers proximity but also accounts for each robot’s initial position and the size of the priority zones, thereby reducing travel overhead and enabling early completion of high-priority areas without significantly impacting overall coverage efficiency.

B. Sensitivity Analysis

In this subsection, we validate whether our method can scale effectively with the number of robots and whether its behavior can be controlled by adjusting the priority weights of zones. All experiments are conducted on the *estate1* with 5 robots as the default configuration, unless otherwise specified.

Figure 6 shows the relationship between zone coverage latency and the number of robots. We observe that the overall latency decreases as the number of robots increases, indicating that our algorithm can effectively utilize additional robots to improve coverage speed. However, when comparing the performance to an ideal scaling baseline (calculated as $y_{sol}^5 \times \frac{5}{num_robot}$, where y_{sol}^5 denotes the zone coverage latency of actual solution with five robots), we notice that the performance gap increases with more robots. This ideal scaling assumes perfect parallelization and no additional coordination overhead, which is rarely achievable in practice. The widening gap suggests that while our method remains effective at larger scales, its efficiency diminishes slightly due to increased complexity in robot coordination and zone assignments.

Figure 7 evaluates whether zone coverage can be expedited by increasing priority weights. We perform a sensitivity analysis on the smallest zone (Zone 1) and the largest zone (Zone 27) in the map, using different numbers of robots. For Zone 1, we observe that coverage time decreases rapidly as its priority weight increases, but it converges quickly, showing diminishing returns beyond a certain weight. In contrast, for Zone 27, which is significantly larger, the convergence is more gradual, and the benefit of increasing the priority weight depends more

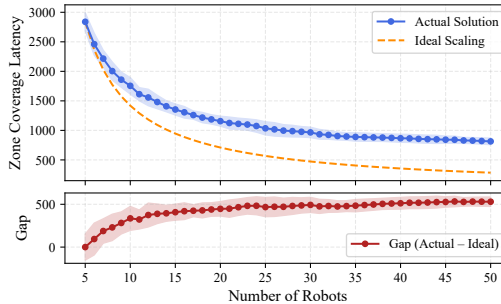
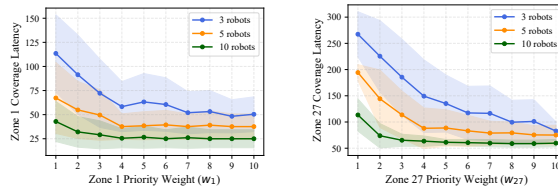


Fig. 6. Sensitivity analysis on the number of robots. Top: zone coverage latency versus the number of robots, along with the ideal scaling baseline. Bottom: the gap between the actual solution and the ideal baseline.



(a) Zone 1 (the smallest zone) (b) Zone 27 (the largest zone)

Fig. 7. Sensitivity analysis on priority weights. Zone coverage latency under different priority weights and numbers of robots.

strongly on the number of robots. Also, these results suggest that our method allows flexible control over zone coverage behavior by adjusting priority weights, validating that the algorithm can adapt to user-specified preferences.

C. Evaluation of Zone Assignment Strategies

In this subsection, we validate whether the proposed greedy zone assignment combined with local search can produce high-quality assignments for multiple robots. Experiments are conducted on the *estate1* layout, with five robots as in Section V-B. The left plot in Figure 8 shows the change in zone coverage latency over local search iterations for different initialization and operator selection strategies. We compare greedy against random initialization and evaluate two operator variants: cosine and static schedules. In the cosine schedule, we set the period to 10% of the total number of iterations, resulting in each operator’s selection probability oscillating smoothly between 1.0 and 0.0 throughout each period. In the static schedule, the move or swap operator is selected uniformly at random. Greedy initialization consistently provides a better starting point than random initialization. As

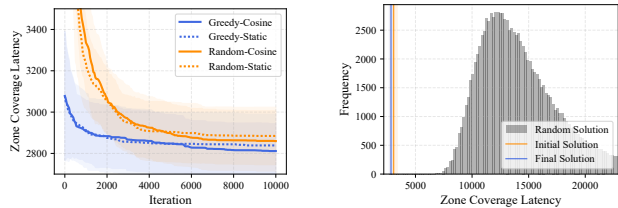


Fig. 8. **Results of zone assignment optimization.** Left: Zone coverage latency over local search iterations for different initialization and operator selection strategies. Right: Distribution of zone coverage latency for random assignments, with vertical lines indicating the initial greedy solution and the final solution after local search.

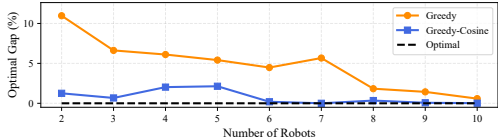


Fig. 9. **Comparison with the optimal IP solution on zone assignment.** Optimality gap of zone coverage latency versus the number of robots for initial greedy solution, the final solution after local search, and the optimal IP baseline on `outdoor1`.

the number of local search iterations increases, the solutions further improve and eventually converge. However, greedy initialization consistently produces better final solutions than random initialization. Also, the cosine schedule achieves better results than the static schedule for both initialization methods.

The right plot in Figure 8 shows the distribution of zone coverage latency for random assignments, illustrating the substantial performance gap between random and our solutions. The vertical lines indicate the latency of the initial greedy assignment and the final solution after local search refinement. These results demonstrate the effectiveness of combining a strong initial assignment with local search, supporting the ability of our method to find high-quality zone assignments.

Figure 9 compares the proposed solutions with the optimal IP baseline [33]. The results show that combining greedy initialization with local search achieves near-optimal performance, demonstrating the effectiveness of the proposed algorithm. As the number of robots increases, the optimality gap narrows because each robot is responsible for fewer zones. In such cases, a simple greedy assignment becomes nearly optimal, as most robots are assigned to the closest zone. Due to computational complexity in solving the IP formulation, this comparison was conducted only on maps (e.g., `outdoor1`) with a relatively small number of zones.

D. Evaluation of Robot Workload Balancing

To quantitatively assess workload balancing, we compute the Max-to-Mean Ratio (MMR) of the path lengths assigned to individual robots, defined as the ratio between the maximum individual workload and the mean workload across all robots, where values closer to 1 indicate better balance. Table II reports the mean and standard deviation of MMR for different team sizes. Across all cases, the MMR values of PA-MCPP remain comparable to those of MSTC*, with differences (Δ) within statistical variation. For larger teams (e.g., 20 robots), the variance increases (0.06 ± 0.23), indicating that some trials

TABLE II
COMPARISON OF MAX-TO-MEAN RATIO FOR WORKLOAD BALANCING

# of Robots	Max-to-Mean Ratio (\downarrow)		
	MSTC*	PAMCPP	Diff. (Δ)
5	1.01 ± 0.01	1.01 ± 0.0	-0.01 ± 0.01
10	1.04 ± 0.04	1.05 ± 0.03	0.01 ± 0.05
15	1.19 ± 0.13	1.20 ± 0.08	0.01 ± 0.15
20	1.21 ± 0.15	1.27 ± 0.18	0.06 ± 0.23

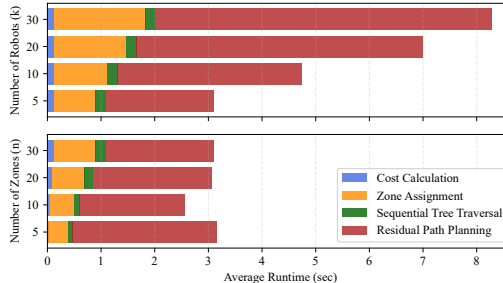


Fig. 10. **Average runtime breakdown by algorithmic stage.** Top: Mean execution time for varying numbers of robots k , with the number of zones fixed at 30. Bottom: Mean execution time for varying numbers of zones n , with the number of robots fixed at 5.

exhibit less balanced workloads. This variability arises from the inherent characteristics of the MSTC*, where maintaining perfectly balanced workloads becomes increasingly difficult as the number of robots grows. Our integration introduces only marginal additional imbalance beyond this baseline behavior. Overall, PA-MCPP maintains workload balance comparable to MSTC* while achieving its primary performance gains.

E. Computational Time Analysis

Figure 10 presents the average runtime breakdown of the proposed algorithm into four major computational stages. Across all tested configurations, residual path planning constitutes the dominant portion of the computation time, with its contribution becoming more pronounced as the number of robots increases. The runtime of cost calculation, zone assignment, and sequential tree traversal exhibit gradual growth with increasing k or n , reflecting the additional pairwise computations and assignment complexity. However, residual path planning does not scale monotonically with n . When n is small (e.g., $n = 5$), the number of uncovered nodes after the initial coverage is relatively large, causing the residual path planning step to require substantially more processing than for $n = 10$ or 20. For larger n (20–30), this effect saturates and results in comparable planning times. Overall, the number of robots has a greater impact on computation time than the number of zones, mainly due to the larger branching factor and increased complexity in assignment and traversal.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed PA-MCPP, a two-phase framework that considers both prioritized zone importance and overall coverage efficiency. The method combines greedy zone assignment with local search refinement, followed by residual area coverage to ensure completeness. Experiments show that it effectively reduces priority-weighted latency while maintaining competitive makespan, demonstrating scalability

and controllability through sensitivity analyses. However, a primary limitation of the current framework is the locally updated anchor strategy, which may lead to suboptimal zone connections due to its myopic design and the one-to-one assignment between zones and robots, which may hinder scalability in environments that contain large but few prioritized zones. As future work, we plan to explore more flexible assignments that allow multiple robots to share a single zone.

REFERENCES

- [1] X. Zhang, H. Qin, F. Wang, Y. Dong, and J. Li, "Lamma-p: Generalizable multi-agent long-horizon task allocation and planning with lm-driven pddl planner," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025.
- [2] H. Kim, K. Lee, J. Park, J. Li, and J. Park, "Human implicit preference-based policy fine-tuning for multi-agent reinforcement learning in usv swarm," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2025, pp. 18 653–18 659.
- [3] X. Gao, R. Xu, J. Li, Z. Wang, Z. Fan, and Z. Tu, "Stamp: Scalable task- and model-agnostic collaborative perception," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [4] J. Li, X. Liu, B. Li, R. Xu, J. Li, H. Yu, and Z. Tu, "Comamba: Real-time cooperative perception unlocked with state space models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- [5] B. Ai, M. Jia, H. Xu, J. Xu, Z. Wen, B. Li, and D. Zhang, "Coverage path planning for maritime search and rescue using reinforcement learning," *Ocean Engineering*, vol. 241, p. 110 098, 2021.
- [6] J. Wu, L. Cheng, S. Chu, and Y. Song, "An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning," *Ocean engineering*, vol. 291, p. 116 403, 2024.
- [7] S. Velhal, S. Sundaram, and N. Sundararajan, "A decentralized multi-robot spatiotemporal multitask assignment approach for perimeter defense," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3085–3096, 2022.
- [8] S. Bajaj, S. D. Bopardikar, E. Torng, A. Von Moll, and D. W. Casbeer, "Multivehicle perimeter defense in conical environments," *IEEE Transactions on Robotics*, vol. 40, pp. 1439–1456, 2024.
- [9] A. B. Asghar, S. Sundaram, and S. L. Smith, "Multi-robot persistent monitoring: Minimizing latency and number of robots with recharging constraints," *IEEE Transactions on Robotics*, 2024.
- [10] A. Bolu and Ö. Korçak, "Adaptive task planning for multi-robot smart warehouse," *Ieee Access*, vol. 9, pp. 27 346–27 358, 2021.
- [11] Z. Wang, Y. Wang, Z. Wu, H. Ma, Z. Li, H. Qiu, and J. Li, "Cmp: Cooperative motion prediction with multi-agent communication," *IEEE Robotics and Automation Letters*, 2025.
- [12] L. Wang, M.-A. Lavoie, S. Papais, B. Nisar, Y. Chen, W. Ding, B. Ivanovic, H. Shao, A. Abuduweili, E. Cook, et al., "Trends in motion prediction toward deployable and generalizable autonomy: A revisit and perspectives," *arXiv preprint arXiv:2505.09074*, 2025.
- [13] Y. Wang, X. Huang, X. Sun, M. Yan, S. Xing, Z. Tu, and J. Li, "Uniocc: A unified benchmark for occupancy forecasting and prediction in autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2025.
- [14] Z. Wang, S. Singh, M. Pavone, and M. Schwager, "Cooperative object transport in 3d with multiple quadrotors using no peer communication," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1064–1071.
- [15] J. Tang, Z. Mao, and H. Ma, "Large-scale multi-robot coverage path planning on grids with path deconfliction," *IEEE Transactions on Robotics*, 2025.
- [16] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 735–741.
- [17] W. Dong, S. Liu, Y. Ding, X. Sheng, and X. Zhu, "An artificially weighted spanning tree coverage algorithm for decentralized flying robots," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1689–1698, 2020.
- [18] J. Tang, C. Sun, and X. Zhang, "Mstc*: Multi-robot coverage path planning under physical constraint," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 2518–2524.
- [19] J. Tang and H. Ma, "Mixed integer programming for time-optimal multi-robot coverage path planning with efficient heuristics," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6491–6498, 2023.
- [20] I. Vandermeulen, R. Groß, and A. Kolling, "Turn-minimizing multi-robot coverage," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1014–1020.
- [21] J. Lu, B. Zeng, J. Tang, T. L. Lam, and J. Wen, "Tmstc*: A path planning algorithm for minimizing turns in multi-robot coverage," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5275–5282, 2023.
- [22] T. Kusnur, S. Mukherjee, D. M. Saxena, T. Fukami, T. Koyama, O. Salzman, and M. Likhachev, "A planning framework for persistent, multi-uav coverage with global deconfliction," in *Field and Service Robotics: Results of the 12th International Conference*, Springer, 2021, pp. 459–474.
- [23] H. Song, J. Yu, J. Qiu, Z. Sun, K. Lang, Q. Luo, Y. Shen, and Y. Wang, "Multi-uav disaster environment coverage planning with limited-endurance," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 10 760–10 766.
- [24] M. Jager and B. Nebel, "Dynamic decentralized area partitioning for cooperating cleaning robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, IEEE, vol. 4, 2002, pp. 3577–3582.
- [25] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "Darp: Divide areas algorithm for optimal multi-robot coverage path planning," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 3, pp. 663–680, 2017.
- [26] V. G. Nair and K. Guruprasad, "Gm-vpc: An algorithm for multi-robot coverage of known spaces using generalized voronoi partition," *Robotica*, vol. 38, no. 5, pp. 845–860, 2020.
- [27] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 77–98, 2001.
- [28] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 3852–3857.
- [29] D. Krupke, "Near-optimal coverage path planning with turn costs," in *2024 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, SIAM, 2024, pp. 118–132.
- [30] R. Mitra and I. Saha, "Online concurrent multi-robot coverage path planning," *arXiv preprint arXiv:2403.10460*, 2024.
- [31] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, "The complexity of the travelling repairman problem," *RAIRO-Theoretical Informatics and Applications*, vol. 20, no. 1, pp. 79–87, 1986.
- [32] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994, p. 163–171.
- [33] M. E. Bruni, S. Khodaparasti, I. Martínez-Salazar, and S. Nucamendi-Guillén, "The multi-depot k-traveling repairman problem," *Optimization Letters*, vol. 16, no. 9, pp. 2681–2709, 2022.
- [34] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 2415–2420.
- [35] R. Han, Y. Wen, L. Bai, J. Liu, and J. Choi, "Age of information aware uav deployment for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2705–2715, 2021.
- [36] S. Poudel and S. Moh, "Priority-aware task assignment and path planning for efficient and load-balanced multi-uav operation," *Vehicular Communications*, vol. 42, p. 100 633, 2023.
- [37] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin, "Efficient planning of informative paths for multiple robots," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, ser. IJCAI'07, San Francisco, CA, USA, 2007, pp. 2204–2211.
- [38] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha, "Min-max tree covers of graphs," *Operations Research Letters*, vol. 32, no. 4, pp. 309–315, 2004.
- [39] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: his life, work, and legacy*, 2022, pp. 287–290.
- [40] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.