A Hierarchical Multi Robot Coverage Strategy for Large Maps With Reinforcement Learning and Dense Segmented Siamese Network

Yihang Huang, Yanwei Wang, Zeyi Li, Haitao Zhang, and Chong Zhang

***Abstract*—Complete coverage of multiple robots for a large map is an important collaborative planning task, which is widely used in disaster search and rescue, forest fire prevention, resource exploration, and other fields. It generally focuses on coverage completion with less robot (mostly drone) occupation and higher time efficiency. However, as the map becomes larger, most existing works will fail to remain near-optimal due to escalating complexity. In this letter, we formulate the problem as two levels of traveling salesman problem (TSP) to reduce the complexity, where the lower level is single-robot local coverage planning with preset TSP solutions, and the higher level is multiple TSP (mTSP) global planning executed by multiple robots. To better adapt to dynamic scenarios, we apply a distributed multi-agent reinforcement learning (MARL) framework that allows efficient online computation, and a dense segmented Siamese network (DSSN) to achieve efficient and effective solutions. We show that compared to existing advanced methods, our strategy can effectively solve the problem with coverage costs decreased by over 30% average and the execution time reduced to seconds in large maps. Furthermore, our network DSSN achieves an additional improvement with random settings to the previous mTSP architecture. We also discuss the influence of different robot densities and separated block sizes on the results, and demonstrate the adaptability to irregular and large-scale obstacles.**

***Index Terms*—Path planning for multiple mobile robots or agents, reinforcement learning, planning, scheduling and coordination, complete coverage, large maps.**

I. Introduction

SELF organized clustering of robots exhibits strong efficiency and anti-interference in complex mission environments. The coverage path planning (CPP) in a certain area is one of the key problems included in such missions [1]. CPP is widely used in various clustering applications such as geographic mapping [2], disaster search and rescue [3], forest fire prevention [4], resource exploration [5], and so on.

Received 9 September 2024; accepted 1 November 2024. Date of publication 19 November 2024; date of current version 6 December 2024. This article was recommended for publication by Associate Editor Q. Dou and Editor J. Kober upon evaluation of the reviewers' comments. *(Yihang Huang and Yanwei Wang are joint first authors.) (Corresponding author: Haitao Zhang.)*

Yihang Huang, Yanwei Wang, and Haitao Zhang are with the State Key Laboratory of Precision Space-time Information Sensing Technology, Tsinghua University, Beijing 100084, China (e-mail: huang-yh21@mails.tsinghua.edu.cn; wywTHU24@163.com; zhanghaitao@mail.tsinghua.edu.cn).

Zeyi Li is with the National Key Laboratory of Electromagnetic Space Security, Chengdu 610000, China (e-mail: 1563636860@163.com).

Chong Zhang is with the Tsinghua University, Beijing 100084, China (e-mail: chozhang@ethz.ch).

Digital Object Identifier 10.1109/LRA.2024.3502067

On concern of different objectives and environmental conditions, many significant works aimed at CPP have been carried out in recent years, which could be subdivided into the following four aspects: (1) static or moving target detection [6], [7], (2) tracking under obstacles [8], (3) network communication [9], and (4) blind area exploration [10]. These efforts achieved good results in their respective fields, however, they usually focused on the reconnaissance of a portion of several given points, as well as limited to a small map area.

For some specific tasks, it is rarely possible to provide points of interest and target characteristics since terrain changes, like life detection in an avalanche or rapid image reconstruction of disaster areas [11]. It is required that robots can complete full coverage of relevant areas with the highest efficiency, in order to gain more valuable rescue time. In these tasks, large-size regional maps are often set up at the kilometer level [6], and we usually pay attention to the time efficiency and the spatial coverage rate to evaluate the implementation [12].

In such a task, a grid map is commonly used as the graphical model, where the map is uniformly divided to small grids. Several classic methods have been proposed to complete the traversal of all grids, including the regular traversal method [13], the pheromone method [14], and the metaheuristic algorithm [8], etc. These algorithms executed quickly and intuitively since little additional interference was set when modeling. However, once the map grows larger, the time cost in these methods increases linearly with the area [15], and the coverage rate drops as well. The Voronoi diagram [16] was early proposed to solve the problem by area division. It reduced the computation difficulty to some degree but its partitioning method was fixed manually and restricted its performance. The DARP algorithm [17] has achieved better results since it learned to divide the map more flexibly through centralized optimizing and finished traversal in each divided area by one robot. Unfortunately, DARP is still hard to converge on large-scale maps because the division process was executed on the whole map and greatly influenced by its shape. With the rapid development of reinforcement learning, several works [3], [18], [19], [20] have also been introduced to pursue better performance through training, while suitable reward settings and complex models cause new challenges as well.

Multiple traveling salesman problem (mTSP) is a classic model widely applied in task allocation while easily causing an NP-Hard problem [21]. From another perspective, the mTSP

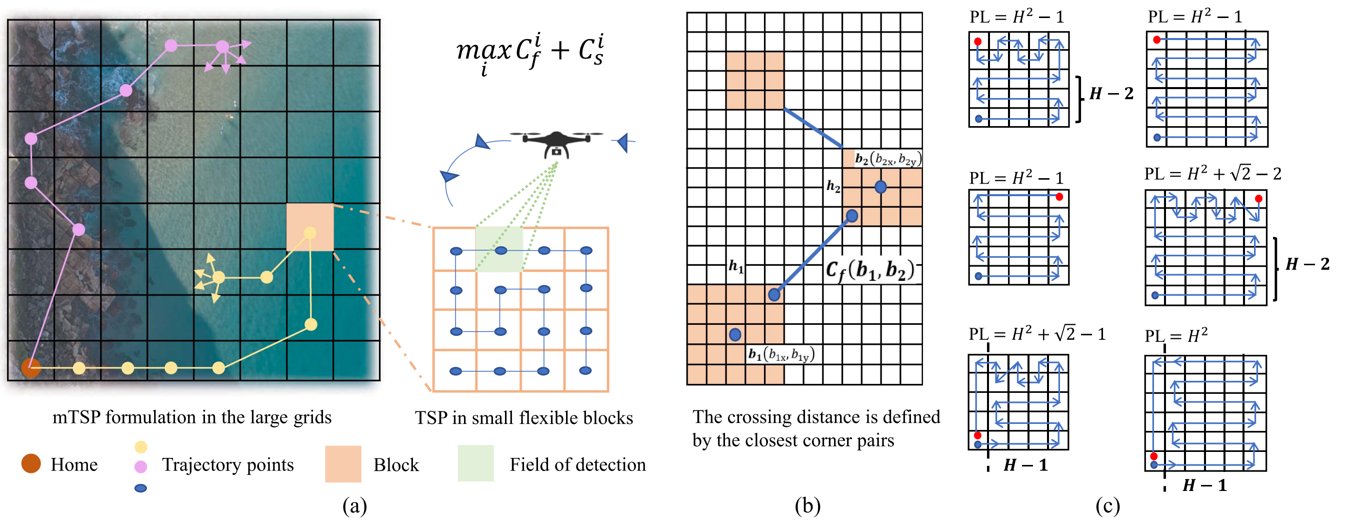

Fig. 1. Basic strategy and calculation process in regular environments. (a) Multiple robots cooperate in large grids and traverse in small blocks; (b) Corner distance calculation between blocks; (c) Enumeration of different situations for traversing in blocks.

seems to provide a potential way to solve the CPP task if we could divide the map and reduce the number of grids properly since its calculation time is also approximately linear with the number of target points. Inspired by that, in this letter, the task of complete coverage is modeled as a variant of the mTSP with preset TSP, as shown in Fig. 1(a). Meanwhile, on account of possible obstacles in maps, a Separated by Upper Right Expansion (SURE) algorithm is also proposed to do a division process to avoid falling into the NP-Hard problem. Multiple robots are set to cooperate to do the mTSP process in a fixed large-size map more efficiently, and each robot would finish preset TSP in chosen small-size blocks. All maps are guaranteed to be 2D rectangles to improve the convergence performance of the model, similar to DARP, and it would still not lose the generality of the method.

In order to pursue the extremity of coverage time, we also apply a decentralized multi-agent reinforcement learning(d-MARL) framework to execute mTSP to further improve the convergence and dynamicity, inspired by DAN [22]. The main contributions of this article are as follows:

1) We propose a novel strategy for solving CPP aimed at large maps by combining mTSP, preset TSP, and the Separated by Upper Right Expansion (SURE) method, which is adaptive to large environments with complex irregular obstacles and decreases the coverage costs by over 30% average towards the advanced methods.
2) Dynamic adaptability is well achieved by introducing the d-MARL framework, which also helps reduce the execution time to seconds. A dense segmented Siamese network (DSSN) with various reward settings is proposed and it can adapt to different random vectors without changing the network structure, ultimately suitable for varying inputs of the robot.

II. Problem Formulation

A. Regular Environments

Since expanding the scale as much as possible is one of the main objectives discussed in this method, the target environment is defined as a rectangular grid map, and the irregularity of the terrain is ignored respectively. The smallest unit of the grid is called a cell, whose size in a real environment is determined by the robot's sensing ability, such as the field of detection (FOD) of a drone. FOD can be regarded as a constant for a specific robot, so the number of cells reasonably determines the size of maps.

The global map is subsequently divided into $M \times N$ blocks, each block containing $H \times H$ cells. H is regarded as constant among different blocks in a regular and unobstructed environment. By extracting the center of each block as a target point, multiple robots would traverse these target points and a single robot would traverse within the block, as shown in Fig. 1(a). Such a division rule is beneficial for convergence, and a discussion of more general cases will be given in Section B. Here gives the definition of the coverage problem formulated by mTSP.

Definition 1: Robots in queue $(R_1, R_2, \ldots, R_n)$ starting from an assembly point A, collaborate to complete access to block series $(B_1, B_2, \ldots, B_m)$, and ultimately return to point A, with no overlap between blocks chosen by any two robots. For robot R_i, C_s^i is taken as the search distance within blocks, and C_f^i represents the distance crossing. The optimization goal is to minimize the maximum cost among all robots:

$$\min \max_i C_f^i + C_s^i \tag{1}$$

For the sake of approximate optimization in training, it is considered that the moving cost is defined as the distance between the closest corner points of two blocks, while that of searching a single block is taken as the number of cells inside:

$$C_f(b_1, b_2) = \sqrt{\max\left(|b_{1x} - b_{2x}| - h, 0\right)^2 + \max\left(|b_{1y} - b_{2y}| - h, 0\right)^2} \tag{2}$$

$$C_s(B_1) = h_1^2 + \sqrt{2} - 1, C_s(B_2) = h_2^2 + \sqrt{2} - 1 \tag{3}$$

$$h = \frac{h_1 + h_2}{2} \tag{4}$$

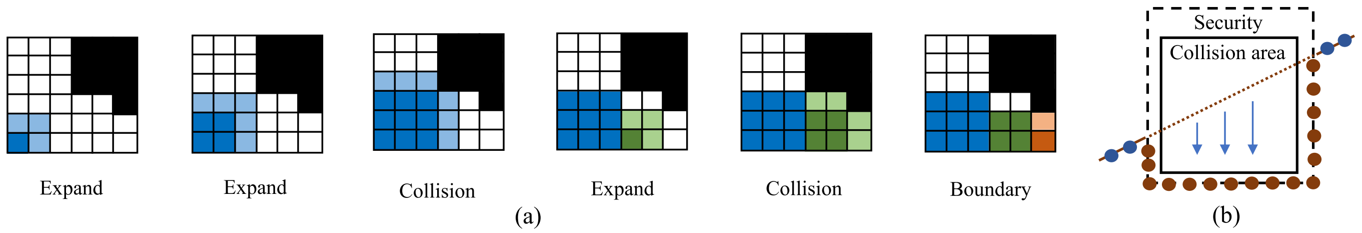

Fig. 2. Two main steps of prepossessing in irregular environments with obstacles. (a) The Separated by Upper Right Expansion (SURE) algorithm; (b) The collision area avoidance process.

Where h_1 and h_2 are the sizes of two blocks, h denotes the average, and b_1 and b_2 are the respective center coordinates, as shown in Fig. 1(b).

To illustrate the high rationality of this approximation, it should be noted that in block with size h_j, there must be a path for a robot entering and exiting from any corner, while the real path length is no larger than $h_j^2 + \sqrt{2} - 1$. Fig. 1(c) gives examples of path solutions to traverse within blocks with H equal to 5 and 6. Generally, for $h_j \in N^+, h_j \geq 2$, the range of minimum path length (PL) should be:

$$h_j^2 - 1 \leq PL \leq h_j^2 + \sqrt{2} - 1, h_j = 2k+1, k \in N^+$$
$$h_j^2 - 1 \leq PL \leq h_j^2, h_j = 2k, k \in N^+ \tag{5}$$

Using the upper bound of the above equation as a cost could encourage robots in training to choose a more balanced solution and the max error is as low as $\sqrt{2}$. It can be foreseen that under the above rules, the optimized result tends to be consistent in the number of blocks that each robot needs to access, and robots with fewer blocks should be responsible for visiting ones that are farther away from the assembly point.

B. Irregular Environments With Obstacles

When the target environment contains some uninterested areas, no-fly zones, or the task is executed by the ground mobile robot, such obstacle areas should not be ignored. However, the number and the size of obstacle areas vary for different environments, and this randomness will cause difficulties in block partitioning. At the same time, itâs essential to avoid the robotâs path passing through obstacle areas.

This article proposes a greedy expansion Algorithm, Separated by Upper Right Expansion (SURE), to solve the problem of region division, which could quickly form block sequences for any number of randomly located obstacle areas.

Fig. 2(a) represents the process of the SURE method. Starting from the cell in the left-bottom corner, do checking and an expansion process cell by cell. Only when the cell is outside all the collision areas or does not belong to any existing blocks, set it as the first member and find the largest block to be formed in the upper right direction, otherwise skip.

The global target area will be divided into blocks of varying sizes after all cells are checked. The construction of a block reflects the radiation principle during the coverage process, which means that the robot should expand the access to the current neighboring area as much as possible before accessing the next node, and it saves a lot of costs. However, the range of the block should not be too large, otherwise, the problem will unexpectedly collapse into a simple TSP that cannot utilize the collaborative ability of multiple robots. Furthermore, the positions of the target point sequences extracted from the blocks are now arranged randomly. It would increase the optimization difficulty, so a more effective model needs to be developed in the follow-up mTSP process.

Although the obstacles have been removed from the robot's choice by the SURE algorithm, it is still possible for the trajectory line to pass through the obstacle area during the robot's movement across two blocks, which is incorrect. In this letter, a control point extrapolation method similar to the EGO-planner algorithm [23] is used by setting a slightly wider safety zone for each obstacle area to avoid this problem, as shown in Fig. 2(b). After completing the route planning of all robots, the method will sequentially check whether a trajectory line has entered the safety zone. If so, subdivide the portion of the trajectory line within the safety zone into several control points with a certain spatial resolution, and move the control points to the edge to form a new collision-free trajectory. It is worth reminding that the constraints in flight control such as moving angular acceleration and turning radius are not considered in the strategy, because this process is mostly focused on coverage planning at the decision-making level.

III. Decentralized Reinforcement Learning

A. The Dense Segmented Siamese Network

As mentioned in the above algorithm, optimizing mTSP and generating effective block-aligned results are the keys to solving the problem. In this section, we adopt a distributed multi-agent reinforcement learning(d-MARL) [22] framework to execute mTSP and design a dense segmented Siamese network, DSSN, to guide the block selection.

Specifically, d-MARL is a stepper decision-making framework in time series where robots are relatively parallel and independent throughout their action paths without interference. After completing a block-to-block transfer and the coverage of all cells within the current block, robots will make a new access choice. To avoid the robot being forced to access blocks that are far away due to the lack of available blocks remaining at the end of the time series, we allow the robot to finish and return to the assembly point earlier.

In the selection of blocks, the DSSN network is shared by each robot, both in terms of structure and parameter weights. The DSSN network model is shown in Fig. 3. Its backbone

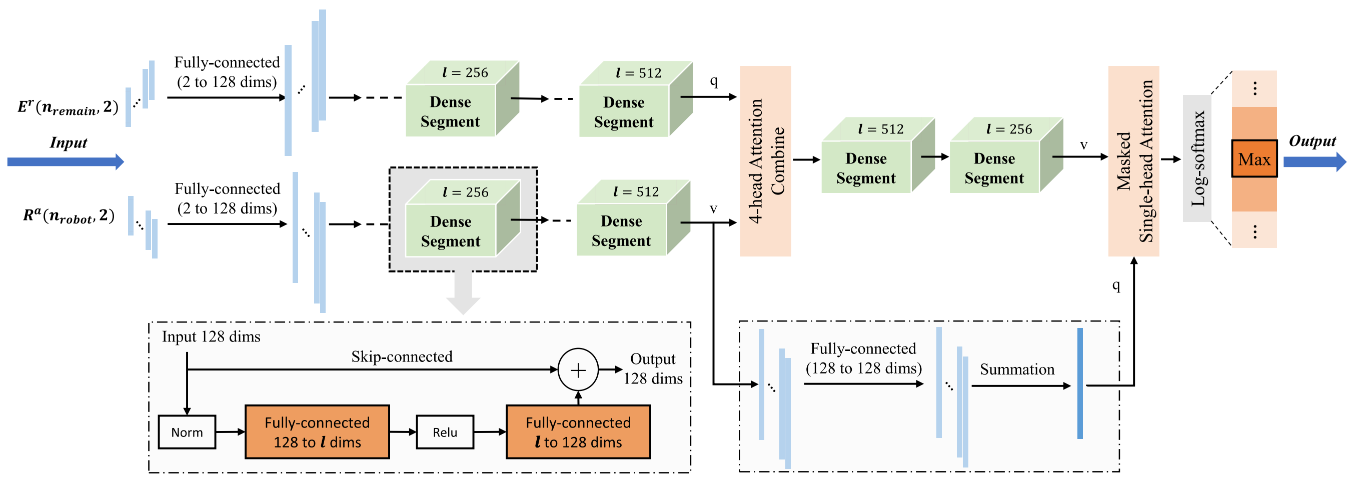

Fig. 3. Main framework of Dense Segmented Siamese Network (DSSN).

is a dual-branch neural network [24], which is used to extract environmental feature vectors C_{env} and robot feature vectors C_{rob} respectively:

$$\begin{aligned} C_{env} &\leftarrow Branch\left(E^r\left(n_{remain}, 2\right)\right) \\ C_{rob} &\leftarrow Branch\left(R^a\left(n_{robot}, 2\right)\right) \end{aligned} \tag{6}$$

In the equation, n_{remain} is the current number of unreachable blocks, n_{robot} is the remaining number of robots; E^r is a $n_{remain} \times 2$ matrix as one input, representing the relative positions between the remaining block target points and the current point; R^a is a $n_{robot} \times 2$ matrix as the other input, representing the absolute position coordinates of the robots.

The structure of both branches is the same, consisting of two stacked tight-structural segments and a fully connected layer:

$$Branch(input) = Dense_{512}\left(Dense_{256}\left(Fully(input)\right)\right) \tag{7}$$

For the (7), The composition of $Dense_h$ is shown in Fig. 3. It is described as dense because the fully connected layer is still the main module in the structure [25], which can extract the feature more sufficiently, different from the definition described in DenseSiam [26]. The first layer uses the ReLU as the activation function and the normalization and the residual calculation [27] are added to the process. l represents the length of the intermediary dimension between two layers.

A 4-head attention layer [28] is used for the joint and output of two feature branches, where the query input of this layer is taken as the environmental feature vector C_{env}, the value input is the robot feature vector C_{rob}, and the weight is calculated using the softmax function. The t-th head and the output of the attention layer are calculated as:

$$Q_t = W_t^Q C_{env}^t, K_t = W_t^K C_{rob}^t, V_t = W_t^V C_{rob}^t \tag{8}$$

$$Head_t\left(Q_t, K_t, V_t\right) = softmax\left(\frac{Q_t^T K_t}{\sqrt{d_t}}\right) V_t \tag{9}$$

The output of the 4-head attention layer is then calculated through the dense segments of two different dimensions and used as the value input for the single-head attention layer. The query input is provided by the environment feature vector through a linear fully connected layer and summation calculation to make full consideration of the environment, as shown in Fig. 3. A global mask [29] is added in the single-head attention layer, and if one block has already been visited, its similarity will be set as $-\infty$. At last, the output is calculated by log-softmax where the block with max possibility is chosen. The DSSN is novel and specially designed for the CPP. With the dense segments and the attention mechanism well constructed, it can achieve near-optimal performance in both random-setting training and a specific fixed environment.

B. Reward Fixation

In this work, the setting of rewards will directly affect the convergence performance of reinforcement learning. As an optimization goal, the maximum path length of the robot is the most important component of the reward. For small and medium-sized maps, block partitioning is generally not performed, and all cells are used as target points for the mTSP process. The reward is set to a negative number of the maximum path length:

$$R_1(\pi) = -\max_{i\in\{1,\ldots,n\}}\left(C_f\left(\pi^i\right)\right) \tag{10}$$

where π means global policy, π^i is the policy of robot R_i, and n is the number of robots executing the task.

For large-scale maps, as the robot's travel length not only includes the block transfer distance but also the search distance within the block, this part of the distance should be added. In a rule environment, blocks are usually set to the same size H. Therefore, the search distance is the same:

$$R_2(\pi) = -\max_{i\in\{1,\ldots,n\}}\left(C_f\left(\pi^i\right) + H^2 + \sqrt{2} - 1\right) \tag{11}$$

For irregular environments with obstacle areas, the search distance within blocks will no longer be the same due to varying block sizes. To facilitate the tedious calculation process, the weighted average of all block search distances is used as an alternative:

$$h_a = \frac{\sum_{j=1}^{m} n_j h_j^2}{\sum_{j=1}^{m} n_j} + \sqrt{2} - 1 \tag{12}$$

h_j is the possible size of blocks, m is the total number of blocks after division, and n_j is the number of blocks with size h_j.

In order to minimize the additional path increase caused by the robot's movement path passing through obstacle areas during the optimization process as much as possible, this increment is also included as a penalty term in the calculation of rewards:

$$R_3(\pi) = - \max_{i \in \{1,\ldots,n\}} \left(C_f \left(\pi^i \right) + h_a \right) - \frac{\sum_{k=1}^{t} \Delta C_o^k(\pi)}{n} \tag{13}$$

$\Delta C_o^k(\pi)$ denotes the distance difference of each path passing through obstacle areas on policy π, and t is the number of the above situations that occur.

IV. Experiments

A. Set up

The DSSN model uses AdamW [30] as the optimizer in training, with a weight decay of 0.98 and an initial learning rate of 5e-6. The learning rate also decays with a rate of 0.96 after every 256 epochs.

We use the following four experiments to verify the performance of the proposed strategy and network. Subsection B works on the performance comparison of DSSN and the advanced method DAN. Subsection C observes the variation effect with different robot densities and checks how block size affects the convergence performance. Subsection D evaluates the dynamic ability to further adapt to real applications. Finally, subsection E shows the complete process of our strategy in irregular large maps with obstacles and compares it to several existing great methods to further show its effectiveness.

The ideal limit is represented as the following three metrics. For a map of $M \times N$ in subsections B and C, the ideal is calculated as two times the farthest distance between two corners of the map.

$$E_{\rightarrow} = 2 \times \sqrt{(M-1)^2 + (N-1)^2} \tag{14}$$

In subsection E, the ideal limit is simply a combination of two parts. The first is calculated by dividing the number of unoccupied cells by the number of robots n. The second is the average or the maximum distance from an unoccupied cell transferring back to the start.

$$Alloc = \frac{\sum_{unoccupied} Cells}{n} \tag{15}$$

$$I_{ave} = Alloc + Ave_{back}, I_{max} = Alloc + Max_{back} \tag{16}$$

B. Network Performance

Fig. 4(b) shows the network performance for middle-scale maps in environments without obstacles, with grid numbers of 100 and 256 fixed. The majority of points in this training are randomly set, except the corner point to calculate the optimal limit $E_{\rightarrow}$, which is displayed by a dotted line. Our proposed network DSSN outperforms the advanced method DAN for each two maps, showing a good ability to reach the optimal in training. The DAN is chosen for comparison because DSSN and DAN are the only two networks proposed able to train in random environments using the d-MARL framework.

TABLE I
Robot Density Test Results in Different Maps

Map Size	625	729	900	1024	1296	1600	2025
Rob=15	84.3	90.0	109.1	103.2	139.0	178.1	223.9
Rob=10	93.1	104.9	129.7	125.5	172.5	217.3	304.9
Rob=5	141.2	171.0	198.7	208.3	292.5	371.0	484.5

TABLE II
Block Density Test Results in 2048 Map

Block Size	1×1	2×4	4×4	4×8	8×8	8×16
Max length	524.9	391.4	380.6	378.1	386.2	389.8

C. Robot Density and Block Density Test

It is undeniable that the ratio between the number of robots and the size of the map will significantly affect the convergence of the algorithm. This section uses variable controlling to test and explore how robot density applies influence on convergence and provides the setting of the optimal value in such a task.

Table I shows the results tested by the overall randomly trained model with the changes of map sizes (obstacle rate is set to around 34%) and robot numbers. It can be seen that the more robots applied, the slower the maximum length grows with the size of the map. Fig. 4(c) shows a further test with a density variation range of [0.051, 0.156]. when the density is greater than 0.1, the degree of approximation to $E_{\rightarrow}$ is around 95%, which duplicates the good performance. It is worth noting that in this experiment, a density of 0.1 represents 10 robots in a 100-size map. We believe the reason is that the number of robots is equal to the number of targets on each edge of the map and the algorithm makes it easier to find the optimal to evenly allocate all targets to different robots from an efficiency perspective.

In the curves where the number of robots changes in Fig. 4(c), more robots (11 or even 12) are assigned, but the convergence results actually decrease. This is because heavy mobile robots cannot bring more effective benefits to the results, but increase the computational burden of the model.

We further explore the impact of block sizes on the final performance of the strategy using a uniform partitioning approach in a large size of 2048. It is executed with 8 robots by the overall randomly trained model, as shown in Table II. As the block size increases, it generally shows a downward trend of max length. The main time cost in this strategy comes from the movement between blocks. Allowing the robot to cover as many adjacent areas as possible in a reasonable time will help improve the results, and the increase in block size leads to a decrease in target points, which also reduces the learning difficulty of the model.

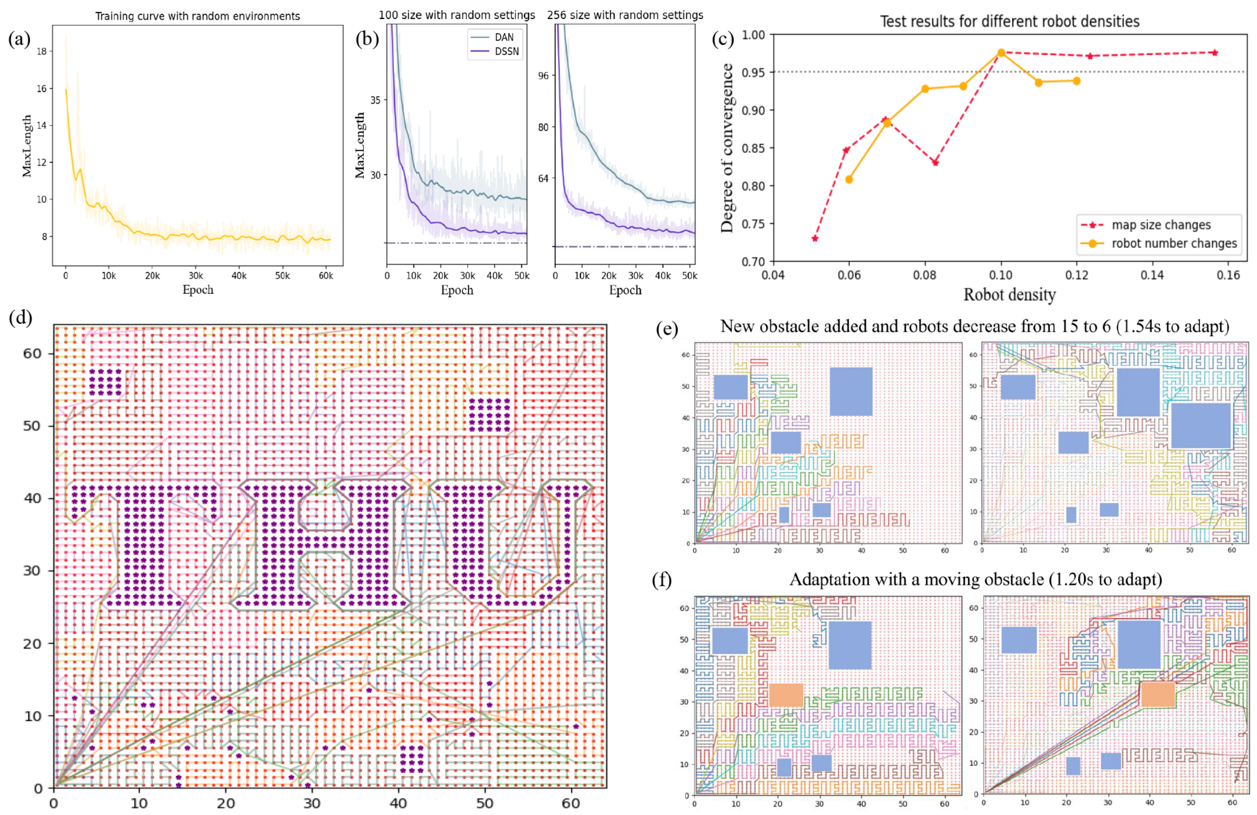

Fig. 4. (a) Overall training curve with random settings. (b)Network training comparison with DAN in different middle-scale maps with random settings. (c) Model test results in comparison with different robot densities. (d)Results of the algorithm with irregular obstacles (THU, marked as purple *). (e) Quick dynamic adaptation after the occupying robots decreases to 6 and a dynamic obstacle is added at step 130. (f) Quick dynamic adaptation with a dynamic obstacle moving.

D. Dynamic Evaluation

To better verify the good dynamic adaptability of our strategy, we change the task settings after executing several steps by reducing the number of robots and randomly adding or moving an obstacle, as we can see in Figs. 4(e) and (f). Several robots have been first set to execute in a 4096 map of 5 obstacles. In Fig. 4(e), after 130 steps, we decrease the occupying robots to 6 and add a new obstacle, and a new solution is calculated within 1.54 seconds. In Fig. 4(f), when an obstacle is moving, a new solution is also calculated within 1.20 seconds to adapt. This advantage of our strategy is helpful to better cope with unexpected situations, such as some robots losing their ability to work unexpectedly or the obstacles being repositioned during the searching time.

E. Large Map Results and Comparison

To test the adaptability to randomly distributed areas, a comparison is made with several advanced algorithms in recent years. The training curve of the overall model used in the test is shown in Fig. 4(a). The test map varies with random sizes and random obstacle distribution. The comparative algorithms include the heuristic learning GA algorithm, the DAN method [22], the MSTC* method [31], the GA combined with our SURE algorithm, and the DAN combined with SURE algorithm. It is worth noting that other advanced methods like DARP have also been tested in our experiment. However, it failed to find a solution for the large maps in a time-consuming process (>2h). As shown in Table III, The map sizes are 400, 1024, 4096, and 10000, respectively, and the collision value represents the proportion of obstacle areas to the total area. The maximum robot execution time (Max) in a 100% complete coverage, average execution time (Ave), the ratio of the above two (Rate), and computation time (Time) to evaluate the performance. Max and Ave are the most significant criteria. The smaller Max and Ave mean shorter time and less total energy it takes to complete a coverage.

From Table III, it can be seen that in large-size experiments of 1024, 4096 and 10000, the proposed model achieves the optimal results in both Max and Ave. Besides, our strategy obtains 36.20% average improvement of Max compared to MATC* and 33.97% compared to DAN. In the middle-size map of 400, the reduction of target points makes the optimization process of the other algorithms easier, and our strategy also achieves results that are very close to the best. Significantly, the total computation time including map partitioning, mTSP, and TSP process is around several seconds in our strategy, an order of magnitude less than that of GA methods. Our strategy has also proved to be suitable for complex environments with irregular obstacles, which can be found in Fig. 4(d). Besides, Our strategy is also suitable for any proportion of obstacles (even larger than 50%), as expressed in Table IV.

TABLE III
COMPARISON RESULTS WITH EFFECTIVE METHODS

Map size	400				1024				4096				10000			
Setting	Collision:10.00% Robot:10				Collision:5.18% Robot:10				Collision:8.89% Robot:15				Collision:13.04% Robot:15			
Criterion	Max	Ave	Rate	Time	Max	Ave	Rate	Time	Max	Ave	Rate	Time	Max	Ave	Rate	Time
GA	95.1	83.6	1.14	417s	482.6	435.6	1.11	609s	–	3372	–	1.2h	–	15174	–	1.5h
DAN	83.7	81.2	1.03	3.49s	197.2	194.4	1.01	7.42s	605.0	596.1	1.02	128s	1618.0	1570.1	1.03	29m
MSTC*	100.5	94.6	1.06	0.13s	252.5	237.9	1.06	0.47s	603.5	586.9	1.03	5.87s	1341.9	1293.8	1.04	7.52s
GA(**+Sure**)	75.7	67.1	1.13	131s	184.6	151.6	1.22	134s	528.9	394.1	1.34	271s	1151.9	1070.1	1.08	416s
DAN(**+Sure**)	83.1	76.7	1.08	1.92s	190.0	162.8	1.17	2.79s	508.7	394.5	1.29	3.39s	971.4	831.6	1.17	5.23s
Proposed	**76.5**	**69.9**	**1.09**	**1.63s**	**153.0**	**146.8**	**1.04**	**2.19s**	**431.5**	**380.6**	**1.13**	**2.81s**	**796.4**	**765.0**	**1.04**	**5.11s**
Ideal limit	64.3	50.5	–	–	142.4	120.9	–	–	339.3	296.9	–	–	721.2	656.5	–	–

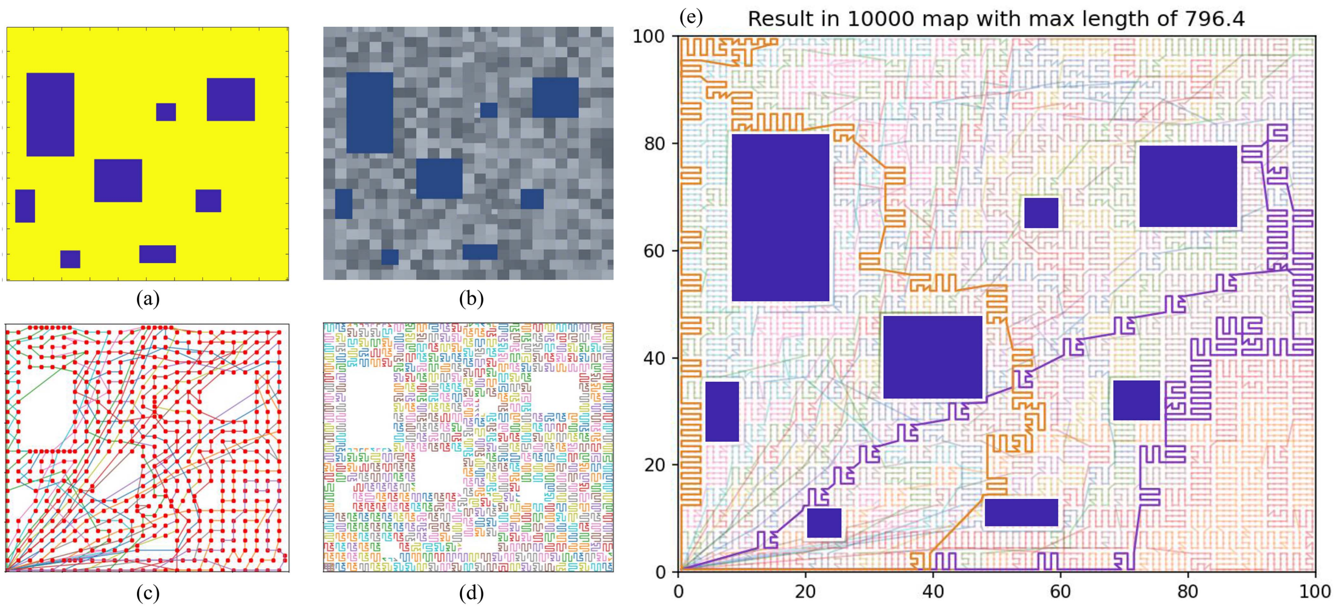

Fig. 5. Main process and the final results of the proposed strategy. (a) The random distribution of obstacles in the environment; (b) The block-partitioned result after the SURE algorithm; (c) The mTSP process calculated by the DSSN network; (d) Different TSP trajectories in different blocks; (e) The final trajectories combined in the whole map.

TABLE IV
TEST OF A LARGE PROPORTION OF OBSTACLES

Map size	2500			
Collision rate	21.16%	37.16%	49.12%	50.4%
Max searching time	354.1	388.3	289.1	301.1
Cal. time	3.76s	3.49s	3.59s	2.68s
Map size	10000			
Collision rate	13.04%	22.48%	31.92%	52.56%
Max searching time	796.4	820.2	755.2	675.0
Cal. time	5.11s	4.84s	5.42s	4.73s

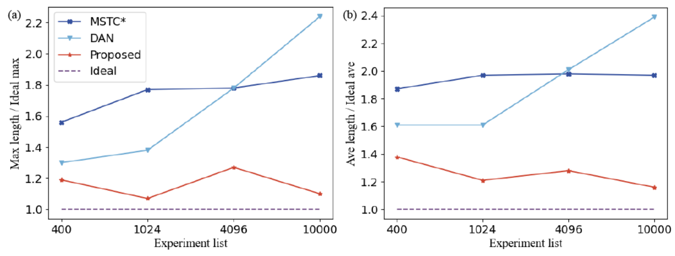

Fig. 6. The degree of approximation between results and (a) ideal max; (b) ideal ave in different experiments.

Fig. 5 shows the results of each step of the model after a single execution in a 10000-size map. Among them, the obstacles in the initial area are set randomly and our strategy can adaptively allocate paths for each robot. Fig. 5(d) shows the different TSP trajectories adapting to different blocks by considering the entering and the exiting cells in each block and the relative position between two successive blocks. Fig. 5(e) is the final trajectories combined in the whole map, where the maximum and minimum movement are marked with purple and orange bold lines. Since the models are all pre-trained, this process does not require long iterations and only takes milliseconds in our trial. Additionally, Fig. 6 further represents the comparison of results of the degree of approximation to the ideal limit in large maps. Most importantly, only one pre-training is sufficient for different map sizes with random obstacles and different robot numbers, demonstrating strong real-world applicability.

V. Discussion and Conclusion

This letter focused on the task decision-making layer. The coverage was completed through the joint modeling of mTSP and preset TSP to decouple the two actions of crossing and searching, which aligns with the application expectations of real scenarios. Even more details like 3D height are hidden, the traversal method in mTSP and TSP could be easily extended to 3D maps.

In addition, it is possible to set a random number of executing robots, map sizes, and the number of obstacles during training. This advantage also gives our model the ability to execute in real time. Of course, there is still room for improvement, such as a significant amount of time being spent on the robot's transferring without having an effect on coverage. In subsequent research, we would consider incorporating the transfer process into the optimization model to pursue better performance in real-world scenarios.

References

[1] T. M. Cabreira, L. B. Brisolara, and F. J. Paulo R, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[2] J. Valente, D. Sanz, J. Del Cerro, A. Barrientos, and M. Á. de Frutos, "Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields," *Precis. Agriculture*, vol. 14, pp. 115–132, 2013.

[3] D. Hachiya, E. Mas, and S. Koshimura, "A reinforcement learning model of multiple UAVs for transporting emergency relief supplies," *Appl. Sci.*, vol. 12, no. 20, 2022, Art. no. 10427.

[4] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Fault-tolerant cooperative navigation of networked UAV swarms for forest fire monitoring," *Aerosp. Sci. Technol.*, vol. 123, 2022, Art. no. 107494.

[5] H. Zhang, J. Cheng, L. Zhang, Y. Li, and W. Zhang, "H2GNN: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3435–3442, Apr. 2022.

[6] S. Swain, P. M. Khilar, and B. R. Senapati, "A reinforcement learning-based cluster routing scheme with dynamic path planning for mutli-UAV network," *Veh. Commun.*, vol. 41, 2023, Art. no. 100605.

[7] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective UAV path planning for search and rescue," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5569–5574.

[8] Z. Qadir, M. H. Zafar, S. K. R. Moosavi, K. N. Le, and M. P. Mahmud, "Autonomous UAV path-planning optimization using metaheuristic approach for predisaster assessment," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12505–12514, Jul. 2022.

[9] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5533–5540, Jul. 2021.

[10] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 610–617, Apr. 2019.

[11] S.-W. Cho, J.-H. Park, H.-J. Park, and S. Kim, "Multi-UAV coverage path planning based on hexagonal grid decomposition in maritime search and rescue," *Math.*, vol. 10, no. 1, p. 83, 2021.

[12] K. Kumar and N. Kumar, "Region coverage-aware path planning for unmanned aerial vehicles: A systematic review," *Phys. Commun.*, vol. 59, 2023, Art. no. 102073.

[13] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021.

[14] J. Zelenka and T. Kasanicky, "Outdoor UAV control and coordination system supported by biological inspired method," in *Proc. 23rd IEEE Int. Conf. Robot. Alpe-Adria-Danube Region*, 2014, pp. 1–7.

[15] R. S. Nilsson and K. Zhou, "Method and bench-marking framework for coverage path planning in arable farming," *Biosyst. Eng.*, vol. 198, pp. 248–265, 2020.

[16] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, "Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 3, pp. 1453–1468, Jul. 2021.

[17] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "Darp: Divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robotic Syst.*, vol. 86, pp. 663–680, 2017.

[18] A. A. Adepegba, S. Miah, and D. Spinello, "Multi-agent area coverage control using reinforcement learning," in *Proc. 29th Int. Flairs Conf.*, 2016.

[19] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV coverage path planning under varying power constraints using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1444–1449.

[20] P. T. Kyaw, A. Paing, T. T. Thu, R. E. Mohan, A. V. Le, and P. Veerajagadheswar, "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem," *IEEE Access*, vol. 8, pp. 225945–225956, 2020.

[21] J. Chen, C. Du, Y. Zhang, P. Han, and W. Wei, "A clustering-based coverage path planning method for autonomous heterogeneous UAVs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25546–25556, Dec. 2022.

[22] Y. Cao, Z. Sun, and G. Sartoretti, "Dan: Decentralized attention-based neural network for the minmax multiple traveling salesman problem," in *Proc. Int. Symp. Distrib. Auton. Robotic Syst.*, 2022, pp. 202–215.

[23] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.

[24] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 539–546.

[25] L. Melas-Kyriazi, "Do you even need attention? A stack of feed-forward layers does surprisingly well on imagenet," 2021, *arXiv:2105.02723*.

[26] W. Zhang, J. Pang, K. Chen, and C. C. Loy, "Dense Siamese network for dense unsupervised learning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 464–480.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[28] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[30] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[31] J. Tang, C. Sun, and X. Zhang, "MSTC*: Multi-robot coverage path planning under physical constrain," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 2518–2524.