Multi-Agent Collaborative Exploration through Graph-based Deep Reinforcement Learning

Tianze Luo*, Budhitama Subagdja†, Di Wang‡ and Ah-Hwee Tan*
*School of Computer Science and Engineering
†ST Engineering-NTU Corporate Laboratory
‡Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY)
Nanyang Technological University
Email: tianze001@e.ntu.edu.sg, budhitama@ntu.edu.sg, wangdi@ntu.edu.sg, asahtan@ntu.edu.sg

***Abstract*—Autonomous exploration by a single or multiple agents in an unknown environment leads to various applications in automation, such as cleaning, search and rescue, etc. Traditional methods normally take frontier locations and segmented regions of the environment into account to efficiently allocate target locations to different agents to visit. They may employ ad hoc solutions to allocate the task to the agents, but the allocation may not be efficient. In the literature, few studies focused on enhancing the traditional methods by applying machine learning models for agent performance improvement. In this paper, we propose a graph-based deep reinforcement learning approach to effectively perform multi-agent exploration. Specifically, we first design a hierarchical map segmentation method to transform the environment exploration problem to the graph domain, wherein each node of the graph corresponds to a segmented region in the environment and each edge indicates the distance between two nodes. Subsequently, based on the graph structure, we apply a Graph Convolutional Network (GCN) to allocate the exploration target to each agent. Our experiments show that our proposed model significantly improves the efficiency of map explorations across varying sizes of collaborative agents over the traditional methods.**

***Index Terms*—Graph convolutional networks; Reinforcement learning; Multi-agent map exploration; Multi-robot system**

I. Introduction

Autonomously and efficiently exploring an environment is one of the fundamental research in autonomous robotics such as mowing and cleaning [1], [2], search and rescue [3], reconnaissance [4], mowing, and autonomous deployment [5]. Compared to single agent exploration, exploring environment bymultiple agents can be advantageous in several aspects, such as faster task accomplishment and more fault tolerance [6].

However, the core challenge in multi-agent exploration is how to coordinate the agents behaviors to achieve efficient and effective exploration. Traditionally, for tackling this problem, several exploration algorithms based on the map information have been developed [6]–[8]. In general, a team of robots share common information and a centralized or decentralized algorithm control the exploration strategies for the robots.

The research was partially supported by the ST Engineering NTU Corporate Lab through the NRF corporate lab@university scheme.

The multi-agent map exploration methods evolve as the development of the idea that dispatches agents to different areas for exploration, which significantly improves the exploration efficiency [6], [9]. Methods such as Voronoi multi-agent map exploration [10] utilized this idea and achieved fast map exploration. Following the idea, in our paper, we developed a map segmentation method, which can efficiently partition the map into separate regions. Therefore, robots can explore the environment based on the regions in the map. To achieve effective coordination, agents should be assigned to separate and non-overlapping regions, so that each agent will not repeatedly explore regions that have been covered by other agents. Based on the regions, we further construct a graph representation of the regions, based on which we can analyse the map through a topological approach.

On the other hand, applying machine learning to improve the tasks allocation in multi-agent exploration remains a difficult problem. Deep learning model like Convolutional Neural Network (CNN) and Deep Reinforcement Learning models have been applied in complex domains like video classification, speech recognition, playing Atari games to name a few with remarkable performance [11]–[13]. Meanwhile, directly applying RL on multi-agent map exploration usually lead to the curse of dimensionality problem [14], i.e. the action space is huge upon completing map exploration.

In this paper, we present a graph-based multi-agent reinforcement learning method to address the multi-agent map exploration with deep reinforcement learning. Similar to the traditional approach [10], the environment is partitioned into separate segments. From the segments, a graph representation is constructed wherein each node in the graph corresponds to a segmented region (e.g room, corridor etc.) and every edge represents a path connecting the regions with a weight value indicating the path distance. A deep reinforcement learning model is employed to learn the efficient strategy to allocate agents to different nodes or regions. We adapt the Deep Q Network for learning with the graph-based representation and construct a graph-based multi-agent learning model named MAG-DQN, for the multi-agent environment exploration prob-

lem.

We conduct experiments showing that the proposed MAG-DQN can learn superior allocation strategy that outperforms other baseline models of multi-agent task allocation. We also find that the information shared among the agents in the centralized MAG-DQN make learning better compared to a decentralized version of the model.

II. Proposed Method

A. Problem Formulation

In this paper, we consider the multi-agent task allocation problem. Suppose a team of agents enter an environment with hazards, where the layout is given to the agents. The agents are required to efficiently detect the entire environment to search for victims. The problem and settings are formulated as follows:

Map: The map is two-dimensional grid map, in which each cell either represents the empty space or obstacle.

Agent: Each agent can freely move on the empty cells, and can detect its surrounding within s cells. All agents move continuously until the entire map is detected.

Goal: The ultimate goal is to obtain the minimal time spent, upon completing detecting the entire environment. As each agent moves continuously, we can just calculate the traveling distance per agent, as an indicator for the time spent.

To tackle the problem, we propose a novel graph-based reinforcement learning method. The method consists of two parts. In the first part, we present a hierarchical clustering method to segment the map into small segments, where we construct a sparse graph based on the segment centres. In the second part, we apply the MAG-DQN method to automatically learn the coordination of the agent team. The overall structure of the model is shown in Figure 1.

B. Hierarchical Clustering for Graph Topology Representation of a Map

To offer a good graph topology representation of a map, we propose a novel hierarchical map clustering method to construct a graph representation for a given map. The graph representation reflects the map information in a topology structure, based on which the graph-based reinforcement learning method learns to assign exploration targets to individual agents.

Our method starts with randomly sampling n points $\{(x_i, y_i) \mid i \in [1, n]\}$ on the empty area of the map, and these points are to be clustered by two-layer hierarchical Adaptive Resonance Theory (ART) network.

The ART network [15] is a vector classifier which accepts an input vector and classifies it into one of the categories depending upon which of the stored pattern it resembles the most. Here we adapt the ART architecture [15]: we modify the matching criteria to be Euclidean distance based. Each time an input point $\{(x_i, y_i) \mid i \in [1, n]\}$ is sampled, its Euclidean distance to every cluster's centroid is measured. This is conducted by an ART neural network to find the nearest cluster (the closest match) under a criteria Cri_1. In Cri_1, if the matching value of a cluster is found to be above a certain threshold (vigilance), the highest one will be selected as the cluster that include the point which leads to the update of its centroid. If no match is found (all clusters are too far), a new cluster is created with the point under the evaluation is set as the centroid.

Each cluster is represented by $\{(\hat{x}_j, \hat{y}_j) \mid j \in [0, m]\}$, where $\hat{x}$ and $\hat{y}$ are the coordinates of the centroid. If a satisfied centroid is found, we merge the input with the corresponding cluster and redefine the centroid. If no match is found, we create a new node in the lower layer of the ART, which takes the coordinate of the input point. The clustering from the lower level layer to the upper-level layer is a similar process but under matching criteria Cri_2. In Cri_2, a cluster is also selected or created according to a vigilance threshold. However, the selection of an upper level cluster only occurs if there is no obstruction (e.g walls, doors, obstacles) between the lower level centroid and the selected upper level central point.

The centroid of a cluster can be updated as follows:

$$\hat{x}^{'} = (N * \hat{x} + x) / (N + 1), \hat{y}^{'} = (N * \hat{y} + y) / (N + 1) \quad (1)$$

where $\hat{x}^{'}$ and $\hat{y}^{'}$ denote the updated coordinate of the centroids, and N represents the number of elements within a cluster. The hierarchical self-organized segmentation structure and complete algorithm are shown in Figure 2 and Algorithm 1 respectively.

Algorithm 1 Segmentation Algorithm

```
Pre-process the map
Detect and mark the doors on the map
Randomly drop n dots in the map
for i in dots do
    for j in LowerLevelCluster do
        if Satisfy C_1 then
            Update Centroid of j and break
        else
            Create new LowerLevelCluster and break
        end if
    end for
end for
for i in LowerLevelCluster do
    for j in UpperLevelCluster do
        if Satisfy C_2 then
            Update Centroid of j and break
        else
            Create new UpperLevelCluster and break
        end if
    end for
end for
Construct the graph G based on the segment centers
Return segment centers and graph G
```

From map to graph representation. After we perform the clustering, we shall get a set of centroids of regions $\{(x_i, y_i) \mid i \in [1, n]\}$, where x_i and y_i represent the location of the region center i on the map. We then compute the path distance $\{d_{i,j} | i, j \in [1, n]\}$ between any two nodes using the A^* method, which is a fast and efficient path planning method for the two dimensional map. The path distance between any

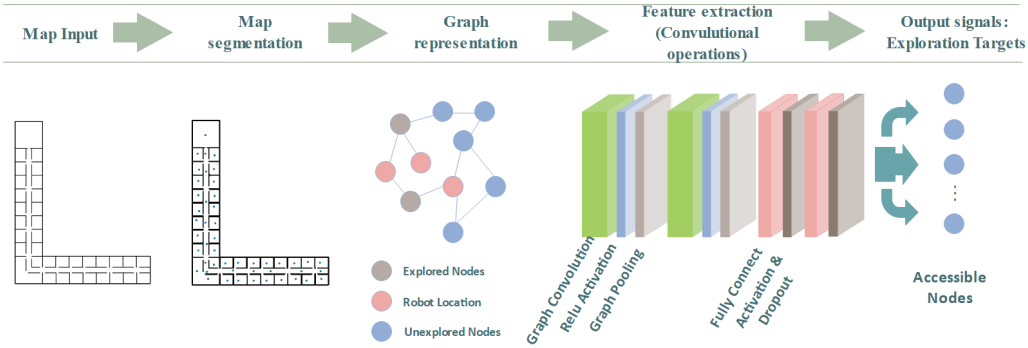


Fig. 1: The processing pipeline of MAG-DQN.

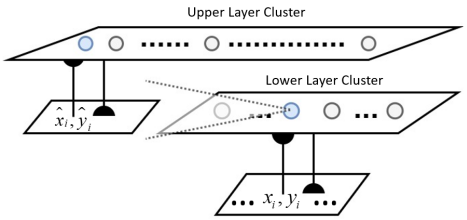


Fig. 2: The structure of Hierarchical Self-Organized Map Segmentation. Given an input point (x_i, y_i), if it is matched to the nearest centroid in the lower layer cluster, the coordinate of that centroid is updated. If no match is found, a new node will be created. The matching and updating process from the lower layer cluster to the upper layer cluster is similar but under a different criteria of matching.

two nodes are taken as the weight of the edge between them. The weight of the edge is calculated by the equation:

$$W_{i,j} = \begin{cases} \frac{d_{i,j} - d_{min}}{d_{max} - d_{min}}, & d_{i,j} <= \lambda, \\ 0, & d_{i,j} > \lambda. \end{cases}$$

The λ is a adjustable parameter to control the density of the graph. We set $\lambda = 0.15$ in our method as we want relatively sparse graph in which a segment only connects to its surrounding segments.

Therefore, the map is transformed to an undirected and weighted attributed graph G = (V, E, W), where V = $\{v_1, v_2, ..., v_n\}$ is the set of n nodes. $E \subseteq V \times V$ is the set of edges, and the weight attribute $W_{i,j}$ represents the weight value of the edge (v_i, v_j). The higher the weight value $w_{i,j}$ of vertices v_i and v_j, the nearer the two vertices in the environment.

Sparsity. The graph G is designed to be a sparse graph for simplicity purpose. Each node of the graph connects to its adjacent nodes, and if two nodes are far from each other, there should be no connection between them.

C. GCN and Multi-agent Graph-based DQN

In this subsection, we present the Multi-agent Graph-based DQN (MAG-DQN) model, which involves graph convolutional networks and multi-agent deep Q-learning.

Spectral GCN. The spectral graph convolutional networks (Spectral GCN) performs graph convolutional operations in the spectral domain [16].

Firstly, we compute the normalized Laplacian matrix of the graph G by the equation:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

Where $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{D}$ is an $n \times n$ diagonal matrix where $\mathbf{D}_{i,j} = \sum \mathbf{W}_{i,j}$.

Since the graph Laplacian L is a real positive semi-definite matrix, we can compute the a set of orthonormal eigenvectors $\{u_i | i \in [1, n]\} \in \mathbb{R}^n$ through eigen-decomposition $L = U\Lambda U^T$, where $U = [u_0, ..., u_{n-1}] \in \mathbb{R}^{n \times n}$ is the the set of eigenvectors, also Known as the Fourier basis for the graph Laplacian. $\Lambda = diag([\lambda_0, ..., \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$ is the diagonalized real non-negative eigenvalues, corresponding to eigenvectors.

We follow the spectral convolutions on graphs method [16], where the localized convolutional filters are defined in Equation 2:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k\left(\tilde{\Lambda}\right) \tag{2}$$

$T_k(x), k = [0, ..., K-1]$ is the kth order Chebyshev polinomial $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$, which defines the k-hop convolutional filters. The convolutional operation with normalization can be defined in Equation 3:

$$y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda})x \tag{3}$$

where the input vector x is the value of each node of the graph G , and $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$ is the normalized diagonal matrix. In the following MAG-DQN method, the input vector x is represented by the state vector s.

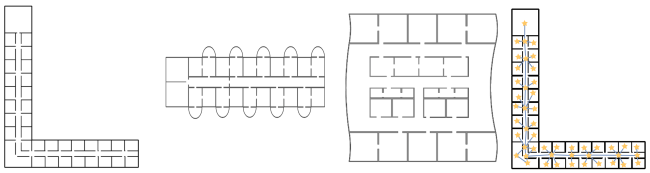


Fig. 3: Figure (a) to (c) shows the map environments for the experiments. (d) The graph generated by the hierarchical clustering from (a) (stars represent segment centers).

Multi-agent deep Q-learning The deep Q-learning, also known as DQN is a famous deep reinforcement learning method, and has been applied in multiple agents environments [17], [18]. DQN usually makes use of the CNN to predict the action-value given a state input and based on the Bellman equation, the Q value function for any agent at state s with action a can be written recursively as $Q^{\pi}(s,a) = \mathbb{E}_s[r(s,a) + \gamma \mathbb{E}_{a' \sim \pi}[Q^{\pi}(s^{'}, a^{'})]]$. The loss function for the agent can be therefore derived as: $\mathcal{L}(\theta) = E_{(s,a,r,s')}[r + \gamma \underset{a'}{max} Q(s^{'}, a^{'}, \tilde{\theta}) - Q(s,a,\theta)]$

We consider multi-agent DQN on the graph G, in which the system involves n agents and each agent i can obtain its state $s_{i,t}$ at time t. At each given state s for agent i, we compute the Q value by considering other agents actions $A = [a_0, ..., a_{i-1}, a_{i+1}, ..., a_{n-1}]$, and A is integrated into state s for computing $Q_i^{\pi_i}(s_i, a_i)$ value for agent i. Here, $Q(s_i, a_i) = F(G, s_i, a_i)$, where F denotes the GCN. The state s_i observed by the agent i is the set of nodes' states, which contain whether the nodes are unexplored, explored, occupied by other agents or occupied by the current agent i. The optimized goal is to minimize the loss function for each agent in the agent team
$\mathcal{L}(\theta) = \sum_{i=1}^{n} E_{(s_i, a_i, r_i, s_i')}[r_i + \gamma \underset{a_i'}{max} Q_i(s_i^{'}, a_i^{'}, \tilde{\theta_i}) - Q_i(s_i, a_i, \theta_i)]$.
In this method, each agent i equips an individual GCN to compute its own $Q_i(s_i, a_i, \theta_i)$ value, and the goal for each agent is to obtain its own optimal Q value function. The model can be categorized as the decentralized learning model.

Centralized learning and decentralized action.

The limitation of the decentralized learning method is that by optimizing the performance of each agent may not achieve optimal results for the overall agent team. Therefore, consider the agent team as a whole

We consider each agent in the team shares the same learning system, i.e. Q-value network, which is updated in a centralized basis by considering all the agents' performance:

$$\mathcal{L}(\theta) = E_{(s_i, a_i, r_i, s_i', i \sim n)}[r + \gamma \underset{a'}{max} Q(i, s_i^{'}, a_i^{'}, \tilde{\theta} - Q(i, s_i, a_i, \theta))] \quad (4)$$

In this way, the system can learn from all the agents' experience, and obtain better performance than the decentralized learning system. The detailed multi-agent Q-learning algorithm is explained in the Algorithm 2.

Algorithm 2 MAG-DQN

```
Initialize experience replay memory M to capacity N
Initialize Graph G
for episode e = 1 to M do
    for step t = 1 to T do
        for agent i = 1 to N do
            Construct state s
            With probability ε select random node v ∈ G as action a_{i,t}
            Otherwise select a_{i,t} = arg max Q(s_t, a_t)
            Execute action a_{i,t} and update state s_{t+1}
            Store transition (s_t, a_{i,t}, r_t, s_{t+1}) to memory
            Sample random minibatch B of transition (s_j, a_{i,j}, r_j, s_{j+1})
        from M
            Update θ by SGD for B with equation (4)
        end for
    end for
end for
Return θ
```

III. Experiments

In this section, we describe our experiments on multi-agent map exploration using MAG-DQN, compared to frontier-based method, graph-based greedy method and graph-based Hungarian method.

A. Environment

To evaluate our proposed method, we use the real-world indoor maps from the ROS room dataset [19], which is a publicly available data set. Examples of maps used in the experiments are shown in Figure 3.

The map is presented as a grid-based map, which is stored as an $n \times m$ matrix, where n is the height and m is the length. Each cell in the grid is assigned to one of the following states:

- **Unexplored**: This cell has not been explored by any agent.
- **Obstacle**: This cell is occupied by obstacles that agent cannot pass.
- **Explored**: This cell is explored by at least one agent.

In the experiment, we assume each agent is equipped with a 360° Lidar with the radius $r = 10$ cells, such that during the exploration, each robot can explore the local area within a radius of r in one step. Map (a) has the size of 145×122 cells, map (b) has the size of 60×180 cells, and map (c)'s size is 107×115 cells. Based on this environment configuration, our proposed model constructs the graphs from the maps and performs multi-agent exploration training.

As for generating graph from a map, the nodes of the graph represent the segment centres, and each edge of the graph represents the path and the distance between two nodes. Figure 3 (d) shows an example of generating segment centres using the hierarchical segmentation methods as described in Section 3.1.

B. Experimental Settings

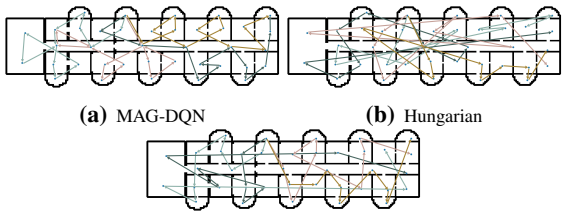
We implement the MAG-DQN method and conduct experiments on the four maps in Figure 3. For each map, we conduct experiments with one to ten agents. We conduct experiments with various parameter settings, and choose the following parameter values:

Hierarchical clustering and graph building. We apply σ= 0.05, λ= 0.8 and γ= 0.15 for the hierarchical clustering method in Algorithm 1. γ is the main parameter to adjust the size of the segments. Here we take $\gamma = 0.15$, so that under the pre-defined sensing range, when a robot moves to a segment center, it is able to explore the whole segment.

MAG-DQN settings The parameter settings for MAG-DQN are as follows. Learning rate: 0.01, Gamma: 0.7, and epsilon greedy: decay from 0.95 to 0.05 in 3000 episodes. Each episode contains at most 100 steps. If within 100 step, the agent team has not completely explored the whole graph, the episode is forced to stop.

TABLE I: The average distance traveled by each agent in three different maps using graph Q-learning with frontier based method, graph-based frontier method, Hungarian method, and MAG-DQN. The numbers in boldface indicate the best performance with the same number of agents

Number of agents	1	2	3	4	5	6	7	8	9	10
Frontier	1121.26	654.15	494.42	412.03	362.95	333.72	302.24	282.54	266.25	263.59
Graph-based Frontier	**846.73**	454.8	370.37	308.62	276.67	263.59	245.07	227.23	223.29	205.99
Hungarian	**846.73**	**431.36**	**339.38**	293.52	270.26	247.21	226.66	222.49	214.07	201.88
MAG-DQN	951.23	517.23	372.83	**287.83**	**266.91**	**243.39**	**218.54**	**210.31**	**197.34**	**188.89**



(a) MAG-DQN **(b)** Hungarian

(c) Graph-based frontier

Fig. 4: motion trace of four agents during the exploration on map in Figure 3b, (a) MAG-DQN, (b) Hungarian method, (c) Graph-based frontier method.

C. Experiments on Baseline Methods

To evaluate our proposed method, we conduct experiments with the same settings for the baseline methods: frontier based coordinated multi-agent map exploration method [6], graph-based frontier method and graph-based Hungarian method. The first one is grid-based method and the other two are graph-based method, which use the same graph generated by the proposed graph construction method.

Frontier based method. [6] The agent team shares the global information, and each agent selects its movement target based on a generally used utility function [6]: $\underset{(t_1,\ldots,t_n)}{\arg\max} \sum_{i=1}^{n}[U(t_i|t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_n) - \beta(V_{t_i}^i)^2]$, where $U_{t_i}^i$ represents the utility of a frontier point t_i for robot i, and $V_{t_i}^i$ represents the cost of robot i travel to the frontier point t_i.

In each decision making cycle, an agent chooses the most favorable frontier point on the map as the target and moves to that target at a constant speed. When the agents thoroughly explore the entire map, the simulation stops and the average traveling distance made by all the agents are recorded as the performance indicator: shorter traveling distance indicate better performance.

Frontier-based graph method. Since there is no well defined frontier-based exploration method for multi-agent exploration on graphs, we define the cost function for agents selection frontier nodes.

$$C(i, v_i) = \begin{cases} d(i, v_i) & unexplored \\ \infty & explored \end{cases}$$

During exploration, each agent selects the frontier nodes with least cost, i.e., the nearest unexplored node. After an agent explores a frontier node, other agents cannot list this node as a target, but can still pass through this node. In the end when the map is fully explored, we take the average travel distance for each agent as the performance measure for this method.

Hungarian method. The Hungarian method [20] is a well-known method to solve multi-agent task allocation problem, and it is often used in multi-agent map exploration problem (reference here). In this experiment, we adopt the Hungarian method to the graph-based multi-agent map exploration problem.

In each step, we use the Hungarian method to allocation agents to a unexplored node, meanwhile the allocation minimize the total travel distance for the agent team in this movement, and the objective function for each allocation is according to $\underset{(t_1,\ldots,t_n)}{min} \sum_{i=1}^{n}[D(i, t_i)]$, where t_i is the unexplored node assigned to agent i. Through this iteration, in each step, the agent team can greedily select the current best action.

IV. Results and Discussion

In this section, we discuss the performance of MAG-DQN, compared with frontier-based method, graph frontier method and Hungarian method.

We conduct 20 trails on each map with number of agents from one to ten, with different starting points for the agent team in each trail. We take the traveling distance per agent as the result of the trial and average the 20 trails' results as the final results for the agent team, which are shown in Table I. Although the MAG-DQN method obtain longer travel distance than the baseline methods when the agent team size is small, the MAG-DQN can learn effective collaborations and outperforms baseline methods when the number of agents in a team is greater than three.

Another interesting finding is that the graph-based methods outperform the frontier-based method. This is mainly because the graph generated by the hierarchical clustering method obtains the essential spatial information of the map, wherein the graph, each node represents a individual region and each edge represents the connection between the two regions. Therefore, collaborative exploration methods based on the graph can achieve better performance than the methods based on the grid map.

We analyze the coordination of the agents through their movement trace during exploration. Figure 4 shows the agents' movement traces with the MAG-DQN method, graph-based frontier method and graph-based Hungarian method. We use the four agents' movement trace to illustrate the difference in the exploration strategies of each method. In this experiment, each robot starts from the same starting point, and only when the map is fully explored, the exploration process will stop.

It can be observed that with the MAG-DQN method, the agents are allocated to explore different regions of the

environment. The interference among agents, i.e. the overlap of the trace, is less than the other two methods. However in Figure (b) and (c), the movement trace is relatively messy and sometimes the agents need to travel a long distance to move to the desired region. Therefore, the task allocation of the agents can be observed more efficient compared to the Graph-based Frontier method and the Hungarian method.

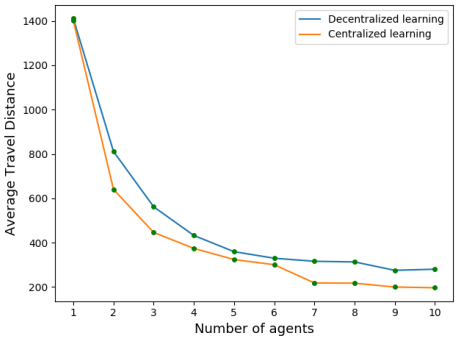


Fig. 5: The performance of the centralized learning MAG-DQN compared with decentralized learning MAG-DQN with one to ten agents. The distance travelled are averaged over the last 100 episode among the 800 episodes training on the map in Figure 3b.

We also compare our centralized learning model with a decentralized learning model, wherein each robot maintains its own deep Q network and does not share the knowledge learned with the others. Each agent starts from a pre-defined entrance and the configurations are the same as the centralized learning model. Figure 5 shows the performance comparison between the centralized learning MAG-DQN with the decentralized learning MAG-DQN. The results show the average performance of last 100 episodes, where the training contains 800 training episodes. The results clearly indicate that, within a certain training episode, the performance of the centralized learning model is better than the decentralized one.

V. Conclusion and Future Work

In this paper, we present a novel deep reinforcement learning model for multi-agent exploration task allocation. It integrates a deep reinforcement learning model that processes a graph representation as the input, with a hierarchical self-organized clustering method to construct the graph, which provides an end-to-end pipeline from situation map to exploration strategies. In our experiments, we show that the proposed method can learn an efficient strategy to allocate the agents to visit and explore particular regions in the environment.

In addition, the experiments show that the centralized learning model can efficiently improve the training efficiency compared with decentralized learning model. Meanwhile, the model is shown to possess good generalization abilities across different number of agents in the system.

In the future, we plan to enhance the graph convolutional model for adaptation to varying graphs. We also plan to investigate more reinforcement learning methods for solving coordinated multi-agent tasks. We will compare their performance and develop methods that are more suitable to combine them with graph convolutional networks models.

References

[1] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 20–96, 2007.

[2] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "Ba*: an online complete coverage algorithm for cleaning robots," *Applied intelligence*, vol. 39, no. 2, pp. 217–235, 2013.

[3] M. Bernard, K. Kondak, I. Maza, and A. Ollero, "Autonomous transportation and deployment with aerial robots for search and rescue missions," *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.

[4] B. Yamauchi, "Autonomous urban reconnaissance using man-portable ugvs," in *Unmanned Systems Technology VIII*, vol. 6230. International Society for Optics and Photonics, 2006, p. 62300S.

[5] G. Tuna, V. C. Gungor, and K. Gulez, "An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters," *Ad Hoc Networks*, vol. 13, pp. 54–68, 2014.

[6] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[7] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Aaai/Iaai*, 2000, pp. 852–858.

[8] W. Burgard, M. Moors, and F. Schneider, "Collaborative exploration of unknown environments with teams of mobile robots," in *Advances in plan-based control of robotic agents*. Springer, 2002, pp. 52–70.

[9] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[10] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1160–1165.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7548–7555.

[15] G. A. Carpenter and S. Grossberg, *Adaptive resonance theory*. Boston, MA: Springer, 2016, pp. 1–17.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[17] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[18] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 443–451.

[19] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1019–1026.

[20] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.