2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Balancing Efficiency and Unpredictability in Multi-robot Patrolling: A MARL-Based Approach

Lingxiao Guo, Haoxuan Pan, Xiaoming Duan, Jianping He

***Abstract*—Patrolling with multiple robots is a challenging task. While the robots collaboratively and repeatedly cover the regions of interest in the environment, their routes should satisfy two often conflicting properties: i) (efficiency) the time intervals between two consecutive visits to the regions are small; ii) (unpredictability) the patrolling trajectories are random and unpredictable. We manage to strike a balance between the two goals by i) recasting the original patrolling problem as a Graph Deep Learning problem; ii) directly solving this problem on the graph in the framework of cooperative multi-agent reinforcement learning. Treating the decisions of a team of agents as a sequence input, our model outputs the agents' actions in order by an autoregressive mechanism. Extensive simulation studies show that our approach has comparable performance with existing algorithms in terms of efficiency and outperforms them in terms of unpredictability. To our knowledge, this is the first work that successfully solves the patrolling problem with reinforcement learning on a graph.**

I. Introduction

Patrolling is the task of persistently visiting locations of interest for surveillance and monitoring purposes. It has a wide range of applications such as smart city [1] and smart defense [2]. A Patrolling strategy, which schedules the visits to different areas by the robots, is essential for the effective execution of the patrolling task. On one hand, the patrolling strategy should be efficient in that the inter-visit times to any area of interest by some robot must be small [3]. On the other hand, the patrol routes should be unpredictable so that an adversary capable of observing the behavior of the patrolling agents cannot accurately predict the future locations of the robots [4], [5]. The adoption of multiple robots in a patrol task is appealing since many locations can be covered simultaneously. However, the design of patrol strategies for a robot team is notoriously difficult and complex. In this paper, we formulate the patrolling problem on a graph, and provide a multi-agent reinforcement learning (MARL) approach for the design of patrolling strategies, which desirably satisfy both the efficiency and unpredictability requirements.

Related work: a) Robotic patrolling. Multi-robot patrolling has been studied extensively in the literature [6]–[8]. A commonly used performance criterion for designing the patrolling strategies is the idleness (a.k.a latency or refresh time), which is the time duration between consecutive visits to locations [3], [9]. Two main ideas are adopted to achieve the coordination among multiple robots: cycle-based solutions [10] and partition-based solutions [11]. In the former, robots are placed equidistantly on a cycle that touches all locations in the environment; in the latter, the environment is partitioned into multiple subregions and each robot patrols one subregion independently. Specific robotic roadmaps such as line graphs and ring graphs are considered in perimeter patrol problems [12], [13]. Besides theoretical studies, heuristic optimization algorithms such as the ant colony optimization algorithm are also utilized [14]. Portugal and Rocha propose a Concurrent Bayesian Learning Strategy (CBLS) in [15], [16]. They develop a probabilistic model to represent the suitability of traveling to a neighboring location from the current location of the robot, and Bayesian learning is used to estimate the probability function. Using the idleness as reward functions, a different line of work adopts learning-based techniques to generate patrolling strategies. Santana et al. use tabular Q-learning to learn patrolling strategies [17], and Jana et al. further replace the Q-table with function approximation [18].

The aforementioned patrolling strategies are usually designed specifically to minimize the idleness and do not take into account unpredictability, which could make them vulnerable to malicious attacks. Some recent works consider various unpredictable strategies based on Markov chains [4], [5], [19] and heuristics [20]. Despite the numerous studies on patrolling problems, existing algorithms usually focus only on one aspect of the problem, either efficiency or unpredictability. In this paper, we aim to design strategies that perform well by both criteria based on MARL approaches.

b) Learning on graphs and MARL. Learning on graphs is a non-trivial task due to the non-Euclidean and flexible structure of graphs. Reinforcement learning manages to solve some combinatorial optimization problems on graphs such as TSP [21] and CSP [22]. Notably, Kool et al. in [23] propose a novel network architecture based on an encoder-decoder structure. The encoder part of this network employs attention to encode the information of all nodes in the graph, and the decoder adopts attention to obtain a query and uses the query to score each node and obtain the probability distribution of the next choice. This work demonstrates the power of reinforcement learning in solving graph problems, and we build our work based on it.

One recent enlightening work in the MARL literature

This work is supported in part by Shanghai Pujiang Program under Grant 22PJ1404900.

L. Guo is with the Department of Civil Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. email: glx15534565855@sjtu.edu.cn.

H. Pan, X. Duan and J. He are with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. email: {panhaoxuan, xduan, jphe}@sjtu.edu.cn .

The source code of the simulation experiments is available at https://github.com/glx15534565855/MARL_to_solve_patrol.

is Heterogeneous-Agent Mirror Learning [24], which is a framework that can spawn numerous cooperative game solving algorithms with monotonicity and Nash convergence guarantees. Algorithms such as Heterogeneous-Agent Trust Region Policy Optimization (HATRPO), Heterogeneous-Agent Proximal Policy Optimization (HAPPO) [25], and multi-agent Transformer (MAT) [25] are all instances of mirror learning. The MAT reveals that the multi-agent advantage function decomposition theorem [26] can be characterized by the decoder in the transformer, with a transferable multi-agent decision-making sequence model. In this paper, we draw on a variant of the decoder architecture in the MAT to model the collaborative decision-making of the patrolling agents.

Contributions: Despite recent advancements in graph learning and MARL, solving a MARL problem on graphs is still a non-trivial task, especially in situations involving continuous online decision-making. We tackle this issue with a carefully designed sequence model, equipped with a graph feature extraction encoder layer and a variant of the MAT decoder. We show that our model achieves good performance in terms of both efficiency and unpredictability in multi-robot patrolling tasks via extensive simulation studies.

Organization: We formulate the multi-robot patrolling problem in Section II. Then, we present the main results including the details of the neural network structure and the training procedure in Section III. Extensive simulation studies are reported in Section IV. Section V concludes the paper.

II. Preliminaries and Problem Formulation

A. Robotic patrolling

We model the patrolling environment with M patrolling locations by a weighted directed graph $G = (V, E, W)$, where $V = \{v_1, \cdots, v_M\}$ is the set of vertices (patrolling locations) and each $v_m \in V$ has a Euclidean coordinate (x_m, y_m), $E \subset V \times V$ is the set of directed edges between vertices, and the edge weights $W = [w_{ij}]$ indicate the travel times along edges. The robot team consists of N robots, and each robot makes a decision about which node to visit next when it reaches a node. We set a fixed total duration T for the patrolling task for training purposes.

The *efficiency* of patrolling strategies is measured using the average global idleness (AGI) defined as follows. Let the instantaneous node idleness (INI) of a node $v_m \in V$ at time t be the time elapsed since the last visit to the node by any robot, i.e., $\text{INI}(v_m, t) = t - t_l$, where t_l is the last time v_m is visited by a robot before time t. The instantaneous global idleness (IGI) is then the average of all nodes' INI's, i.e., $\text{IGI}(t) = \frac{1}{M}\sum_{m=1}^{M} \text{INI}(v_m, t)$. Finally, AGI is defined to be the average of IGI's over the task time duration T, i.e.,

$$\text{AGI} = \frac{1}{T}\sum_{t=0}^{T} \text{IGI}(t).$$

For *unpredictability*, we define a metric called the global visit time entropy H_G. Let the node visit time entropy $H(v_m)$ be the entropy of the distribution of the nodes' inter-visit times defined by

$$H(v_m) = -\sum_i p_m(t^i_{\text{interval}}) \log p_m(t^i_{\text{interval}}), \tag{1}$$

where $p_m(t^i_{\text{interval}})$ is the probability that the inter-visit time to node v_m is equal to t^i_{interval}. In the empirical evaluations, the entropy is computed based on a coarser distribution where inter-visit times are grouped in increments of 10 time steps. The global visit time entropy H_G is then the sum of $H(v_m)$ over all nodes in the graph, i.e., $H_G = \sum_{m=1}^{M} H(v_m)$.

B. Patrolling as MARL

To formalize the N-agent patrolling problem in the framework of multi-agent reinforcement learning (MARL), we define a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$ where

- $\mathcal{S} = \{(o^1, o^2, \cdots, o^N)\}$ is the set of states for N agents where $o^n = (o^{n_1}, o^{n_2}, \cdots, o^{n_M})$ constitutes agent n's state, and each component o^{n_m} of o^n for $n \in \{1, \cdots, N\}$ and $m \in \{1, \cdots, M\}$ provides the information about vertex v_m to the n-th agent as
$$o^{n_m} = (x_m, y_m, \text{NI}(v_m), X^n, Y^n). \tag{2}$$
In (2), (x_m, y_m) and (X^n, Y^n) are the coordinates of the *patrol node* v_m and *agent* n, respectively, and $\text{NI}(v_m)$ is the idleness of node v_m.
- $\mathcal{A} = \{(a^1, a^2, \cdots, a^n, \cdots, a^N)\}$ where a^n denotes the coordinate of the next node selected by the agent n to reach, i.e., $a^n = (X^n_{\text{next}}, Y^n_{\text{next}})$.
- P is a deterministic transition function.
- R is the reward function, which will be given in Section III-B.

At each decision round, the agent will choose among the neighboring nodes of the current location and move there. Therefore, its action space is state-dependent and variable.

III. Main Results

A. Network structure

In this subsection, we present the structure of the proposed network. Overall, the network is divided into two parts, the encoder and the decoder. The encoder uses the Graph Attention Networks (GAT) [27] to encode the agents' state information, producing a GAT feature layer, which is then used to predict the critic and as the feature representation to help the actor make a decision. The Multi-Agent Transformer (MAT) decoder, which is similar to the decoder in [26], outputs the patrol nodes selected by agents in sequence through multi-head attention (MHA) blocks. The overall structure is shown in Fig. 1 and Fig. 2.

1) Encoder: We borrow the basic encoder structure from [23], where the attention encodes the nodes on a fully connected graph and each node needs to query all other nodes. In our case, the patrol map is generally not a complete graph and instead has a specific topology, and the attention on nodes that are not neighbors will cause unnecessary interference and is a waste of computational resources. Thus, we only allow each node to query its neighbors, and we replace the

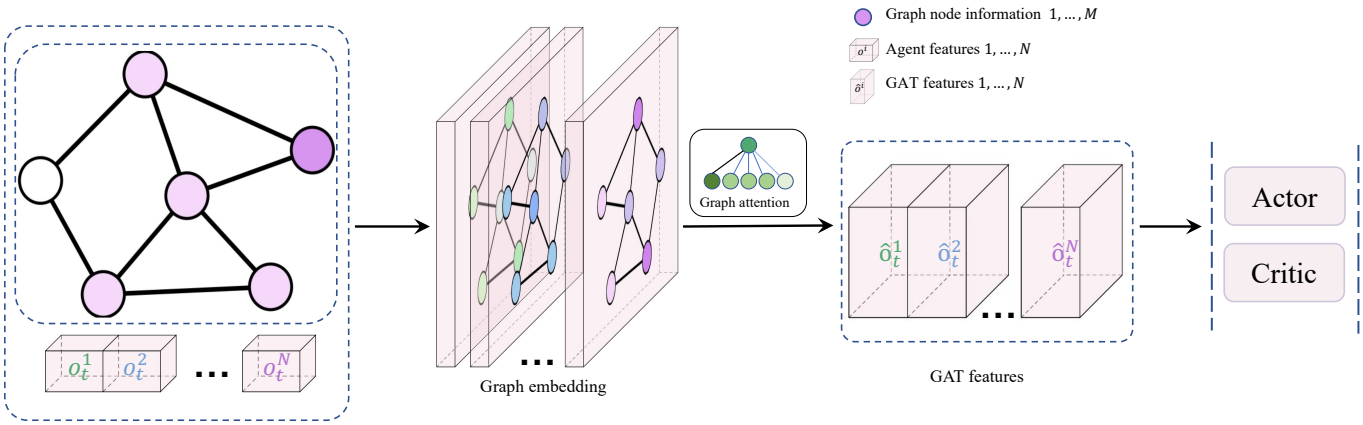


Fig. 1. Graph encoder takes the agents' state information (2) as inputs, and computes the GAT features which is followed by the actor and critic.

attention layer with the GAT [27]. For a specific agent n, each component o^{n_m} of the agent's state in (2) is first encoded by a layer of full connectivity to obtain a d_h dimensional hidden vector. These M hidden layer vectors then enter the GAT to complete further encoding, where the graph information is the topology of the patrol map constituted by the patrol nodes in o^{n_m}'s. Later, the GAT-encoded vectors will enter the decoder to complete the decoding operation. The dimension of the hidden layer vectors remains the same during the whole process. The pipeline of the encoding process is shown in Fig. 1.

GAT Block: We use the GAT for the nodes to share the local feature vectors with neighboring nodes and use the MHA of K heads to capture more relationships between nodes. Given the hidden layer vectors $h_1, \cdots, h_M$ encoding some agent n's state, where h_i is the encoded state component o^{n_i}, the learnable attention value $\alpha_{i,j}^k$ gives how much weight h_i gives to the feature vector h_j at the k-th head, and it can be calculated by

$$\alpha_{i,j}^k = \frac{\exp(\text{ReLU}(\sigma(W^k h_i, W^k h_j)))}{\sum_{s\in\mathcal{N}_i}\exp(\text{ReLU}(\sigma(W^k h_i, W^k h_s)))},$$

where $\mathcal{N}_i$ is the index set for the neighboring nodes of node v_i, $W^k \in \mathbb{R}^{d_h\times d_h}$ is a learnable linear transformation matrix, and $\sigma : \mathbb{R}^{d_h} \times \mathbb{R}^{d_h} \to \mathbb{R}$ is a single-layer feedforward neural network with real-valued output. Once the attention values are obtained, the embedded state component $\hat{o}^{n_i}$'s can be computed by

$$\hat{o}^{n_i} = \frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{i,j}^k W^k h_j.$$

Finally, we obtain the output of the GAT block $\hat{o}^n = (\hat{o}^{n_1}, \cdots, \hat{o}^{n_M})$.

The GAT block implemented in this paper consists of the following sub-layers: a GAT layer that adopts an MHA mechanism, a fully connected feedforward layer with ReLU activation functions and another fully connected feedforward layer. Each sub-layer adds a skip-connection [28] and layer normalization [29]. As shown in Fig. 1, after encoded by the GAT block, the hidden layer vectors then enter the fully connected feedforward layer to obtain an estimate of the current state value.

2) Decoder: The decoder we use combines the advantages of attention shown in [26] and [23], and can be applied to both synchronous and asynchronous decision scenarios. In order to illustrate the network details in an organized manner, we will first present the network output process in the synchronous decision scenario, and then describe how to extend the autoregressive output mechanism to be compatible with both synchronous and asynchronous decision scenarios.

Synchronous decoder : Our decoder adopts the idea that agents make decisions in sequence [26]. In the synchronous scenario, all the agents arrive at the patrol nodes at the same time and make the next round of patrol nodes selection. Under the autoregressive mechanism, the agents make decisions sequentially in each round, and each agent outputs its own action based on actions of other agents who have already chosen an action. Suppose it is agent n's turn to make a decision. We use an action buffer to store the actions $a_1 \sim a_{n-1}$ that have been made by agents in the current round. The action buffer first enters an MHA block for encoding. After that, M copies of the embedded actions form a new tensor whose dimension matches that of the encoded state information $\hat{o}^n = (\hat{o}^{n_1}, \hat{o}^{n_2}, \ldots, \hat{o}^{n_M})$ by the GAT, and it enters the next MHA block along with $\hat{o}^n$. The output of the MHA block is the hidden state $(h'_1, h'_2, \ldots, h'_M)$. Then, the context embedding h^c is obtained by $h^c = (\frac{1}{M}\sum_{m=1}^{M} h'_m, h'_l)$ [23], where l is the index of the node v_l the agent is located at. We further compute a new context embedding by using the MHA mechanism. The query is from h^c, and the keys and values come from $(h'_1, h'_2, ..., h'_M)$, but only vectors corresponding to neighboring nodes of v_l are chosen, i.e.,

$$q^c = W^Q h^c \quad k_i = W^K h'_i, \quad v_i = W^V h'_i, \quad i \in \mathcal{N}_l,$$

where W^Q, W^K and W^V are trainable parameters. The vectors q^c, k_i's and v_i's enter a single-layer attention network and we obtain a new vector $\hat{h}^c$ [23]. Then we use it as a probability query, and compute probability keys from h'_i for $i \in \mathcal{N}_l$:

$$\hat{q}^c = W^{Q'}\hat{h}^c, \quad k'_i = W^{K'} h'_i,$$

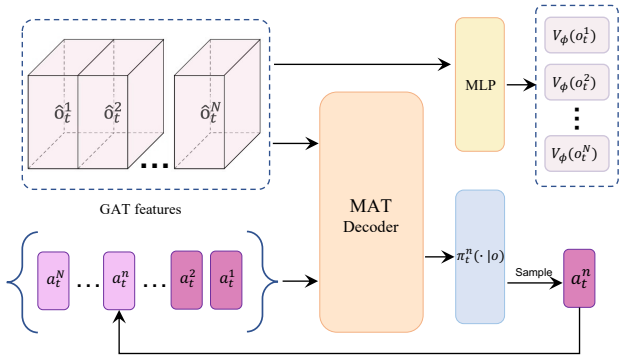


Fig. 2. The decoder uses the GAT features extracted by the encoder and a sequence of actions as input. The top part of this figure indicates the critic mechanism, and the bottom part shows the process of sequential decision-making.

where $W^{Q'}$ and $W^{K'}$ are trainable parameters. Then we compute the compatibilities u_i by:

$$u_i = C \cdot \tanh\left(\frac{\hat{q}^{c\top} k_i'}{\sqrt{d_h}}\right), \quad i \in \mathcal{N}_l,$$

where C is a constant. Finally, the output probability vector is computed by the softmax function.

Asynchronous decoder: In the asynchronous scenario, while some agents are still patrolling on the edges and are unable to make new decisions, others reach the patrol nodes and need to make the next node selection. Patrolling with asynchronous policies is closer to the real-world scenario. In order to adapt to asynchronous decision making, the autoregressive mechanism of the decoder needs to be improved as follows. In a decision round as some agents reach the patrol nodes, the decisions of agents on edges are fixed as their most recent decisions. The action buffer will first store these actions. Then, the action buffer is fed to the decoder to determine the new actions for those agents at vertices in an autoregressive manner. Note that this modification is not needed in the synchronous scenario because all agents are either on the edges or at the vertices. The asynchronous autoregressive mechanism automatically degenerates into the synchronous one when the agents patrol in a synchronous manner. Thus, the network is compatible with both synchronous and asynchronous decision-making scenarios.

B. Training

Our goal is to maximize the patrol efficiency over time, i.e.,

$$\max_{\pi} J(\pi) = \max_{\pi} \sum_{t=0}^{T} \mathrm{IGI}(t).$$

Naturally, we design a reward function $R(o_t, a_t) = -\mathrm{IGI}(t)$. After agents take a joint action, they receive a real-time performance evaluation of their joint action.

The encoder network, whose parameters we denote by ϕ, plays the role of the critic network to estimate the value $V_\phi\left(o_t^n\right)$ of agent n's current state. It is trained with Bellman equation loss:

$$L_{\text{Encoder}}(\phi) = \frac{1}{TN} \sum_{n=1}^{N} \sum_{t=1}^{T} \left[R\left(o_t, a_t\right) + \gamma V_{\bar{\phi}}\left(o_{t+1}^n\right) - V_\phi\left(o_t^n\right)\right]^2,$$

where the parameter $\bar{\phi}$ of a target network is introduced to make the value representation more stable [30] and $\gamma \in (0,1)$ is a discount factor in the training process. The decoder, whose parameters is denoted by θ, serves as the actor of the model. We train the decoder using the Proximal Policy Optimization (PPO) [31] with the loss function

$$L_{\text{Decoder}}(\theta) = \frac{1}{TN} \sum_{n=1}^{N} \sum_{t=1}^{T} \min\left(r_t^n(\theta)\hat{A}_t, \operatorname{clip}\left(r_t^n(\theta), 1 \pm \epsilon\right)\hat{A}_t\right),$$

where $r_t^n(\theta)$ is the importance sampling ratio defined by

$$r_t^n(\theta) = \frac{\pi_\theta^n\left(a_t^n \mid o_t^n, a_t^{1:n-1}\right)}{\pi_{\theta_{\text{old}}}^n\left(a_t^n \mid o_t^n, a_t^{1:n-1}\right)},$$

and $\hat{A}_t$ is an estimate of the current joint advantage function. One can take $\hat{V}_t = \frac{1}{N}\sum_{n=1}^{N} V_\phi\left(o_t^n\right)$ as an estimate of the current joint value function and approximate $\hat{A}_t$ by the temporal difference error:

$$\hat{A}_t = R\left(o_t, a_t\right) + \gamma V_{t+1} - V_t.$$

C. Performance analysis

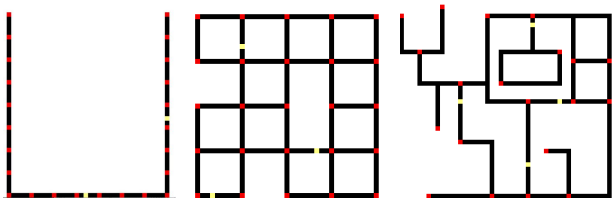
The monotonic improvement guarantee and Nash Equilibrium: During training process, agent optimizes the PPO objective based on the actions of previous agents' decisions. Thus, the monotonic improvement guarantee follows from that of HAPPO [25]. It has been shown in HAPPO that the sequential update scheme allows the joint strategy to converge to a Nash equilibrium [25]. This provides theoretical guarantees for the optimization of patrol efficiency [25, Theorem 2, 3].

In terms of unpredictability, since the decoder outputs a probability vector over the set of actions, the agents select locations to visit next in a random manner. The randomness of the agents' decisions at the individual level further results in greater unpredictability at the team level. We evaluate the unpredictability of the team behavior using the metric defined in (1) in the simulation section.

IV. Simulation

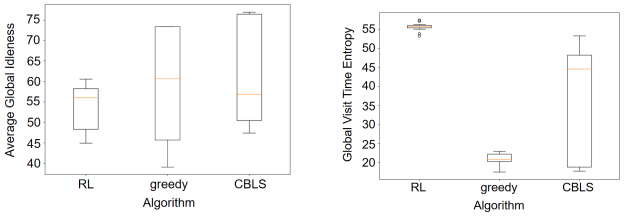
A. Simulation setup

We generate the patrol maps using the grid world environment developed in [32]. The maps are then abstracted into graphs, on which we run our algorithm directly. Agents make decisions at the level of the graph and take specific actions on the grid world map. We test our algorithms on three types of maps as shown in Fig. 3: the strip map, the grid map and the irregular map. On all these maps, the yellow squares indicate the agents, while the red ones represent patrol nodes.



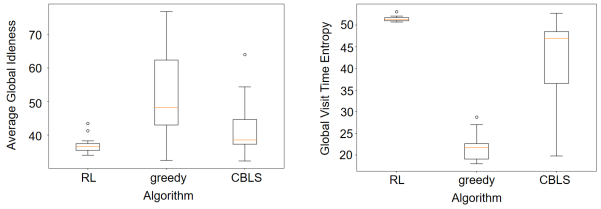
(a) strip map (b) grid map (c) irregular map

Fig. 3. Patrol maps in the simulation where the red squares are patrol nodes and yellow squares are patrol agents.



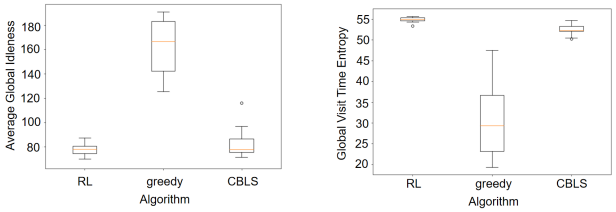
(a) strip map AGI (b) strip map H_G

Fig. 4. Strip map results: all three methods have similar performance in terms of the patrol efficiency. However, in terms of unpredictability, RL based method outperforms the other two by a large margin.



(a) grid map AGI (b) grid map H_G

Fig. 5. Grid map results: the greedy approach performs badly in AGI and H_G. Our approach beats CBLS in both criteria.



(a) irregular map AGI (b) irregular map H_G

Fig. 6. Irregular map results: the greedy approach shows bad performance in terms of both AGI and H_G compared with the other two methods. Our approach beats CBLS by a small margin.

The distances between patrol nodes are integer-valued. We conduct the training with $T = 1000$ steps per episode and test our algorithm with $T = 5000$ steps per episode.

A decision round begins whenever a robot reaches its target node, and all robots on nodes make decisions sequentially. The strip map and the grid map are used to test the performance of our algorithm in the synchronous decision scenario, where the distances between all pairs of neighboring patrol nodes on each map are the same. Thus, in every decision round, all the robots arrive at the vertices simultaneously and make decisions. The irregular map is used to test the performance of the model in the asynchronous decision scenario. At each decision round, only the robots who reach the vertices need to make decisions. As a reference for the convergence rate of the training procedure, our algorithm converges after about 200 episodes for 4 agents in the irregular map.

B. Comparison studies

We compare our algorithm with two other existing algorithms. One is the greedy method, in which each agent greedily selects the node with the largest idleness among its neighbors to visit next. Another one, named CBLS algorithm [15], [16], is an existing distributed cooperative patrolling algorithm. For testing, we conduct the patrolling experiments for 50 times on each map where the agents are randomly placed on the map initially, and we collect the node idleness and inter-visit times statistics. We then report the resulting AGI and the entropy of the patrolling strategies in box plots.

Strip map: Fig. 4 shows the results of 2 agents patrolling on a strip map with 24 nodes. It can be seen that, in terms of the patrol efficiency, RL achieves the minimum AGI on the strip map on average, though the margin is not significant. In addition, the AGIs of the greedy method and CBLS have large variances, which implies that their performance fluctuates. To have better insights for the greedy method, let us consider the case when the two agents start from the midpoint and the right end of the strip map, respectively. In the beginning, the agent at the midpoint has a 0.5 probability of going left and a 0.5 probability of going right; while the agent at the end can only choose to travel down. Obviously, going right is a worse choice as it would increase the inter-visit times for the nodes on the left part of the map. In fact, if the right is chosen at the beginning, the greedy method will fall into a bad cyclic behavior, where the two agents patrol at a small distance apart all the time. In terms of unpredictability, the greedy method performs worst since it almost follows a deterministic pattern. The CBLS algorithm, although considering the degree of nodes and the number of historical visits in the decision, is hand-crafted based on intuition and cannot guarantee high randomness either. In contrast, the randomness of the neural network can well prevent the agents from falling into bad cyclic behaviors and does little harm on the patrol efficiency.

Grid map: Fig. 5 shows the results of 3 agents patrolling on a grid map with 25 nodes. In terms of the patrol efficiency, RL achieves comparable performance with CBLS and both outperform the greedy method. In terms of unpredictability, the H_G of the greedy method is always low, and the H_G of CBLS has a large variance. In contrast, the H_G of RL is always at a high value, and the strategy achieves good unpredictability.

Irregular Map: The map consists of 33 nodes, and 20 of them are set as patrol nodes. A total of 4 agents are placed on the map. As shown in Fig. 6, in terms of both the patrol

efficiency and unpredictability, RL achieves comparable performance with CBLS and far outperforms the greedy method.

From the simulation studies summarized above, we can see that our method achieves good performance in terms of both the patrol efficiency and unpredictability criteria in all three maps representing very different topologies, while the greedy method and CBLS behave well in only some of them. This shows that our method has desirable generalizability in the sense that it can be applied in different patrolling scenarios.

V. Conclusion

This paper studies the multi-robot patrolling problem on a graph utilizing the multi-agent reinforcement learning (MARL) techniques. We first formulate the problem in the framework of MARL. To directly learn the patrolling strategies on graphs, we propose novel neural network structures that take into account the sequential nature of the problem. The patrolling strategies designed using our approach achieves a balance between the efficiency and unpredictability as evidenced by the simulation studies. Our strategy can also be applied to scenarios where robots make decisions asynchronously.

References

[1] S. Witwicki, J. C. Castillo, J. Messias, J. Capitan, F. S. Melo, P. U. Lima, and M. Veloso, "Autonomous surveillance robots: A decision-making framework for networked multiagent systems," *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 52–64, 2017.

[2] P. Fazli, A. Davoodi, and A. K. Mackworth, "Multi-robot repeated area coverage," *Autonomous Robots*, vol. 34, no. 4, pp. 251–276, 2013.

[3] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Beijing, China, Sep. 2004, pp. 302–308.

[4] M. George, S. Jafarpour, and F. Bullo, "Markov chains with maximum entropy for robotic surveillance," *IEEE Transactions on Automatic Control*, vol. 64, no. 4, pp. 1566–1580, 2019.

[5] X. Duan, M. George, and F. Bullo, "Markov chains with maximum return time entropy for robotic surveillance," *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 72–86, 2020.

[6] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre, "Recent advances on multi-agent patrolling," in *Brazilian Symposium on Artificial Intelligence*, Sao Luis, Maranhao, Brazil, Sep. 2004, pp. 474–483.

[7] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 894–903, 2019.

[8] N. Basilico, "Recent trends in robotic patrolling," *Current Robotics Reports*, vol. 3, no. 2, pp. 65–76, 2022.

[9] A. Machado, G. Ramalho, J. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," in *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, Bologna, Italy, Jul. 2002, pp. 155–170.

[10] P. Afshani, M. de Berg, K. Buchin, J. Gao, M. Loffler, A. Nayyeri, B. Raichel, R. Sarkar, H. Wang, and H.-T. Yang, "On cyclic solutions to the min-max latency multi-robot patrolling problem," *arXiv*, 2022. [Online]. Available: http://arxiv.org/abs/2203.07280

[11] V. Sea, A. Sugiyama, and T. Sugawara, "Frequency-based multi-agent patrolling model and its area partitioning solution method for balanced workload," in *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Delft, The Netherlands, Jun. 2018, pp. 530–545.

[12] N. Agmon, G. A. Kaminka, and S. Kraus, "Multi-robot adversarial patrolling: Facing a full-knowledge opponent," *Journal of Artificial Intelligence Research*, vol. 42, no. 1, pp. 887–916, 2011.

[13] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis and approximation algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 592–606, 2012.

[14] H. Chen, T. Cheng, and S. Wise, "Designing daily patrol routes for policing based on ant colony algorithm," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-4/W2, pp. 103–109, 2015.

[15] D. Portugal and R. P. Rocha, "On the performance and scalability of multi-robot patrolling algorithms," in *IEEE International Symposium on Safety, Security, and Rescue Robotics*, Kyoto, Japan, Nov. 2011, pp. 50–55.

[16] D. Portugal and R. P. Rocha, "Cooperative multi-robot patrol with Bayesian learning," *Autonomous Robots*, vol. 40, no. 5, pp. 929–953, 2016.

[17] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch, "Multi-agent patrolling with reinforcement learning," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, USA, Jul. 2004, pp. 1122–1129.

[18] M. Jana, L. Vachhani, and A. Sinha, "A deep reinforcement learning approach for multi-agent mobile robot patrolling," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 4, pp. 724–745, 2022.

[19] X. Duan and F. Bullo, "Markov chain-based stochastic strategies for robotic surveillance," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 243–264, 2021.

[20] C. D. Alvarenga, N. Basilico, and S. Carpin, "Time-varying graph patrolling against attackers with locally limited and imperfect observation models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, Nov. 2019, pp. 4869–4876.

[21] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L. Rousseau, "Learning heuristics for the TSP by policy gradient," in *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Delft, The Netherlands, Jun. 2018, pp. 170–181.

[22] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 142–13 155, 2021.

[23] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *The International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[24] J. G. Kuba, X. Feng, S. Ding, H. Dong, H. Wang, and Y. Yang, "Heterogeneous-agent mirror learning: A continuum of solutions to cooperative MARL," *arXiv*, 2022. [Online]. Available: http://arxiv.org/abs/2208.01682

[25] J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust region policy optimisation in multi-agent reinforcement learning," in *The International Conference on Learning Representations*, Virtual, Apr. 2022.

[26] M. Wen, J. G. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, "Multi-agent reinforcement learning is a sequence modeling problem," in *Neural Information Processing Systems*, New Orleans, LA, USA, Dec. 2022.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *The International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv*, 2016. [Online]. Available: https://arxiv.org/abs/1607.06450

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[32] A. Baskaran, "rl_multi_agent," *GitHub*. [Online]. Available: https://github.com/amrish1222/rl_multi_agent