International Journal of Intelligent Robotics and Applications (2022) 6:724–745
https://doi.org/10.1007/s41315-022-00235-1

REGULAR PAPER

A deep reinforcement learning approach for multi-agent mobile robot patrolling

Meghdeep Jana[1] · **Leena Vachhani**[1] · **Arpita Sinha**[1]

Received: 21 June 2021 / Accepted: 30 March 2022 / Published online: 4 May 2022

Abstract
Patrolling strategies primarily deal with minimising the time taken to visit specific locations and cover an area. The use of intelligent agents in patrolling has become beneficial in automation and analysing patterns in patrolling. However, practical scenarios demand these strategies to be adaptive in various conditions and robust against adversaries. Traditional Q-learning based patrolling keeps track of all possible states and actions in a Q-table, making them susceptible to the curse of dimensionality. For multi-agent patrolling to be adaptive in various scenarios represented using graphs, we propose a formulation of the Markov Decision Process (MDP) with state-representations that can be utilised for Deep Reinforcement Learning (DRL) approaches such as Deep Q-Networks (DQN). The implemented DQN can estimate the MDP using a finite length state vector trained with a novel reward function. Proposed state-space representation is independent of the number of nodes in the graph, thereby addressing scalability to graph dimensions. We also propose a reward function to penalise the agents for lack of global coordination while providing immediate local feedback on their actions. As independent policy learners subject to the MDP and reward function, the DRL agents formed a collaborative patrolling strategy. The policies learned by the agents generalise and adapt to multiple behaviours without explicit training or design to do so. We provide empirical analysis that shows the strategy's adaptive capabilities with changes in agents' position, non-uniform node visit frequency requirements, changes in a graph structure representing the environment, and induced randomness in the trajectories. DRL patrolling proves to be a promising patrolling strategy for intelligent agents by potentially being scalable, adaptive, and robust against adversaries.

Keywords Multi-agent · patrolling · Markov decision process · Reinforcement learning · Deep learning

1 Introduction

Patrolling is the task of travelling through or around an area at regular intervals for monitoring and supervising important locations. Patrolling algorithms aim at simulating this task with time duration for every visited location to the extent possible. Thus, using multiple patrolling agents to improve patrolling is advantageous in covering multiple patrol locations while the duration can be further minimised compared to the single agent. The patrolling problem has applications in various scenarios, such as surveillance and vigilance for police patrolling in a particular block or neighbourhood. Recent developments suggest that patrolling problem can aid by minimising human intervention during disasters or pandemics by coordinating multiple robots for search and surveillance in high-risk areas (Maza et al. 2011). Hence, developing patrolling strategies for such realistic scenarios becomes of importance. The patrolling scenario is represented as a graph where patrolling locations are nodes, and the connections are represented as edges.

In an autonomous multi-agent patrolling (MAP) system, each agent is assigned the next location to visit once it finishes the previous assignment. However, coordination and collaboration of these multiple agents to optimally patrol the area is a challenge. Also, realistic scenarios in patrolling demand the system to be adaptive of environment changes, goal requirements and against adversaries. Earlier works on MAP strategies can be categorised as deterministic and stochastic strategies. Deterministic strategies such as reactive agents (Machado et al. 2002), single cycle Travelling

✉ Meghdeep Jana
meghdeepj@gmail.com

[1] Autonomous Robots and Multi-agent Systems Laboratory, Systems and Control Engineering, Indian Institute of Technology Bombay, Mumbai, India

Salesman Problem (TSP) agents (Chevaleyre et al. 2004), or negotiator mechanism-based agents (Menezes et al. 2006). While deterministic strategies claim performance under ideal conditions, stochastic strategies work for representing realistic scenarios, as mentioned above. Existing stochastic strategies include probabilistic Multilevel Sub-graph Patrolling (MSP) Portugal and Rocha (2011) and Reinforcement Learning (RL) strategies (Santana et al. 2004; Mao and Ray 2014; Lauri and Koukam 2014). Due to the absence of a training data-set, RL has provision for unbiased exploration when searching in solution space through experience.

This work investigates a Deep Reinforcement Learning (DRL) approach for multi-agent patrolling to address scalability to any graph dimension. Our approach is to identify multi-agent patrolling requirements and reflect these requirements in terms of reward function or solutions. Further, an interesting observation on the behaviour of patrolling agents is observed. We observe that the RL agents have learned cyclic trajectories to be most efficient in maximising the reward. A similar result has been observed in work by Almeida et al. (2004) where single-cycle patrolling outperformed other State-of-the-art approaches. However, an investigation for adaptive capabilities with changes in graph structure by extension to TSP-based cyclic strategy is unsuitable because the formed Hamiltonian cycle is graph structure specific. In this work, we contribute in the follows ways:

- To tackle scalability and curse of dimensionality in RL-based multi-agent patrolling, we propose a Markov Decision Process (MDP) formulation with state representations that can be utilised for DRL approaches such as Deep Q-Networks (DQN). Due to the state estimation using Deep Neural Networks, the state-space dimensions are no longer dependent on the number of nodes in the graph and scales well with the number of agents.
- We introduce a novel reward function to penalise the agents for lack of global coordination while providing immediate local feedback on their actions.
- We analyse the patrolling algorithm for attaining cyclic patterns while coordinating multiple patrolling agents. This steady-state of a cyclic pattern is analysed in terms of the time interval between successive visits of locations. Whether reaching the steady-state is important or not for a patrolling strategy depends on the use-case. A method to induce randomness to disturb the steady-state is also proposed to mitigate the vulnerability of cyclic patterns from the perspective of adversarial attacks that may observe repetitive actions due to the cyclic pattern.
- We show that the proposed method can satisfy the exhaustive requirements of a good patrolling strategy using multiple agents as in (1) it is adaptive to changes in agents' position, (2) it is deployable for any dimension of underlying graph structure representing the environment, (3) it can cater to the non-uniform requirements on node visit frequency, and (4) it is capable of inducing randomness in the agents' trajectories.

The structure of the remainder of this paper is as follows. A review of previous literature on patrolling is presented in Sect. 2. A formal definition of the patrolling problem in Sect. 3, few concepts of reinforcement learning are introduced in Sect. 4. We then introduce our deep reinforcement learning model for the multi-agent patrolling problem in Sect. 5. Lastly, in Sect. 6, the model is empirically analysed and evaluated in a multi-agent patrolling simulation environment for representative graphs and several agents and environment conditions.

2 Related work

The pioneering work on the patrolling problem by Machado et al. (2002) evaluates several architectures for MAP with various configurations such as agent communication, perception etc. The defined 'node idleness' influenced the agents' decisions and two efficient strategies were concluded: (i) Conscientious Reactive (CC) (ii) Negotiator Cognitive Coordinated (Neg-CC). The CR agents greedily move to the neighbouring node with maximum idleness, where each agent keeps track of the node idleness values individually. The Neg-CC agents access the node idleness values from a central blackboard, and a negotiator mechanism helps resolve node conflicts by rerouting other agents to a different node. A theoretical analysis of Multi-agent TSP cyclic patrolling strategy was proposed by Chevaleyre et al. (2004). A dynamic graph partitioning based MAP algorithm was introduced by Elor and Bruckstein (2009); Portugal and Rocha (2010). A model-free Tabular Q-learning based RL approach for the patrolling task was introduced by Santana et al. (2004). Comparative results on communication architectures of Grey Box Learner Architecture (GBLA) and Black Box Learner Architectures (BBLA) are also presented by Santana et al. (2004).

A survey on *Multi-agent patrolling* (Almeida et al. 2004) compared several state-of-the-art patrolling algorithms based on their patrolling performance normalised by the number of agents. Algorithms such as single cycle TSP (SC), Conscientious Reactive (CR), Heuristic Pathfinder Cognitive Coordinated (HPCC), Negotiation Mechanisms (Neg-CC), and RL approaches were considered. HPCC and SC algorithms outperformed Q-learning based GBLA. Both HPCC and SC need tuning and initialisation for different graphs and agent configurations. Like SC, HPCC also has limitations in huge graphs because of path-finding complexity. GBLA showed to be more adaptive than both HPCC and SC. However, the performance of GBLA is limited by

its training cost and dimensionality. The CR and Neg-CC strategies resulted in collaborative patrolling strategies that aren't limited by the graph structure or agent configurations.

More recent work on patrolling is focused on the autonomous discovery of the patrolling strategy by the agents. The algorithm proposed by Wiandt and Simon (2018) focuses on autonomous partitioning of nodes in the graph for patrolling. Uncertainties in the MAP were modelled using Partially Observable Markov Decision Process (POMDPs) Chen et al. (2015). Hu and Zhao (2010) developed an RL solution to the Multi-agent patrolling policy. Other works include reducing the curse of dimensionality in RL algorithms and improving the adaptive capabilities of agents. The extended-GBLA architecture by Lauri and Koukam (2014) gives importance to the local information in the RL state-representation to reduce state-dimension. The "worst-case idleness" is selected as the reward function that performed better than the visit idleness reward function. It is also shown that the RL approaches have an advantage over deterministic strategies as Extended-GBLA has better adaptive capabilities even with changes in agent population and graph topology. Also that RL agents can be randomly initialised anywhere in the graph, unlike SC or HPCC. Also, strategies have been developed to address the patrolling of significant locations in the environment (Baglietto et al. 2009).

Patrolling strategies are challenged by external adversarial agents that intend to attack unattended nodes in the graph. Full-knowledge adversaries are aware of the patrolling policy and can exploit that information, hence making deterministic strategies more vulnerable to them (Agmon et al. 2011). Further, it develops a patrolling strategy focusing on the detection of adversaries at weak locations. Another direction (Elmaliach et al. 2009) on non-uniform frequency requirements has been explored to address the handling of events at specific locations that disrupt the patrolling strategy. The possibility of using reinforcement learning-based methods on practical implementations using small-size vehicles have also been explored. Recent examples are DRL based single-agent aerial drone patrolling by Piciarelli and Foresti (2019).

The MAP problem needs to address formation of a collaborative policy as compared to the single-agent counterparts. Further investigation to address scalability with graph dimension has direct benefits in solving multi-agent scenarios. In a recent work(Luis et al. 2021), Deep RL for state estimation in a graph environment by multiple agents has been developed. The extension to multi-agent is not straightforward due to scalability in state-space and the formation of a collaborative policy. In addition, the overall behaviour for multi-agent policies in the form of steady-state observations would be beneficial for various use-cases. The policy's adaptation to various configurations, behaviours and against adversaries is also of interest in practical scenarios.

The Extended-GBLA's state dimensions depend on the number of nodes and scales exponentially with the nodes in the graph and the number of communicating agents. Existing MAP approaches have utilised the tabular Q-learning approach, which requires storing and keeping track of all the possible state-action value pairs that do not scale well with increasing nodes and communicating agents. Moreover, the number of states will not be finite if *idleness* is introduced in the state representation unless assumed. A review by Nguyen et al. (2018) also suggests that dimensionality is one of the main concerns for implementing multi-agent Deep Reinforcement Learning systems. Therefore, this work sets out the main objective to address dimensionality concerns for MAP. Moreover, a structured method must account for various parameters and configurations in the MAP problem for agents to learn the strategy.

3 Patrolling problem

The premise of the patrolling task is represented as a graph $G = (V, E)$ (Chevaleyre et al. 2004; Machado et al. 2002), where nodes $v \in V$ are relevant locations to patrol and edges $e \in E$ are the connections or path between two nodes. Two nodes are said to be neighbours or adjacent if there exists an edge connecting the two nodes. The degree of a node of a graph is defined as the number of neighbours to that node. Defining $d = \max_{v \in V}(\text{degree of node } v)$. Let the total number of nodes be $|V|$, the time elapsed be t, the total time of episode be T, and the number of visits to a node v is $N(v)$. For MDPs, the agent's configuration is defined as states and agents make transitions to and from various states through performing actions. Several metrics to evaluate multi-agent patrolling have been formulated (Santana et al. 2004; Lauri and Koukam 2014), based on the *idleness* concept. The *Instantaneous Node Idleness* (INI) of a node at a time instant is given as the time steps elapsed since the last visit to the node by any patrolling robot. Various metrics are defined based on INI in Sect. 5.

The multi-agent patrolling problem is to bind multiple agents to cover the complete graph by visiting each node $v \in V$ at regular intervals. In particular, the problem is to find a graph patrolling strategy (π) for two or more agents to continually minimise the time gap between successive visits to the same node (or idleness) for each node in the graph. This criteria can be defined using an objective or cost that the patrolling strategy needs to minimise. Also, as discussed in Sects. 1 and 2, we also investigate some more challenges in MAP, including dimensionality, adaptability and adversaries. In particular, we attempt to answer the following questions: How can large dimensional states of the MDP be dealt with by forming unbiased patrolling strategies with minimum assumptions? Can function approximation

techniques such as Deep Q-Networks (DQN) address this high dimensional state space through state-space estimation? Moreover, would multi-agent collaboration with independent DQN learners be an effective strategy? Can a suitable MDP maximise the performance of DRL agents such that the DQN method improves the agents' adaptive capabilities regardless of different graph structures, non-uniform frequency requirements, and against adversaries by inducing randomness in its trajectories?

In this regard, we investigate the problem of finding a graph patrolling strategy Π by partitioning the problem into the following sub-problems:

P1 Evaluate the metrics derived from INI to compare various multi-agent patrol strategies and model parameters to form the MDP for an efficient patrolling that minimises these metrics.
P2 Given K agents, formulate a collective strategy $\pi = \{\pi_k\}$ for each $(k = 1, 2, ..., K)$ such that each agent k learns the policy π_k independently but collaboratively optimises the criterion for the same objective defined in problem P1.
P3 Modify the policy obtained from P2 to induce randomness in node visits, giving it protection against external adversaries with minimum effect on the metrics defined in problem P1.

The scope of this paper is to develop a graph patrolling strategy, as defined above, for multi-agent patrolling, where each agent is a mobile robot with the following setting:

- The environment is similar to a city or campus with junctions of connecting lanes and represented as a graph G. Each junction (node in the graph G) can have four connecting lanes ($d = 4$), and the information relevant to the underlying graph is known to patrolling strategy.
- Each mobile robot (agent) has a finite speed and takes non-zero time to travel to the next node after deciding a node. Further, there is a non-zero delay between the decision and the impact of the decision on the criteria.
- The agents are assumed to have the same properties and dynamics. Moreover, the time taken to travel any edge in the graph is the same for each agent. This is not a constraint for designing a patrolling strategy but is primarily assumed for comparison purposes.

4 Preliminaries

This section describes the established concepts and techniques in Reinforcement Learning used to derive the proposed MDP model. We utilise the Temporal Difference (TD)-Learning method as a model-free technique to find the optimal policy. Further, the technique of Deep Q-Networks explained in this section is utilised as a function approximator for state estimation in the proposed model described in Sect. 5. A brief on the concepts of RL and DQN is presented next for completion.

4.1 Overview on reinforcement learning

Reinforcement Learning (RL) is an area of Machine Learning (Sutton and Barto 2018) that deals with the problem of finding an optimal series of actions that software agents should take in an environment to maximise a long term cumulative reward.

The RL agent's interaction with the environment is realised in discrete time steps. Let an agent be at a state $S_t = s$ at time t. The agent takes an action $A_t = a$ at state (s) under a policy (π). The agent receives a reward $r_t = r$ from the environment as it transitions to the next state $S_{t+1} = s'$ at time $t + 1$. If defined, then the agent's interaction with the environment terminates at time $t = T$ where $S_T = s_T$ would be terminal state of the agent. The *return* (R_t) is defined by the sum of consequent rewards discounted by a factor (γ) and given by

$$R_t = \sum_{j=0}^{T} \gamma^j r_{j+t+1} \tag{1}$$

RL states are represented in an MDP, and the state to state transitions follow the Markov property. The change in state due to actions may be governed by state transition probabilities $P_{ss'}(s'|s, a)$ which is the probability of the agent transitioning to state s' when action a is taken from the current state s. Each transition is associated with an expected reward $P_r(r|s, a, s') = \mathbb{E}[r|s, a, s']$. In MDP, each state is associated with a state-action value function $(Q(s, a))$. Due to the MDP's stochastic nature, the state-action value function represents the cumulative expected *return* starting with that state for actions taken as per policy (π).

$$Q_\pi(s, a) = \mathbb{E}[R_t \mid S_t = s, A_t = a, \pi] = \mathbb{E}\Big[\sum_{j=0}^{T} \gamma^j r_{j+t+1} \mid S_t = s, A_t = a, \pi\Big] \tag{2}$$

Policy (given by (2)) is a solution to the MDP, which maps agent's probability or likelihood of taking an action a from state s. An optimal policy maximises the agent's cumulative long term reward. Using state-action values, agents can act optimally by choosing the action from $Q^*(s, \cdot)$ with the highest value at each state.

$$Q^*(s, a) = \max_\pi Q_\pi(s, a) \tag{3}$$

4.2 Epsilon-greedy exploration

Untrained agents have to choose between exploration and exploitation while interacting with the environment and finding the optimal policy. Exploration allows the agent to take random actions and improve its value function estimate. In exploitation (Sutton and Barto 2018) an agent greedily chooses the action that maximises the reward based on the estimate so far. Epsilon-Greedy is a method for the agent to choose whether to explore or exploit at a given time step. The agent decides to explore i.e. execute a random action (a) at state (s) with a probability of (ϵ) and exploit with a probability of $(1-\epsilon)$. A common strategy is to start the learning process with a high value of (ϵ) and gradually decay it towards zero as the agent's estimation of the value function improves.

4.3 Deep Q-networks (DQN)

If the Q-values are finite, they can be stored as a lookup table throughout the Q-learning process. However, it is inefficient to represent them in a tabular format for continuous or infinite state space systems. Deep Neural Networks (Mnih et al. 2013) can hence be used as function approximators to estimate the Q-functions from the observed/experienced states, and rewards.

Deep Q-Networks use fixed-length representations of the state space to feed-forward and estimate the associated state to action mapping (Q-Function). DQNs estimate a parameterised form of the state-value function $Q(s,a;\theta)$ with parameter θ that are the weights of the neural network. It is coupled with experience replay during the learning process. In *experience replay* the agent's experience tuple $e_t = (s,a,r,s')$ at each time step t is stored in a set of tuples. The experience tuple is the agent's interaction with the environment at that time-step. During the parameter update, the network samples mini-batches of experience tuples at random from the stored replay memory.

Vanilla Deep Q-Learning uses the same parameters to select and evaluate an action, leading to learning instability. The Deep Q networks proposed by Mnih et al. (2015) makes use of two identical neural networks but with different weights θ and θ^-. The first network, called the policy network, is used to select an action as per the learned policy from the input state, and the second network, the target network, is used to evaluate the action. In Temporal Difference learning (TD-Learning), one-step look ahead is used to approximate the $Q(s,a)$ by the target network. The first set of weights are learned directly from the experience replay, and the second set of weights can be learned by copying from the first network after every fixed number of steps. To improve learning stability, we soft-update by a factor of τ instead of copying directly from the first set of weights.

DQN system block diagram

As shown in Fig. 1, both the training cycle and testing cycle happen simultaneously as Q-learning is an online technique. The policy network estimates the $Q(s,a;\theta)$ value from state s input and the target network estimates the $Q(s',a';\theta^-)$ value from the s' input, where a' is the action at state s' as per policy π. Q_{target} is the one step look-ahead of the Q-value which is the TD target. The objective of the DQN process is the minimise the error between Q_{policy} and Q_{target}. Both these values and reward from the replay memory are used to calculate the DQN loss, which is the mean squared error (MSE) between Q_{target} and Q_{policy}. In this process both networks iteratively try to converge towards the optimal Q-value (Q^*) with a learning rate α.

$$Q_{target} = r + \gamma * \max_{a'}(Q(s',a';\theta^-)) \tag{4}$$

$$Q_{policy} = Q(s,a;\theta) \tag{5}$$

$$Loss = MSE\_Loss(Q_{policy}, Q_{target}) \tag{6}$$

4.4 Metrics

The goal is to evaluate and compare strategies, analyse the framework and model parameters based on a metric derived from the node idleness values as defined in problem P1 in Sect. 3. The analysis is done in episodes, i.e. length of the simulation and observation. The system resets to the initial state after an episode ends. The reward function and state-space representation of the MDP model is based on these metrics. Extending from the INI defined in Sect. 3, other evaluation metrics defined are as follows:

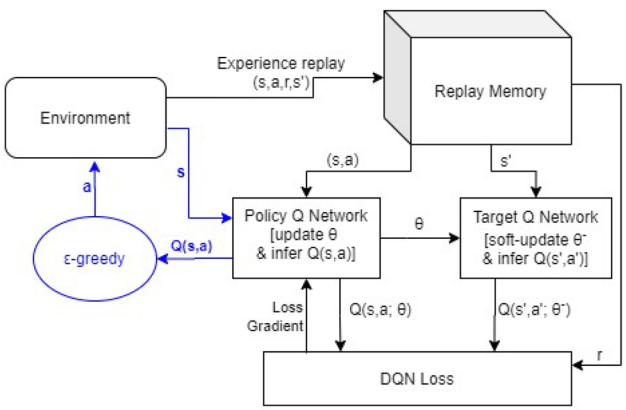

Fig. 1 The DQN process. Blue colored block represents testing or inference cycle while black colored blocks represent training cycle; figure replicated from Li et al. (2020)

– **Node Visit Idleness (NVI)** for a particular node (v) is the series of INI logged at each visit of that node over the experiment's length.

$$NVI(v,i) = INI(v) \text{ at } i^{th} \text{ visit; for each } i = 1,2,3...N(v) \tag{7}$$

– **Average Node Visit Idleness(ANVI)** of a node is given by the NVI averaged over number of visits for that particular node.

$$ANVI(v) = \sum_{i=0}^{N(v)} \frac{NVI(v,i)}{N(v)} \tag{8}$$

It indicates the frequency of visits at each node throughout the experiment. This metric provides insight into the patrolling strategy's performance for priority nodes that require more frequent visits than other nodes.

– **Global Average Node Visit Idleness(GANVI)** is the ANVI averaged over all the nodes in the graph.

$$GANVI = \sum_{i=0}^{|V|} \frac{ANVI(i)}{|V|} \tag{9}$$

This provides an overall time-averaged performance estimate of the patrolling strategy over the entire experiment.

– **Instantaneous Global Idleness (IGI)** is the Instantaneous node idleness averaged over all the nodes in the graph at a particular time t. This provides an overall performance estimate of the patrolling strategy in real-time. This metric and NVI will be used to evaluate the testing performance of adaptive behaviours.

$$IGI(t) = \sum_{i=0}^{|V|} \frac{INI(t,i)}{|V|} \tag{10}$$

– **Average Global Idleness (AGI)** is the The IGI averaged over the whole simulation time. This metric will be used as the episode score to evaluate the training performance of various parameters and definitions.

$$AGI = \sum_{t=0}^{T} \frac{IGI(t)}{T} \tag{11}$$

5 Proposed model

The multi-agent patrolling problem is aimed to minimise node visit idleness for all the nodes in the graph. Thus, it is advantageous to utilise a function of idleness as reward feedback from the environment to influence the agents' learning of optimal patrolling strategies. The patrolling environment is described as a Markov Decision Process (MDP) to facilitate the learning of an optimal policy, where the agent decides the next node to visit from the graph G. This section describes the model that proposes the solutions to the objectives P1–P3 defined in Sect. 3. Referring to Fig. 2, the MDP (for solving P1) and DQN (for solving P2) tuned upon evaluation of the chosen metric performance. Multiple configurations and parameters are accounted for in the initial design, and a suitable configuration is selected iteratively based on the metric. Triggers for random trajectory switching or other adaptive behaviours are induced (solves P3) on the learned policy to get the final patrolling strategy. The DQN's ability to act as a universal function approximator to estimate the state-space enables us to induce an external trigger to which DRL agents are expected to react accordingly.

5.1 The MDP framework

If the multi-agent system is considered a single centralised agent, it leads to an exponentially large number of states and actions. Thus, considering all possible states' dimensions, it is advantageous to define the MDP with multiple decentralised agents that learn independently (Portugal and Rocha 2014). For developing the MDP framework, the state s- action a pair is described to come up with a policy π as described in Sect. 4.1.

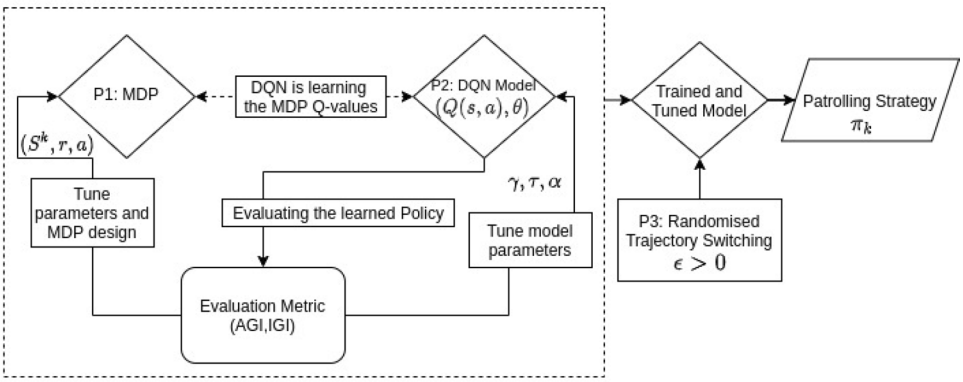

Fig. 2 The proposed model for each agent as independent learner; the chosen idleness metric from evaluating the policy after an episode is used to tune the MDP and DQN Model: S^k and α refer to state representation and learning rate respectively

5.1.1 Agent and action space

Each agent is a physical agent as described in Sect. 3. As the agent is performing high-level planning, the task is to decide on the next node visit from the current node. The action decision to move to a node will be taken based on the agent's long term expected *return* as described in (2). Thus the decision to move to an adjacent node is based on the policy (π), which is expected to maximise this *return* (Marier et al. 2010). The action space dimension is equal to d (= 4 for grid), the maximum degree of the graph. The actions are mapped as moving to each neighbouring node. Each adjacent node can be uniquely associated with an action at a particular node. a_1 to a_4 are the actions that take the robot from a node to the adjacent node to the right, straight, left and u-turn respectively and the directions are with respect to the agent's heading direction (Fig. 3).

5.1.2 State representation

As the idleness values governing the patrolling task are dynamic and can be influenced by other agents, idleness is required to provide complete information of the agent's state in a Markov decision process. The agents take decisions based on the four Q-values (as d=4) corresponding to each action at a state. Markov property states that an agent's action decision at any state must not depend on its previous states. Work by Walsh et al. (2008) suggests using variants of DQN to tackle the non-stationary problem. However, we attempt to form the MDP so that the state-space representation has enough information to make optimal state-to-state transitions through DQN in a non-stationary environment. Inclusion of idleness in the state-space leads to infinitely many states as idleness values are not bounded. Even with assumptions and bounds on idleness, the state-representations for tabular Q-learning formulated in Santana et al. (2004), Lauri and Koukam (2014) end up having prohibitively large number of possible states and parameters that grow with the number of nodes in the graph or agent population. Keeping track of all possible states in a Q-table thus becomes inefficient.

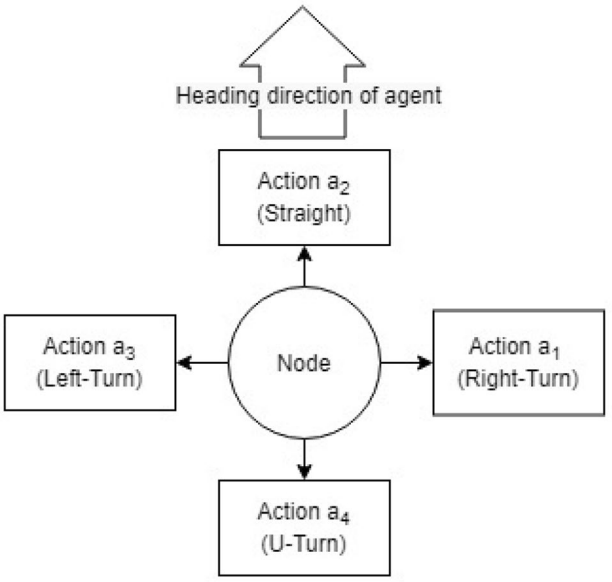

Fig. 3 The action space of the agent for a grid where $d = 4$

The proposal is to implement the local information at a node to represent a state to form the state-space in the MDP and estimate the next state using a function approximator such as in DQN. Through this, we can alleviate the curse of dimensionality and estimate the state-space using finite parameters and minimum assumptions.

Based on the above definition, several fixed-length state vectors that rely on local information are proposed. These state representations contain two sets of information. First, the information related to the decision-making agent, i.e. its node location and neighbour nodes' idleness values. Secondly, other agents' information in the graph, which is equivalent to agents communicating through state representation. This information will be useful in achieving collaboration through communication. Various state representations [S1 to S5] are defined as follows. Let k be the decision making agent which is traversing from source node (v_s^k) to target node (v_t^k) and l be another agent in the environment ($l \neq k$).

S1 **Nodes as State-space , No State Communication**: Agents do not share idleness values centrally, each agent keeps an independent track of the nodes' idleness values. The k^{th} agent only utilise the source node (v_s^k) and the next approaching node (v_t^k) while traversing on the edge between (v_s^k) and (v_t^k) as state-representation. Hence, the state representation is given by:

$$S1^k = [v_s^k, v_t^k] \tag{12}$$

S2 **Local Idleness, No State Communication**: Agents do not share idleness values centrally, each agent keeps an independent track of the nodes' idleness values. Agent utilises its source-target nodes and the idleness of neighbours of (v_t^k) as state-space. There is no information of other agents in the state space. Let the neighbour nodes of v_t^k be defined as v_1^k to v_d^k. From our action space, $v_d^k = v_s^k$. Hence, the state representation is:

$$S2^k = [v_s^k, v_t^k, INI(v_1^k), INI(v_2^k), ..., INI(v_{d-1}^k), INI(v_s^k)] \tag{13}$$

S3 **Shared Idleness, No State Communication**: S3 differentiates from S2 in the way agents are keeping track of the idleness values. For S3, the agents are not keeping

track of node idleness independently but the values are updated centrally. Node idleness values are shared centrally by all agents. Agent utilises its source-target nodes and the idleness of neighbours of (v_t^k) as state-space., and there is no information of other agents in the state space.

$$S3^k = [v_s^k, v_t^k, INI(v_1^k), INI(v_2^k), ..., INI(v_{d-1}^k), INI(v_s^k)] \tag{14}$$

S4 **Shared Idleness, Node Communication**: Node idleness values can be accessed centrally by all agents. Agents utilise its own to-from nodes and neighboring idleness of (v_t^k) as their state-representation. Also, other agents' source and target node positions are utilised in S4. Below is an example of two agent case where second agent's information is available in the state-vector.

$$S4^k = [v_s^k, v_t^k, INI(v_1^k), INI(v_2^k), ..., INI(v_{d-1}^k), INI(v_s^k), v_s^l, v_t^l] \tag{15}$$

With more agents in the environment, each agent's information is appended to this representation. Length of this representation is $4 + 2K$ (K =no. of agents).

S5 **Shared Idleness, Node and Idleness Communication (Full Communication)**: Global idleness values can be accessed centrally by all agents. Agents utilise its nodes and neighbouring idleness as their state-representation. Other agents communicate both their source-target nodes and neighbouring idleness values. Below is an example of a two-agent case where the second agent's information is available in the state vector.

$$\begin{aligned} S5^k = &[v_s^k, v_t^k, INI(v_1^k), INI(v_2^k), ..., INI(v_{d-1}^k), INI(v_s^k), \\ &v_s^l, v_t^l, INI(v_1^l), INI(v_2^l), ..., INI(v_{d-1}^l), INI(v_s^l)] \end{aligned} \tag{16}$$

Length of this representation is $(d + 2)K$ (K=*no.ofagents*).

In Fig. 2, the a state representation is referred by S^k for k^{th} agent which takes up either $S1^k$, $S2^k$, $S3^k$, $S4^k$, or $S5^k$ representation. These representations are compared and analysed in Sect. 6. The AGI score for patrolling policy learned with each representation is compared and the state-representation S^k for the action space is decided as defined in objective problem P1. Thus, providing a solution to problem P1 as defined in Sect. 3.

When agents are independent learners, the challenge of creating cooperative agents is addressed by proposing a reward function for the MDP explained next.

5.2 Proposed reward function

The proposed reward function is utilised as feedback from the environment (block P1 in Fig. 2) for updating the Q-values as given in (4). The idea is to design a globally shared reward r that serves two purposes. Firstly, the reward function must ensure that the agents are getting feedback on the cooperative functionality of their actions, and secondly, the reward function should give immediate feedback on the impact of the agent's respective actions on the environment. It is worth noting that the choice of reward function is crucial as it will have an impact on the global performance on the learning policy. If the agents optimise a selfish reward function, it might harm the rewards of other agents and, hence, the globally optimal policy. A reward function that only gives the global performance to the agents might suffer from unstable or slow learning of agents as the impact of their actions is not significant on such a reward.

Penalised Idleness: Proposed reward function penalises agents to have higher global average idleness even if the visit idleness was high, thus giving the agents direct feedback on collaboration. Therefore, proposed reward (r) is given to the agent if it visits a node (v) at time (t)

$$r = \frac{INI(t, v)^p}{IGI(t)} \tag{17}$$

It is worth noting that the parameter p decides the factor by which INI influences the reward. Greater the value of p, more will be the effect of INI on the reward than IGI. We further analyse the effect and optmial value of parameter p in Sect. 6.

The reward for greedily approaching nodes will be lower if the global idleness value is high. Therefore, to maximise its reward, the agent must approach a node with the higher INI but also collaborate to minimise IGI, resulting in a robust policy. If more than one agents visit the same node, the agents that visit the node second and beyond will end up with a significantly lower value of INI in their reward, thus the learned collaborative policy is expected to prevent node conflicts. In case the agents chose to visit the same node at the same time during the training phase, the lower level plan of the simulation environment handles the conflict by stopping the agents momentarily and letting them pass through the node one-by-one. The action space, state representation $S1^k$, $S2^k$, $S3^k$, $S4^k$, or $S5^k$, and reward function r form our MDP model elements for multi-agent patrolling. Based on this solution to problem P1, the agents need to learn a policy for this MDP that optimises our metrics as defined in problem P2. The proposed method to find the policy that solves problem P2 is defined next.

5.3 DRL system

As the state-space is defined as a local parameterised representation, function approximation is utilised to estimate the state-action value pair (Q(s, a)). As the actions are discrete and finite based on the problem definition in Sect. 3, Deep

Q-Network is a suitable method for TD-Learning with state estimation. The previous subsections present the proposed method for learning the optimal policy using DQN. The convergence of the DQN losses and the early stopping criteria defined through the minimum AGI metric provides the solution to problem P2. We further analyse this solution to problem P2 in Sect. 6.

5.3.1 DQN framework

The state vector, reward, action, and next state are sampled from the experience replay during the training process. The state vector defined in Sect. 5.2 is fed as input to the DQN that estimates the Q(s, a) and Q(s', a') values (refer Fig. 1). In addition, from the defined reward function the TD-loss is calculated and the Q-values are updated based on (4), (5) and (6). This iterative process continues for multiple episodes, and if the DQN Loss converges towards zero, the parametric estimation of $Q(s, a;\theta)$ by the DQN eventually converges towards the Q*(s, a) as defined in Sect. 4.

As we are dealing with physical agents, these have finite linear velocities. Hence there will be a delay between the agent's action decision and its impact on the environment. Hence, there is a delay in reward feedback to the agent because it takes time to reach a particular node after deciding to go there. This delay is addressed in the experience replay as the experience tuple's current reward is synchronised with the previous action decision while storing the experience in the memory buffer.

5.3.2 DQN architecture

As input to the DQN is an array of integers, a Multi-layer Perceptron architecture with custom model parameters can estimate the Q-values. The number of hidden layers, units in each layer, variants of Rectified Linear Unit (ReLU) for activation function, batch size, optimiser and learning rates are varied and empirically determined. Both policy and target networks share the same architecture. The neural network architecture is as shown in Fig. 4:

The input layer dimension is the dimension of the state-space representation i.e. $2, d+2, d+2, d+2K, (d+2)K$ for $S1^k, S2^k, S3^k, S4^k$ and $S5^k$ respectively and K=number of agents. This parameter is resolved in Sect. 6. The output layer dimension is the number of actions that the agent can take.

5.3.3 Learning and tuning

The training procedure is conducted as follows.

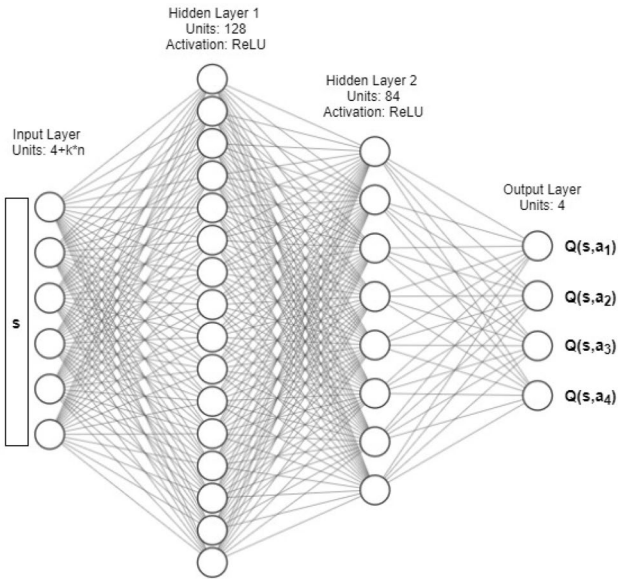

Fig. 4 Deep Q-Network architecture for DRL Agents. It takes fixed-length state representation as input and outputs the estimated Q-values. Hidden layers are fully connected with (ReLU) activation

- The stored experience replay is sampled randomly in mini-batches and fed to the DQN system defined in Sect. 4 (Fig. 1).
- The training is done episodically. The agents are initialised randomly in the graph at the start of each episode, and the episode runs for fixed time steps, iteratively estimating Q(s, a) and minimising the DQN Loss by learning the parameters θ. Here, we define n as the number of episodes elapsed since the training procedure began.
- The random exploration probability, epsilon (ϵ), as defined in Sect. 4 for epsilon-greedy exploration is decayed from 1.0 to 0.005 during training. This decay will ensure that agents initially improve their Q(s, a) estimation and then gradually converge towards the optimal policy. The random exploration probability $\epsilon(n)$ at episode n decays with the number of episodes passed while training, it is defined as: $\epsilon(n) = \epsilon_{start} * 0.992^{(n)}$
- The training keeps track of the minimum of AGI of the elapsed episodes. If AGI is less than the minimum at the end of an episode, it is updated, and the model parameters are saved. If the minimum has not been updated for 50 episodes, the training is stopped, and the parameters of best AGI performance is saved.

The MDP, RL and DQN model parameters are tuned empirically based on the AGI performance.

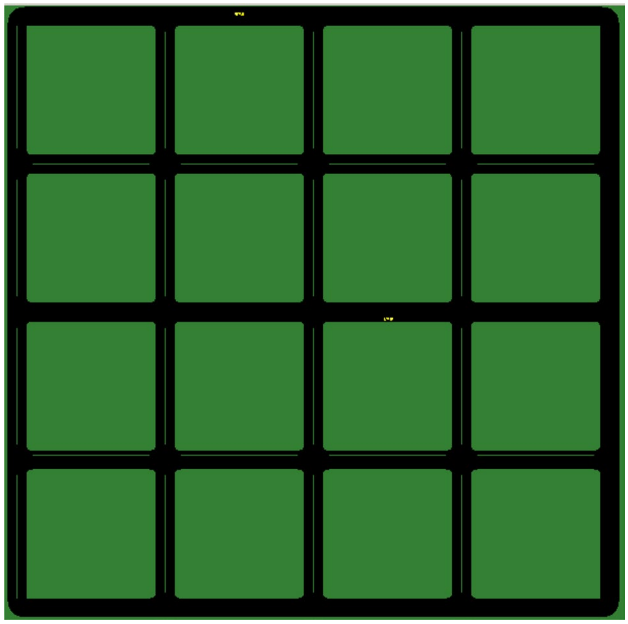

Fig. 5 5×5 symmetric grid

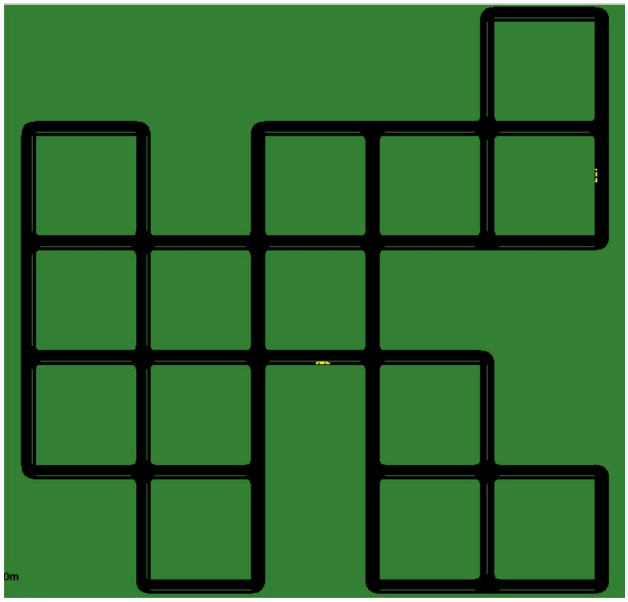

Fig. 6 30-node asymmetric grid

5.4 Randomised trajectory switching

The results of this optimal policy form the basis of our problem objective P3. If there are adversaries present in the environment, the repetitive patterns of deterministic strategies can be exploited as agents are traversing a particular node in set intervals (Agmon et al. 2011). Inducing randomness in the patrolling strategy reinforces our patrolling strategy against adversaries, especially if they observe the strategy for a long duration and know the formed policies.

While the non-zero value of epsilon has been used in the training phase, the idea here is to utilise the epsilon greedy strategy for randomised exploration during the patrolling procedure. A non-zero but small value of ϵ will trigger a random neighbouring node traversal at irregular intervals during patrolling. This will help break cycles and create randomised patrolling patterns, making the patrolling strategy (π_k) difficult to predict even on observing the strategy in action for a considerable amount of time, thus proposing a solution to the problem P3 defined in Sect. 3. This random action should have a minimum effect on the AGI performance. Different values of ϵ are analysed against the AGI metric and a suitable value is selected as per problem objective P3.

5.5 Adaptive capabilities

The proposed patrolling strategy should generalise well to various conditions defined in objective problem P2 in Sect. 3. The hypothesis is that the trained DQN models would adapt to at least the following two conditions.

5.5.1 Adaptive to any graph structure

The MDP model, which encompasses both local information of the deciding agent and communication from other agents, makes it feasible to adapt to various graph structures. The DRL agents trained on a specific graph structure adapt and perform the patrolling task on a graph that is different in symmetry and number of nodes.

5.5.2 Adaptive to strategies with priority nodes

Priority nodes are particular nodes in the graphs that require a shorter time between visits (smaller node visit idleness). The DRL agents follow the training procedure as defined in Sect. 5.4.3. These trained agents can be informed about the priority nodes from the Apparent Node Idleness (ANI) values during patrolling. DRL agents perceive the priority node to have a higher INI value than the actual INI, and all other nodes are perceived to have the actual INI values. We

Table 1 Parameters selected in the final proposed model from evaluation (refer Sects. 4,5)

Optimizer	Loss	α	γ	Memory buffer	Batch size	τ
Adam	MSE	7.5e-4	0.95	1e+5	32	0.001

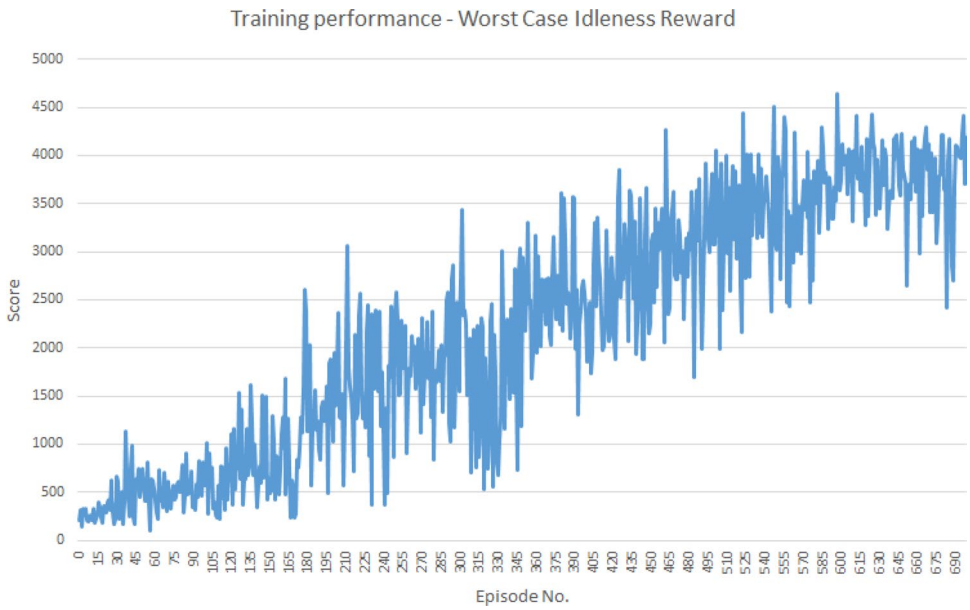

Fig. 7 Training performance of worst case idleness reward in multi-agent case. The score (AGI) is diverging due to less learnability

get the ANI by scaling the INI by a factor, we have chosen the factor to be 3.

$$ANI(v) = \left\{ \begin{array}{ll} 3 * INI(v), & \text{for } v = \text{ priority node} \\ INI(v), & \text{for } v \neq \text{ priority node} \end{array} \right\} \quad (18)$$

The AGI metric and NVI metric based on actual idleness for a particular node is analysed and compared for two cases in which agents are informed whether it is a priority node or not. The hypothesis is that when DRL agents are informed of priority nodes during patrolling by simply providing the ANI values, they adapt and reduce the NVI for that node without training to do so.

6 Empirical results and analysis

An exhaustive set of results are obtained by simulating a patrolling scenario based on Simulation of Urban MObility (SUMO) Krajzewicz et al. (2002) and TraCI python library.

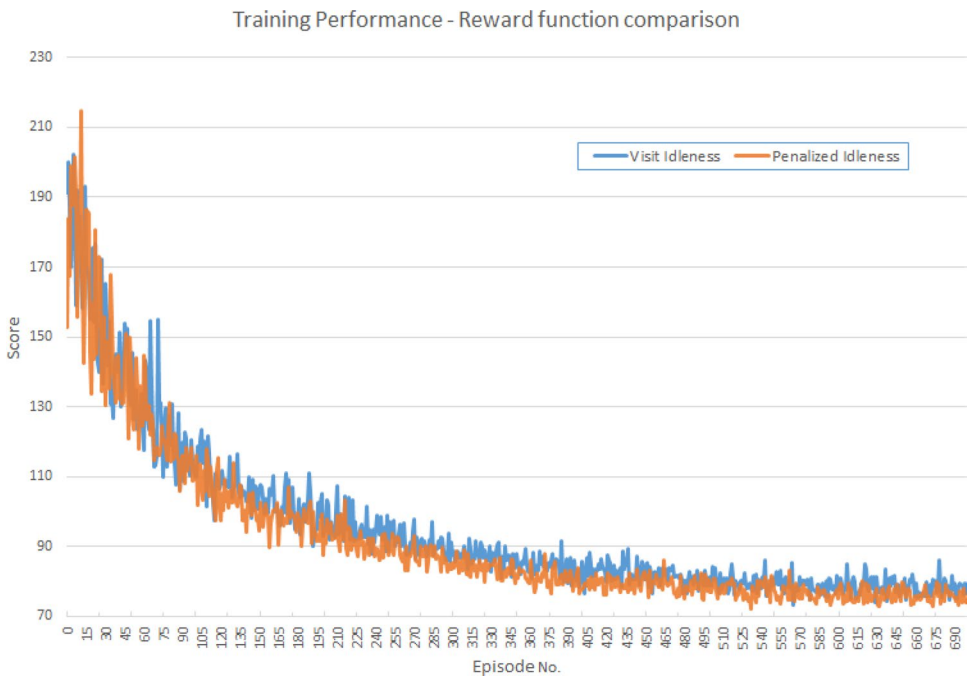

Fig. 8 Training performance of other two reward functions in multi-agent case. The score (AGI) is decreasing and converging

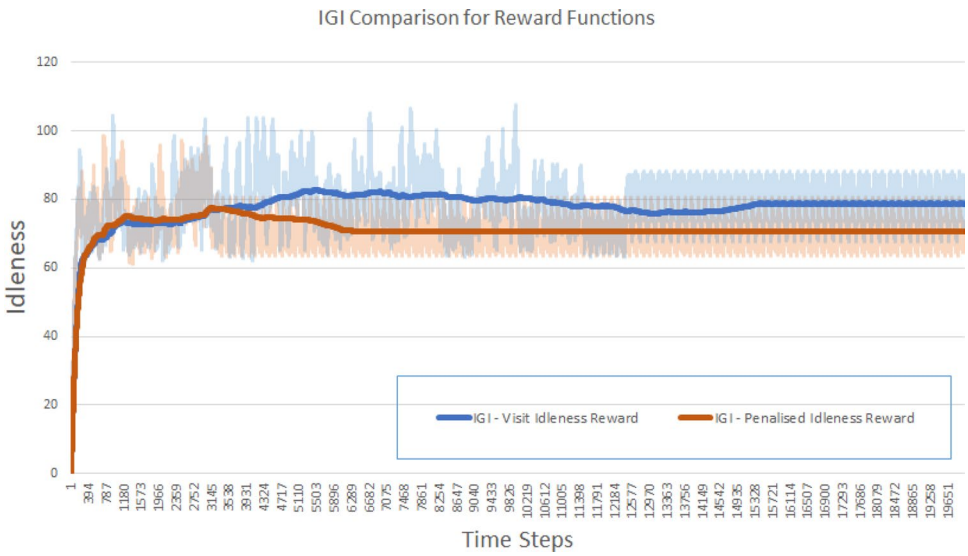

Fig. 9 Testing performance of reward functions R1 and R2 in multi-agent case. Penalised visit idleness performs better than node visit idleness reward

The agents were set to take 10 secs to traverse an edge. The Episode duration is varied between 3000 to 10000 time-steps during training. The results are obtained to evaluate the proposed model for learning patrolling strategy. The obtained results are aimed to demonstrate the following:

- Through the suitable MDP model, the DRL agents are able to learn independent policies and form a collaborative patrolling strategy.
- The strategy formed by DRL agents are cyclic in nature. Further, agents are able to reform randomised trajectory switching to break cyclic patterns making it robust against adversaries.
- Finally, DRL agents adapt to various behaviours such as different graph structure and non-uniform frequency requirements without explicitly training to do so.

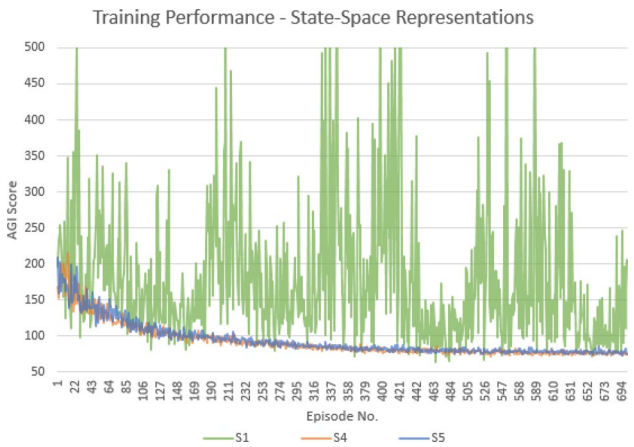

Fig. 10 Training performance of various state-space representations in multi-agent case

The metrics used to evaluate are the IGI and AGI. The NVI is utilised to analyse priority nodes and visit randomness, and all metrics are defined on the SUMO simulation's discrete time steps. The various state representations, reward functions, and learning parameters are compared and applied on two different graphs; *Graph 1:* a symmetric 5×5 grid graph and *Graph 2:* an asymmetric 30-node grid graph that would resemble a city/campus-like environment. Agents move with constant velocity in the simulator and take about ten time-steps to traverse an edge. To evaluate the strategies' collaborative nature, they are tested on single, two, and three agent scenarios. The agents are initialised at random nodes at the beginning of each episode, both during training and testing (Figs. 5 and 6).

6.1 Model parameters

Various model architectures, the learning rate, batch-size, discount factor, and soft update factor are tuned throughout the experimental analysis to optimise general cases' performance. The parameters are tuned individually to achieve optimal idleness metric performance while isolating the one being tuned from the rest of the parameters. The AGI values for the corresponding RL and DQN Model parameters are tabulated in Tables 3 and 4. Nearly all possible combinations of near-optimal parameter values are empirically analysed.

The architecture and tuned parameters provided optimal empirical performance to various reward functions, state representations, and the number of agents— tabulated in Table 1. The final DQN model architecture can generalise and learn for the various experimental cases mentioned previously.

The reward function and state-representation are chosen from an empirical analysis of all the pair's possible

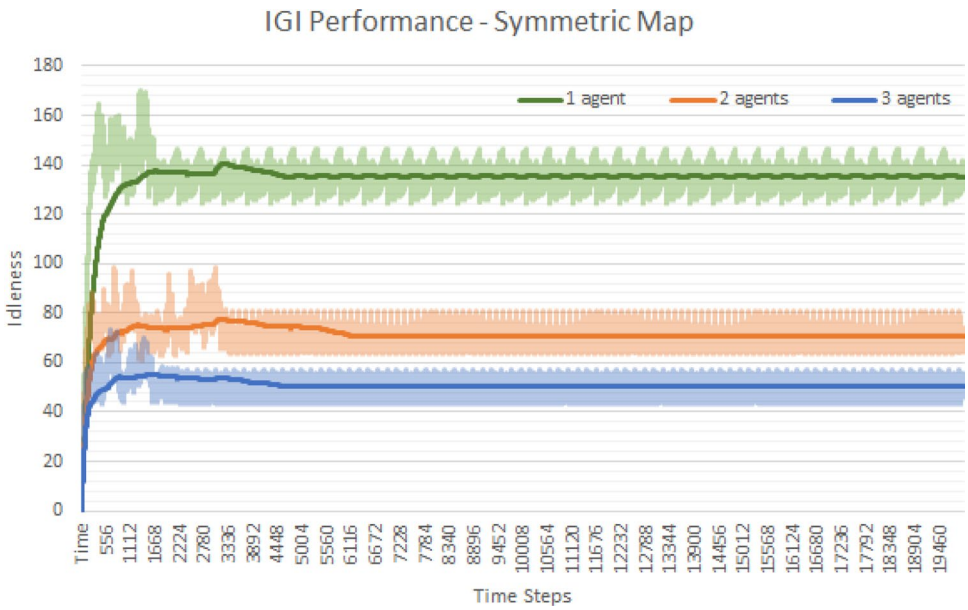

Fig. 11 Comparison using IGI performance of RL agent(s) in symmetric grid graph

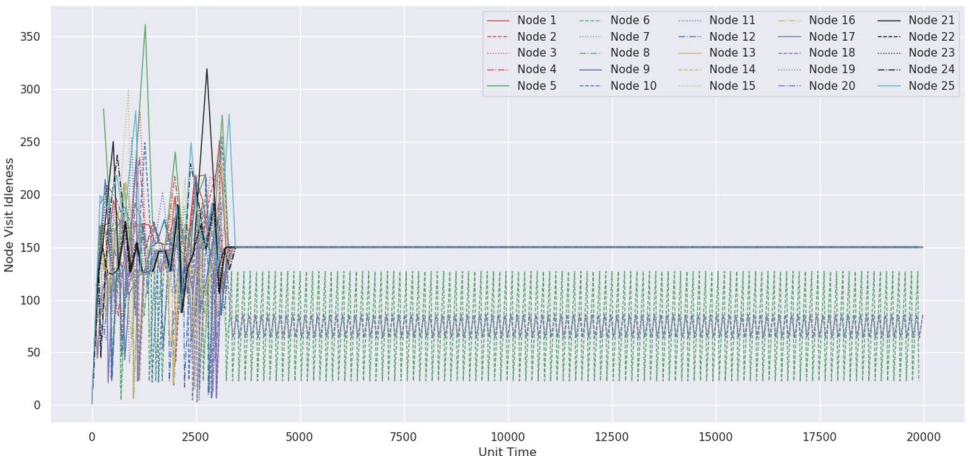

Fig. 12 Node Visit Idleness of all the nodes are either periodic or constant when agents reach convergent trajectories

combinations. The data of the empirical analysis is presented in Table 2. Based on that, the comparison data shown in Fig.9 for reward function is measured by varying the reward function definitions only while keeping other MDP definitions and parameters at optimal values. Similarly, the comparison is made for state-space representation in Sect. 6.3.

6.2 Selection of reward function

To select an appropriate reward function, the performance is compared on multi-agent scenarios using the proposed reward function and the reward functions used in the existing literature. The multi-agent scenario is taken to evaluate the independent learning and collaborative policies. The three reward functions are evaluated based on agent training feasibility and idleness performance, and they are as follows. The score is the AGI metric for each episode, as defined

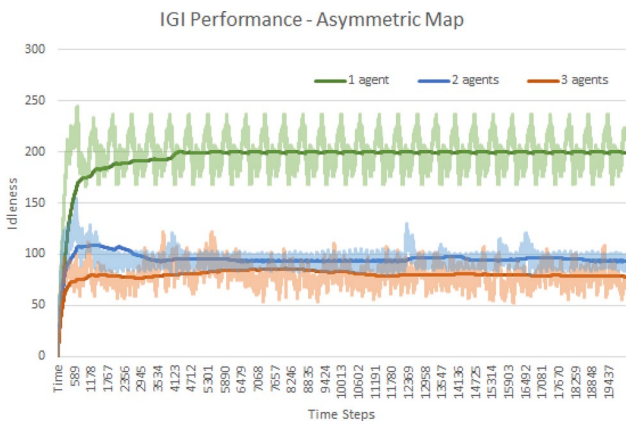

Fig. 13 Comparison using IGI performance of RL agent(s) in asymmetric grid graph

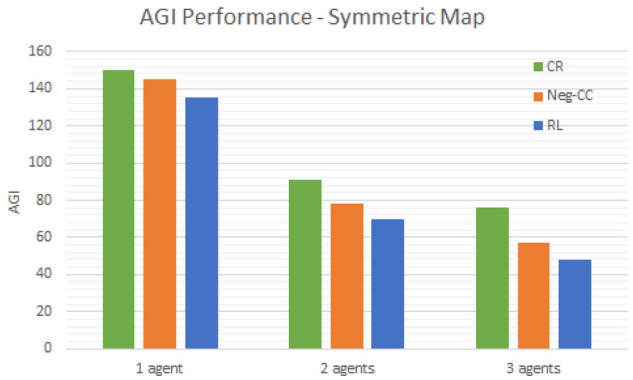

Fig. 14 Comparison using AGI performance with other approaches in symmetric grid graph

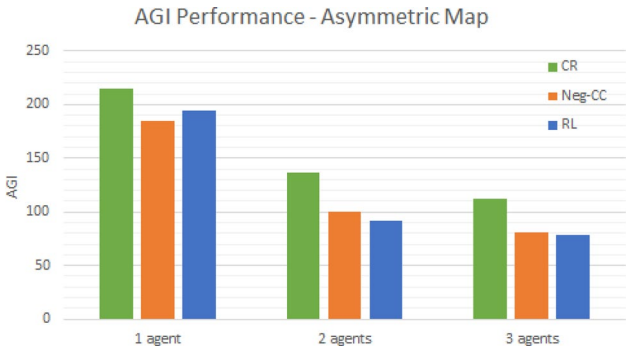

Fig. 15 Comparison using AGI performance with other approaches in asymmetric grid graph

in Sect. 5.1. The convergence of the score to a minimum value is required, as mentioned in problem objective P2. We investigate and compare the use of various reward functions (R1–R3). These reward functions are as follows:

R1 Proposed Penalised Visit Idleness: as defined in equation (17). The optimal value of p (=1.5) is determined empirically (refer Table 2). Thus, the final reward function is now given by (19).

$$r = \frac{INI(t, v)^{1.5}}{IGI(t)} \tag{19}$$

R2 Visit idleness (Santana et al. 2004): Agents are fed the reward as the NVI of the node it has just traversed as a consequence of their action. It does not have direct feedback on the collaboration between agents.

R3 Worst-Case Idleness (Lauri and Koukam 2014): The Maximum Idleness recorded during the whole episode is used as a penalty reward to all the agents.

The results of the reward functions are shown in Figs. 7, 8 and 9. As shown in Fig. 7, the reward function R3, worst case idleness, did not converge to a stable policy, whereas the rewards R1 and R2 undergo stable learning (refer Fig. 8). The unstable learning of reward R3 could be due to the lack of significant direct impact of the agent's action on the reward as the worst-case idleness remains constant for most of the time during an episode.

The IGI performance of reward R1 and R2 are further compared during testing for multi-agent cases, and the results are shown in Fig. 9. Reward R1 provided better IGI performance than reward R2 for both the graphs. In penalised idleness reward, the agent would not get a high reward for higher node visit idleness if the IGI metric is not reduced significantly, thus directly impacting the agents to

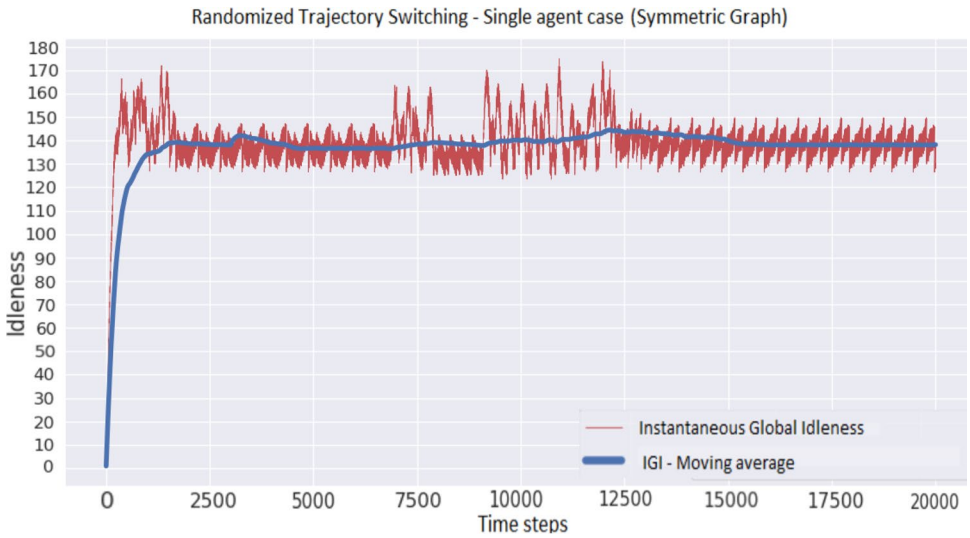

Fig. 16 IGI of single agent case in symmetric graph undertaking randomised action selection with probability of 0.001. The agent switches trajectory intermittently and traverses the nodes in a different order, notice the shape of the periodic sections

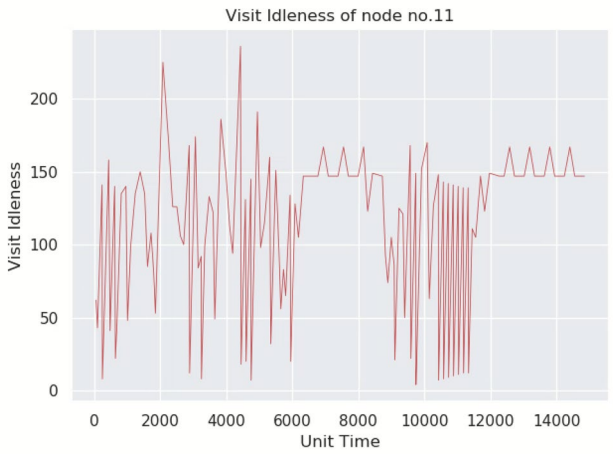

Fig. 17 NVI of a node while two agents perform randomised trajectory switching in the symmetric grid graph. The agents break out of formed periodicity and bring randomness in the visits

act collaboratively, instead of selfish actions that have an indirect impact on collaboration.

6.3 State representation

Various state-representations (S1–S5) of the MDP as defined in Sect. 5 that the DQN utilises are evaluated. The agents are not able to converge to stable policies with state representation S1. The reason could be a lack of correlation between the node values and the dynamically changing idleness reward arising due to the multi-agent environment's non-stationary nature. The following considerations included the neighboring idleness values in state-representations mentioned in Sect. 5. These are evaluated based on the performance, as shown in Fig. 10.

The state-representation S4 gave the best AGI performance for multi-agent cases in both graphs. Agents were able to resolve conflicts and collaborate to decrease idleness metrics. Lack of communication in representations S2 and S3 were leading to conflict of interest. Only the agent that traverses the node first will get a high reward, and other agents will not correlate with the lower reward with its state-space representation in the absence of inter-agent communication. Interestingly, the representation S5 gave a slightly worse performance than S4. The reason could be that the learning agent incorrectly correlates its rewards with the neighbouring idleness values that other agents are communicating. This analysis of the MDP elements and model parameters solves P1 as defined in Sect. 3 (Fig. 11).

6.4 Dimension

The state vector (the input layer to the DQN) with all the agents communicating has $4 + 2K$ (where K is the number of agents) units, and the output layer has d units. Based on the DQN model architecture as mentioned in Sect. 5, the number of parameters or elements required for the DRL model (D) is calculated as follows:

$$D = (4 + 2K) * 128 + 128 * 84 + 84d + 128 + 84 + 1558 \tag{20}$$

$$D = 11477 + 256K + 84d \tag{21}$$

For action space dimension (d) = 4 the number of parameters (N) is given by

$$D = 11813 + 256K \tag{22}$$

The number of DRL parameters is linear in the number of agents and degrees and independent of the number of nodes in the graph. No bounds are set to the number of nodes or

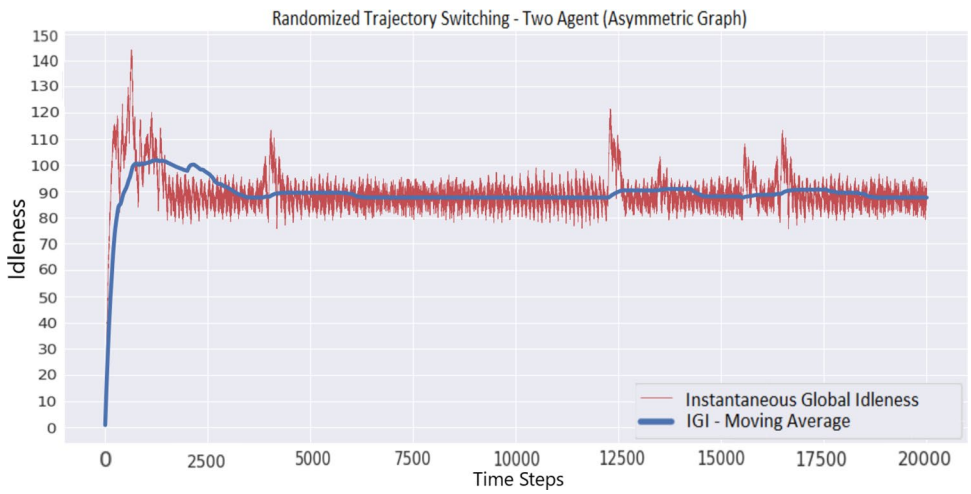

Fig. 18 IGI of multi agent case in asymmetric graph performing randomised trajectory switching

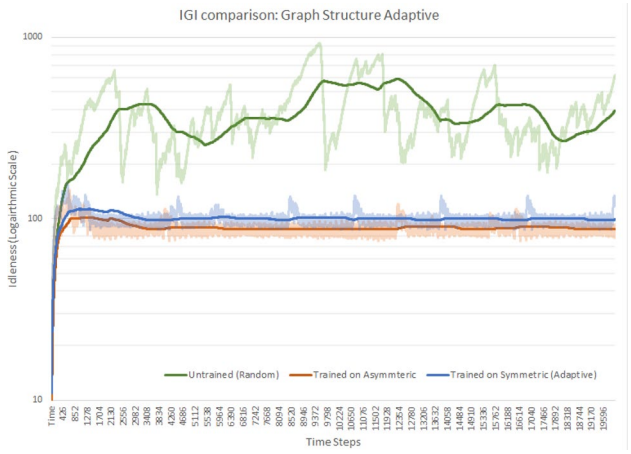

Fig. 19 IGI performance of adaptive agents in multi agent case for asymmetric graph. Random (untrained) agents showed IGI in the range of (400–800) whereas adaptive agents showed convergent IGI of around 100, which is similar to the trained agents' performance of around 90

idleness, making the model potentially scalable to large graphs.

6.5 Comparative performance

The RL approach is compared with two state-of-the-art non-RL patrolling algorithms. a) The Conscientious Reactive (CR) (Machado et al. 2002) and b) Cognitive Coordinating with Negotiator Mechanism (Negotiator-CC). The DRL policies were found to converge in under 700 episodes. The agents learned that cyclic trajectories gave the best AGI performance for the two graphs during training, as evident in Fig. 12 and as shown in the video (Online Resource 1 and 2) (Figs. 13, 14, 15).

Based on the comparative study by Almeida et al. (2004) on grid-type graphs, the following observations are made: The Negotiator-CC outperformed GBLA in grid-type graphs but had adaptive limitations. The Deep-RL approach performed better than the Negotiator-mechanism CC approach in multi-agent cases. The RL-based GBLA though more adaptive, is limited by the total possible states in the MDP model. With an increasing number of agents and nodes, the dimension of the Q-table becomes too large. Using DQN for state estimation, the policies can scale to larger graphs, numerous agents, and have adaptive capabilities. Hence, the proposed RL approach demonstrated improvements and a broader scope of applicability as compared to the existing ones (Figs. 16, 17).

6.6 Randomised trajectory switching

Upon applying the proposed randomised exploration technique, the agent would change its path and converged to a different cyclic trajectory as seen in Fig. 18 and as shown in the video (Online Resource 4). The agents performed this procedure without harming the AGI value, thus solving P3 as defined in Sect. 3.

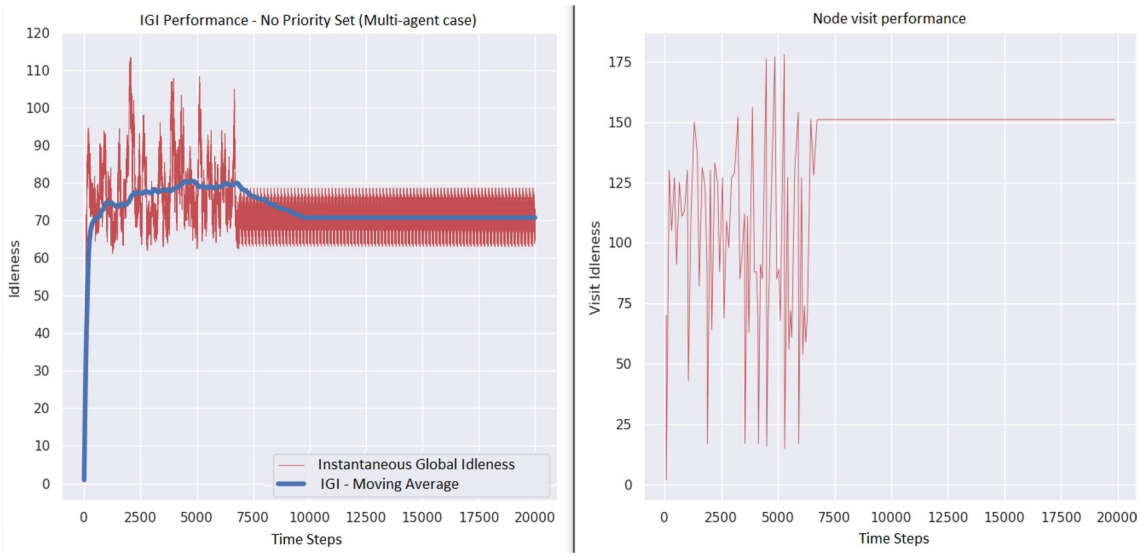

Fig. 20 IGI and Node visit idleness of Node no. 11 in multi agent case without priority being set

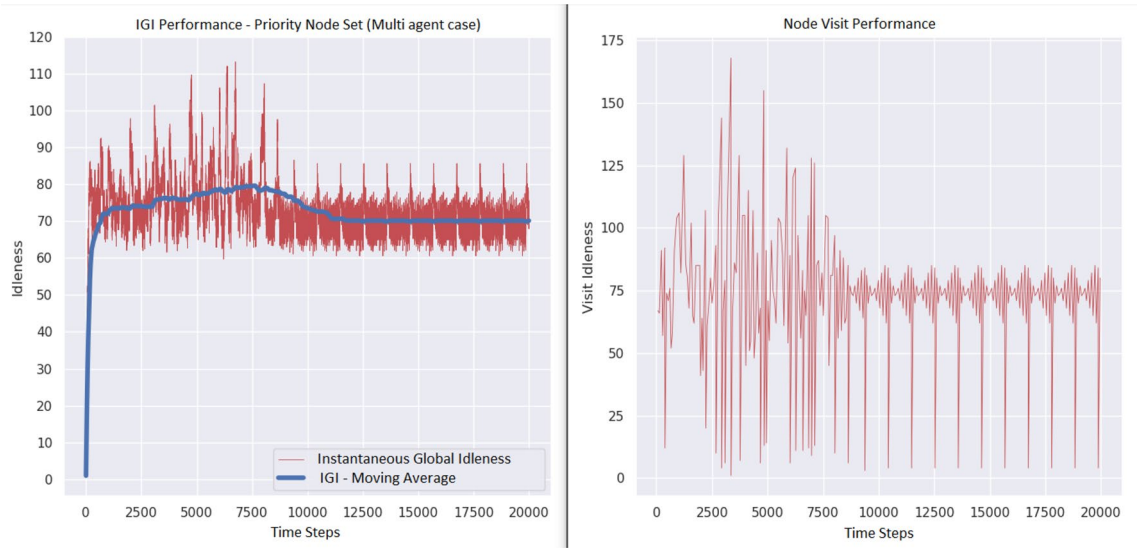

Fig. 21 IGI and Node visit idleness of Node no. 11 in multi agent case with priority set, the node visit idleness reduces from 150 to around 75 when convergent policy is achieved

This randomised trajectory switching based patrolling makes it difficult for any adversarial agents to predict the patrolling agent's trajectory or recognise any patterns if it forms one.

6.7 Adaptive

The system's adaptive nature is further evaluated. The DQN state estimation makes the agents adapt to various scenarios without explicitly training the agents in them. The first observation is that the agents can adapt to the graph structure. Agents trained in the 5×5 symmetric grid performed almost as good as agents explicitly trained on the 30-nodes asymmetric graph, as shown in Fig. 19.

Secondly, the agents can adapt to graphs with priority nodes without training to do so. Through the method mentioned in sect. 5, the agents took shorter time between visits for those nodes evident from the NVI performance shown in Figs. 20, 21 and the video (Online Resource 3).

7 Conclusions

This paper presents a novel Deep Reinforcement Learning approach for multi-agent mobile robot patrolling. The DRL algorithm based on Deep Q-Network is implemented to make the agents select the best actions in a non-stationary patrolling environment. The state representation suitable for DQN estimation and a novel reward function used by the agents to independently learn and collaborate in a graph is proposed. Empirical analysis shows that the proposed reward function is suitable for DRL and outperforms the earlier developed reward functions. RL approaches' main drawback is that the curse of dimensionality is alleviated with the new state-space representation and Neural Network function approximation. As expected from a DRL approach, the agents can adapt to changes in the graph structure and non-uniform frequency requirements for priority nodes without training to do so. Though the patrolling policies learned by agents are inherently stochastic, it is observed that the agents formed cyclic trajectories to cater to the optimal policy. To protect themselves from adversaries, the DRL agents can switch trajectories upon induced random exploration without affecting the AGI during testing time, making it difficult for external agents to predict patrolling agents' behaviours.

As the Deep RL patrolling agents learn a high-level planning policy, simulation transfer to physical robots for real-life applications is achievable. The future work involves the investigation of directed edges and design of the RL System accordingly. We will also consider the case of agent society in which the policy has to adapt to online removal and

agents' addition during testing time. A further possibility of DRL approaches such as Double-DQN, Advantage Actor-Critic can also be explored.

Appendix

A. List of notions

1. $d = \max_v$(degree of node v)
2. $t =$ time steps elapsed in the episode
3. $T =$ the total time of episode
4. $|V| =$ total number of nodes in the graph
5. $N(v) =$ Total visits to node(v) during an episode
6. $INI(v)$ of a node at a time instant is given as the time steps elapsed since the last visit to the node by any patrolling robot
7. Node Visit Idleness(NVI)

$$NVI(v,i) = INI(v) \text{ at } i^{th} \text{ visit; for each } i = 1,2,3...N(v) \tag{23}$$

8. Average Node Visit Idleness(ANVI)

$$ANVI(v) = \sum_{i=0}^{N(v)} \frac{NVI(v,i)}{N(v)} \tag{24}$$

9. Global Average Node Visit Idleness(GANVI)

$$GANVI = \sum_{i=0}^{|V|} \frac{ANVI(i)}{|V|} \tag{25}$$

10. Instantaneous Global idleness (IGI)

$$IGI(t) = \sum_{i=0}^{|V|} \frac{INI(t,i)}{|V|} \tag{26}$$

11. Average Global idleness (AGI)

$$AGI = \sum_{t=0}^{T} \frac{IGI(t)}{T} \tag{27}$$

12. Apparent Node Idleness (ANI)

$$ANI(v) = \left\{ \begin{array}{ll} 3 * INI(v), & \text{for } v = \text{ priority node} \\ INI(v), & \text{for } v \neq \text{ priority node} \end{array} \right\} \tag{28}$$

13. Proposed Reward function (r)

$$r = \frac{INI(t,v)^{1.5}}{IGI(t)} \tag{29}$$

14. Exploration decay function: $\epsilon(t) = \epsilon_{start} * 0.992^{(n)}$
15. Total Model Parameters (D) = $D = 11477 + 256K + 84d$)

Table 2 MDP model empirical analysis results

Exp. no.	State space	Reward function	p	AGI range
1	S2	R2	–	90–95
2	S2	R1	1.5	92–100
3	S3	R2	–	90–93
4	S3	R1	0.5	89–95
5	S3	R1	1	85–92
6	S3	R1	1.5	82–91
7	S3	R1	2	90–93
8	S4	R2	–	72–80
9	S4	R1	0.5	71–82
10	S4	R1	1	69–78
11	S4	R1	1.5	**67–77**
12	S4	R1	2	70–80
13	S5	R2	–	80–83
14	S5	R1	0.5	81–85
15	S5	R1	1	79–82
16	S5	R1	1.5	76–82
17	S5	R1	2	80–83

Bold values indicate the best AGI performance for the experiment scenario

B. MDP model selection

The state space representations (S2–S5) as defined in Sect. 5 and the Reward Functions (R1,R2) and parameter p for R1 are compared based on AGI Metric. The empirical data collected over various experimental runs is presented in Table 2.

The best performance range is obtained for state-representation S4 and reward function R1 with $p = 1.5$ (Exp. no. 11). This analysis provides the solution to objective problem P1 in defining the MDP model elements.

C. Reinforcement learning parameter tuning

This section and D present the data of 27 experiments numbered Exp. 1 to 27 in Tables 3 and 4. The analysis is done to tune the RL and DQN parameters based on the AGI performance.

The proposed model's epsilon-greedy decay, reward function, discount factor, and time per episode RL parameters are concluded (Exp.No. 9, 19,22,24 for each case). The parameters in question were varied individually for each experiment, and the AGI performance was noted. Combining the parameter values that gave the best AGI performance was selected for the proposed RL model.

Table 3 DQN parameters tuning

Exp. no.	No. of episodes	T	Eps-start	Eps decay rate	Eps-end	Reward	γ	AGI range	Best AGI
2 Agent: Symmetric graph									
1	500	6000	0.9	0.991	0.003	R2	0.95	112–135	112.42
2	500	6000	0.9	0.991	0.0025	R2	0.95	98–106	98.06
3	500	6000	0.9	0.991	0.003	R2	0.95	83–95	83.32
4	500	6000	0.9	0.991	0.0025	R2	0.95	92–106	92.87
5	500	6000	0.9	0.991	0.006	R2	0.95	76–83	76.5
6	500	6000	0.9	0.991	0.003	R2	0.95	72–82	72.132
7	600	6000	0.9	0.991	0.0025	R1	0.95	70–79	70.7839
8	700	6000	0.95	0.992	0.006	R1	0.95	72–80	72.035
9	800	6000	0.93	0.992	0.005	R1	0.95	67–75	**67.345**
10	800	6000	0.93	0.993	0.005	R1	0.99	110–140	110
11	800	6000	0.95	0.993	0.001	R1	0.95	86–95	86.3
12	800	6000	0.93	0.992	0.004	R1	0.95	68–75	68.75
13	800	6000	0.93	0.992	0.004	R1	0.95	72–79	72.35
14	1000	6000	0.93	0.993	0.005	R1	0.95		
2 Agent: Asymmetric graph									
15	700	7000	0.93	0.992	0.005	R1	0.95	150–160	152.5
16	700	6000	0.9	0.992	0.003	R1	0.95	114–122	114.74
17	800	6000	0.93	0.993	0.004	R1	0.95	106–120	106.535
18	700	7000	0.93	0.992	0.004	R2	0.95	91–104	91.25
19	700	7000	0.93	0.992	0.005	R1	0.95	86–92	**86.5**
3 Agent: Symmetric graph									
20	550	7000	0.93	0.992	0.004	R2	0.95	51–54	51.13
21	550	7000	0.93	0.991	0.004	R2	0.95	49–53	49.435
22	600	7000	0.93	0.992	0.005	R1	0.95	48–55	**48.75**
3 Agent: Asymmetric graph									
23	500	7000	0.93	0.991	0.005	R2	0.95	79–89	79.81
24	700	7000	0.93	0.992	0.005	R1	0.95	77–85	**77.56**
25	600	7000	0.93	0.991	0.004	R1	0.95	80–86	80.32
26	700	7000	0.93	0.992	0.004	R1	0.95	78–85	78.63
27	700	7000	0.93	0.992	0.004	R1	0.96	79–89	79.81

Bold values indicate the best AGI performance for the experiment scenario

Table 4 DQN parameters tuning

Exp no.	Batch size	L1 Units	L2 Units	L3 Units	α	Optimizer	AGI range	Best AGI
2 Agent: Symmetric graph								
1	32	192	128	64	0.01	SGD	112–135	112.42
2	32	128	84	–	0.01	SGD	98–106	98.06
3	32	192	128	64	0.00075	Adam	83–95	83.32
4	64	192	128	64	0.00075	Adam	92–106	92.87
5	32	128	64	–	0.00075	Adam	76–83	76.5
6	32	128	84	–	0.00075	Adam	72–82	72.132
7	32	128	84	–	0.00075	Adam	70–79	70.7839
8	32	128	84	–	0.00075	Adam	72–80	72.035
9	32	128	84	–	0.00075	Adam	67–75	**67.345**
10	32	128	84	–	0.00075	Adam	110–140	110
11	32	128	84	–	0.00075	Adam	86–95	86.3
12	32	128	84	–	0.00075	Adam	68–75	68.75
13	32	128	84	–	0.00075	Adam	72–79	72.35
14	32	128	84	–	0.00075	Adam		
2 Agent: Asymmetric graph								
15	32	128	84	–	0.00075	Adam	150–160	152.5
16	32	128	84	–	0.00075	Adam	114–122	114.74
17	32	128	84	–	0.00075	Adam	106–120	106.535
18	32	128	84	–	0.00075	Adam	91–104	91.25
19	32	128	84	–	0.00075	Adam	86–92	**86.5**
3 Agent: Symmetric graph								
20	32	128	84	–	0.00075	Adam	51–54	51.13
21	32	128	84	–	0.00075	Adam	49–53	49.435
22	32	128	84	–	0.00075	Adam	48–55	**48.75**
3 Agent: Asymmetric graph								
23	32	128	84	–	0.00075	Adam	79–89	79.81
24	32	128	84	–	0.00075	Adam	77–86	**77.56**
25	32	128	84	–	0.00075	Adam	80–86	80.32
26	32	192	128	64	0.00075	Adam	78–85	78.63
27	32	128	84	–	0.00075	Adam	79–89	79.81

Bold values indicate the best AGI performance for the experiment scenario

D. Deep Q-network model parameter tuning

The DQN model parameters batch size, hidden layer units, learning rate and optimiser for the proposed model are concluded (Exp.No. 9, 19,22,24 for each case). Similarly, The model parameters in question were varied individually for each experiment, and the AGI performance was noted. Combining the parameter values that gave the best AGI performance was selected for the proposed DQNetwork model.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s41315-022-00235-1.

Author contributions MJ, LV, and AS conceptualised and designed the work. MJ programmed the experiment simulation, processed the experimental data, performed the analysis, drafted the manuscript and designed the figures. LV and AS were involved in planning and supervised the work. LV and AS aided in interpreting the results and worked on critical revisions of the manuscript. MJ, LV, and AS discussed the results and commented on the manuscript.

Funding The authors did not receive support from any organisation for the submitted work.

Data availability Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organisation or entity with any financial or non-financial interest in the subject matter or materials discussed in this manuscript.

References

Agmon, N., Kaminka, G.A., Kraus, S.: Multi-robot adversarial patrolling: facing a full-knowledge opponent. J, Artif. Intell. Res. **42**(1), 887–916 (2011)

Almeida, A., Ramalho, G., Santana, H., Tedesco, P., Menezes, T., Corruble, V., Chevaleyre, Y.: Recent advances on multi-agent patrolling. In: Proceedings of the 17th Brazilian Symposium on Artificial Intelligence, pp. 474–483. São Luis, Maranhão, Brazil (2004). https://doi.org/10.1007/978-3-540-28645-5_48

Baglietto, M., Cannata, G., Capezio, F., Sgorbissa, A.: Distributed Autonomous Robotic Systems 8, chap. Multi-Robot Uniform Frequency Coverage of Significant Locations in the Environment, pp. 3–14. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00644-9_1

Chen, S., Wu, F., Shen, L., Chen, J., Ramchurn, S.D.: Multi-agent patrolling under uncertainty and threats. PLOS ONE **10**(6), 1–19 (2015). https://doi.org/10.1371/journal.pone.0130154

Chevaleyre, Y., Sempe, F., Ramalho, G.: A theoretical analysis of multi-agent patrolling strategies. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1524–1525. New York, NY, USA (2004)

Elmaliach, Y., Agmon, N., Kaminka, G.A.: Multi-robot area patrol under frequency constraints. Ann. Math. Artif. Intell. **57**(3), 293–320 (2009)

Elor, Y., Bruckstein, A.: Multi-a(ge)nt graph patrolling and partitioning. In: 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 52–57. Milan, Italy (2009). https://doi.org/10.1109/WI-IAT.2009.125

Hu, Z., Zhao, D.: Reinforcement learning for multi-agent patrol policy. In: Proceedings for 9th IEEE International Conference on Cognitive Informatics (ICCI'10), pp. 530–535. Beijing, China (2010)

Krajzewicz, D., Hertkorn, G., Feld, C., Wagner, P.: Sumo (simulation of urban mobility); an open-source traffic simulation. pp. 183–187 (2002)

Lauri, F., Koukam, A.: Robust multi-agent patrolling strategies using reinforcement learning. In: Siarry, P., Idoumghar, L., Lepagnot, J. (eds.) Swarm Intell. Based Optimiz., pp. 157–165. Mulhouse, France (2014)

Li, L., Xu, Y., Yin, J., Liang, W., Li, X., Chen, W., Han, Z.: Deep reinforcement learning approaches for content caching in cache-enabled d2d networks. IEEE Internet of Things J. **7**(1), 544–557 (2020)

Luis, S.Y., Reina, D.G., Marín, S.L.T.: A multiagent deep reinforcement learning approach for path planning in autonomous surface vehicles: The ypacaraí lake patrolling case. IEEE Access **9**(2021), 17084–17099 (2021). https://doi.org/10.1109/ACCESS.2021.3053348

Machado, A., Ramalho, G., Zucker, J.D., Drogoul, A.: Multi-agent patrolling: an empirical analysis of alternative architectures. In: Proceedings of the 3rd International Workshop on Multi-Agent Systems and Agent-Based Simulation, pp. 155–170. Bologna, Italy (2002). https://doi.org/10.1007/3-540-36483-8_11

Mao, T., Ray, L.E.: Frequency-based patrolling with heterogeneous agents and limited communication. CoRR arXiv:1402.1757 (2014)

Marier, J.S., Besse, C., Chaib-draa, B.: Solving the continuous time multiagent patrol problem. In: 2010 IEEE International Conference on Robotics and Automation, pp. 941–946 (2010)

Maza, I., Caballero, F., Capitán, J., de Dios, J.R.M., Ollero, A.: Experimental results in multi-uav coordination for disaster management and civil security applications. J. Intell. Robot. Syst. **61**(1–4), 563–585 (2011). https://doi.org/10.1007/s10846-010-9497-5

Menezes, T., Tedesco, P., Ramalho, G.: Negotiator agents for the patrolling task. In: J.S. Sichman, H. Coelho, S.O. Rezende (eds.) Advances in Artificial Intelligence - IBERAMIA-SBIA 2006, pp. 48–57. Ribeirao Preto, Brazil (2006)

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR arXiv:1312.5602 (2013)

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–33 (2015). https://doi.org/10.1038/nature14236

Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. CoRR arXiv:1812.11794 (2018)

Piciarelli, C., Foresti, G.L.: Drone patrolling with reinforcement learning. In: Proceedings of the 13th International Conference on Distributed Smart Cameras, pp. 1–6. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3349801.3349805

Portugal, D., Rocha, R.: Msp algorithm: Multi-robot patrolling based on territory allocation using balanced graph partitioning. In: Proceedings of the ACM Symposium on Applied Computing, pp. 1271–1276. Sierre, Switzerland (2010). https://doi.org/10.1145/1774088.1774360

Portugal, D., Rocha, R.: A survey on multi-robot patrolling algorithms. In: Camarinha-Matos, L.M. (ed.) Technological Innovation for Sustainability, Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS, pp. 139–146. Costa de Caparica, Portugal (2011)

Portugal, D., Rocha, R.P.: Cooperative Multi-robot Patrol in an Indoor Infrastructure, pp. 339–358. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-10807-0_16

Santana, H., Ramalho, G., Corruble, V., Ratitch, B.: Multi-agent patrolling with reinforcement learning. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1122–1129. IEEE, New York, NY, USA (2004)

Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, New York, NY (2018)

Walsh, T., Nouri, A., Li, L., Littman, M.: Learning and planning in environments with delayed feedback. Auton. Agents Multi-Agent Syst. **18**, 83–105 (2008). https://doi.org/10.1007/s10458-008-9056-7

Wiandt, B., Simon, V.: Autonomous graph partitioning for multi-agent patrolling problems. In: Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 261–268. Poznan, Poland (2018)

Meghdeep Jana obtained his bachelor's degree in Mechanical Engineering from the Indian Institute of Technology Guwahati, India in 2019. He was a Project Research Assistant under Prof. Leena Vachhani and Prof. Arpita Sinha at Autonomous Robots and Multi-agent Systems Lab, Indian Institute of Technology Bombay, India. He is currently completing his Master's Degree from Carnegie Mellon University, USA. His research is centred on Robot Learning and Intelligent Planning-Control.

Leena Vachhani received her PhD from IIT Madras, India, in embedded robotics. She is currently Professor at the Systems and Control Engineering Group, IIT Bombay, Mumbai. She has guided PhDs in embedded control and robotic applications, including multi-agent mapping, exploration, patrolling, and coverage. She is a faculty advisor of the AUV-IITB team since its inception in 2010. Her current research interests are perception modelling for single and multi-agent applications, edge computing for IoT, and multi-agent systems in IoT frameworks. She is currently leading the activity of establishing the Technology Innovation Hub (TIH) at IIT Bombay under the National Mission on Interdisciplinary Cyber-Physical Systems.

Arpita Sinha is a faculty in Systems and Control Engineering at IIT Bombay since 2009. Before that, she was a postdoctoral fellow at Cranfield University, UK, for a year. She obtained her bachelor's degree from the Electrical Engineering Department at Jadavpur University in 2001, her master's degree from the Electrical Engineering Department at IIT Kanpur in 2003, and her doctoral degree from the Aerospace Department at IISc Bangalore in 2007. Her research interest lies at the intersection of robotics and control systems. She works on guidance and control of autonomous vehicles, multiple vehicle coordination, and distributed decision making.