

Q-MARL: A QUANTUM-INSPIRED ALGORITHM USING NEURAL MESSAGE PASSING FOR LARGE-SCALE MULTI-AGENT REINFORCEMENT LEARNING

KHA VO AND CHIN-TENG LIN

ABSTRACT. Inspired by a graph-based technique for predicting molecular properties in quantum chemistry – atoms’ position within molecules in three-dimensional space – we present Q-MARL, a completely decentralised learning architecture that supports very large-scale multi-agent reinforcement learning scenarios without the need for strong assumptions like common rewards or agent order. The key is to treat each agent as relative to its surrounding agents in an environment that is presumed to change dynamically. Hence, in each time step, an agent is the centre of its own neighbourhood and also a neighbour to many other agents. Each role is formulated as a sub-graph, and each sub-graph is used as a training sample. A message-passing neural network supports full-scale vertex and edge interaction within a local neighbourhood, while a parameter governing the depth of the sub-graphs eases the training burden. During testing, an agent’s actions are locally ensembled across all the sub-graphs that contain it, resulting in robust decisions. Where other approaches struggle to manage 50 agents, Q-MARL can easily marshal thousands. A detailed theoretical analysis proves improvement and convergence, and simulations with the typical collaborative and competitive scenarios show dramatically faster training speeds and reduced training losses.

1. INTRODUCTION

The recent success of reinforcement learning (RL) in sophisticated cooperative/ competitive strategy games, such as DOTA [11], StarCraft II [34], ATARI [29], AlphaGo [41, 42], and poker [30, 6] has raised the question of whether these algorithms can be generalised into scenarios involving a much larger number of agents. The main challenge is that the size of the joint space of actions in multi-agent RL (MARL) scenarios increases exponentially with each new agent. So, at hundreds or thousands of agents, finding optimal individual policies for each agent to complete their objectives becomes computationally infeasible. Moreover, training any single agent without taking the behaviour of other agents into account would violate the compulsory Markov property assumption of the environment’s stationarity [40, 8, 16]. From this violation, a cascade of other problems follows, including suboptimal credit assignments [2], imbalanced exploitation/exploration [28], combinatorial explosions [7], and policy overfitting [35]. Hence, despite the aid of advanced machine learning techniques like deep neural networks

[32], designing algorithms to solve the non-stationarity of MARL remains the pivotal problem in RL [36].

In MARL, non-stationarity problems are not straightforward to address, especially once the number of agents exceeds just a few dozens. For example, combinatorial explosion means that one cannot naively view the joint state and action space of all agents as a single space and then implement single-agent algorithms. This also precludes the use of central controller-like algorithms that distribute actions to agents [26, 15, 33, 12]. Some researchers have attempted to override the Markov stationarity property by making a few assumptions about the environment, such as a common reward system [47, 4]. However, with too many agents, the state and action spaces become too large to learn trustworthy policies and can even fail to converge to a local optimum [24, 25].

In competitive and collaborative MARL scenarios, the actions of agents are correlated. The degree of influence these correlations have is governed by the *neighbourhood* information each agent contains. For instance, in a simulated soccer game, the action of a striker with the ball is strongly affected by the surrounding agents, i.e., the opponent’s defenders and the teammate’s mid-fielders, but weakly affected by the agents farther away, i.e., the teammate’s goalkeeper and defenders. Similarly, in a cyber-security scenario, connected computers form a graph network, where a malicious attack must find the intermediate agents on its way to the final computer target. These examples demonstrate that focusing on appropriate information is more efficient than using all information that can possibly be retrieved from the environment.

1.1. Related Works.

1.1.1. *Issues in Centralised Learning.* The collaborative and competitive nature of MARL leads to a handful of difficult challenges, the most notable of which is the possibility of an extremely large joint state/action space. In fact, early research [24, 25] on value-based approaches found this problem to be intractable at a certain number of agents. In looking for solutions, some researchers turned to assumptions about the environment. For example, Wang and Sandholm [47] assumed a common reward system for all agents so the value function could be estimated locally (see also [4]). This simple assumption makes training trivial; however, it also violates the stationarity characteristic required for RL [40, 8, 16].

Deep learning has also been explored for answers by embedding neural networks into the policy learning process as a viable function approximator [26, 15, 33, 12]. The general approach here is to implement a common reward system [33, 15] to control each agent’s learning with a central manager [12, 26]. However, combinatorial explosion prevents all of these approaches

from scaling to large environments. Hence, to reduce the joint state/action spaces and speed up the learning process, some frameworks strike a balance between full-scale and local training by allowing communication between agents. A full-scale training using the attention mechanism for collaborative tasks with two policy networks in an end-to-end architecture was proposed and named ATOC [20]. This work is an extension of the multi-agent deep deterministic policy gradient (MADDPG) [26]. However, in MADDPG, the authors also assumed the existence of fully connected agents in the communication network. TarMAC [10] provides an architecture for targeted continuous communication via soft attention that allows agents to select other appropriate agents to communicate with at each time step. Both methods show that incorporating the attention mechanism improves performance with complete centralised small-scale scenarios but, again, large-scale scenarios are still beyond computational limits.

1.1.2. Agent Communication. One typical approach that aims to achieve a balance between full-scale training and local training to accelerate the process is to allow partial communication between agents. Communication in RL is an emerging field of research for collaborative tasks in a partially observable environment. Reinforced inter-agent learning (RIAL) and differentiable inter-agent learning (DIAL) were proposed in [13] to discover the communication protocols between agents. These methods employ a neural network to determine each agent’s Q values, as well as the communication message to all other agents in the next time step. As a result these methods work by centralised learning and decentralised execution. The centralised learning is not feasible in large-scale scenarios, i.e, the authors of RIAL/DIAL only experimented on small scenarios with up to 4 agents. Memory-driven multi-agent deep deterministic policy gradient (MD-MADDPG) [38] was built on top of the classic MADDPG [26] by adding a shared memory as a communication protocol across agents. This shared memory is processed by each agent before taking an action, and is dependent on each agent’s private observation. Similar to RIAL/DIAL, MD-MADDPG also lacks a mechanism to deal with large-scale environment, because the training flow will explode if the number of agents exceeds a few dozens. The authors of MD-MADDPG only conducted experiments in toy problems with up to 2 agents. Similarly, Dropout-MADDPG [21] was inspired by the dropout technique in supervised learning [43] and MADDPG, but also suffered from the *curse of agent count*. Although the dropout mechanism implemented on communication messages had some positive effects such as improved training speed and steady-state performance, it cannot compensate the great drawback of the infeasibility in large scenarios, as all experiments only involved less than 10 agents. CommNet [44] provides continuous communication through a simple broadcasting channel. In this strategy, an averaged message from all agents in one network layer is used as input for the next layer. However, although CommNet has been shown to work well in collaborative tasks, it does not

consider competitive scenarios or scenarios with heterogeneous agents. Designed as a robust approach to combat and capture games, and StarCraft in particular, BiCnet [37] is a vectorized extension of the actor-critic approach that employs a bi-directional recurrent communication framework. However, since full-scale training is required, feasible scenarios are limited to a maximum of 50 agents.

1.1.3. Approaches for big-scale scenarios. Of the impact-based strategies to tackle large scenarios, mean-field MARL [48] approximates the mean effect of neighbouring agents. This method has two variants, one policy-based (mean-field actor-critic) and the other value-based (mean-field Q-learning). Although applicable to large-scale scenarios with homogeneous agents, this approach only considers the influence of neighbours at a very broad level; the nuanced information is lost. Therefore, in simple scenarios with a diverse range of heterogeneous agent types, such as speaker-listener [26], the mean-field approach has difficulty representing the dominant influence of the speaker.

Jaques et al. [18] is the other impact-based framework. This solution involves thoroughly assessing social influence across all agents by comprehensively simulating the alternative actions each agent could have taken. Actions with the greatest influence over other agents are detected and encouraged later in the process. This strategy is good for predicting degrees of influence across diverse agent types, but it does not establish a strong connection between the causal prediction and the real target. In other words, the causal analysis is unsupervised and can be prone to failure in scenarios where the probability of successful actions during training are low.

More recently, there have been some attempts to integrate the concept of local neighbourhoods into MARL through attention [19, 3] and relevance graph embedding [27]. Since these approaches are typical to be compared with our proposed method with regards to the relatively large number of agents, we will go into more details with some experimental comparison in Section 3.3.

1.2. Inspiration from Quantum Chemistry. Predicting molecular properties is regarded as one of the most challenging problems in quantum chemistry [14], in part because the atomic symmetry of molecules demands a graph-based mechanism that can deal with the isomorphic invariance of an atom’s position in three-dimensional space. Example problems have even been used as the basis of world-class machine learning competitions, with thousands of entrants across the globe [9]. As a team who finished in the gold-medal zone of one of these competitions in 2019 (rank 10/2749) [9], we

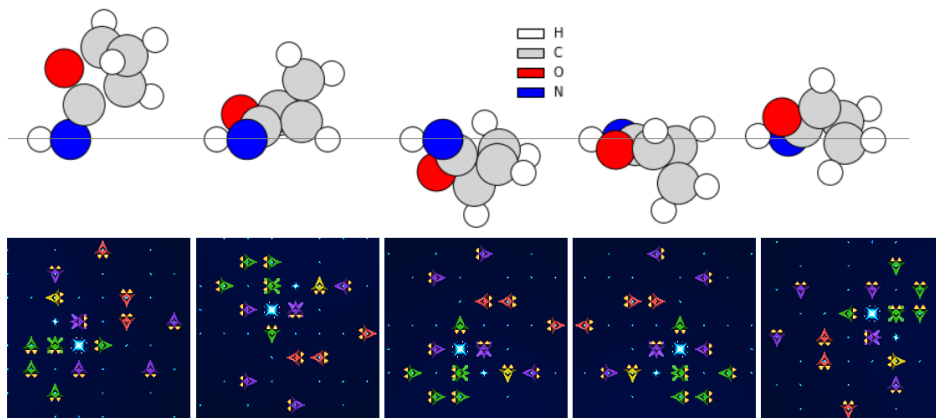


FIGURE 1. Inspiration for MARL from quantum chemistry. Top row: illustrations of five differently placed counterparts of a same molecule C_4H_5ON . The *scalar coupling constant* [9] between any pair of atoms in the molecule is invariant with respect to the coordination of the molecule. Bottom row: illustration of five differently placed scenes of the same relative structure in a 4-player Halite game [17, 1]. The optimal action of each agent should be invariant with respect to the scene’s rotation/symmetry.

were intrigued to find some interesting parallels between quantum chemistry and MARL as follows:

- Consider that the atoms in a molecule in quantum chemistry are equivalent to the agents in an interactive game in MARL. In both scenarios, the relative pair-wise distances between entities are crucial.
- Rotational invariance and graph isomorphism are equivalent and are key, in that randomly rotating a molecule is equivalent to changing the coordinate origin in a MARL game, and placing entities in an equivalent symmetric way should give the same outcome.
- The number of entities is not fixed, rather it differs given the context.
- The influence of one entity over another is governed by proximity.

These similarities prompted us to wonder whether state-of-the-art methods in quantum chemistry might be state-of-the-art for MARL. However, transposing a working concept from quantum chemistry to MARL poses multiple technical challenges. The first being how to reformulate a supervised learning problem with a fixed dataset in quantum chemistry into an RL problem to be solved by self-playing with multiple agents given infinite simulating data.

In this paper, we propose a framework called Q-MARL (Quantum-MARL), to fully decentralise learning, inspired by state-of-the-art works from quantum chemistry. Our graph-based approach is to address large-scale MARL in joint collaborative-competitive scenarios. The model is capable of capturing nuanced influence across agents within a neighbourhood by efficiently decomposing the large original environment into sub-graphs. The key characteristic of our approach is the invariance of the model with respect to the rotation or isomorphism of graphs representing neighborhoods of agents. At each time step, each agent is able to observe only its own information, i.e., state and reward, and the information of agents in its neighbourhood. Stated differently, the neighbouring information forms a time-varying graph, where each vertex is an agent and an edge between two agents indicates a temporary neighbouring bond between them. The agents do not necessarily form a full graph in which any vertex can be accessed by any vertex. The time-varying property of the graph guarantees convergence to global optimality while substantially reducing the size of the problem. Our graph-based approach addresses this problem by not discriminating each agent's identity. A message-passing mechanism is used as the main backbone of the graph network, and all vertices and edges can fully interact with each other in a sub-graph. By accumulating the gradient of each specific agent in all sub-graphs that it belongs to, training becomes more stable and converges faster. The graph network is coupled with a λ - *step* actor-critic RL algorithm [23] and is deployed in a decentralized fashion, i.e., at test time, each agent acquires its own information and the neighbourhood information, forms a graph, and executes an action from the trained model without the existence of a central manager.

1.3. Research Gaps and Contributions. Given the literature review in Section 1.1, the following research gaps that need to be fulfilled are summarized as follows.

- Previous works that proposed decentralized training schemes in big-scale scenarios only use shared model parameters with respect to each neighborhood, thus actions of agents within a neighborhood are independent. This is not desired. There lacks a centralized training approach within each decentralized neighborhood.
- To the best of our knowledge, there were no works dealing with the graph's rotation and isomorphism to tackle overestimation and improve generalization in MARL. We were inspired by this idea from quantum chemistry where molecules' rotational invariance and graph isomorphism play an important role in model design. The advent of this innovative idea also tailors with the research gap made in the previous point: a dynamic graph representing a network of agents with varying size/topology/positions is the most generalized and can be viewed as a centralized fashion within each neighborhood.

- There lacks an approach to fully centralize each local neighborhood sample. Suppose after decentralizing the full scenario into overlapped neighborhoods, the action of each agent is desired to be influenced by the actions all other agents it communicates to. This is not the case in most works in literature.

The contributions of our work can be generally summarized as follows.

- Formulate a general MARL scenario as a graph-based problem, where homogeneous agents' policies can be employed in a decentralized fashion. Convergence to global optimality is guaranteed and proved by assuming a time-varying graph transition.
- Propose a graph-based model based on a message-passing neural network mechanism, where all vertices and edges can interact with each other. The rotational invariance of the input vertices is a crucial factor for the convergence analysis mentioned in the first point.
- Implement our proposed method in typical multi-agent scenarios, with comparison to recent state-of-the-art algorithms in the literature. Our results suggest that given an existing algorithm, the convergence speed is drastically enhanced by using the sub-graph embedding, and the decentralized framework is preserved with promising global reward performance. Our method is therefore well tailored to large-scale competitive and collaborative multi-agent tasks.

The rest of this paper is organised as follows. Section 2 presents theoretical aspects of the work, including the convergence analysis of policy gradient in MARL (Section 2.1), graph-based formulation of the problems (Section 2.2), and neural message passing architecture (Section 2.3). The experiment scenarios and results are presented in Section 3.1 and Section 3.2 respectively, followed by the comparison with other studies in Section 3.3. Finally, Section 4 draws key brief conclusion points.

2. GRAPH-BASED MULTI-AGENT REINFORCEMENT LEARNING

2.1. Background and Problem Setting. Consider an environment with N agents, each having a finite state space \mathcal{S}^i and a finite action space \mathcal{A}^i . The joint state and action spaces of all agents are denoted as $\mathcal{S} = \Pi_i \mathcal{S}^i$ and $\mathcal{A} = \Pi_i \mathcal{A}^i$, respectively. In each time step t , agent i is in an instantaneous state S_t^i , performs action A_t^i drawn from its current policy $\pi^i(a^i|s)$ which is parametrised by θ^i , then transitions to a new state S_{t+1}^i and receives a reward R_{t+1}^i . From a global perspective, the joint state and joint action of all agents at time step t , are denoted as $S_t = (S_t^1, \dots, S_t^N)$ and $A_t = (A_t^1, \dots, A_t^N)$, respectively.

The parametric policy for each agent is formulated as

$$(2.1) \quad \pi^i(a^i|s) = \mathbb{P}\{A_t^i = a^i \mid S_t = s, \theta^i\}$$

This policy indicates the probability of agent i taking action a^i given the joint state s and the agent's policy parameters θ^i at time step t . This formulation expresses the multi-agent scenario as a single-agent scenario, with the joint state and action at each time step as the state of action of a single agent. The sample of joint return at time step t can be formulated simply as the averaged return of all agents, i.e., $R_t = \frac{1}{N} \sum_i R_t^i$. The optimal joint policy $\tilde{\theta} = (\tilde{\theta}^1, \dots, \tilde{\theta}^N)$ is obtained by maximizing the expectation of the long-term discount reward

$$(2.2) \quad \tilde{\theta} = \underset{\theta}{\text{maximize}} \lambda(\theta) \doteq \mathbb{E}[G_t | \theta]$$

where $G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$ denotes the instantaneous long-term reward and $\gamma \in (0, 1]$ is a discount factor influencing the impact of near-future actions. We hereby present the extended policy gradient in a multi-agent environment.

2.2. Theoretical Foundations.

Theorem 1. *The gradient of the global reward $\lambda(\theta)$ with respect to any local policy parameter θ^i of agent i can be derived as*

$$(2.3) \quad \nabla_{\theta^i} \lambda(\theta) = \mathbb{E}_{\pi} \left[G_t \nabla_{\theta^i} \ln \pi^i(A_t^i | S_t) \right],$$

where \mathbb{E}_{π} denotes the expectation of the random variables inside the brackets following the joint policy π .

Proof. Conventional policy gradient (PG) theorem [46] states that the gradient of the long-term discounted return with respect to the joint policy θ is

$$(2.4) \quad \begin{aligned} \nabla \lambda(\theta) &= \sum_{s \in \mathcal{S}} d_{\pi}(s) \sum_{a \in \mathcal{A}} q_{\pi}(s, a) \nabla \pi(a | s) \\ &= \mathbb{E}_{\pi} \left[G_t \frac{\nabla \pi(A_t | S_t)}{\pi(A_t | S_t)} \right]. \end{aligned}$$

The first equality is derived as the original result of PG [46], while the second equality holds by replacing the variables s and a with stochastic samples S_t , A_t and obtaining the expectation, as explained in [45]. Since by definition $\pi(A_t | S_t) = \prod_i \pi^i(A_t^i | S_t)$, then from (2.4), the proof is straightforward by taking the gradient with respect to only local parameters θ^i , given the fact that all individual policies are conditionally independent, i.e., $\nabla_{\theta^i} \ln \pi^j(A_t^j | S_t) = 0 \ \forall j \neq i$, and $\frac{\nabla_{\mathbf{x}} f(\mathbf{x})}{f(\mathbf{x})} = \nabla_{\mathbf{x}} \ln f(\mathbf{x}) \ \forall f$. \square

Here, we introduce two widely used terms in RL - the *state-value* function $v_{\pi}(s) \doteq \mathbb{E}_{\pi}[G_t | S_t = s]$ and the *action-value* function $q_{\pi}(s, a) \doteq \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$. The state-value function expresses the expectation of long-term reward in a specific state s following policy π , while the action-value function expresses the expected long-term reward when starting in state s , performing action a , and following π . The policy gradient theorem of graph-based MARL suggests that the gradient of the global reward with respect to the

local policy parameter θ^i of agent i is independent of the other agents' policies and requires only the unbiased estimate G_t of q_π . However, in our decentralised graph-based MARL we do not have access to G_t during training. Therefore, each agent is coupled with a local estimate $\hat{q}^i(s, a)$, which is parametrized by w^i . Further details are provided in Section 2.3.

Theorem 2. *An agent's policy θ^i can be updated locally via the following procedure*

$$(2.5) \quad \theta^i \leftarrow \theta^i + \alpha \delta^i \nabla_{\theta^i} \ln \pi^i(A_t^i | S_t),$$

where $\delta^i = \hat{q}^i(S_t, A_t) - v(S_t, A_t^{(-i)})$ and $A_t^{(-i)}$ denote the actions of all agents who are independent of agent i at time step t .

Proof. From Theorem 1, it is evident that the sample $G_t \nabla_{\theta^i} \ln \pi^i(A_t^i | S_t)$ serves as a stochastic estimate following the Monte Carlo gradient ascent algorithm [39]. With a proper learning rate α , the update procedure for θ^i is

$$(2.6) \quad \theta^i \leftarrow \theta^i + \alpha G_t \nabla_{\theta^i} \ln \pi^i(A_t^i | S_t).$$

From a general perspective, G_t is λ steps in the future and difficult to retrieve. Thus, this value can be substituted with a semi-gradient target $G_t \approx R_{t+1} + \gamma v(S_{t+1})$ [45]. However, R_{t+1} is still inaccessible locally, and the true global state-value function v is unknown. Thus, an alternative unbiased estimate of G_t must be obtained. To this end, during training, each local agent is coupled with its own action-value network that serves as a critic, denoted by $\hat{q}^i(s, a)$ and parametrized by w^i . According to the definition of the action-value function, \hat{q}^i is a direct unbiased estimate of G_t . Moreover, adding a baseline offset quantity $v(S_t, A_t^{(-i)})$ to G_t preserves the update procedure and stabilizes the training process by reducing the target variance since from (2.4), we have $\sum_{a^i} v(s, a^{(-i)}) \nabla_{\theta^i} \pi^i(a^i | s) = v(s, a^{(-i)}) \nabla_{\theta^i} \sum_{a^i} \pi^i(a^i | s) = 0$ because $\sum_{a^i} \pi^i(a^i | s) = 1$. \square

The baseline can be computed by sweeping through all possible actions a^i combined with the instantaneous samples of other agents A_t^j , $j \neq i$ as

$$v^i(S_t, A_t^{(-i)}) = \sum_{a^i} \pi^i(a^i | S_t) \cdot \hat{q}^i\left(S_t, (A_t^{(1)}, \dots, A_t^{(i-1)}, a^i, A_t^{(i+1)}, \dots, A_t^{(N)})\right).$$

Graph-based Formulation

According to Theorem 2, the global state S_t must be accessed in each time step to train every individual agent. Yet, this is a difficult task in large-scale MARL scenarios because, as discussed, an agent cannot retrieve the global state during testing due to specific environmental specifications. Decomposing the full environment into smaller sub-environments, each one specific to an individual agent, offers multiple benefits. First, agents require far less but suitable training data in each time step which accelerates the training

process, plus the acquired data for each agent is concentrated on the local environment, which means irrelevant or spurious interactions with other uncorrelated agents can be discarded leaving only quality data. Second, each agent’s decision can be determined by an ensemble of its actions in N sub-graphs \mathcal{G}^i during each time step, i.e., from its actions in all the neighbourhoods it belongs to, which makes for better outcome and also helps to stabilise the training process.

We model the full environment as an undirected graph $\mathcal{G}(V, E)$ consisting of a set of vertices V and a set of edges E . Each vertex in V represents one agent in the environment, and each edge in E is a bidirectional connection between two agents. For each agent, there is one sub-graph \mathcal{G}^i based on the perspective of agent i . All other agents in the neighbourhood of agent i are present in \mathcal{G}^i . As a result, multiple sub-graphs can contain the same agent.

Some modifications to the vanilla RL algorithm have to be made. Without loss of generality, all decomposed sub-graphs have identical roles in learning from the global perspective. In other words, the sub-graphs of one time step in MARL can be thought of in the same way as one-batch data samples in supervised learning. This treatment allows a single model to be built for the whole environment. The model takes each sub-graph as input and stochastically performs back-propagation on each agent in the sub-graph. The model has two major components: the main policy $\pi(a|s)$ and the action-valued critic $\hat{q}(s, a)$. With a slight abuse of notation, π and \hat{q} operate only on sub-graphs; hence, the action variable a and state variable s represent the joint action and joint state of all agents in a sub-graph. The output of the model is agent-wise, which means each agent j in sub-graph \mathcal{G}^i is assigned an action according to a probability policy $\pi(a_j^i|s^i)$. The pseudo-code of our algorithm presented in Algorithm 1, where we have replaced s^i with \mathcal{G}^i for notational convenience. The graph decomposition process is illustrated in Figure 2.

Theorem 3. *The ensemble action of each agent i aggregated from all relevant sub-graphs that contain it, denoted as A^i in Algorithm 1, is statistically improved from any single action A_j^i from any sub-graph that contains agent i .*

Proof. Suppose for a specific agent i , we draw each scenario circumstance (y, \mathbf{x}) from the probability distribution P to form a sub-graph \mathcal{G} containing agent i , where y is the optimal action for agent i and \mathbf{x} is the input feature vector of agent i in \mathcal{G} . Let $\pi(\mathbf{x}, \mathcal{G})$ be the learner’s output action that is performed on \mathcal{G} , then subsequently, the ensemble action averaged on all possibilities of \mathcal{G} can be defined as follows:

$$(2.7) \quad \pi_{\text{ensemble}}(\mathbf{x}, P) = \mathbb{E}[\pi(\mathbf{x}, \mathcal{G})].$$

Algorithm 1 Graph-Based MARL

```

1: Initialize: Graph policy model  $\pi(a|s)$  with parameters  $\theta$ 
2: Initialize: Action-value function  $\hat{q}(s, a)$  with parameters  $w$ 
3: Input: learning rates  $\alpha_\theta, \alpha_w$ 
4: for each episode do
5:   for each time step do
6:     if episode end criteria met then: break
7:      $\mathcal{G}^i \leftarrow \text{decompose}(S)$ 
8:     Sample  $A_j^i \sim \pi(\mathcal{G}^i)$  for all relevant  $j$  in each  $\mathcal{G}_i$ 
9:     Ensemble action  $A^i = \sum_j A_j^i$  for each  $i$ 
10:    Perform actions  $A = [A^1, \dots, A^N]$  to the environment
11:    Observe  $\mathcal{G}' = [\mathcal{G}'^1, \dots, \mathcal{G}'^N], R = [R^1, \dots, R^N]$ 
12:    for each  $i$  do
13:      for each relevant  $j$  in  $\mathcal{G}^i$  do
14:         $v \leftarrow \sum_j \pi(a_j|\mathcal{G}^i) \cdot \hat{q}(\mathcal{G}^i, [A_1^i, \dots, a_j^i, \dots, A_{\text{end}}^i])$ 
15:         $v' \leftarrow \sum_j \pi(a_j|\mathcal{G}'^i) \cdot \hat{q}(\mathcal{G}'^i, [A_1^i, \dots, a_j^i, \dots, A_{\text{end}}^i])$ 
16:         $\delta \leftarrow R^i + \gamma v' - v$ 
17:         $w \leftarrow w + \alpha_w \delta \nabla_w \hat{q}(\mathcal{G}^i, A^i)$ 
18:         $\theta \leftarrow \theta + \alpha_\theta \delta \nabla_\theta \ln \pi(A_j^i|\mathcal{G}^i)$ 

```

We denote the random variables of \mathbf{x}, y as \mathbf{X}, Y , which are independent of \mathcal{G} . The averaged error e on a weak single action $\pi(\mathbf{x}, \mathcal{G})$ over all possibilities of \mathcal{G} can be calculated by

$$\begin{aligned}
e &= \mathbb{E}_{\mathcal{G}} \mathbb{E}_{\mathbf{X}, Y} [Y - \pi(\mathbf{X}, \mathcal{G})]^2 \\
(2.8) \quad &= \mathbb{E}_{\mathcal{G}} \mathbb{E}_{\mathbf{X}, Y} [Y^2] - 2\mathbb{E}_{\mathbf{X}, Y} [Y] \mathbb{E}_{\mathcal{G}} [\pi(\mathbf{X}, \mathcal{G})] + \mathbb{E}_{\mathbf{X}, Y} \mathbb{E}_{\mathcal{G}} [\pi(\mathbf{X}, \mathcal{G})]^2 \\
&= \mathbb{E}_{\mathbf{X}, Y} [Y^2] - 2\mathbb{E}_{\mathbf{X}, Y} [Y \pi_{\text{ensemble}}] + \mathbb{E}_{\mathbf{X}, Y} \mathbb{E}_{\mathcal{G}} [\pi(\mathbf{X}, \mathcal{G})]^2.
\end{aligned}$$

On the other hand, the error produced by π_{ensemble} is computed as

$$\begin{aligned}
e_{\text{ensemble}} &= \mathbb{E}_{\mathbf{X}, Y} [Y - \pi_{\text{ensemble}}(\mathbf{X}, P)]^2 \\
(2.9) \quad &= \mathbb{E}_{\mathbf{X}, Y} [Y^2] - 2\mathbb{E}_{\mathbf{X}, Y} [Y \pi_{\text{ensemble}}] + \mathbb{E}_{\mathbf{X}, Y} [\mathbb{E}_{\mathcal{G}} \pi(\mathbf{X}, \mathcal{G})]^2.
\end{aligned}$$

Since $\mathbb{E} Z^2 \geq \mathbb{E}^2 Z \forall Z$ we yield

$$(2.10) \quad e \geq e_{\text{ensemble}}$$

The gap of e and e_{ensemble} is dependent on the variance of $\pi_{\text{ensemble}}(\mathbf{X}, P)$ as

$$\begin{aligned}
(2.11) \quad e - e_{\text{ensemble}} &\geq \mathbb{E}_{\mathbf{X}, Y} \mathbb{E}_{\mathcal{G}} [\pi(\mathbf{X}, \mathcal{G})^2] - \mathbb{E}_{\mathbf{X}, Y} [\mathbb{E}_{\mathcal{G}}^2 \pi(\mathbf{X}, \mathcal{G})] \\
&= \mathbb{E}_{\mathbf{X}, Y} [\text{Var}_{\mathcal{G}} \pi(\mathbf{X}, \mathcal{G})].
\end{aligned}$$

□

Thus, we expect to obtain an improved action by ensembling single actions from the population of sub-graphs \mathcal{G} drawn from P that contains a specific agent i . This is achieved by performing the inference of all agents in each sub-graph after the environment decomposition step, then aggregating the actions of agent i from the sub-graphs it belongs to.

2.3. Neural Message Passing. Our graph-based model architecture is inspired by the neural message-passing (NMP) networks of quantum chemistry [14]. We argue that a large-scale MARL environment is similar to atomistic space in quantum chemistry, where each agent in a MARL scenario is equivalent to an atom in a molecule, and the neighbouring information across agents is equivalent to the inter-atomistic interactions between atoms. The architecture of our proposed network is illustrated in Figure 3. The feature vector for the state of each agent i in the graph is denoted as \mathbf{s}_i , and the edge, i.e., the neighbouring information between two agents i, j is denoted as \mathbf{z}_{ij} . The hidden features of agent i at layer l are dispersed to its neighbours $j \in \mathcal{N}(i)$ by a vertex update block \mathcal{V} as

$$(2.12) \quad \mathbf{s}_i^{l+1} = \mathcal{V}(\mathbf{s}_i^l, \mathbf{z}_{ij}^l).$$

The edge update process for \mathbf{z}_{ij}^l is dependent on itself and the resulting updated vertices at its two ends $\mathbf{s}_i^{l+1}, \mathbf{s}_j^{l+1}$ as

$$(2.13) \quad \mathbf{z}_{ij}^{l+1} = \mathcal{E}(\mathbf{s}_i^{l+1}, \mathbf{s}_j^{l+1}, \mathbf{z}_{ij}^l).$$

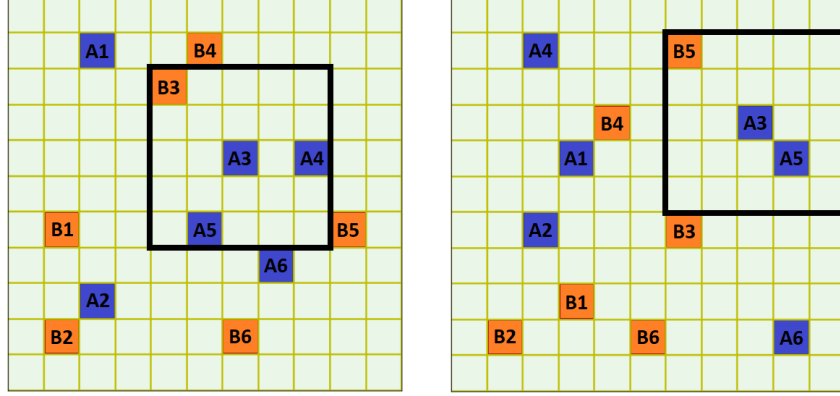
The input features for the network can either be the raw properties of each vertex or their spatial positions. In this architecture, the interaction block is crucial to capturing the relative correlation of adjacent vertices. The input states, or observations, of all agents are denoted as $\mathbf{s} = (\mathbf{s}_i)_{i=1}^N$. We define h_{lin} and h_{rel} as fully connected neural network layers with linear and ReLU [31] activation functions, respectively. These layers are the backbone for all message-passing blocks in the framework. The weight matrix \mathbf{W} of this layer provides a linear combination of input features, biased by \mathbf{b} as

$$h_{\text{lin}}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}.$$

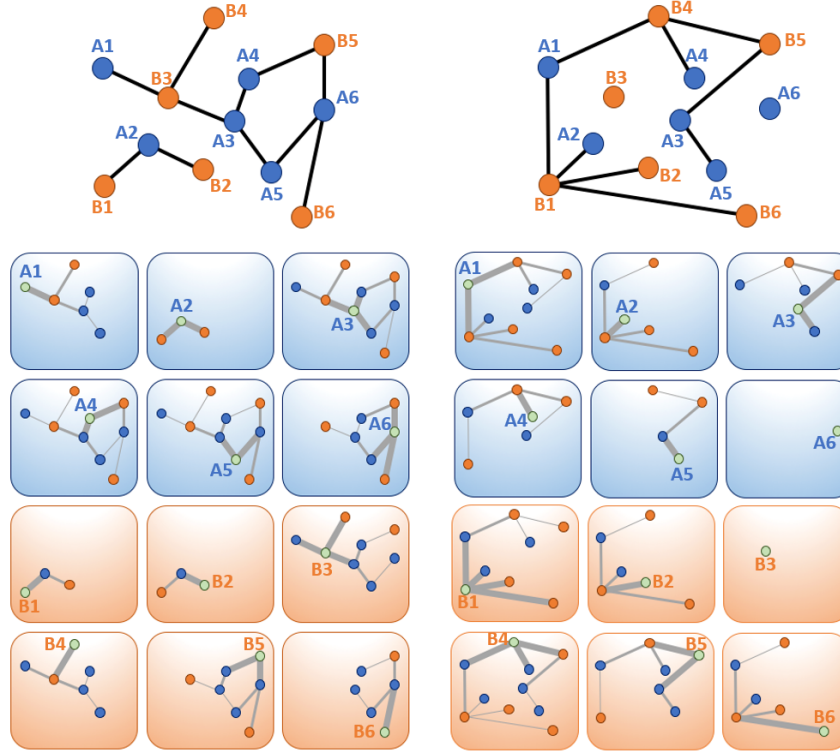
In h_{rel} , a ReLU activation function [31] is applied to yield

$$h_{\text{rel}}(\mathbf{x}) = \max(0, \mathbf{W}\mathbf{x} + \mathbf{b}).$$

The summation block \sum_j indicates the sum of all edges \mathbf{z}_{ij} in the neighbourhood of i .



(A) Scenario Displacement



(B) Graph Formulation

FIGURE 2. An example of how the graph decomposition process differs for two different time steps. The blue and orange vertices represent two homogeneous groups of agents, i.e., two teams. There are 12 sub-graphs for each time step, each formed from the perspective of a different individual agent as indicated by the green vertex, and extending to its 3rd-degree neighbours. This depth of degree, and therefore the complexity of the model, is controlled by a hyperparameter.

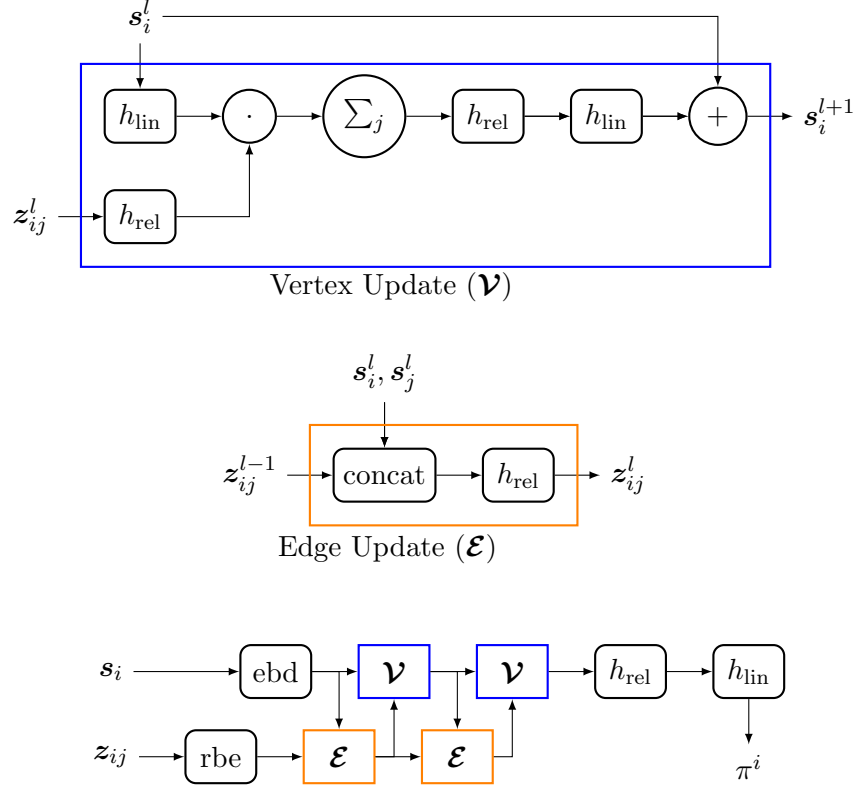


FIGURE 3. The proposed neural message-passing (NMP) architecture. The full architecture consists of recurring vertex update (\mathcal{V}) blocks and edge update (\mathcal{E}) blocks. The smaller blocks (fc, rl, concat, ebd, rbe, \cdot , and $+$) are described in Section 2.3.

Radial Basis Expansion (rbe): Since one-hot encoding is the preferred representation for categorical features with neural networks, we argue that a similar formulation is also preferable for the scalar distance. Hence, the distance between two agents d_{ij} is encoded as an n_{max} -dimensional feature vector z_{ij} using a radial basis function, with the n -th element as

$$(2.14) \quad z_{ij,n} = \exp\left(-\frac{(d_{ij} - n\Delta d)^2}{\Delta d}\right),$$

where Δd is the increment step size. Depending on the RL environment setting, we set $n\Delta d$ equal to the maximum view range of all agents, and $n_{\text{max}} = 10$ is the chosen dimension.

Training Setting: The Adam optimizer [22] with an initial learning rate of 0.01 is implemented. A reduce-on-plateau schedule is used for decaying the learning rate with respect to training progression: the learning rate is

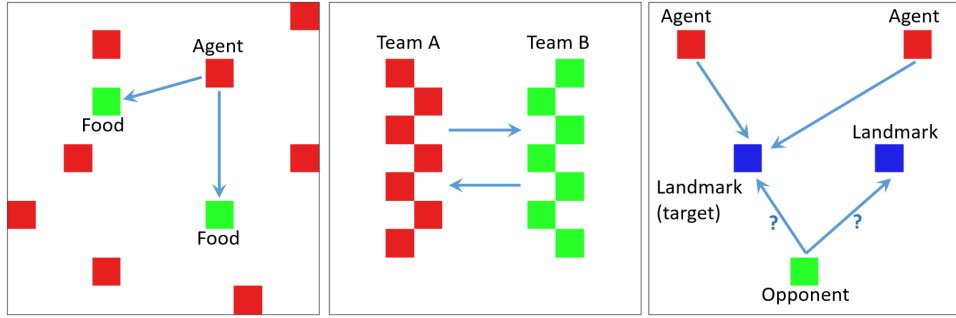


FIGURE 4. Illustrations of the three MARL scenarios considered in this paper. From left to right: Jungle, Battle, Deception.

decreased by 5% for every 10 consecutive batches with no reward improvements.

3. EXPERIMENTS AND RESULTS

To verify our theories and assess Q-MARL’s performance in situ, we conducted experiments in typical MARL scenarios and compared the results to those in the literature.

3.1. Scenarios. The three grid-world MARL scenarios we conducted were commonly-used: *jungle*, *battle*, and *deception* [26]. In all three scenarios, each agent occupies one cell in a grid-world environment. Overlapping occupants are allowed in any cell. At any time step, agents can apply their actions to the environment and change the relative properties accordingly. The action of an agent are governed by its learned policy, with input states as its local observation, which is a 3×3 rectangle surrounding the cell it occupies. Homogeneous agents share the same policy. Two agents are directly connected in the graph if one is present in the local observation of the other. The action space is common for all three scenarios. At each time step, an agent has 5 actions to choose from: move up, move down, move left, move right, and stay idle. Actions are only eligible if they do not result in the agent occupying an already-occupied cell, say by a wall or other obstacle. An illustration of the different types of physical interactions between agents and with the environment is provided in Figure 4. The different rules, goals, and reward schemes associated with each of the three scenarios follows. The interactions between agents and rewarding schemes vary across the 3 scenarios are described as follows.

Jungle: In this scenario, agents must search for stationary foods placed in the environment, and they are allowed to attack and kill each other in the process. The objective is to maximize the long-term total amount of

food consumed (reward) for all agents. This is ostensibly a collaborative scenario that poses a difficult social-dilemma, where agents must find ways to obtain food without killing each other. The short-term rewards for killing are attractive but detrimental to the long-term rewards. We formulated the specific environment and reward scheme as follows. In each time step, an agent receives an instantaneous reward +1 for sitting next to a food cell, and 0 otherwise. If an agent is adjacent to at least one other agent three times, then it is killed and removed from the environment. The Food cells, however, permanently exist.

Battle: In this scenario, two teams, each with N agents, are placed randomly in the environment. The goal is to have the most agents alive at the end of the episode, pre-defined as a number of time steps. Each agent from a particular team can be killed if surrounded by at least 3 agents from the opposing team. Since this is a collaborative-competitive game, where winning or losing is the ultimate goal, no instantaneous reward is assigned to any agent in any time step. Instead, the reward follows a simple positive scalar of +1 when a team wins, and -1 when it loses. The main challenge to winning an episode is developing an effective collaboration strategy for connected agents.

Deception: In this scenario, N home team agents and 1 adversary are randomly placed in the environment, along with a few stationary landmarks, one of which is the target that both teams are trying to reach. The home agents know which landmark is the target, while the adversary does not. No attacking or killing is allowed in this scenario. At the end of an episode, the home team is rewarded +1 if both the following requirements hold: the adversary agent has not found the target landmark and there is at least one home agent occupying the target landmark, and -1 otherwise. The adversary is co-trained with the home team to achieve its maximized reward of correctly occupying the target landmark (which it does not know). The major challenge in this scenario is that home agents tend to be attracted to the target landmark, therefore, the adversary agent can track the behaviour of the home team to identify that target, resulting in a low reward for the home team.

3.2. Results. In the Jungle scenario, Q-MARL captured crucial information about the position of food cells and the surrounding situation. Some examples of the typical policies trained are depicted in Figure 5. (Note that these plots are from the perspective of one agent simply for demonstration purposes). Overall, the strategy for a team of eight agents was to fix the positions of some agents near known food cells, while others continued searching for more. As the example in the left panel of Figure 5a shows, even though the food is not present in Agent 1’s local view (the light blue

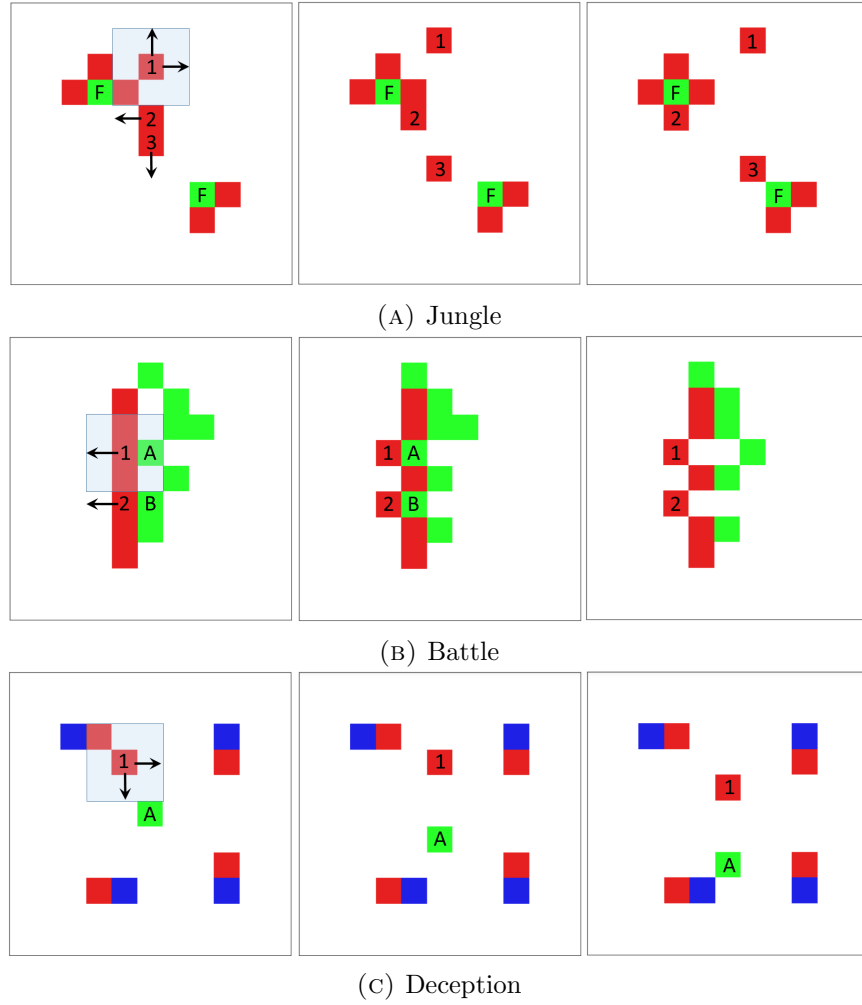


FIGURE 5. Illustration of typical trained behaviour in the three MARL scenarios, described in detail in the text of Section 3.

rectangle), the recommended actions (marked as black arrows) encourage it to move away and look for additional food cells. This behaviour comes as a result of aggregating the local views of other agents who have already partially surrounded that food. Agent 2 is encouraged to move toward the food so as to block the remaining open cell, while Agent 3 is also encouraged to continue searching in new directions. The middle and right sub-figures illustrate the next two time steps of the scenario.

In the Battle scenario (Figure 5b), the home team’s agents (red) attempt to maintain a straight line formation to avoid being surrounded and killed. Giving the agents flexibility to explore different types of attack and defence

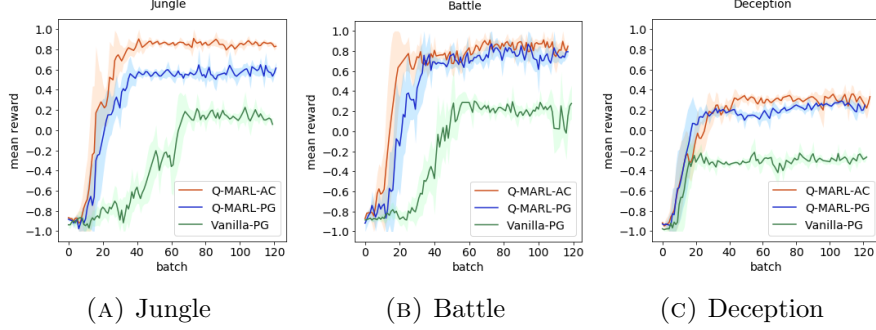


FIGURE 6. Performance comparison of the mean training reward between Vanilla-PG with Q-MARL-AC and Q-MARL-PG.

formations meant they could find a smart strategy for killing their opponents. The middle panel shows, Agents 1 and 2 moving toward the left so as to lure opposing Agents A and B into a trap and kill them. Again, this strategy is the result of considering the graph information of all neighbouring agents.

The strategy the home team developed in the Deception scenario was to try and occupy all landmarks so as to deceive the adversary (A). This behaviour is an improved version of the naive tactic that optimizes the short-term reward by ignoring non-target landmarks. Conventionally, the adversary can develop its own scavenging strategies to search for a landmark. The probability of a landmark that is not covered by home agents being a non-target is high in the naive training case. Therefore, the adversary is discouraged from occupying that landmark and continues to search for others. Since the home agents and adversary agent are trained simultaneously, any changes in the home team’s policy lead to an improved adversary policy. This dilemma of co-evolving adversarial policies is effectively resolved by Q-MARL. Interestingly, we observed a special characteristic in the Deception scenario that was not present in Jungle and Battle: the smaller disconnected local sub-graphs were prominent during training. Each sub-graph represents a cluster of agents at one landmark. As a result, by means of NMP, all agents that discovered a specific landmark by chance, regardless of whether it was a target or not, were encouraged to occupy the landmark. This improved behaviour camouflaged the target landmark, because there was nothing distinctive for the adversary to discover.

The results for the different variants of Q-MARL compared to vanilla-PG is depicted in Figure 6. In each sub-figure, the x -axis is the number of training batches, each consisting of 100 games, and the y -axis is the mean reward of the batch. In all games, there were distinct differences in performance

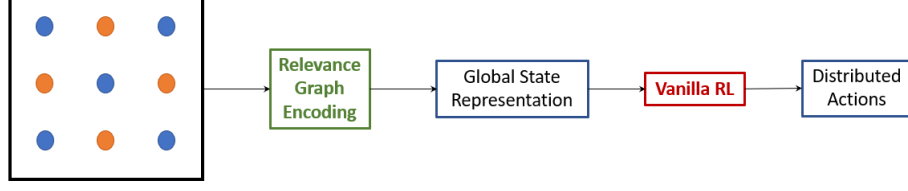
between three algorithms.

In Jungle, Q-MARL-AC won 91.16% of the games. Food cells are randomly placed in the environment for each game; hence, this scenario is able to let the algorithm stochastically improve the learning process, as the agents must develop their own strategies to search for food using only their local views, i.e., follow four walls clockwise until meeting another agent or meeting a food cell. This scenario therefore witnessed the peak performance for Q-MARL-AC (90.94%), compared to Q-MARL-PG (64.86%) and vanilla-PG (22.66%). In Battle, Q-MARL variants outperformed vanilla-PG (30.00%), although the gap between Q-MARL-PG (87.13%) and Q-MARL-AC (90.16%) was not as large as that in Jungle. Similarly, in Deception, Q-MARL-AC (35.90%) performed slightly better than Q-MARL-PG (28.91%), and both greatly surpassed vanilla-PG (−21.87%).

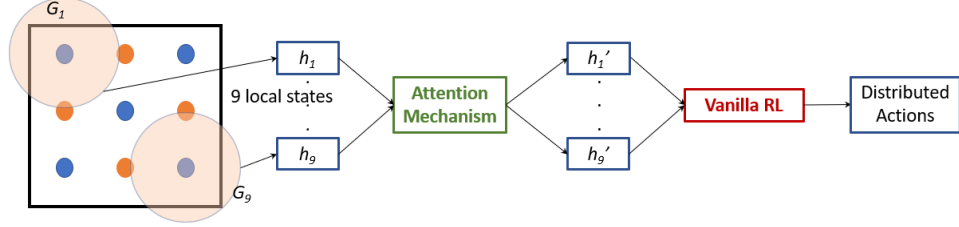
3.3. Comparison to Contemporary Methods. In this section, we compare our results to observations in the literature from two perspectives: the theoretical strengths and weaknesses of the approaches’ capability for problem solving, and the empirical performance numbers. Three recent works on graph-based RL make for suitable comparison as follows.

- **Relevance Graph Encoding (RGE)** by Malysheva *et al.* (2018) [27]: RGE produces *relevance graphs* that describes the relationship between agents and environment objects, given the relevance graph of the previous time step, and recent states/actions from all agents. At each specific time step, a new graph that represents the whole environment (with all agents) is retrieved and fed into the RL algorithm. There are two major disadvantages of this approach. First, because it uses information about all the agents in each time step, it does not scale larger environments. In fact, the largest experimental scenario includes four agents. Second, it uses previous states and actions for learning, which violates the fundamental Markov property required for RL. A high-level overview of this approach is provided in Figure 7a. The distinctive contrast between our work and Malysheva’s is that our method decomposes the full graph into sub-graphs and distributes actions to individuals as deliverables. This helps to generalise Q-MARL to larger environments and significantly reduces training burden.

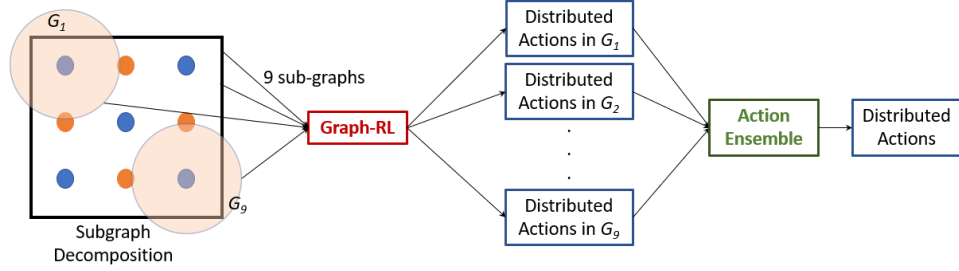
- **Single/Multi-head Attention** by Jiang *et al.* (2020, [19]), Agarwal (2019, [3]): These two works rely on communications between agents in a local neighbourhood, as does Q-MARL. However, these works require agent indexing and a full adjacency matrix for the subsequent attention mechanism. Each agent extracts its own local state (h_i) then learns its dependency with neighbouring states via a convolutional network to produce *better* individual states h'_i . The convolved states (h'_i) are then fed into a vanilla RL algorithm. Agarwal [3] proposed single-head attention. Jiang [19] subsequently



(A) Relevance Graph Embedding [27]



(B) Single-head Attention and Multi-head Attention [3, 19]



(C) Our Work

FIGURE 7. High-level summary of related state-of-the-art studies in graph-based MARL

improved on this with a multi-head mechanism that simply concatenates several attentional heads before feeding them into a non-linearity function, based on a shallow multi-layer-perceptron (MLP).

The biggest difference between all these strategies and our approach, is the principle design for how neighbourhood information is learned. Instead of extracting local representative states for individual agents and improving upon them through an attention mechanism, we feed raw features from the perspective of any given agent into the graph network. Our method therefore offers more capacity for learning and, hence, better generalisation since it can yield more subtle dependencies across neighbouring agents. Further, each output sample from the graph network is a set of actions from all agents involved in a sub-graph. As the same agent may present in multiple sub-graphs, this leaves room for a subsequent action ensembling step for more robust action decisions. High-level summary diagrams of these studies as

TABLE 1. Generalisation performance in the three scenarios Jungle, Battle, and Deception. Each method was trained with the number of agents indicated at the top of the left column. The trained policy was then applied to a larger scenario, as shown in the right column.

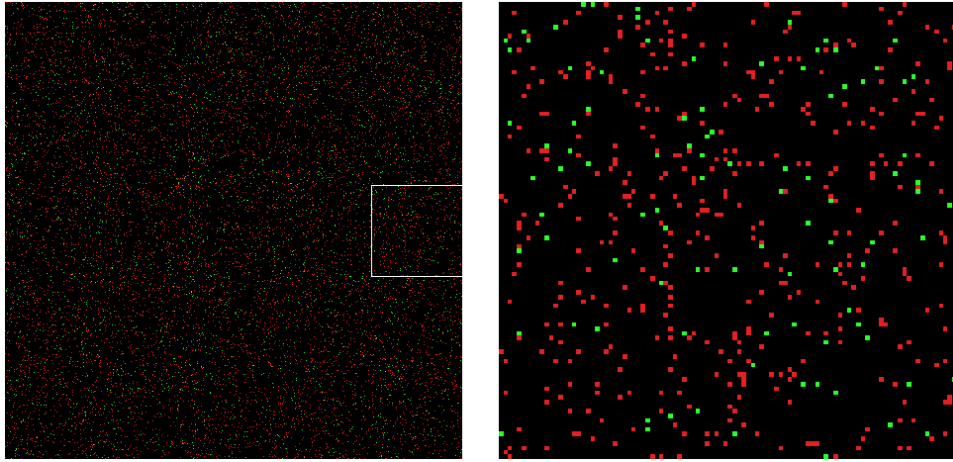
Method	Jungle		Battle		Deception	
	$N = 4$	$N = 8$	$N = 14$	$N = 28$	$N = 1$	$N = 2$
RGE [27]	0.952	0.667	0.733	0.681	0.252	0.210
	0.943	0.638	0.752	0.689	0.278	0.207
	0.951	0.614	0.710	0.694	0.246	0.222
SHA [3]	0.861	0.746	0.817	0.699	-0.029	-0.294
	0.884	0.779	0.815	0.726	-0.185	-0.284
	0.893	0.753	0.818	0.708	0.006	-0.188
MHA [19]	0.905	0.780	0.894	0.716	0.107	-0.113
	0.897	0.779	0.870	0.704	0.091	-0.091
	0.897	0.777	0.885	0.729	0.152	-0.087
Q-MARL	0.925	0.911	0.955	0.848	-0.163	-0.196
	0.930	0.917	0.960	0.863	-0.144	-0.164
	0.922	0.926	0.960	0.849	-0.127	-0.155

well as our work are provided in Figure 7.

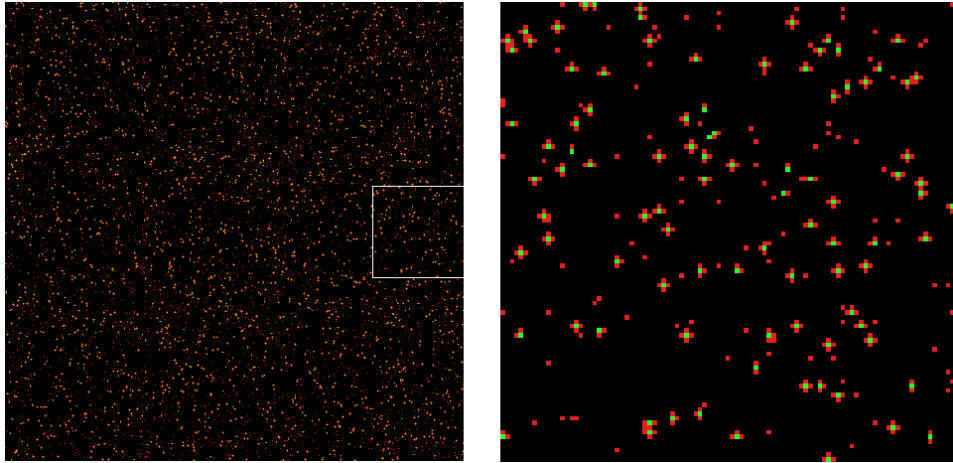
In terms of empirical performance, we assessed how well each algorithm generalised to a larger environment and the training speed. As shown in Table 1, each scenario was trained with a small number of agents, then tested with double that number. Known as *curriculum learning* [5], this is an effective method for evaluating the generalisation of an algorithm.

Our observations of the results follow:

- It is evident with all methods that performance suffered when moving from the smaller training environment to the larger testing environment.
- In the Jungle scenario, Q-MARL showed the best generalisation ability, despite RGE received the highest training rewards. The explanation is that a full graph embedding better captures the global information than a series of sub-graphs - an approach that is only feasible in small scenarios.
- In the Battle scenarios, Q-MARL amassed the most rewards in both scenario sizes.
- Q-MARL failed miserably at the Deception scenario, accumulating the least rewards of all during training and performing only marginally better than SHA in testing. We attribute its inadequacy



(A) Full initial environment (left) and the zoomed region (right) corresponding to the white square box in the left figure.



(B) Full environment after 1000 time steps performed by a trained model (left) and the zoomed region (right) corresponding to the white square box in the left figure.

FIGURE 8. An illustration of a big scale Jungle scenario with 10,000 agents (red dots) and 2500 foods (green dots).

to the underlying nature of the Deception task, where separate sub-graphs are unable to spread agents to separate landmarks.

To evaluate training speed, we tested all methods in each of the different scenarios with 10, 100, 1000, and 10000 agents. Figure 8 shows an example from the Jungle scenario and Figure 9 shows the time (in seconds) required to update 100 episodes for each setting. From the result, we find that at $N > 100$ agents, the computational burden exploded for RGE, while all graph-based approaches (SHA, MHA, Q-MARL) remained feasible. Strikingly,

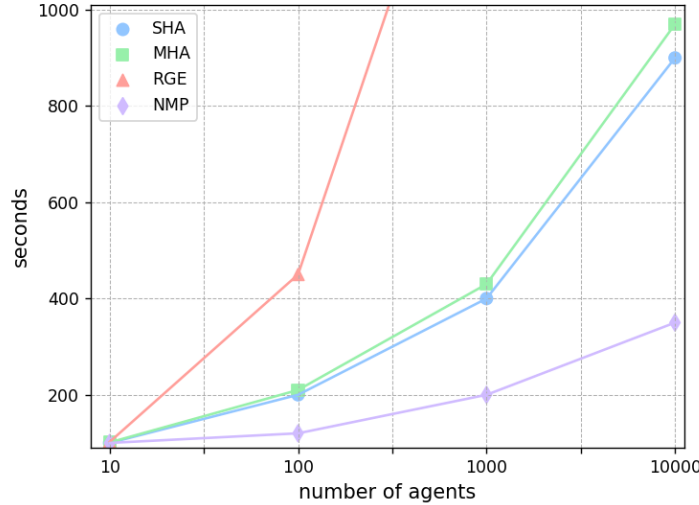


FIGURE 9. Training speed comparison. The horizontal axis represents the number of agents (10, 100, 1,000, and 10,000 agents). The vertical axis indicates the time (in seconds) required to update 100 episodes, averaged on three scenarios (Jungle, Battle, Deception).

as the number of agents increased, Q-MARL performed better and better as compared to SHA and MHA. This is because the graph decomposition procedure prunes substantially more unnecessary information about agents that are far away as the number of entities grows.

4. CONCLUSION

The graph-based approach to MARL presented in this paper addresses the long-standing issue of scalability with this paradigm. At each time step, the full environment is decomposed into multiple sub-graphs, each consisting of a limited number of agents that form a local neighbourhood. Conventional RL algorithms, such as policy gradient [46] or actor-critic [23], learn by using decomposed sub-graphs as training samples. The graph decomposition and architecture design were inspired by state-of-the-arts from quantum chemistry. Experiments with typical RL scenarios support our theoretical proof and illustrate that our graph-based approach can substantially reduce training time and training loss in terms of performance, and can also significantly better generalise when transitioning from a small curriculum scenario to a larger one. Q-MARL repository is available for download at https://github.com/cibciuts/NMP_MARL.

ACKNOWLEDGEMENT

This work was supported in part by the Australian Research Council (ARC) under discovery grant DP220100803 and DP250103612 and ITRH grant IH240100016. Research was also sponsored in part by the US Office of Naval Research Global under Cooperative Agreement Number ONRG - NICOP - N62909-19-1-2058. We would also like to thank the anonymous reviewers for their constructive comments on this work.

REFERENCES

- [1] Adam, Addison Howard, Elsie K, Harikrishna Menon, katieamazing, Myles O'Neill, Sam Harris, and snaaar. Halite by two sigma. <https://www.kaggle.com/competitions/halite>, 2020.
- [2] Adrian Agogino and Kagan Tumer. Unifying Temporal and Structural Credit Assignment Problems. In *Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems*, 2004.
- [3] Agarwal Akshat, Kumar Sumit, and Sycara Katia. Learning transferable cooperative behaviors in multi-agent teams. In *International Conference in Machine Learning (ICML), Workshop on Learning and Reasoning with Graph-Structured Data*, 2019.
- [4] G. Arslan and S. Yüksel. Decentralized q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62(4):1545–1558, April 2017.
- [5] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the International Conference in Machine Learning (ICML)*, 2009.
- [6] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.
- [7] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.
- [8] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, March 2008.
- [9] CHemistry, Mathematics in Phase Space (CHAMPS), and Kaggle. Predicting molecular properties. <https://www.kaggle.com/competitions/champs-scalar-coupling>, 2019.
- [10] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication, 2018.
- [11] Open AI Five. Openai five defeats dota 2 world champions. <https://openai.com/five>, 2018.
- [12] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16*, pages 2145–2153, USA, 2016. Curran Associates Inc.
- [13] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016.
- [14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17*, pages 1263–1272. JMLR.org, 2017.

- [15] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing.
- [16] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. 2017. <http://arxiv.org/abs/1707.09183>.
- [17] Addison Howard and Sam Harris. Halite by two sigma - playground edition. <https://www.kaggle.com/c/halite-iv-playground-edition>, 2020.
- [18] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning, 2018.
- [19] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference in Learning Representation (ICLR)*, 2020.
- [20] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS’18, pages 7265–7275, USA, 2018. Curran Associates Inc.
- [21] Woojun Kim, Myungsik Cho, and Youngchul Sung. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. *CoRR*, abs/1902.06527, 2019.
- [22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.
- [23] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, pages 1008–1014. MIT Press, 2000.
- [24] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, 2000.
- [25] Michael L. Littman. Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55 – 66, 2001.
- [26] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, pages 6382–6393, USA, 2017. Curran Associates Inc.
- [27] Aleksandra Malysheva, Tegg Taekyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman. Deep multi-agent reinforcement learning with relevance graphs. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS’18, 2018.
- [28] Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.
- [29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015.
- [30] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

- [31] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.
- [32] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. 2018.
- [33] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2681–2690. JMLR.org, 2017.
- [34] Vinyals Oriol, Babuschkin Igor, and Chung Junyoung. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. <https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii>, 2019.
- [35] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 443–451, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.
- [36] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. 2019.
- [37] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games, 2017.
- [38] Emanuele Pesce and Giovanni Montana. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109, 09 2020.
- [39] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Publishing, 3rd edition, 2016.
- [40] Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365 – 377, 2007.
- [41] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484, 2016.
- [42] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354, 2017.
- [43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
- [44] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2244–2252. Curran Associates, Inc., 2016.
- [45] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*, 2nd edition, MIT Press. 2018.

- [46] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.
- [47] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 1603–1610, Cambridge, MA, USA, 2002. MIT Press.
- [48] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5571–5580, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

INDEPENDENT RESEARCHER

Email address: `khahuras@gmail.com`

URL: `https://khavo.ai`

UNIVERSITY OF TECHNOLOGY SYDNEY (UTS)

Email address: `chin-teng.lin@uts.edu.au`