Efficient Multi-Agent Reinforcement Learning Using Clustering for Many Agents

In-Chang Baek,[1] Kyung-Joong Kim[2,*]

[1]Department of Computer Science and Engineering, Sejong University, South Korea
[2]Institute of Integrated Technology, Gwangju Institute of Science and Technology, South Korea
[1]bic4907@gmail.com, [2]kjkim@gist.ac.kr

Abstract

Recently, multi-agent research systems have been used in the field of reinforcement learning to manage with cooperative agents. Simultaneously managing a large number of agents is challenging, so various approaches are being considered. Specifically, the application of high-dimension data methodologies is a significant challenge in the research of many-agent problems, as complexity increases exponentially with the number of agents. Furthermore, policy convergence can be difficult as the contribution of each agent is unclear. In this work, we flexibly decomposed a multi-agent problem into sub-multi-agent tasks using a clustering method, and applied this technique to a hierarchical structure. After abstracting the movements of units through hierarchical approach, a group's action space and micro-control tasks were mapped onto high- and low-level actions, respectively. We demonstrated our method through combat scenarios in the StarCraft video game. Our method successfully decomposed a complex multi-agent problem into homogeneous sub-tasks, and showed the advantage of making the training process efficient and inexpensive.

Introduction

In recent years, the field of reinforcement learning (RL) has developed remarkably, with applications being found in video games and real-world problems. Dealing with a single agent is the most important task in RL; and the primary challenge is in finding a balance between exploration and exploitation. Recent studies have focused on solving complex video game problems that are transferable to the real-world. In particular, long duration scenarios with unclear, delayed rewards (Kulkarni et al., 2016; Kaelbling, 1996; Bellemare et al 2016) are unavoidable problems for researchers, and the capacity to manage problems with high dimensionality is important. These problems persist, and continued research aiming to reduce dimensionality is critical if data-efficiency is to be increased.

Dealing with multiple agents has introduced a new challenge as the convergence of multi-agent problems in traditional scenarios is unreliable, which makes training difficult. Non-stationary concurrency problems occur when multiple agents are executed simultaneously (Hernandez-Leal et al., 2017). Previous multi-agent studies have attempted to transplant existing RL methods into a centralized paradigm (Tan, 1993), and this approach has improved several multi-agent algorithms by increasing the collaboration between agents (Foerster et al., 2016).

As multi-agent methods have been applied to real-world applications, researchers have shown an increased interest in addressing problems with many agents. Most related works focus on the multi-agent credit assignment problem, which is concerned with reward distribution and devising a scalable architecture that is less-affected by the number of agents (Nguyen et al., 2018; Chang, Ho, and Kaelbling, 2004). In this problem, the joint-action set increases with the number of agents, and the learning complexity increases exponentially when actions are combined.

A hierarchical framework was used to solve high-dimension multi-agent system problems. Several studies have reported that action- and state-space abstraction are good methods for reducing learning complexity, which increases learning efficiency. Through this concept, we demonstrate that more flexible decomposition methods are required in sensitive applications, such as autonomous vehicles (Shalev-Shwartz et al., 2016), and traffic routing problems (Ye et al., 2015; Wiering et al., 2000). This was implemented by substituting a grid-like method with a clustering algorithm for dynamic grouping. Following the work of (Stanescu and Buro, 2018), the agents' action probabilities were spatially correlated which increased the elegance of macro-actions.

In the next section, we present a review of studies that have inspired our method. We then describe our proposed technique, which expands upon the hierarchical RL

[*]Corresponding Author

framework meta-controller (Kulkarni et al., 2016). Experimentally, this new system is demonstrated in three StarCraft combat scenarios (i.e., a representative multi-agent testbed was focused on cooperative formation), with a focus on many-agent cooperative formation, and we assess the efficiency of our method compared to the vanilla multi-agent RL with qualitative discussion of each scenario.

Related Work

Exploration of high-dimension state spaces has been an important issue in deep-RL literature. Many studies examine effective exploration in complex video games with delayed and sparse reward signals. However, in multi-agent settings, training numerous agents is a challenging problem that produces high-dimension state spaces and many joint-action combinations. Much of the literature addresses high-dimension learning in single-agent RL fields where learning components are decomposed with some examples. However, decomposing a multi-agent problem necessitates consideration of agent-property homogeneity. In this section, we will examine the complexity of multi-agent RL when many agents are computed, and discuss studies that have attempted to decrease the learning dimensionality of multi-agent systems.

Centralized Learning

Training cooperative agents is an ongoing challenge in multi-agent reinforcement learning (MARL). Historically, tabular RL methods concentrated on single-agent frameworks; however, these methods are insufficient for collaborative tasks, as illustrated by independently executed agents that follow the Dec-POMDP framework (Oliehoek and Amato, 2016). In response, Tan presented Independent Q-Learning (IQL) that centralizes independent Q-functions for each agent, and shows that decentralized learning causes non-stationary problems owing to unstable convergence in multi-agent settings (Tan, 1993). The centralized learning concept formed the basis of a paradigm in multi-agent literature where a centralized Q-function, Q_{tot} was trained. This strategy is useful when solving non-stationary problems as partial observations of each agent are included during training.

The problem of multi-agent credit-assignment modulates the contribution of each agent considered in centralized learning. The significance of this effect increases with the number of joint-action sets, which complicates learning-processes. This problem is well-studied in the multi-agent field.

Foerster et al. proposed COMA, a counterfactual based actor-critic method, to estimate fictitious individual rewards distributed from a team-reward signal (Foerster et al., 2018; Lowe et al., 2017). More elegantly, value decomposition network (VDN) represented optimal individual value function through back-propagation of the centralized Q-function (Sunehag et al., 2018; Rashid et al., 2018). These methods are well-suited to solving the credit-assignment problem, as determining individual rewards through *reward-shaping* is inefficient and unrealistic in complex video games.

Neural network structures have been designed to improve communication between agents operating with *decentralized execution*. CommNet (Sukhbaatar et al., 2016) reported that a communication-specialized model, particularly one using recurrent neural networks (RNNs) (Hochreiter et al., 1997; Cho et al., 2015), was well-suited to multi-agent problems. Furthermore, BiCNet (Peng et al., 2017) used bi-directional long short-term memory (LSTM) as a communication layer to exchange vectorized messages between units, and showed that it is scalable to an arbitrary number of agents. The aforementioned techniques are appropriate for a large number of agents; however, most multi-agent research focuses on the precise control of a small number of agents, so new techniques are required to move a large number of agents effectively.

Temporal Abstraction

When controlling many agents, large-scale maps increase dimensionality of exploration such that agent distribution can cause combinatorial explosion of the state-dimension, and increasing the joint-action set further complicates scalability.

Hierarchical reinforcement learning (HRL) (Kulkarni et al., 2016; Pang et al., 2018; Ghavamzadeh et al., 2006) is a method for efficient RL exploration, which decomposes a training session into several steps. In particular, this technique is best applied to long-trajectory problems (e.g., a broad range of path planning problems). In multi-level HRL, high- and low-level policies teach abstracted actions for global- and local-range actions, respectively. The abstracting process is focused on the modular learning component, and unnecessary details are eliminated.

In the MARL domain, *temporal abstraction* is useful for the formation of agents, and high-level decisions concentrate on their movement actions. Tang et al. utilized this structure in a three-player basketball game; players were located with a high-level multi-agent policy, and replacing independent actions with single-agent policy (Tang et al., 2018). This demonstrated that the low-level tasks executed by agents were homogeneous and, therefore, could be replaced with a single policy. To scale this problem for many agents, a large action space must be abstracted, and homogeneity of grouped agents assumed.

Spatial Decomposition

The abstraction of minor tasks for many-agent problems leads to improved performance and efficiency. Stanescu and Buro presented a concept for the decomposition of multi-agent systems (Stanescu and Buro, 2018) that was demonstrated in a many-agent environment (Zheng et al., 2017) and focused on the cooperative formation of agents. In their work, the action space was decomposed into multi-level

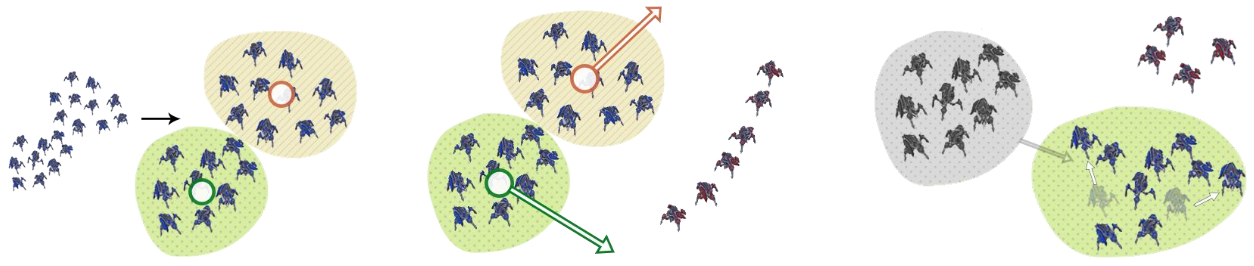
(a) Clustering of units (b) High-level action (c) Low-level action

Figure 1: An implementation of flexible decomposition. (a) K-Means groups the ally units and returns the center coordinates of each cluster. (b) A multi-agent algorithm determines the long-range group-actions. and (c) each unit moves short-distances with any micro-control method.

layers using a HRL architecture. Specifically, a rectangular map was divided into gridded sectors. This approach was successful due to the homogeneity nearby sectors, and this concept is applicable in most multi-agent environments.

As discussed, grid-based methods are better-suited to monotonic environments than complex ones (such as StarCraft). In this work, we propose a method for controlling many agents in more elegant and flexible fashion by dividing the action-space, with consideration given to the current formation, and with the goal of maximizing expected future rewards.

Proposed Method

Flexible Action Decomposition Using Clustering

In this section, we propose a non-linear decomposition method that is suitable for high-dimension state and action spaces. More delicate decompositions were processed by a method that could respond to the dynamic formation of agents, and was expanded to various attributes before MARL was applied. In machine-learning, spatially scattered data are organized into sensible groups by clustering. As discussed, this approach is also well-suited to spatial correlation. Here, we used the well-known K-means algorithm for the decomposition (Jain, 2010), which is capable of spatial correlation, as its clustering methodology is based on the distance between data. This algorithm has been used on large amounts of data, and can be used to process very large real-world multi-agent tasks. Similarly, Justesen et al. applied K-means to UCTCD (a family of the Monte Carlo tree-search) for micro-control management in the StarCraft combat simulator, and reported that it effectively decreased the expansion of MCTS (Justesen et al., 2014). We also expect to decrease unnecessary exploration to increase model learning speed.

Abstraction of Large Action Space

To reduce the dimensionality of our approach, we considered temporal abstraction of the HRL framework. This approach results in a more stable convergence of large-scale problems, but reduces the detail of learning components. Typically, in the counterbalance between efficiency and detail, the former is preferable in many-agent problems.

Following this method, a large multi-agent task we decomposed into several homogeneous multi-agent tasks. According to Peng et al., a *zero-sum stochastic game* (SG) between N agents occupying action space A, and M opponents of occupying action space B, in a state space S, with transition function τ , can be described as a tuple $(S, \{A_i\}_{i=1}^{N}, \{B_i\}_{i=1}^{M}, \tau, \{R_i\}_{i=1}^{N+M})$ and we assume learning complexity $\mathbb{C} = S \times A^N \times B^M$ as a *reward function* of SG (Peng et al. 2017). From this expression, it is easily verified that learning complexity $\mathbb{C}$ increases with the number of units. Thus, an increase in the number of agents results in an increase in complexity. The decomposition of A^N to reduce complexity will be discussed in the next phase.

$$\mathbb{C}_{dec} = S \times ({A_{grp}}^{k} \times {A_{ind}}^{N/k}) \times B^M \tag{1}$$

According to Equation 1, when the enemy's action space B and the number of enemies M are fixed, A^N can be decomposed into ${A_{grp}}^{k}$ and ${A_{ind}}^{N/k}$. The abstracted action space A_{grp} is applied to a group as a high-level action that represents global-range movement, and A_{ind} is a low-level action that is executed by individual units within each group as a local-range action. In cooperative combat scenarios, groups within clusters are homogeneous as they have a similar number of agents, and the shared goal of winning the game. Therefore, a single model must be trained for micro-control, making this an inexpensive process.

Learning Architecture

Utilising the homogeneity of these groups, we propose a hierarchical architecture for MARL. We have assigned the many-agent and micro-control problems to the high- and lower-layer respectively. In this architecture, the high-level policy selects a lower-level policy that is expected to produce a greater sum of rewards. To overcome the problem of delayed-rewards, we designed the high-level policy to have a longer duration than the low-level policy (Tang et al. 2018; Kulkarni et al., 2016). In this paper, we focus on the high-level execution of the HRL structure, and other micro-control tasks can be found in the referenced literature (Usunier et al., 2016; Rashid et al., 2019). Our implementation for the StarCraft video game is summarized in the following sequence. This procedure is performed for each S_t

1) *Clustering*: K-means groups the alive units according to their two-dimensional (2-D) position, and returns the group-labelled units and cluster centers. Then local observations are generated to identify the members of each group (a technique commonly used in the Dec-POMDP framework).
2) *Inference*: a multi-agent algorithm is executed with local observations (scatter graph), and some additional centralized global-state data (health and position of all units) to enable cooperative action. 2-D vectors are required for movement in StarCraft.
3) *Execution:* the cluster centers and 2-D vectors from previous steps are summed with a constant coefficient that has been optimized for global-movement. The target position of each group persists for a *frame-skip* duration, and the units move close to the target individually for each frame.

This solution is motivated by the techniques employed by human players when controlling large numbers of units in RTS games. It is essential that units are grouped for them to be controlled in a limited timeframe. When compared to other multi-agent systems, this approach of treating spatially related units as a “troop” is an inexpensive model.

Experiments

StarCraft Multi-Agent Task

Real-time strategy (RTS) games have characteristics that can be transferred to real-world environments because of their real-time nature, concurrent party actions, and numerous strategies. StarCraft (a typical RTS game) is often used by machine learning researchers as a testbed. The development company (Blizzard) provides the application programming interface (API) that enables the StarCraft II video game to be used for this purpose (Vinyals et al., 2017; Samvelyan et al., 2019), so that various scenarios can be tested. Micro-control tasks are investigated by many researchers, as there is a simultaneous goal of cooperation and competition (Churchill et al., 2016). Additionally, the game has a high-dimensional state and action space according to units’ type and quantity, so it is well-suited to experiments regarding the balance of exploration and exploitation in the context of RL.

Micro-control tasks were expanded for our purpose by adding more units to increase the dimensionality of the learning process. An increase in unit numbers maintains the credit-assignment problem, and increases the delay in the return of a reward signal. Here, we designed a fully-observable (i.e., except ‘fog of war’) multi-agent environment as we are focused on high-level agent actions.

Experiment Setup

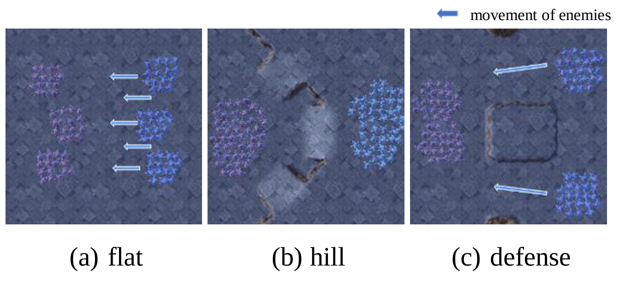

Figure 2: Images of the three scenario maps

Three scenarios were prepared to test our proposal. Through testing a simple multi-agent algorithm, we designed three maps to estimate the cooperation of agents. Figure 2 shows the three scenarios created as StarCraft game maps.

(a) Flat has the enemies vertically surround the allies on a flat map. The game result for this map is impacted by establishing an early formation (e.g., a crane-wing formation) before combat commences. This is a more complex problem than other maps as there are no obstacles.

(b) Hill is designed to test solutions to simple cooperative problems. A limited number of units can pass the narrow hill at one time. For this reason, the groups can divide into three directions by mutual agreement so that the enemy is surrounded, and the units are victorious. Therefore, group communication has an influence on this game result.

(c) Defense is a scenario that presents a randomly chosen number of enemies approaching in two directions, with several quantities considered. The allies’ goal is to place two appropriately sized groups that take into account the number of enemies incoming in each direction.

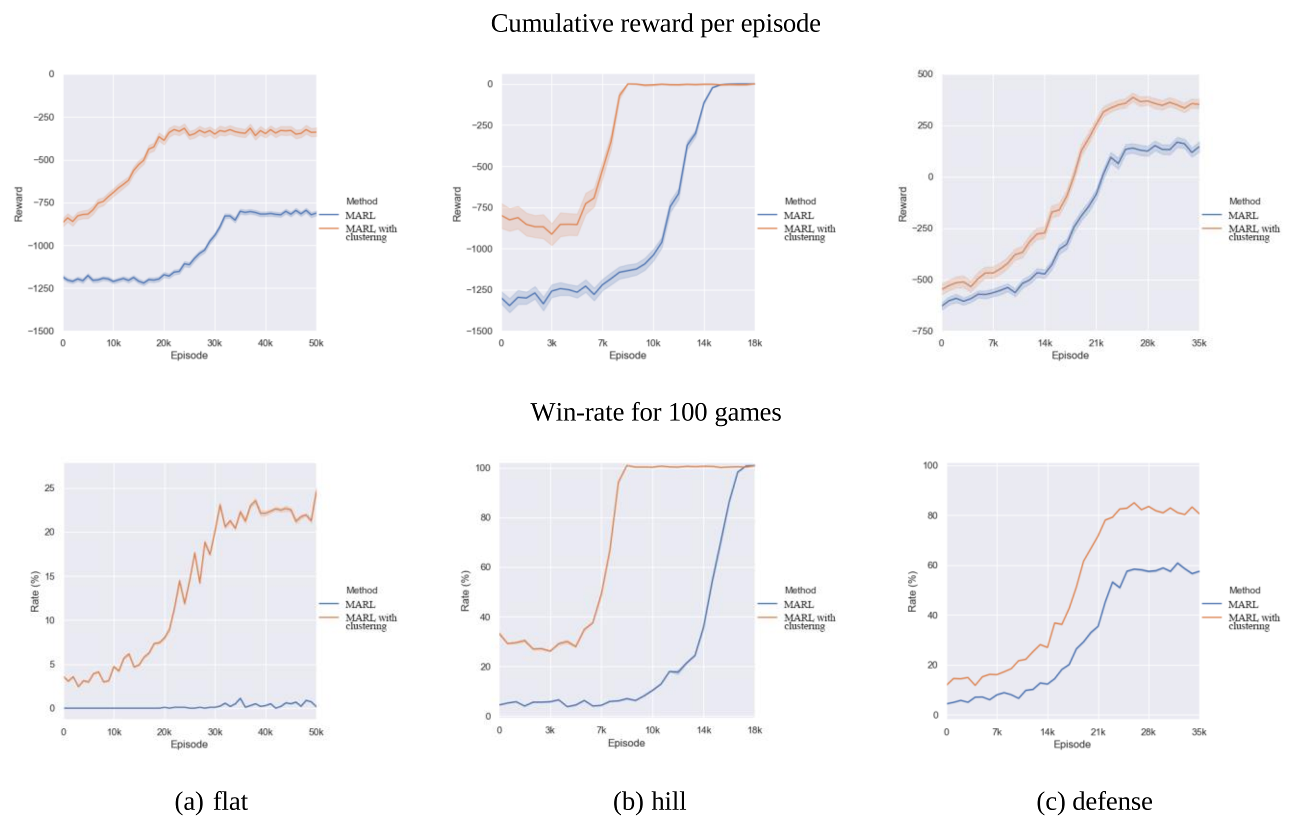

Figure 3: The reward and winning rate results for three scenarios.
(Shaded regions show the error rate when the slide window was set to 1000.)

Local-Obs (Image) 84 × 84 × k	Global-Obs (Image) 84 × 84 × 2
Concatenation each Local-Obs(1), Global-Obs(2)	
Conv(4, stride-1)-16 + AvgPool(2)	
Conv(3, stride-1)-16 + AvgPool(2)	
Conv(3, stride-1) + BN	
Dense(600) + BN	
Dense(300)	
Bi-LSTM(300, length-k)	
Dense(300) + BN	
Dense(150) + BN	
Dense(150) + BN	
Dense(4) + Tanh	

Table 1: Model Architecture. BN: Batch-norm, Conv(4, stride-1)-16: **4 × 4** convolution, 16 channels and stride 1. ReLU as an activation function, ***k***: The number of units that are alive.

In the three scenarios above, the target was for agents to select the appropriate formation for each situation. Teams of non-skilled 45 Stalkers (a.k.a Dragoons in StarCraft I) were chosen for each experiment. These units were chosen because their long *weapon-cooldown* time allows formations to be changed during combat.

Results

The three scenarios were each tested twice, once our proposed method and once assuming that each unit is an independent agent. Convergence took approximately seven hours in a distributed environment using 8 CPUs. The performance on each map is shown in Figure 3.

Experimental Result

In Figure 3, the cumulative reward per episode used in RL is plotted along with the percentage win-rate of the previous 100 games. As the reward increases, our agent is minimizing the enemy attack, and maximizing the ally attack. At the start of the training period, differences between the two methods mean that our technique succeeded in reducing the dimension, because the movement of the grouped unit produced a reward variation.

The **(b) hill** environment was easier to train than others, because of the relatively small state dimension. Our agents focus on the location of the aisle and timing of the attacks, rather than how many agents could pass through the aisle. The figures show that our process converged twice as fast as the conventional method in this scenario.

The state dimension of the **(a) flat** scenario was large because there were no obstacles, making this the most complex scenario to learn, and also demonstrating the efficiency of our method. Training using the individual unit method converged slowly due to unnecessary exploration. Our dimensional decomposition solves this problem because our agents focus on group formation, rather than how individual units can be assembled. At the start of the scenario, the enemy units form a crane wing so our agent took several training episodes to start to win. Once the training process converged on a solution, our agent dynamically created a form that was responsive to the enemies' formation.

The **(c) defense** scenario created a situation that could not have been predicted. A group remained on standby in the middle of the fork, and moved to assist when one formation showed the potential of losing. Although the distinction of agent units is less precise, this approach eases the training of multi-agent communication (i.e., we reduced the length of the RNN communication layers).

Discussion

In comparison to our approach, the non-combined method took more episodes to converge. The **(a) flat** scenario had a greater dimensionality that others, and was solved more efficiently by our abstraction method because our agents focus on large-scale multi-agent problems such as formation, and the grouping of agents reduces the scope of the exploration required. Stanescu and Buro reported that the performance of abstraction improves as the number of agents increases (Stanescu and Buro, 2018). Similarly, the number of groups increases nonlinearly for our method, and it is efficient in many-agent situations.

Through the experimental process we observed that dealing with a large number of agents requires an increased training resource. For example, the independent-agent experiments required significantly more memory to save their trajectories. Additionally, the training time for our method was approximately half that of the individual-agent model. Our approach has the further advantage that using fewer agents reduces the demand for computing resources. This result demonstrates that researchers now have access to environments that use many agents.

Conclusion and Future Work

Applying RL to video games with complex dimensions is not trivial. Also, various multi-agent issues arise as the number of agents increases. In this work, we have taken a hierarchical approach to simplifying the learning complexity to solve StarCraft combat simulation problems. We focused on the meta-controls at the top level of the hierarchy architecture. Large problems were divided into homogeneous tasks, and the problems were solved easily by a multi-agent system. This approach reduces the complexity of MARL, reduces the cost of many-agent problems, and can be used without any clustering or multi-agent algorithms.

In this study, units were grouped using basic K-means. However, if the unit type was varied, various strategies are required depending on the unit composition. It would be necessary to consider a more rigorous clustering process to solve these more complex problems. This may require the provision of other information (e.g., health, influence) to the clustering algorithm, or the implementation of a trainable clustering method that can be combined with RL.

Acknowledgements

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (grant no. 2017R1A2B4002164).

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2019.

References

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *In Advances in Neural Information Processing Systems* (pp. 1471-1479).

Chang, Y. H., Ho, T., & Kaelbling, L. P. (2004). All learning is local: Multi-agent learning in global reward games. *Advances in Neural Information Processing Systems* (pp. 807-814).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated feedback recurrent neural networks. *International Conference on Machine Learning* (pp. 2067–2075).

Churchill, D., Preuss, M., Richoux, F., Synnaeve, G., Uriarte, A., Ontañnón, S., & Čertický, M. (2016). StarCraft bots and competitions. *Encyclopedia of Computer Graphics and Games*, (pp. 1-18).

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. *32nd AAAI Conference on Artificial Intelligence*.

Ghavamzadeh, M., Mahadevan, S., & Makar, R. (2006). Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13 (2), (pp. 197–229).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9 (8), (pp. 1735–1780).

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31 (8), (pp. 651–666).

Justesen, N., Tillman, B., Togelius, J., & Risi, S. (2014). Script- and cluster-based UCT for StarCraft. *2014 IEEE Conference on Computational Intelligence and Games*, (pp. 1–8).

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, (pp. 237–285).

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems* (pp. 3675–3683).

Nguyen, D. T., Kumar, A., & Lau, H. C. (2018). Credit assignment for collective multiagent RL with global rewards. *Advances in Neural Information Processing Systems*, (pp. 8102–8113).

Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized POMDPs* (Vol. 1). Springer International Publishing.

Pang, Z. J., Liu, R. Z., Meng, Z. Y., Zhang, Y., Yu, Y., & Lu, T. (2018). On reinforcement learning for full-length game of starcraft. *arXiv preprint* arXiv:1809.09095.

Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint* arXiv:1703.10069.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint* arXiv:1803.11485.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., ... & Whiteson, S. (2019). The StarCraft multi-agent challenge. *arXiv preprint* arXiv:1902.04043.

Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint* arXiv:1610.03295.

Stanescu, M., & Buro, M. (2018). Spatial Action Decomposition Learning Applied to RTS Combat Games. *In AIIDE Workshops*.
Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., ... & Graepel, T. (2018, July). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 2085-2087). International Foundation for Autonomous Agents and Multiagent Systems

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proceedings of the 10th International Conference on Machine Learning* (pp. 330–337).

Tang, H., Hao, J., Lv, T., Chen, Y., Zhang, Z., Jia, H., ... & Wang, L. (2018). Hierarchical deep multi-agent reinforcement learning. *arXiv preprint* arXiv:1809.09332.

Tang, H., Hao, J., Lv, T., Chen, Y., Zhang, Z., Jia, H., ... & Wang, L. (2018). Hierarchical deep multiagent reinforcement learning. *arXiv preprint* arXiv:1809.09332.\

Usunier, N., Synnaeve, G., Lin, Z., & Chintala, S. (2016). Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint* arXiv:1609.02993.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... & Quan, J. (2017). Starcraft II: A new challenge for reinforcement learning. *arXiv preprint* arXiv:1708.04782.

Wiering, M. A. (2000). Multi-agent reinforcement learning for traffic light control. *Machine Learning: Proceedings of the 17th International Conference* (ICML'2000) (pp. 1151–1158).

Ye, D., Zhang, M., & Yang, Y. (2015). A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, 15 (5), (pp. 10026–10047).

Zheng, L., Yang, J., Cai, H., Zhou, M., Zhang, W., Wang, J., & Yu, Y. (2018, April). MAgent: A many-agent reinforcement learning platform for artificial collective intelligence. *In 32nd AAAI Conference on Artificial Intelligence*.