Distributed Methods for Reinforcement Learning Survey

Johannes Czech

Abstract Distributed methods have become an important tool to address the issue of high computational requirements for reinforcement learning. With this survey, we present several distributed methods including multi-agent schemes, synchronous and asynchronous parallel systems, as well as population-based approaches. We introduce the general principle and problem formulation, and discuss the historical development of distributed methods. We also analyze technical challenges, such as process communication and memory requirements, and give an overview of different application areas.

Keywords Distributed methods · Multi-agent systems · Parallelism · Asynchronism · Population-based reinforcement learning

1 Introduction

The increasing amount of data and availability in computing power increases the demand for *Distributed Methods* (DM), particularly because the general setting of *Reinforcement Learning* (RL) is naturally suited for parallelism and multi-agent systems [5]. From a historical perspective, asynchronous methods have appeared very early in research. One of the first use cases of DM for RL was in the context of Dynamic Programming in which a general class of asynchronous iterative methods with convergence guarantees was introduced [2]. Later, this technique was implemented and extended with the *Prioritized Sweeping* algorithm [27] which stores previous experiences to focus the computational effort on important dynamic programming sweeps. Parallel methods for policy iteration have been later investigated by Bertsekas and Yu [3] who used a network of processors to asynchronously update a local policy and cost function, defined on a portion of the state. Next these

J. Czech (✉)
Technische Universität Darmstadt, Darmstadt, Germany
e-mail: johannes.czech@cs.tu-darmstadt.de

B. Belousov et al. (eds.), *Reinforcement Learning Algorithms: Analysis and Applications*, Studies in Computational Intelligence 883,
https://doi.org/10.1007/978-3-030-41188-6_13

computed values were communicated asynchronously between the processors in order to perform the local policy and cost updates.

One major success for supervised learning was achieved in the field of computer vision. The respective work [11] demonstrates the possibility to accurately learn the Imagenet [7] data set in one hour using the convolutional network architecture ResNet-50 [14] with a mini-batch size of 8192 and 256 GPUs. This implies that with a scaling efficiency of 90 %, neural networks which are the core part of many RL algorithms excel at massively parallel computation, while others techniques such as Gaussian Networks fail to scale to high-dimensional state and action spaces [13].

2 Notation of Multi-agent Reinforcement Learning

The extension of *Markov Decision Processes* (MDPs) to multiple agents can be formalized as a *Decentralized Partially Observable Markov Decision Process* (DecPOMDP) or *stochastic game*. It is defined by a tuple $\langle I, S, \{A_i\}, P, R, \{\Omega_i\}, O, h\rangle$ where I is a finite set of agents, S a finite set of states, $A = A_1 \times \cdots \times A_n$ a finite joint action set, $P : S \times A \times S \rightarrow [0, 1]$ a state transition probability function, $R : S \times A \rightarrow \mathbb{R}$ a reward function, $\Omega = \Omega_1 \times \ldots \Omega_n$ the finite joint observation set, $O : \Omega \times A \times S \rightarrow [0, 1]$ the observation probability function and h the maximum number of steps in the environment. Similar to a MDP, each agent i tries to maximize the total cumulative shared reward by following the optimal value joint policy. Whereas the joint value V^{π} is stated as

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{t=0}^{h-1} \gamma^t R(\mathbf{a}^t, s^t)\Big| s, \pi\Big]. \tag{1}$$

The action chosen by each agent can be different and depends on their local observations, but they share a single global reward in a common environment (Fig. 1). Furthermore, it is often useful for the agents to store past observations since the state is only partially observed. Several variants of DecPOMDP exist, where communication between agents is impossible, restricted or associated with a cost. In the case

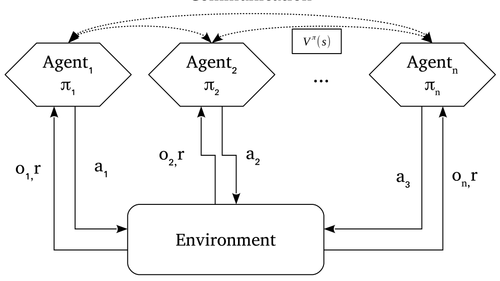

Fig. 1 Scheme of multiple collaborative agents in a common environment. Each agent is governed by its own policy, but they all share the same knowledge about the environment in the form of a joint value function

for instantaneous lossless communication, a DecPOMDP is equivalent to a POMDP. In contrast to MDPs, the exact solution of a DecPOMDP is known to be intractable and *Non-deterministic EXPonential* (NEXP) complete even for only two agents [1].

3 Taxononomy of Distributed Reinforcement Learning

Value based methods (Table 1) rely on either explicitly computing or approximating Q-Values $Q(s, a)$ which express the expected return for a state-action pair. During learning and exploration the Q-Values are updated until convergence as follows

$$Q_t(s, a) := Q_t(s, a) + \alpha[(r_t + \gamma \ \max_{a'} Q_t(s', a')) - Q_t(s, a)], \tag{2}$$

where $\alpha \in [0, 1)$ is the learning rate. In cooperative multi-agent systems, Q-learning has been introduced by Boyan and Littman [4] and Lauer and Riedmiller [21] who considered an ensemble of independently and simultaneously acting agents in a model-free distributed Q-Learning algorithm to maximize the shared discounted sum of rewards without communication between the agents.

By contrast, policy gradient methods update the policy parameters θ by gradient ascent directly, without any intermediate state-action value representation. REINFORCE [35] is the first algorithm of this category. The policy parameters θ are updated with the help of Monte-Carlo trajectories over a full episode

$$\theta_{t+1} = \theta_t + \alpha \ G_t \frac{\nabla_{\theta_t} \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t)}, \tag{3}$$

where G_t, $a_t \sim \pi$ and $\alpha \in [0, 1)$ denote the return, the action sampled from the current policy and the learning rate, respectively. Despite their simplicity, policy gradient methods are known to suffer from high variance. Exemplary algorithms which have been successfully transferred to the distributed domain and that can either be used as a policy gradient method or as the actor part of the Actor-Critic framework include *Deep Deterministic Policy Gradient* (DDPG) [23] and *Proximal Policy Optimization* (PPO) [15, 30].

Actor-Critic (AC) methods combine the ideas of value and policy gradient methods. The critic provides a value $V(s)$ or Q-value $Q(s, a)$, whereas the actor represents the policy $\pi(a|s, \theta)$ and updates its parameters θ based on the feedback of the critic. Moreover, the AC architecture can be represented as a single model with a shared body and two separate heads or two distinct models. The *Asynchronous Advantage Actor-Critic* (A3C) [26] algorithm outlines the first AC method of the parallel, asynchronous RL class.

Table 1 Overview of distributed methods for reinforcement learning. Projects are highlighted in *italic*

	Multi-agent	Parallel		Population based
		Synchronous	Asynchronous	
Optimal Control			Parallel Distributed Numerical Methods (1989) Prioritized Sweeping (1993) Distributed Policy Iteration (2010)	
Value-based Methods	Distributed Q-Learning (2000) Cooperative Q-Learning (1993) *Network Routing* (1993) *Patrolling* (2004)	Ape-X: DQN (2018) R2D2 (2019)	GORILA (2015) 1-step/n-step Q-Learning (2016) 1-step SARSA (2016)	
Policy gradient		DPPO (2017) *OpenAI Five* (2018)		
Actor-Critc Methods	ACCNet (2017) MADDPG (2017) COMA (2018)	A2C (2016) *AlphaZero* (2017) Ape-X: DDPG (2018)	A3C (2016) UNREAL (2017) *Door opening* (2017) *3D Manipulation* (2017) IMPALA (2018)	FTW (2018) *AlphaStar* (2018)

3.1 Multi-agents

The field of *Multi-agent Reinforcement Learning* (MARL) can be divided into different categories. In the first, the agents $i \in I$ work together as a cooperative team, each taking individual action $a_{t,i} \in A$, trying to maximize a common long-term team reward R in a globally shared environment state $s_t \in S$ [13]. In the second, the agents act in a competitive fashion, formalized as a zero-sum game which leads to a minimax equilibrium [32]. In the third, a mix of the above can be used which is described as a general-sum game converging to a Nash equilibrium [10].

As discussed in Sect. 2, multi-agent systems come with additional challenges. Besides the mulit-agent credit assignment problem, value-based techniques are faced with an inherent non-stationary environment, whereas policy gradient methods suffer from an increasing variance with respect to the number of agents.

Several approaches have been proposed to address these issues in AC architectures. For example, Lowe et al. [24] presented an adaptation of the AC-framework which takes action policies of other agents into account, enabling it to successfully learn policies which require complex multi-agent coordination. *Actor-Coordinator-Critic Net* (ACCNet) [25] and *Counterfactual Multi-agent* (COMA) [10] both introduced a centralised critic to estimate the Q-Values, and decentralised actors to optimize the policies of the agent. In order to negate the multi-agent credit assignment problem, a counterfactual baseline marginalises out the action of an agent, while keeping actions of the other agents fixed.

3.2 Parallel Methods

Parallel synchronous methods aim to solve large-scale MDPs with multiple computation units e. g. workers resulting in larger computational power and higher data efficiency. In the simplest case, it can be seen as a scaled up version of a single agent, where each worker learns a common policy on a different instance of an identical MDP, yielding training time reduction and in some cases training stabilization due to larger batch-sizes. Workers in synchronous DMs update their policy in a coordinated fashion at predefined time steps or events, as shown in Fig. 2.

Asynchronous parallel methods (Fig. 3) follow the same general principle, but in contrast exhibit different attributes because asynchronous parameter updates lead to higher sampling quantity and diversity. While first studied by Tsitsiklis [33], they have been later reintroduced in the *Deep Reinforcement Learning* (DRL) setting [26]. Based on empirical experiments, a super-linearity phenomena for training was observed in the original work (Table 2).

However, after studying the system on different environments, the non-stationary, off-policy samples led to problems and in some cases even to non-convergence or divergence [36]. For this reason, a synchronous *Advantage Actor-Critc* (A2C) version was developed which outperforms A3C in most benchmarks.

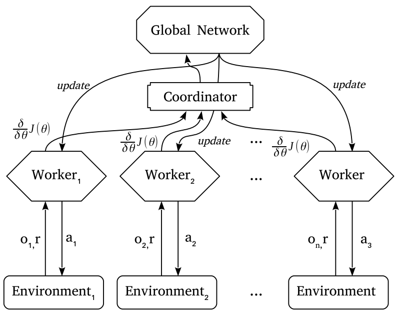

Fig. 2 Parallel synchronous scheme. Workers update their policy in a coordinated fashion at predefined time steps or events

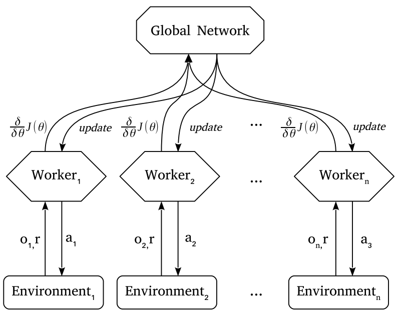

Fig. 3 Parallel asynchronous scheme. Each worker update its policy regardless of the other workers

Table 2 Superlinear training speed-up of parallel asynchronous methods on seven Atari games (Beamrider, Breakout, Enduro, Pong, Q*bert, Seaquest, and Space Invaders). Table from [26]

	Number of threads				
Method	1	2	4	8	16
1-step Q	1.0	**3.0**	**6.3**	**13.3**	**24.1**
1-step SARSA	1.0	**2.8**	**5.9**	**13.1**	**22.1**
n-step Q	1.0	**2.7**	**5.9**	**10.7**	**17.2**
A3C	1.0	2.1	3.7	6.9	12.5

In addition, Jaderberg et al. proposed the *UNsupervised REinforcement and Auxiliary Learning* (UNREAL) framework [19] which uses unsupervised auxiliary tasks to improve the agent in the absence of extrinsic rewards.

3.3 Population-Based

Population-based distributed methods (PBDM) are genetically inspired methods in which agents are divided into multiple groups. The policy of each group is updated independently (Fig. 4), and after a predefined number of iteration steps different populations are compared with each other in a competitive, evolutionary fashion or league system. The *For The Win* (FTW) agent architecture [18] marked the first recent PBDM for DRL and was successfully applied to a first-person-shooter multiplayer game. The same architecture was later adapted to the real-time strategy game Starcraft II [34], with a combination of supervised and self-imitation learning, experience replay as well as policy distillation.

Overall, PBDM appears to be highly valuable in long-term horizon settings, in which a diverse set of mutually exclusive behaviours should be explored.

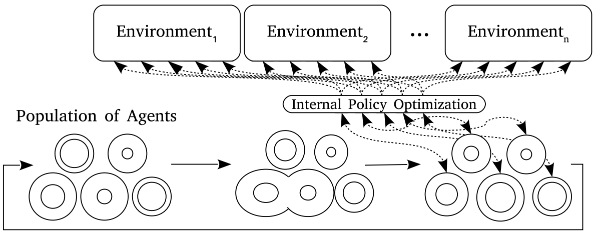

Fig. 4 Population-based scheme in which adversarial populations of agents play thousands of games in parallel on different environments. The collected data is used to discover internal rewards and to optimize their policy respectively

4 Applications

One of the earliest examples of applying distributed methods has been in the field of *network routing* [4] which used a variant of Q-learning to route packets efficiently in irregularly connected communication networks. Later, it was applied in a multi-agent system on a multi sensor network of nodes [6] to fulfill tasks based on a system-wide objective.

Following the recent trend of *Deep Reinforcement Learning* (DRL), RL has been applied on board games, *Multiplayer Online Battle Arena* (MOBA) games and *Real-Time Strategy* (RTS) videogames on large scale systems. The most prominent example in this field is *AlphaZero* [31] which learned the board games Go, Shogi, and Chess using a massively parallel architecture. Its neural networks were trained for 700000 steps with a mini-batch size of 4096 using 5000 first-generation TPUs for generating games and 64 TPUs for training. 128000 CPUs and 256 GPUs were used in *OpenAI Five* [28] for the game of Dota 2 to generate a batch-size of 1048576 observations, allocating $\approx$ 900 years of game experience each day. Despite the collaborative nature of the game, the architecture did not use an explicit communication channel between the neural networks of the heroes (agents), but a hyperparameter which controls the weighting between the individual and team reward function. The policies of the individual heroes were learned with a parallel version of the *Proximal Policy Optimization* (PPO) algorithm [30].

Another field in which distributed methods allow for more efficient learning is robotics, where collecting real data is expensive. For example, Gu et al. [12] learned a variety of 3D manipulation tasks with a collective of robots which gathered experience across multiple robotic platforms asynchronously by using DDPG and a *Normalized Advantage Function* (NAF) algorithm. Additionally, Yahya et al. [37] trained multiple robots to solve a visual-based door opening task. Here, the robots were trained with guided policy search, asynchronous updates, and a replay buffer to store the latest trajectory samples for higher sample efficiency.

5 Discussion

With the help of DM it is possible to achieve better hardware usage, resulting in faster training runs and the possibility to explore a larger set of hyperparameters. This benefit primarily concerns areas where perfect simulation environments are available and in which sampling is relatively cheap, such as video games and other simulated tasks. However, real-system applications, such as robotic tasks, benefit from DM as well, due to a progressive improvement of robotic simulations or by applying it on a small set of physical robots [5, 16].

DM also allowed a significant acceleration of research and the possibility to engage with problems which seemed to be out of reach just few years ago. Furthermore, DM can help to stabilize training due to larger batch-sizes and in some cases even achieving superlinear speed-up.

Nonetheless, these benefits come at the cost of additional coordination, communication overhead and memory requirements [5]. The demand for higher computational resources and energy consumption limits their practicability. For instance, swarm robotics is theoretically an ideal test environment for multi agent systems, but in practice these still rely on handcrafted algorithms today which mostly replicate biological swarm behaviour. Moreover, the non-stationary aspect of asynchronous and multi-agent system can lead to problems. Therefore, traditional MARL algorithms have been primarily applied to simple tasks in the past.

Additionally, by using distributed reinforcement learning techniques, the complexity of RL increases. This results in harder maintenance, debug and most importantly practicability difficulties. Consequently, unified abstraction schemes and programming frameworks are needed. Two exemplary open source frameworks, which aim to offer a solution are the *RLlib* [22] and the *SURREAL* [9] libraries. Nevertheless, the high computational costs for most RL tasks due to low sample efficiency still remains, limiting the research of individuals and most universities. Recent research in DM architectures such as IMPALA [8], Ape-X [17], and R2D2 [20] primarily attempt to deal with this problem. These are characterized by making use of distributed prioritized experience replay [29] and decoupling learning and acting [8].

6 Conclusion

With this survey, we gave a comprehensive overview of the development for DM, starting with a historical perspective until the most recent trends. Moreover, we emphasized the necessity of DM by highlighting their diverse range of applications. Collaborative and asynchronous methods exhibit favorable properties such as superlinear scaling, but come at the cost of either an additional coordination overhead or the risk of instability. Parallel synchronous methods, on the other hand, appear to be the safest approach in terms of scaling while empirically maintaining convergence properties. The sample complexity of RL algorithms is still a concern in order to reproduce results on ordinary computational systems, and is increasingly addressed in today's research. Further, to improve reusability there is a growing demand for general coding schemes of DM.

Overall, DM are a highly promising field in RL research which is expected to make decisive progress in the near future.

References

1. Amato, C., Chowdhary, G., Geramifard, A., Üre, N.K., Kochenderfer, M.J.: Decentralized control of partially observable markov decision processes. In: 52nd IEEE Conference on Decision and Control, pp. 2398–2405. IEEE (2013)
2. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation: numerical methods, vol. 23. Prentice hall Englewood Cliffs, NJ (1989)
3. Bertsekas, D.P., Yu, H.: Distributed asynchronous policy iteration in dynamic programming. In: 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1368–1375. IEEE (2010). 10.1109/ALLERTON.2010.5707073. URL http://ieeexplore.ieee.org/document/5707073/
4. Boyan, J., Littman, M.: A distributed reinforcement learning scheme for network routing. In: Proceedings of the international workshop on applications of neural networks to telecommunications, pp. 55–61. Psychology Press (1993)
5. Bu, L., Babu, R., De Schutter, B., et al.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**(2), 156–172 (2008)
6. Chen-Khong Tham, Renaud, J.C.: Multi-agent systems on sensor networks: A distributed reinforcement learning approach. In: 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 423–429. IEEE (2005). 10.1109/ISSNIP.2005.1595616. URL http://ieeexplore.ieee.org/document/1595616/
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database (2009)
8. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. arXiv preprint arXiv:1802.01561 (2018)
9. Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., Fei-Fei, L.: Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In: Conference on Robot Learning (2018)
10. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
11. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
12. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396. IEEE (2017)
13. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 66–83. Springer (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)

15. Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S., Riedmiller, M., et al.: Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286 (2017)
16. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: Is multiagent deep reinforcement learning the answer or the question? a brief survey. arXiv preprint arXiv:1810.05587 (2018)
17. Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., Silver, D.: Distributed Prioritized Experience Replay. arXiv:1803.00933 [cs] (2018). URL http://arxiv.org/abs/1803.00933. ArXiv: 1803.00933
18. Jaderberg, M., Czarnecki, W.M., Dunning, I., Marris, L., Lever, G., Castaneda, A.G., Beattie, C., Rabinowitz, N.C., Morcos, A.S., Ruderman, A., et al.: Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science **364**(6443), 859–865 (2019)
19. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. CoRR abs/1611.05397 (2016). https://puma.ub.uni-stuttgart.de/bibtex/2b09864ee3f7ad5bfbc34e5ceaf19a10a/dblp. http://arxiv.org/abs/1611.05397. http://dblp.uni-trier.de/db/journals/corr/corr1611.html#JaderbergMCSLSK16
20. Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., Dabney, W.: Recurrent experience replay in distributed reinforcement learning (2018)
21. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: In Proceedings of the Seventeenth International Conference on Machine Learning. Citeseer (2000)
22. Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., Stoica, I.: RLlib: abstractions for distributed reinforcement learning. In: Dy, J., Krause, A. (eds.) Proceedings of Machine Learning Research, Stockholmsmässan, Stockholm, Sweden, 10–15 July, vol. 80, pp. 3053–3062. PMLR (2018). http://proceedings.mlr.press/v80/liang18b/liang18b.pdf. http://proceedings.mlr.press/v80/liang18b.html
23. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
24. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
25. Mao, H., Gong, Z., Ni, Y., Xiao, Z.: Accnet: Actor-coordinator-critic net for" learning-to-communicate" with deep multi-agent reinforcement learning. arXiv preprint arXiv:1706.03235 (2017)
26. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning, pp. 1928–1937 (2016)
27. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine learning **13**(1), 103–130 (1993)
28. Raiman, J., Zhang, S., Wolski, F.: Long-term planning and situational awareness in OpenAI five. arXiv preprint arXiv:1912.06721 (2019)
29. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized Experience Replay. In: Interantional Conference for Learning Representations (ICLR) (2015). URL http://arxiv.org/abs/1511.05952
30. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
31. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **362**(6419), 1140–1144 (2018)
32. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning, pp. 330–337 (1993)
33. Tsitsiklis, J.N.: Asynchronous stochastic approximation and q-learning p. 18 (1994)
34. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

35. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3–4), 229–256 (1992)
36. Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In: Advances in neural information processing systems, pp. 5279–5288 (2017)
37. Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., Levine, S.: Collective robot reinforcement learning with distributed asynchronous guided policy search. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 79–86. IEEE (2017)