A Survey of Scalable Reinforcement Learning

George B. Stone, Douglas A. Talbert, William Eberle
Tennessee Tech University, United States

Abstract

Reinforcement Learning (RL) has been around for some time now, and various issues have been associated with the provision. In the last two years, RL has realized a boost in popularity through deep learning. For instance, RL played a critical role in the DeepMind AlphaGo program that was crucial to beating a top-level Go player in 2016 [1]. However, there is still work to be done despite RL's advancements before the DeepMind AlphaGo program becomes mainstream. For instance, one of the challenges of the program is its ability to multitask. [1] indicate that agents should perform various functions to achieve general AI. However, in the contemporary understanding, multitasking is one of the challenges of AI and RL scalability; for instance, it should not take over 1000 hrs. of different tasks to learn 1000 various tasks. Instead, AI agents must build up a library of general knowledge and learn general skills that are common and applicable across a variety of tasks. However, this ability is currently non-existent in such programs as the Deep Q-Network (DQN). While the DQN has previously been shown to have the ability to play various games, including Atari games, there is often no learning across the tasks. Each of the games is learned from scratch, which is often not scalable.

Keywords: Reinforcement Learning, Scalability, self-recon-figuration, model-based, path-planning, multi-agent

1. Introduction

The Irrespective of being scalable, an AI with the capability to learn general skills also can adapt to new tasks, ensuring dynamic capabilities across dynamic environments [2], [3]. Besides multitasking, RL has reduced capabilities of learning to remember. In many real-world applications, an observation typically captures a small part of the complete environment that can help it achieve the best action. However, for such partially observable settings, an agent should take into account not merely a current observation but also previous actions to determine the best action. For instance, intelligent systems that work to augment the capabilities of human actions should have the ability for remembrance. Consider a program that helps with billing issues for a company. If a customer asks an employee, "what is your outstanding balance?" the program should be capable of providing an answer for the same—it should have the capability of remembering the course of conversation to decipher the account the agent is talking about. Humans can change subjects, loopback and again. By using the maxim, some information can be crucial; however, some other information can be tangential. Therefore, learning a representative compact that only has salient information is crucial.

2. Literature Review

Studies indicate that many RL systems cannot be trained directly and need learning from the fixed logs of the systems' behavior. In many cases, the indicators are that we are deploying an RL approach that replaces a previous control system and logs from an available policy. Like much of the research conducted on deep reinforcement learning [4], real systems often lack separate training and evaluation environments. All the retraining data come from real systems, and agents lack a separate exploration policy during the training. Instead, the agent is required to perform reasonably well and act safely through the learning, and for many systems, it implies that exploration is limited. The resulting data has a low variance—very little of the state space can be covered in the logs. Other researchers have made similar observations, indicating that there is often only one instance of the system, and approaches that instantiate hundreds or thousands of environments to collect more data for distributed training are usually compatible with this setup [5]

Real-world systems are often slow-moving, fragile, or ex- pensive such that the data produced is costly, and policy learning should be data-efficient. However, consider a case where there are offline logs of the system. Learning iterations based on the real system often takes considerably longer, as slower control frequencies can range from 1-hour to multi- month timesteps. Reward horizons could be on the order of months (such as online adverts, drug therapies). Even for situations involving higher-frequency control tasks, learning algorithm tasks should learn quickly from potential mistakes without the need to repeat them multiple numbers of times before fixing them. Learning on real systems often requires that the algorithm is both sample-efficient and performant [6]. Previous studies, however, have suggested various ways through which sample efficiency can be improved, including the use of expert demonstration as a way of bootstrapping the agent rather than learning from scratch. This approach has been previously combined with DQN [7] and achieved success on Atari and combined with DDPG [8] for insertion tasks on robots. There have been model-based deep reinforcement learning approaches where the algorithm plans against a learned transition model of the environment and has shown great promise in improving sample efficiency. A common approach that has been adopted is learning ensembles of transition models and using sampling strategies from models to explore and improve sample efficiency [9].

3. Scalability Issues of RL

Accordingly, there are further concerns about the scalability issues of RL in self-configuration modular robots. For loco- motion by self-reconfiguration tasks, the authors realized that learning problems could be more accessible when the robot has fewer modules as compared to the size of its neighborhood, as it implies that each module will see and have less to learn a policy for a smaller number of states. If we start with two modules and add more incrementally, we can effectively re- duce the problem state-space. The problem, therefore, becomes easily manageable. As a remedy, the authors accordingly suggest the use of an incremental GPAS (IGAPS) algorithm. A second alternative is that instead of parameterizing the policy space using a colossal lookup table where a single parameter represents a single observation-action pair, the policy can be represented compactly as a function of several features defined over the observation-action space of the learning module.

In other literature, other scholars have expressed the challenges associated with exploration when using RL-based systems. [10] indicate that the RL systems are primarily based on trial-and-error processes, where various combinations are adopted to give the best or highest total reward. However, this is associated with two fundamental concerns; one of the challenges is safety. For instance, consider an AI system that would want to learn how to drive a car in real-world settings. In this case, such a system would not attempt random actions as the results could be disastrous. To learn how to drive a system should try things it has not done before; however, even such actions have drastic consequences. Even in simulation-based settings where it can be easier to explore actions, effective exploration has continued to be challenging. For instance, take a case of sparse-reward tasks—for this, it can be highly impossible for an RL agent that basically learns from scratch to stumble on a reward [11]. For instance, consider a car manufacturing assembly with an assembly robot that accesses all the different parts that make up a car and has to learn the process of assembling a car. The chance that the robot can, through random behavior, place the parts in exactly the sample space and achieve positive reward is highly minimal.

However, to deal with the issues of safety as expressed in the previous paragraph, some researchers have continued to consider concepts such as imitation learning, intrinsic motivation, and hierarchical learning. For instance, through imitation learning, a human can easily show what good behavior encompasses. The agent can mimic the portrayal and even improve it. Intrinsic motivation is potentially based on the idea that behavior should not be solely informed by external rewards but can also be driven through internal desires. For instance, in real-world settings, people can simply try an idea due to curiosity. However, an issue with this approach is that reduced internal drivers can easily move agents towards an external reward. Hierarchical learning processes easily decompose tasks into manageable sub-tasks.

4. Model-based RL Approaches

The various recent advances in Reinforcement Learning have proved to be particularly successful in effectively tackling the sample efficiency alongside the approximation of bias that may also be referred to as overestimation bias, stemming from the value of estimates approximated by an effective function approximator [12]. Notably, overestimation bias is a widespread phenomenon and occurs in the value-based methods and may be effectively addressed by utilizing multiple critics in the actor-critic framework [13]. In return, this significantly limits the algorithms’ scalability through the increase in the number of gradient-based updates [14]. Furthermore, the efficient Reinforcement Learning methods’ essential memory complexity tends to increase linearly based on the expressive power of approximators [15] significantly. This, in turn, acts as a hindrance to the scalability of Reinforcement Learning to complex control tasks.

According to [16], world models are effectively used to examine Scalable Reinforcement Learning. Further, the author points out that the study concerning how the various artificial agents may choose specific actions with the primary objective of achieving goals makes rapid progress in significant part due to reinforcement learning [16].

On the flip side, model-based Reinforcement Learning approaches further learn a simplified model of the surrounding world [14]. The world model allows the agent to effectively predict the outcomes of various potential action sequences, which lets it play through the other hypothetical scenarios to come up with informed decisions in new situations, resulting in the reduction of the trial and error necessary for goal achievement [17]. In the past, there have been challenges regarding learning accurate world models and leveraging them in successfully learning new behaviors [12]. On the contrary, recent research such as Deep Planning Network (PlaNet) has significantly pushed the boundaries by learning the accurate world models from various images [18]. Besides, the model- based approaches have been dragged behind by the ineffective or computationally expensive mechanisms used in planning, causing a limitation in their ability to effectively solve complex tasks [15].

The other new agent that showcases Reinforcement Learning's scalability is the Dreamer [16]. The agent in question learns a world model from various images and utilizes it in learning long-sighted behaviors [19]. Dreamer leverages its world model to ensure an efficient learning process equipping it with the necessary behaviors via backpropagation through the various model predictions [14]. Besides, it efficiently learns the required actor-network to ensure the successful forecast actions by propagating gradients of rewards by effective predicted state sequences, which is complex and showcases the scalability of RL, as shown in (Figure 1) below. However, when examining scaling Reinforcement Learning with Seed RL, an RL agent can scale to thousands of machines, which effectively enables training to millions of framers each second besides significantly improving the computational efficiency [17].

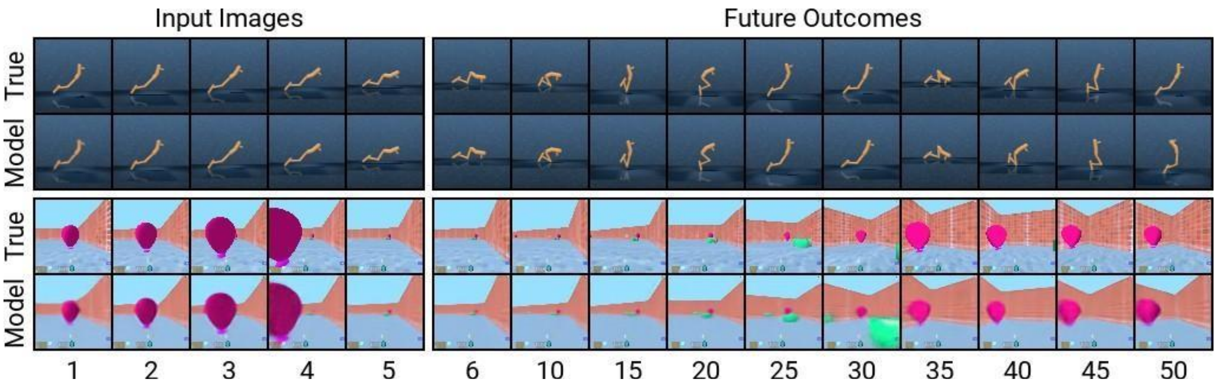

Figure 1. Propagating gradients of rewards

5. Performance of SEED RL

The performance of SEED RL is demonstrated on the popular Reinforcement Learning benchmarks, including Arcade Learning Environment, Google Research Football, as well as DeepMind Lab [16]. Further- more, there is a further demonstration that larger models are crucial in the increase of data efficiency [12]. The code has mainly been open-sourced on Github alongside various examples concerning running on Google Cloud with GPUs [20].

SEED RL is effectively designed to solve the various draw- backs of other architecture RL agents such as IMPALA [17]. The approach makes it easy for neural network inference to be effectively executed centrally by the learner on specific specialized hardware-either GPUs or TPUs [12]. This enables the accelerated deduction besides ensuring the data transfer bottleneck's avoidance by focusing on keeping the various model parameters and state local [21]. When sending multiple observations to the learner at every step of the environment, latency remains low because of a very efficient network library based on the gRPC framework alongside the asynchronous streaming RPCs [16]. Therefore, the learner can acquire up to one million queries within a second on one machine. Hence, the learner can easily be scaled to thousands of cores, such as up to 2048 on Cloud TPUs [14]. Moreover, the number of actors can also be scaled effectively to the tune of thousands of machines that fully utilize the learner and makes it possible for training to be carried out at millions of frames within every second [20].

Using SEED RL to determine scaling Reinforcement Learning requires two state-of-the-art algorithms to be integrated into the SEED RL. Firstly, V-trace is needed, which is a policy gradient-based method that should initially be introduced with IMPALA [17]. Generally, policy gradient methods are crucial in predicting an action distribution from which a particular action may be sampled [13]. However, due to the asynchronous execution of actors as well as the learner in the used SEED RL, the actors' policy slightly staggers behind the learner's policy, becoming off-policy [16]. The other algorithm essential in this process is R2D2, a Q-learning method that plays a vital role in selecting an action primarily based on the future predicted value of the particular action by the utilization of recurrent distributed replay [14]. The approach in question is crucial as it allows running the Q-learning algorithm at scale while also allowing the utilization of the recurrent neural networks that effectively predict various future values based on the acquired information of all the past frames in a single episode [13].

The other means through which scaling Reinforcement Learning is examined is learning by gradient ascent in the policy space. Notably, understanding any task in the self- reconfigurable setting tends to be formally challenging each time [21]. Every module can only observe the environment locally without knowing the global state [13]. Pointedly, such situations are described by the POMDPs and may be effectively optimized by direct policy search [14]. The use of the approach algorithm named Gradient Ascent in the Policy Space in self-reconfiguring modular robots requires some assumptions [17]. The assumptions made regarding each robotic module include the ability to effectively execute a certain number of actions, a local observation field, as well as sufficient computational memory and power to effectively run the algorithm [16].

6. Multi-agent RL

Besides, research over the years has indicated that multi-agent Reinforcement Learning particularly faces substantial scalability issues because of the size of the action and state spaces that are exponentially huge regarding the total number of agents [20]. Over time there has been significant development of efficient path-planning algorithms for multi-agent navigation that may also be used to demonstrate RL scalability [21]. The simple local rules play a significant part in generating emergent flocking alongside other behaviors [22]. Among the mentioned local methods, there has been an active exploration of the social forces model [23]. The pedestrian dynamics point of view illustrates that the various emergent behaviors have been effectively analyzed in some of the most popular microscopic models, such as the social forces, cellular automata, as well as agent-based models [24]. In this regard, RL scalability is demonstrated in the exploration of the stop-and-go waves emerging unidirectional pedestrian traffic [19]. Such explorations present numerical models of the following behavior primarily inspired by the analogy with the traffic alongside an evaluation done at a particular microscopic scale using various real examples [25]. Therefore, the work evaluates the various emergent behaviors.

In this matter, there is significance in introducing a method to clone crowd motion data relevant in animating the crowded scenes [26]. Notably, the RL agents tend to be easily cloned through the provision of various necessary behavioral variabilities by simulating the crowd. Further, the evaluation of the pedestrian models is an essential task for machine learning [27]. The RL agents primarily use the maximum likelihood estimation techniques to calibrate the various parametric models from the various real samples [24]. Pointedly, optimization serves a significant purpose in crowd simulation under various approaches such as Zips Law in crowd simulation, which effectively demonstrates that the model presents the ability to generate collective behaviors [22]. Hence, as a subfield of machine learning, RL can be scaled effectively through the control modules for various purposes of pedestrian simulation systems [21]. Besides, RL has been majorly utilized in motion-graphs-based animation as well as in the creation of various basic agent behaviors.

The RL agents can be effectively learn with rewards from the various actions done during the learning duration [22]. The RL agents' learning procedure can be scaled because of their significant difference from the data-driven agents because they tend to be primarily based on supervised learning techniques [26]. On the flip side, some studies have explored the RL approach in defining self-motivated agents that can get by themselves competently in particular domains [28]. Self- motivation translates to the agent's ability to effectively generate intrinsic rewards through an internal actor-critic model that effectively responds to the environment's various changes, showing RL scalability [21]. The intrinsically motivated RL can be utilized in allowing the artificial actors to extend as well as construct various hierarchies of certain reusable skills that give the agent competent autonomy and can hence be used for RL scalability [19]. Moreover, an intrinsically motivated model-based RL algorithm, which effectively learns various models of the domain's transition dynamics by utilizing the various decision trees, is presented [22].

Notably, the RL system that is primarily designed for the pedestrian simulations may use a simulated pedestrian as an independent agent that effectively interacts autonomously with the environment and the rest of the agents, which creates an independent trajectory and shows the RL scalability [28]. The agents autonomously follow a particular perception-actuation cycle that is effectively synchronized with the other virtual agents [28]. Besides, a special agent is referred to as the environment that takes charge of the physical simulation [21]. A physics machine, the Open Dynamics Engine, may be utilized to compute the virtual pedestrians' various interactions in the virtual world [18]. Each of the virtual agents is controlled by an agent belonging to the virtual world's multi-agent system. Moreover, the sequential decision-making process effectively controls the virtual agent consisting of adapting the particular agent's velocity to the local situation as a real pedestrian would do when in a similar situation. This provides the opportunity for scaling RL [21].

7. Conclusion

Deep Learning (DL), RL, and Deep Reinforced Learning (DRL) have gotten a lot of interest in recent years all around the world. Unlike classic AI algorithms such as expert systems, fuzzy logic, and neural networks, which are academically driven, DL, RL, and DRL are application focused and supported by high-tech firms. Their combination with readily available, big datasets and low-cost parallel computing has enabled them to find genuine applications.

RL has emerged as a promising tool for decision and control as a result of its impressive performance in a wide range of domains such as game play, robotics, and autonomous driving (Li et al., 2019), and there has been resurgence of interest in the utilization of RL in multi-agent systems [29], [30], [31].

Multi-agent reinforcement learning is a new and exciting area of machine learning theory. Because of the rise of the Internet of Things and the edge computing sector, the technical trend is shifting toward distributed systems built of a large number of compute units. MARL might be the key to realizing intelligent systems capable of learning how to collaborate in order to enhance their efficiency. The creation of such algorithms can reduce the requirement for multi-agent systems, such as cloud servers, to interact with a centralized controller, reducing the time necessary to pick an action and, from a reliability standpoint, eliminating the necessity for a single point of failure. Addressing the non-stationarity of the environment, scalability, and the requirement to transition to partially observable settings as major components of a fast- converging, efficient algorithm are the fundamental problems in the creation of MARL algorithms. In recent years, the scientific community has presented a variety of solutions to these difficulties. It was demonstrated that MARL algorithms have been used to a wide range of applications, including traffic light management, autonomous driving, and smart energy grids; nonetheless, the great majority of efforts have employed an independent learning paradigm. It would be fascinating to see how other MARL algorithm typologies perform in real- world applications. When compared to single agent RL, the multi-agent element of Multi-Agent Reinforcement Learning (MARL) introduces extra complications. Scalability is a major issue. Even if individual agents' state or action spaces are tiny, the global state or action space can accept values from a set exponentially larger in size than the number of agents [32], [33], [34]. In many circumstances, the curse of dimensionality renders the task unsolvable. RL techniques such as temporal difference (TD) learning or Q-learning, for example, need the storage of a Q-function whose size is the same as the state-action space, which in MARL is exponentially enormous in n. In a number of scenarios, such scalability concerns have been reported in the literature [35], [36], [37].

To solve the issue of scalability, a potential solution that has arisen in recent years is to take use of problem structure, for example [38], [39], [40]. One possible type of structure is enforcing local interactions, in which agents are connected with a network and interact only with agents in the graph that are close [41], [42], [43]. Such local interactions are common in networked systems such as epidemics, social networks, communication networks, queueing, smart transportation, and smart building systems [34], [44], [45].

The so-called exponential decay property, also known as the correlation decay or spatial, states that the influence of agents on each other decays exponentially in graph distance. The exponential decay feature frequently leads to the possibility for scalable, distributed algorithms for optimization and control, and has been shown to be successful for MARL [46]. While leveraging local interactions and exponential decay in MARL has proved useful, results have been obtained mainly in circumstances where the target is discounted total reward. This is understandable given that results concentrating on average reward, i.e., the payoff in stationarity, are known to be more difficult to get and necessitate alternative methodologies, even in the single agent RL. In many networked system applications, however, the average reward is a more appropriate goal. In communication networks, for example, the most frequent goal is system performance (e.g., throughput) in stationarity. There is also a suggestion for a SAC method that discovers a near-stationary point of $J(\theta)$ in time that scales with the greatest κ-hop neighbor's local state-action space size, which can be substantially less than the overall state-action space size when the graph is sparse. This is maybe the first scalable RL solution with such a proven guarantee for localized control of multi-agent networked systems. Furthermore, the framework underpinning SAC, which includes the truncated Q-function and truncated policy gradient, is a contribution in its own right and may lead to further scalable RL approaches for networked systems, such as TD-variants and Q-learning/SARSA type methods [40].

Scalable representation learning might be enabled via self- supervised reinforcement learning mixed with offline RL. Learned models are valuable because they allow us to make decisions that result in the intended outcome in the world. As a result, self-supervised training aimed at achieving each potential outcome should supply such models with the necessary understanding of how the world operates. Self-supervised RL objectives, such as those in goal-conditioned RL, are closely related to model learning, and achieving such goals is likely to need policies gaining a functional and causal grasp of the environment in which they are placed. However, in order for such strategies to be beneficial, they must be able to be applied at scale to real-world datasets. Offline RL can fill this function since it allows for the utilization of huge, heterogeneous previously acquired datasets. Putting these components together might result in a new class of algorithms that can grasp the environment via action, leading to fully scalable and automated techniques.

8. References

[1] Ghanem, M. C. and Chen, T. M. (2020). Reinforcement Learning for Efficient Network Penetration Testing. Vol. 11, pp. 6–6. DOI: 10.3390/info 11010006.

[2] Schwartz, J. and Kurniawati, H. (2019). Autonomous penetration testing using reinforcement learning. https://arxiv.org/abs/1905.05965 (Access Date: 13 May 2022).

[3] Carrara, N., Laroche, R., Bouraoui, J. L., Urvoy, T., and Pietquin, O. (2018). A fitted-q algorithm for budgeted mdps. European Workshop on Reinforcement Learning (EWRL 2018).

[4] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., and Gruslys, A. (2018). Deep q-learning from demonstrations. Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. https://doi.org/10.1609/aaai.v32i1.11757

[5] Adamski, I., Adamski, R., Grel, T., Jędrych, A., Kaczmarek, K., and Michalewski, H. (2018). Distributed Deep Reinforcement Learning: Learn how to play Atari games in 21 minutes. In International Conference on High Performance Computing. Springer. pp. 370–388.

[6] Jin, F., and Sun, S. (2008). Neural network multitasks learning for traffic flow forecasting. IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence. pp. 1897–1901.

[7] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., and Bellemare, M. G. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

[8] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. Paper presented at 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico.

[9] Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. (2019). Sample-efficient reinforcement learning with stochastic ensemble value expansion. pp. 8224–8234. https://arxiv.org/abs/1807.01675 (Access Date: 13 May 2022).

[10] Arjona-Medina, J. A., Gillhofer, M., Widrich, M. Unterthiner, T., Brand-Stetter, J., and Hochreiter, S. (2018). RUDDER: Return Decomposition for Delayed Rewards. Advances in Neural Information Processing Systems 32 (NeurIPS 2019). ISBN: 9781713807933.

[11] Nassif, R., Vlaski, S., Richard, C., Chen, J., and Sayed, A. H. (2020). Multitask learning over graphs: An approach for distributed, streaming machine learning. IEEE Signal Processing Magazine. vol. 37, no. 3, pp. 14-25, May 2020.

[12] Hua J, Zeng L, Li G, Ju Z., (2021). Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning. Sensors. 21(4):1278. https://doi.org/10. 3390/s21041278.

[13] Martinez-Gil, F., Lozano, M., and Fernández, F. (2017). Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models. Simulation Modelling Practice and Theory. Vol. 74, pp. 117–133, https://doi.org/10.1016/j.simpat.2017.03.003.

[14] Horie, N., Matsui, T., Moriyama, K., Mutoh, A., Inuzuka, N. (2019). Multi-objective safe reinforcement learning: The relationship between multi-objective reinforcement learning and safe reinforcement learning. Artificial Life and Robotics. 24, 352–359. https://doi.org/10.1007/s10015-019-00523-3.

[15] Wu, J., Xu, X., Lian, C., and Huang, Y. (2011).“Multi-robot Formation Control with Kernel-based Reinforcement Learning”. ROBOT. vol. 33, no. 3, pp. 379–384.

[16] Collins, A., and Frank, M. (2012). How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis. European Journal of Neuroscience, vol. 35, no. 7, pp. 1024–1035. doi: 10.1111/j.1460-9568.2011.07980.x.

[17] Barrett, E., Howley, E., and Duggan, J. (2012). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency and Computation: Practice And Experience, vol. 25, no. 12, pp. 1656–1674. https://doi.org/10.1002/cpe.2864.

[18] Toutounji, H., Rothkopf, C.A., and Triesch, J. (2010). Learning hierarchical controllers through reinforcement learning. Front. Comput. Neurosci. Conference Abstract: Bernstein Conference on Computational Neuroscience. doi: 10.3389/conf.fncom.2010.51.00017.

[19] Huh, N., Jo, S., Kim, H., Sul, J., and Jung, M. (2009). Model-based reinforcement learning under concurrent schedules of reinforcement in rodents. Learning and Memory, vol. 16, no. 5:315-323.

[20] Y. Zheng, Y. (2008). Algorithm of stable state spaces in reinforcement learning. Journal of Computer Applications, vol. 28, no. 5:1328-1330.

[21] Alrammal, M., and Naveed, M. (2020). Monte-Carlo Based Reinforcement Learning (MCRL). International Journal of Machine Learning and Computing, vol. 10, no. 2:227-232.

[22] Yücesoy, Y., and M. Tümer, M. (2015). Hierarchical Reinforcement Learning with Context Detection (HRL-CD). International Journal of Machine Learning and Computing, vol. 5, no. 5:353-358. October.

[23] Matzel, L. (1985). Signal properties of reinforcement and reinforcement omis- sion on a multiple fixed-ratio schedule. Animal Learning and Behavior, vol. 13, no. 2:187-193.

[24] Stahl, D. (2003). Action-Reinforcement Learning Versus Rule Learning. SSRN Electronic Journal.

[25] Sun, F., and Z. Zheng, Z. (2009). CMAC application using triangulation in rein- forcement learning," Journal of Computer Applications, vol. 29, no. 3:871–873, 2009.

[26] Vafashoar, R., and Meybodi, M. R. (2019). Reinforcement learning in learning automata and cellular learning automata via multiple reinforcement signals. Knowledge-Based Systems. Vol 169:1-27. DOI: 10.1016/j.knosys.2019.01.021.

[27] Fowler, S. C. and Notterman, J. M. (1974). Reinforcement rate and force proportional reinforcement. Learning and Motivation, Volume 5, Issue 1:80-91. February. DOI: 10.1016/0023-9690(74)90039-3.

[28] Wawrzynski, P. (2015). Control Policy with Autocorrelated Noise in Rein- forcement Learning for Robotics. International Journal of Machine Learning And Computing, vol. 5, no. 2: 91–95.

[29] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Nature 529 (7587): 484-489. January.

[30] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In Proceedings of International Conference on Machine Learning, pages 1329–1338.

[31] Li, D., Zhao, D., Zhang, Q., and Chen, Y. (2019). Enforcement learning and deep learning based lateral control for autonomous driving. IEEE Computational Intelligence Magazine. 14(2):83–98.

[32] Bertsekas, D. P., and Tsitsiklis, J. N. (1996). Neuro-dynamic programming. Athena Scientific Belmont, MA. Vol 5.

[33] Blondel, V. D., and Tsitsiklis, J. N. (2000). A survey of computational complexity results in systems and control. Automatica, 36(9):1249–1274.

[34] Christos H Papadimitriou and John N Tsitsiklis. The complexity of optimal queuing network control. Mathematics of Operations Research, 24(2):293–305, 1999

[35] Zhang, K., Yang, Z., and Başar, T. (2021). Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In Studies in Systems, Decision and Control. Studies in Systems, Decision and Control. Vol. 325:321-384. Springer. DOI: 10.1007/978-3-030-60990-0_12

[36] Kearns, M., and Koller. D. (1999). Efficient reinforcement learning in factored MDPs. In the Proceedings of the 16th international joint conference on Artificial intelligence. Vol 2:740–747, July.

[37] Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research. Vol 19:399–468.

[38] Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. (2021). Mean-Field Controls with Q-Learning for Cooperative MARL: Convergence and Complexity Analysis. SIAM Journal on Mathematics of Data Science. Vol. 3, Iss. 4DOI: 10.1137/20M1360700

[39] Qu, G., and Li, N. (2019). Exploiting fast decaying and locality in multi-agent MDP with tree dependence structure. In the Proceedings of IEEE 58th Conference on Decision and Control (CDC). DOI: 10.1109/CDC40024.2019.9029635.

[40] Qu, G., Wierman, A., and Li, N. (2020). Scalable Reinforcement Learning of Localized Policies for Multi-Agent Networked Systems. Proceedings of the 2nd Conference on Learning for Dynamics and Control, PMLR 120:256-266, 2020.

[41] Mei, W., Mohagheghi, S., Zampieri, S., and Bullo, F. (2017). On the dynamics of deterministic epidemic propagation over networks. Annual Reviews in Control, 44:116–128.

[42] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., and Faloutsos. C. (2008). Epidemic thresholds in real networks. ACM Transactions on Information and System Security (TISSEC), 10 (4):1, 2008.

[43] Alessandro Zocca. Temporal starvation in multi-channel csma networks: an nalytical framework. Queueing Systems, 91(3-4):241–263, 2019.

[44] Vogels, R., van Renesse, R., and Birman, K. (2003). The power of epidemics: Robust communication for large-scale distributed systems. SIGCOMM Comput. Commun. Rev., 33(1):131–135, January.

[45] Zhang R., and Pavone, M. (2016). Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. The International Journal of Robotics Research, 35(1-3):186–203.

[46] Gamarnik, D., Goldberg, D. A., and Weber, T. (2014). Correlation decay in random decision networks. Mathematics of Operations Research, 39(2):229–261, 2014.