Paper:

A Comparison Study of DQN and PPO based Reinforcement Learning for Scheduling Workflows in the Cloud

Jianghang Huang*[1], Huifang Li*[1], Yaoyao Lu*[1], and Senchun Chai*[1]

*[1] School of Automation, Beijing Institute of Technology, Beijing 100081, P. R. China
E-mail: 3220190687@bit.edu.cn; huifang@bit.edu.cn; 1120172631@bit.edu.cn; chaisc97@hotmail.com
[Received 00/00/00; accepted 00/00/00]

Abstract. As a new mode of resource and service provision, cloud computing provides a cost-effective deploying environment for hosting scientific applications. More and more applications modeled as workflows are being moved to the cloud. However, the traditional heuristic and metaheuristic algorithms encounter some problems that cannot be ignored in solving the scheduling problem, such as difficult to get the optimal solution or consuming too much time. In this work, we design the corresponding workflow scheduling model based on Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) algorithm with makespan minimized, and explore the performance difference of DQN and PPO algorithms in workflow scheduling through different experiments. Firstly, a cloud workflow scheduling environment is developed based on the cloud workflow task model. Secondly, according to the DQN and PPO algorithm, the agent models are designed. Then we train the resulting workflow scheduling models. During the training process, the agent (actor) learns and accumulates experience constantly to optimize the scheduling results. Finally, experiments are conducted and results show that both PPO algorithm and DQN algorithm show excellent characteristics in solving the cloud workflow scheduling problem, which also reflects the advantages of deep reinforcement learning in sequential decision optimization.

Keywords: Cloud computing, Workflow scheduling, Reinforcement learning, DQN algorithm, PPO algorithm

1. INTRODUCTION

Through the continuous development of cloud computing technology and increasing market demand, cloud computing has made remarkable strides in academic research and industrial applications [1]. At present, more and more complex applications modeled as workflows are being actively migrated to the cloud. Workflow scheduling is a process of mapping inter-dependent tasks on a set of available resources so that workflow applications are able to complete its execution considering the quality of service (QoS) of cloud service consumers and the efficiency of cloud platforms [2]. However, it is well known that scheduling of workflow is NP-hard [3], and becomes even harder in the cloud due to not only the large size and complexity of workflows but also the elasticity and heterogeneity of cloud resources.

In order to solve the workflow scheduling problem, plenty of heuristic and metaheuristic algorithms have been proposed to find near-optimal scheduling solutions [4]. Heterogeneous Earliest Finish Time (HEFT) and Critical Path on a Processor (CPOP) [5] are two typical list scheduling algorithms. Besides, Yuan et al. [6] is a heuristic algorithm with deadline-constrained cost optimization for workflow scheduling by introducing task priority. Heuristics are experienced-based and provide some specific scheduling rules when selecting a proper resource for each workflow task and have a relatively lower time complexity. Metaheuristic algorithms, such as Genetic Algorithm (GA) [7], Particle Swarm Optimization (PSO) [8], Ant Colony Optimization (ACO) [9], are the high-level problem-independent algorithms which are based on random search techniques, while they require a large number of iterations during the evolution process and has a high computational cost.

Considering the weakness of the heuristic and metaheuristic method, Reinforcement Learning (RL) is an interesting alternative method, where agents can improve scheduling performance through continuous learning. Cui et al. [10] provided an RL-based multiple directed acyclic graphs (DAGs) workflow scheduling scheme to optimize the deadline under given cloud computing resources. Wei et al. [11] regarded the service composition problem as a sequential decision-making process, and solved it by means of a Q-learning algorithm in a dynamic and stochastic cloud environment. Kaur et al. [12] developed a deep Q learning-based heterogeneous earliest-finish-time (HEFT) algorithm, which closely integrates the deep learning mechanism with the task scheduling heuristic HEFT. Aiming at the single objective workflow scheduling problem in cloud environment, this paper designs the corresponding workflow scheduling model based on DQN and PPO reinforcement learning algorithm with the goal of optimizing makespan (workflow execution time), and explores the performance difference of DQN and PPO algorithms in workflow scheduling through experiments.

The rest of the paper is organized as follows. Section 2

is the formulation of cloud resource, workflow and workflow scheduling. Section 3 demonstrates our proposed algorithm. Section 4 is experimental results as well as analysis. Finally, we conclude the paper in Section 5.

2. PROBLEM FORMULATION

2.1. Basic definitions

Scientific workflows are represented as DAGs, for example, a sample scientific workflow is depicted in Fig. 1. In this work, a workflow application can be expressed as DAG, $G = (T, E)$, where $T = \{t_1, t_2, ..., t_n\}$ is a set of n tasks and $E = \{e_{ij} | 1 \leq i,\ j \leq n, i \neq j\}$ is a set of edges which represent the dependencies between tasks. For example, e_{ij} indicates that t_i is the parent task and t_j is the child task. Task t_j cannot be executed until its parental task t_i is finished. A task without a parent task is called an entry task, and a task without a child task is called an exit task. The workflow starts and ends with the entry and exit tasks, respectively.

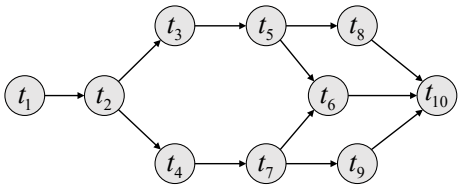


Fig. 1. An example of a workflow

We consider that an IaaS provider offers different types of virtual machines (VMs) to its clients. These VMs have different processing power and price model, which can be represented as $V = \{V_1, V_2, ..., V_m\}$, where m is the number of VMs. These VMs have different processing capacities in terms of Million Instruction per Second (MIPS) which can be denoted by ρ_k, $k \in [1,\ m]$. We believe that all VM categories contain adequate memory to run any workflow tasks and the data transferred in and out of the general storage approach are free.

2.2. Scheduling formulation

In workflow scheduling problems, each task can be allocated to different VMs, and therefore the different processing capability of VMs results in a different performance like makespan. The execution time (ET) of task t_i on resource V_k can be calculated as follows:

$$ET(i,k) = \frac{Z_i}{\rho_k} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (1)$$

where Z_i is the instruction length of task t_i in terms of Million Instruction (MI) and ρ_k is the processing capacity of V_k. The start time $ST(t_i)$and finish time $FT(t_i)$ of task t_i can be calculated as follows:

$$ST(t_i) = \max\left\{avail(V_k), \max_{t_p \in pred(t_i)} (FT(t_p))\right\} \quad (2)$$

$$FT(t_i) = ET(i,k) + ST(t_i) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (3)$$

where $avail(V_k)$ is the available or ready time of V_k on which t_i is scheduled to execute, $pred(t_i)$ is used to denote the set of immediate parents of task t_i. Note that when $t_i = t_{entry}$, $ST(t_{entry}) = 0$. Otherwise, $ST(t_i)$ can be computed as Eq. 2. Then, the makespan can be obtained as follows:

$$makespan = \max_{t_i \in T} \{FT(t_i)\} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (4)$$

where T is a set of tasks in the workflow.

3. REINFORCEMENT LEARNING BASED SCHEDULING

3.1. Reinforcement learning framework

In order to solve the workflow scheduling problems with makespan minimization by RL algorithms, we introduce an RL-based framework, which consists of two parts: an agent and environment, as shown in Fig. 2. The interaction between an agent and environment can be depicted as follows: at each time-step t, the agent first obtains the current state s_t containing all the environmental information and then takes an action a_t. Influenced by the implementation of selected action a_t, the environment changes its state from s_t to s_{t+1} and generates a reward r_{t+1} in return of action a_t.

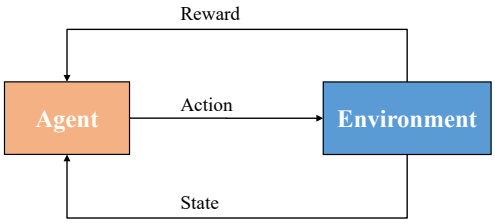


Fig. 2. A framework of reinforcement learning

3.2. DQN-based scheduler

Q-learning is based on the idea of temporal difference (TD) learning and is well suited for solving sequential decision problems. $Q(s,\ a)$ is a parameterized value function to estimate all-action values in all states. At time-step t, after taking action a_t in state s_t and observing the immediate reward r_{t+1} and state s_{t+1}, the update of Q-learning is as follows:

$$\begin{aligned} &Q_{t+1}(s_t,\ a_t) = Q_t(s_t,\ a_t) + \\ &\alpha(r_{t+1} + \gamma \max_a Q_t(s_{t+1},\ a) - Q_t(s_t,\ a_t)) \end{aligned} \quad . \quad . \quad (5)$$

where α is the learning rate. $\gamma \in [0,\ 1]$ is the discount factor that is used to trade off the importance between the immediate reward and later reward.

Based on Q-learning, Mnih et al. [13] proposed a typical algorithm, DQN, which is able to combine RL with a class of artificial neural networks known as deep neural

networks. In DQN, a nonlinear function approximator, such as a neural network, is used to represent the action-value function, so as to solve the storage and representation problem caused by the high-dimensional and continuous states or actions. It uses an online neural network parametrized by θ to approximate the vector of action values for each state, and a target network parameterized by θ^* which is periodically copied from θ to reduce oscillation during training. Meanwhile, DQN using experienced samples drawn from the replay buffer. At time-step t, the parameters θ_t can be updated as follows:

$$\theta_t \leftarrow \theta_t + \alpha \left[Y_t^{\mathrm{DQN}} - Q(s_t,\, a_t;\, \theta_t) \right] \nabla_{\theta_t} Q(s_t,\, a_t;\, \theta_t) \quad (6)$$

where the target value Y_t^{DQN} used by DQN is defined as:

$$Y_t^{\mathrm{DQN}} \equiv r_{t+1} + \gamma \max_a Q(s_{t+1},\, a;\, \theta_t{}^*) \quad . \quad . \quad . \quad . \quad (7)$$

3.3. PPO-based scheduler

PPO, a family of policy optimization methods that use multiple epochs of stochastic gradient ascent to perform each policy update [14]. These methods have the stability and reliability of trust-region methods, and are much simpler to implement. In PPO, parameters of actor (agent) is θ. At each time-step, we can get a reward r, and the final reward for a complete task is R. The actor continuously interacts with the environment, and we get a sequence $\tau = \{s_1,\, a_1,\, s_2,\, a_2,\, ...,\, s_T,\, a_T\}$, where T is the total time-steps.

The reward obtained by the sequence is the sum of the rewards obtained at each stage. Therefore, the expected reward is defined as:

$$\overline{R_\theta} = \sum_\tau R(\tau) p_\theta(\tau) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (8)$$

where $p_\theta(\tau)$ is the probability of sequence τ occurrences, $\overline{R_\theta}$ is the expected reward value, $R(\tau)$ is the sum of the rewards obtained at each stage. Our expectation is to adjust the actor's strategy to maximize the expected reward, so we use the expectation function and use the gradient boosting method to update our network parameters as follows:

$$\nabla \overline{R_\theta} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n s_t^n) \quad . \quad . \quad . \quad . \quad (9)$$

where $\nabla f(\cdot) = f(\cdot) \nabla \log f(\cdot)$, N is the number of all rewards R, and T_n is the total number of timesteps under the corresponding state.

4. EXPERIMENTS AND ANALYSIS

In this section, we first introduce the experiment setting, including resources and workflows. Then the performance of DQN and PPO based RL algorithms are tested respectively for scheduling workflows.

4.1. Experiment setting

For the purpose of model verification, the proposed algorithm is used to schedule parallel workflows in different scientific fields provided by the USC Information Sciences Institute [15]. As shown in Fig. 3, we use five typical scientific workflows: CyberShake workflow which characterizes earthquake hazards in a region, Epigenomics workflow which is used to automate various operations in genome sequence processing, LIGO's Inspiral Analysis workflow which generates and analyzes gravitational waveforms from data collected during the coalescing of compact binary systems, Montage workflow which stitches together multiple input images to create custom mosaics of the sky, Sipht workflow which is used to automate the search for untranslated RNAs for bacterial replicons in the National Center for Biotechnology Information database.

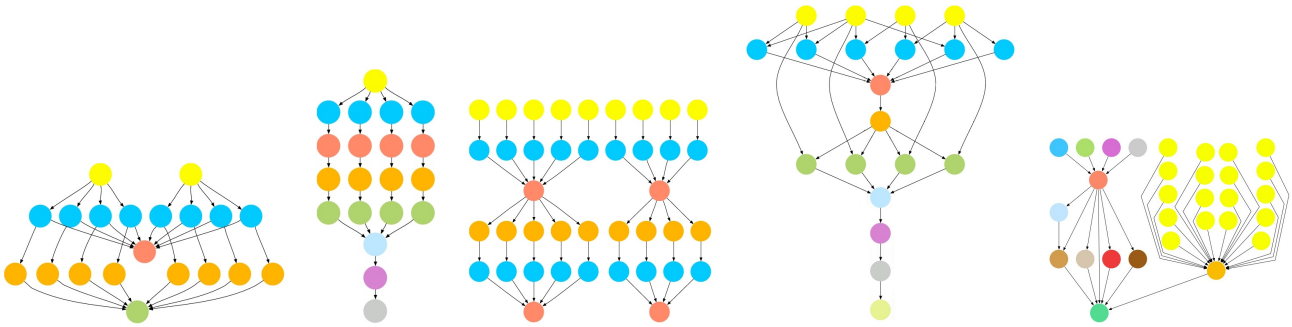
For the cloud platform, we assume that it consists of a cloud service provider and offers different types of VMs, each with different processing capabilities. In this work, the cloud service provider is assumed to provide six different types of VMs that have different configurations as shown in Table 1.

Table 1. Details of VM types from Amazon EC2

v-Type	v-CPU	Memory (GiB)	Performance (MIPS)
t3.medium	2	8	4065
t3.large	2	8	5035
c5.large	4	16	4521
m5.large	4	16	4575
c5n.large	8	32	5105
r5a.large	8	32	4549
a1.4xlarge	8	32	16861

4.2. Performance comparison

The current mainstream methods of RL include value-based and policy-based algorithms. In this work, we first choose and build two typical algorithms DQN and PPO respectively, and then compare their performance in the same experimental environment and with the same experiment configurations. This experiment is divided into two parts. One is five DAG tasks are scheduled separately. The other part is that five DAG task parallel scheduling. The parallel scheduling results of five DAGs are not equal to the simple sum of the scheduling results when they are separately scheduled, because the parallel scheduling principle of five DAGs is different from that of single DAG. In parallel scheduling, tasks in different DAGs can be executed alternately. For example, a task in CyberShake is being executed at a time-step, In the next time-step, a task in the Inspiral may be executed, which greatly improves the utilization of resources and reduces the consumption of time.



(a) CyberShake (b) Epigenomics (c) Inspiral (d) Montage (e) Sipht

Fig. 3. The sample structures of workflow applications

4.2.1. Scheduling of five workflows based on DQN and PPO algorithm

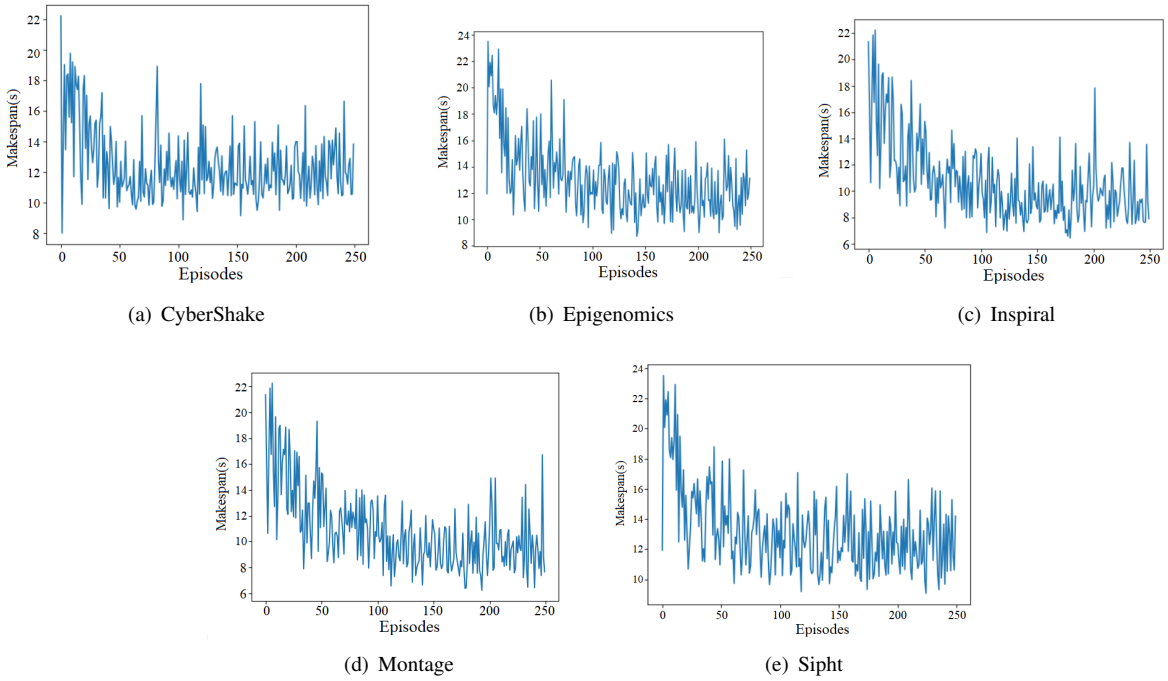
The trend of makespan of different workflows based on DQN and PPO is recorded as shown in Fig. 4 and Fig. 5, respectively. In Fig. 4, we can see that makespan decreases rapidly when the training batch is 0 to 100, which is the exploration period of DQN-based models. When the training batch reaches about 100, the makespan of the five DAGs converges to about 12 seconds, 12 seconds, 10 seconds, 9 seconds and 12 seconds respectively, and remains unchanged when the training batch is more than 100. In Fig. 5, we can see that in the five DAG scheduling results, the makespan of CyberShake and Sipht all decrease rapidly when the training batch is 0 to 100. When the training batch reaches about 100, the makespan of CyberShake and Sipht converges to about 11 seconds and 14 seconds respectively, and remains stable when the training batch is more than 100. Makespan of Epigenomics, Inspiral and Montage converged to 13 seconds, 12 seconds and 11 seconds respectively when the training times are about 200.

4.2.2. Scheduling of five parallel workflows based on DQN and PPO algorithmn

We further test the scheduling performance of DQN and PPO based model under five parallel workflows. During the training process, the trend of makespan under DQN and PPO is recorded as shown in Fig. 6. In Fig. 6 (a), the experimental results show that makespan decreases rapidly when the number of training is 0-200, which indicates that the DQN-based model is approaching convergence when episodes of training reach 200. Then, the DQN-based model continues to explore, and there is a fluctuation between 200-600 episodes. Finally, the makespan of scheduling schedule gradually stabilizes after 600 training times, converges to about 36s. It is worth noting that the curve has a peak when the episodes are about 600. The reason is that we add randomness when the agent chooses the action, which makes the agent choose the action randomly according to a certain probability, and then the scheduling result has a large error with the correct scheduling result. This probability can increase the exploratory in the early stage of training, and avoid falling into the local optimum. Finally, it will decrease with the increase of training batches. In Fig. 6 (b), The experimental results show that makespan decreases rapidly when the training times are about 0-200. At this time, PPO-based model is exploring the scheduling scheme in the right direction. Makespan decreased slowly from 200 to 800, and it was stable when the training times reached 800, and finally converged with about 60s.

When scheduling five DAGs separately, the convergence speed of PPO-based model is not fast, it needs to run many time-steps to get better scheduling results. So, the stability of PPO algorithm is not very good. However, from Fig. 6 (b), we can see that although the scheduling result converges to about 60 seconds, there is still a point that makespan is less than 60 seconds in the scheduling process. In other words, 60 seconds is not the upper limit of the optimization ability of PPO algorithm, PPO algorithm can explore a better scheduling scheme that makes makespan less than 60 seconds, which also shows that we can continue to improve the PPO algorithm, so as to improve the ability of PPO to optimize makespan and find the best scheduling scheme in workflow scheduling. Besides, we can see that the sum of makespan obtained by scheduling five DAGs separately is larger than that obtained by scheduling five DAGs in parallel, which also reflects the advantages of parallel scheduling in improving resource utilization and reducing makespan.

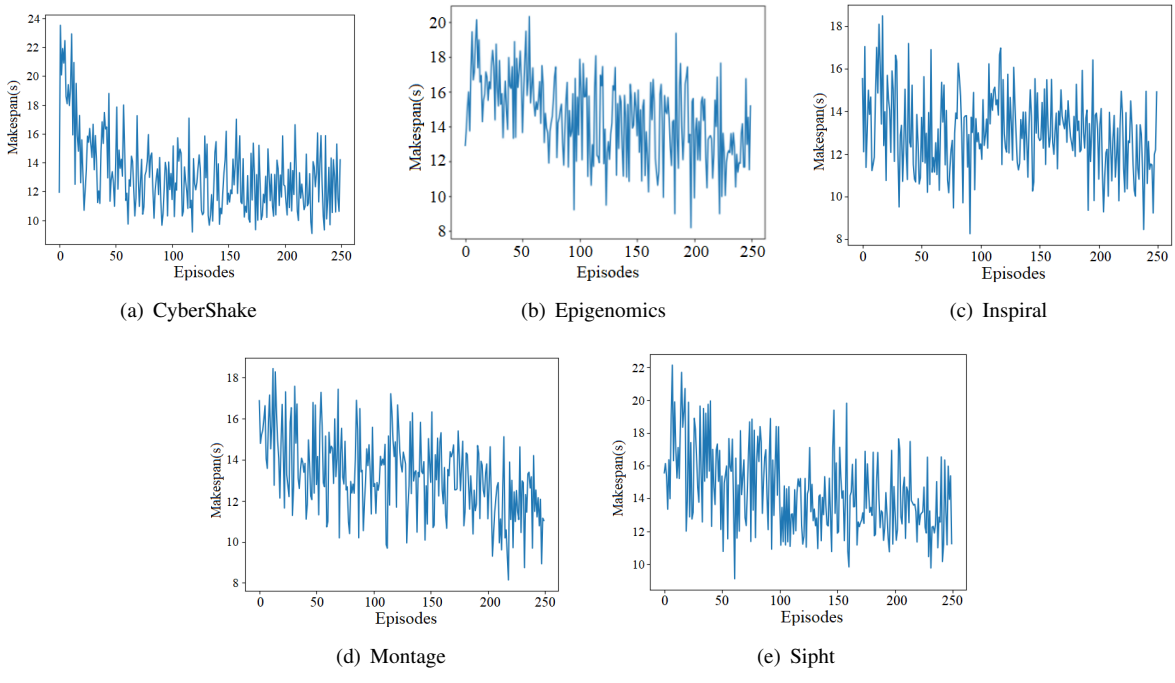
From the above experimental results, we can see that when scheduling five DAGs in parallel, the DQN algorithm converges the makespan of more than 160 seconds to about 40 seconds. When scheduling five DAGs separately, the makespan in the first 20 seconds converges to about 10 seconds, and the convergence speed is very fast, which is due to the good performance of DQN algorithm. At the same time, when running the code, we can get a better solution every time, which shows that the stability of DQN algorithm is also good. In addition, from Fig. 6 (a), we can see that although the scheduling result converges to about 40 seconds, there is a point that makespan is less than 40 seconds in the scheduling process. That is to say, 40 seconds is not the upper limit of



(a) CyberShake (b) Epigenomics (c) Inspiral

(d) Montage (e) Sipht

Fig. 4. Average makespan based on DQN



(a) CyberShake (b) Epigenomics (c) Inspiral

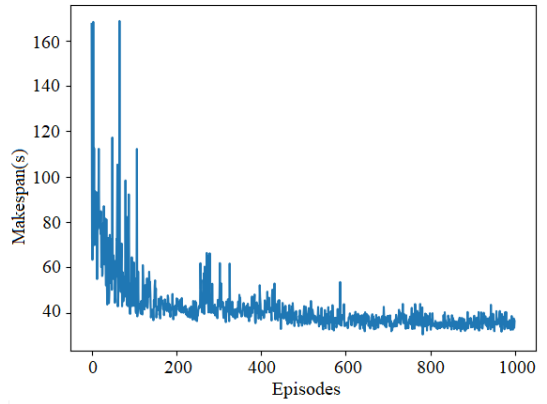
(d) Montage (e) Sipht

Fig. 5. Average makespan based on PPO

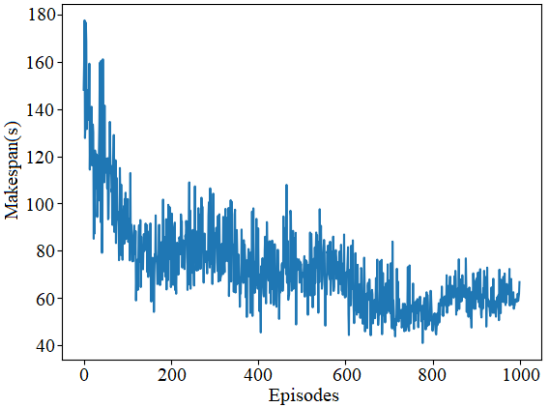
the optimization ability of DQN algorithm. DQN algorithm can explore a better scheduling scheme that makes makespan less than 40 seconds. This also indirectly shows that DQN algorithm has great potential in solving workflow scheduling problems.

5. CONCLUSION

In this work, a comparison study of DQN and PPO based reinforcement learning for scheduling workflows in the cloud is presented. A cloud workflow scheduling environment is designed, and DQN and PPO based agents and their interaction interfaces with the environment are



(a) DQN



(b) PPO

Fig. 6. Average makespan of five parallel workflows based on DQN and PPO

developed. During the training process, the agent learns and accumulates experience constantly, while optimizes the scheduling results. Finally, the near optimal scheduling solutions can be obtained by our well trained reinforcement learning model. Then we conduct a series of comparative experiments to verify its performance and the experimental results show that both PPO algorithm and DQN algorithm have excellent characteristics in solving the cloud workflow scheduling problem, which also reflects the advantages of deep reinforcement learning in solving this kind of optimization problems. In future work, we plan to reduce the possibility for standard DQN to selecte overestimated values and optimize parameters in PPO so as to make more accurate scheduling decisions.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61836001; and in part by the National Key Research and Development Program of China under Grant No. 2018YFB1003700.

References:

[1] X. Geng, Y. Mao, M. Xiong, and Y. Liu, "An improved task scheduling algorithm for scientific workflow in cloud computing environment", Cluster Computing, 22(3):7539–7548, 2019.

[2] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: a survey", Journal of Supercomputing, 71(9):3373–418, 2015.

[3] N. Rizvi and D. Ramesh, "Fair budget constrained workflow scheduling approach for heterogeneous clouds", Cluster Computing, 23(4):3185–3201, 2020.

[4] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments", Concurrency and Computation: Practice and Experience, 29(8):e4041, 2017.

[5] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Transactions on Parallel and Distributed Systems, 13(3):260–274, 2002.

[6] Y. Yuan, H. Li, W. Wei, and Z. Lin, "Heuristic Scheduling Algorithm for Cloud Workflows with Complex Structure and Deadline Constraints", In 2019 Chinese Control Conference (CCC), pp. 2279–2284, 2019.

[7] L. Teylo, U. de Paula, Y. Frota, D. de Oliveira, and L. Drummond, "A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds", Future Generation Computer Systems, 76:1–17, 2017.

[8] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds", IEEE Transactions on Cloud Computing, 2(2):222–235, 2014.

[9] Z.-G. Chen, Z.-H. Zhan, H.-H. Li, K.-J. Du, J.-H. Zhong, Y. W. Foo, Y. Li, and J. Zhang, "Deadline Constrained Cloud Computing Resources Scheduling through an Ant Colony System Approach", In 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI), pp. 112–119, 2015.

[10] D. Cui, W. Ke, Z. Peng, and J. Zuo, "Multiple DAGs workflow scheduling algorithm based on reinforcement learning in cloud computing", In Proceedings of the 7th International Symposium on Computational Intelligence and Intelligent Systems, volume 575, pp. 305–311, Guangzhou, China, 2015, Springer.

[11] Y. Wei, D. Kudenko, S. Liu, L. Pan, L. Wu, and X. Meng, "A reinforcement learning based workflow application scheduling approach in dynamic cloud environment", In Proceedings of the International Conference on Collaborative Computing - Networking, Applications and Worksharing, pp. 120–131, Edinburgh, UK, 2018, Springer.

[12] A. Kaur, P. Singh, R. S. Batth, and C. P. Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud", Software: Practice and Experience, 2020.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning", Nature, 518:529–533, 2015.

[14] Y. Gu, Y. Cheng, C. L. P. Chen, and X. Wang, "Proximal Policy Optimization With Policy Feedback", IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–11, 2021.

[15] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows", Future Generation Computer Systems-the International Journal of Escience, 29(3):682–692, 2013.