Efficient Job Scheduling in Cloud Environments using Reinforcement Learning Actor-Critic Models

Archana Naik
Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore, 560064, India | Visvesvaraya Technological University, Belagavi,590018, India
archananaik05@gmail.com (corresponding author)

Kavitha Sooda
Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore, 560019, India | Visvesvaraya Technological University, Belagavi, 590018, India
kavithas.cse@bmsce.ac.in

Received: 12 June 2024 | Revised: 24 July 2024 | Accepted: 26 July 2024

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.8104

ABSTRACT

Optimized scheduling is an important task in the scheduling of job execution on cloud virtual machines, where optimal resource usage and a shorter makespan have become important features. When scheduling jobs, balancing the workload across all available virtual machines provides optimized performance. Reinforcement learning is a better optimization algorithm due to its adaptability to dynamic environments and balancing exploration and exploitation. To perform optimized balancing of job scheduling, an Actor-Critic-based reinforcement algorithm is applied in this work. The Alibaba cloud dataset is used to analyze the algorithm's performance. Policy constraints are made for assigning the number of tasks to the scheduler. During the learning phase, the rewards turn out to be negative. After the learning phase, the rewards stabilize. The results show that the algorithm is able to produce positive reward points. A 5% reduction in the makespan of job execution demonstrates the improvement in scheduling and resource use.

Keywords-cloud resource scheduling; deep reinforcement learning; learning algorithm; task scheduling

I. INTRODUCTION

Job shop scheduling is a well-researched issue in the fields of operation research and optimization. For several decades, numerous approaches have been put forth to address these issues. Job shop scheduling has been thoroughly researched and employed in virtual machine use, including scheduling and cloud resource optimization, in more recent times. In the realm of cloud computing, job scheduling emerges as a critical challenge owing to the vast pool of resources available and the ever-changing user demands [1]. To satisfy the changing user needs in real time, the key is to efficiently coordinate jobs to make the best use of the available resources. Unlike traditional computing environments, where resources are finite and fixed, the cloud offers scalability and flexibility, enabling users to access resources on-demand and pay only for what they use. Consequently, the scheduling process must be dynamic and capable of swiftly adapting to fluctuations in demand and resource availability. Load balancing assumes a central role in cloud job scheduling, aiming to evenly distribute tasks across available resources to prevent overloading and underutilization. Dynamic load-balancing techniques work better in this situation than their static counterparts. Given the dynamic nature of cloud systems, static algorithms that rely on predetermined rules or thresholds may not be sufficient. In order to ensure optimal performance, dynamic algorithms, on the other hand, continuously analyze system conditions and modify resource allocations in real-time. By continuously evaluating resource usage patterns and adjusting allocations accordingly, these algorithms strive to strike a balance between resource efficiency and cost-effectiveness [2].

Reinforcement Learning (RL) presents a compelling approach for the dynamic scheduling of resources in cloud environments. It offers a sophisticated method to analyze resource utilization and task demands for optimal resource allocation. Through continuous learning and adaptation, RL-based schedulers can respond to changing conditions, such as fluctuations in demand or resource availability, ensuring that resources are allocated optimally at all times [3]. RL allows the artificial intelligence agent to interact with their surroundings and learn the best way to make decisions.

A. Challenges in Job Scheduling in the Cloud

Cloud computing offers a scalable computing infrastructure and a pay-per-use environment. However, the uncertainty and heterogeneity of the job environment pose significant challenges to traditional scheduling algorithms. Efficient job scheduling that minimizes resource usage while enhancing performance is crucial. Additionally, maintaining Quality of Service (QoS) levels during job scheduling is essential. The cloud provides resources according to the QoS requirements specified in Service Level Agreements (SLAs). Therefore, adhering to QoS requirements as per the SLAs is another critical challenge that needs to be addressed.

B. Objectives

- Design and implementation of an RL algorithm tailored to address the unique challenges of job scheduling in the cloud, including the uncertainty and heterogeneity of job environments.
- Applying the RL algorithm to ascertain that job scheduling uses minimal resources while achieving optimal performance.

C. Related Work

An analysis of traditional resource scheduling techniques with an emphasis on the advantages and problems of cloud computing technology is provided in [4]. Resource allocation speed and capability are the main performance metrics for Virtual Machine (VM) resource scheduling. A comparison of traditional VM resource scheduling techniques with Swarm Intelligence (SI)-based methods is performed. The challenges faced during the scheduling process, such as resource consumption, customer happiness, and service provider success, are addressed. A survey of task scheduling algorithms available in the literature is presented in [5]. Maximum scheduling time, overload, computing complexity, and latency are some of the restrictions imposed.

Conventional methods for resolving job-shop scheduling issues often assume a deep understanding of the problem and aim to find global solutions within a particular problem category. However, achieving optimal solutions can require significant computational resources and hardware, particularly when dealing with larger instances of the problem or in environments where conditions frequently change due to the complexity of the problem and the need to explore a vast search space to find the best solution. The work in [6] addresses production scheduling challenges by framing them as multi-agent reinforcement learning problems. It proposes a multi-agent learning algorithm featuring an optimistic inter-agent coordination system, a value function approximation using neural networks, and a data-efficient batch-mode RL.

The scheduling of job shops in real time is thought of as a Markov decision process. The decision to schedule will be learned by the decision-maker by gradually gaining knowledge from the environment. Q-learning is used to implement the value iteration. Q-learning avoids probability estimation and explores only one state at a time [7, 8]. Deep Reinforcement Learning (DRL) is used in the job shop scheduling problem. For improving performance, the agent is provided with information on the minimum duration to complete the job. The dense reward function based on the scheduled area is designed to improve learning. The reward is connected to the make-span minimization function employed by combinatorial optimization problem approaches [9]**.** The Markov Decision Process (MDP), along with scheduling actions, converts the dynamic allocation of flexible job-shop scheduling problems into RL problems. The scheduling actions are formed by the aggregate of the single dispatch rules and weight variables. Decreases in the mean tardy performance criteria serve as a reward function [10, 11]. MDP-based actor critic RL is implemented to minimize workflow completion and total energy consumption [12].

The cloud resource scheduling method based on DRL described utilizes CNN and imitation learning to train the optimal policy for scheduling. DRL with the actor critic algorithm is implemented for job shop scheduling in [13]. The problem is defined as the MDP. Actor and critic networks are the components of the model. Convolutional Neural Networks (CNNs) are used in the construction of these networks. The actor learns the agent's behavior based on its environment and is expected to obtain the maximum reward. The critic network provides value expectations for the statement depending on the action taken by the actor. It is shown that the DRL can deal with unexpected changes in the environment. The unique features of workflow and pipeline applications are taken into consideration for decision-making in automated workflow execution. The infrastructure scaling problem in the cloud is defined as an MDP, and Q learning functions are utilized in learning scaling policies. The workflow scheduling model is based on the Proximal Policy Optimization (PPO) method with makespan reduced, and the Deep Q-Network (DQN) is proposed. The cloud workflow task model serves as the foundation for the development of a cloud workflow scheduling environment. The agent models are built based on the DQN and PPO algorithms. The resultant workflow scheduling models are then trained. To maximize the scheduling outcomes, the agent (actor) continuously learns and gains experience throughout the training process. This RL-based scaling approach uses data on process dependence structures. Improvement from 25% to 55% was observed in comparison to the state of the art concept [14]. The problem of optimizing resource allocation is being framed as Mixed-Linear Integer Programming (MILP) in [15]. This MILP approach is applied to minimize the total cost of the network while also ensuring that the constraints related to distributing the network fragments are satisfied. To reduce congestion and improve load balancing across the network, the nature-inspired Glowworm Swarm Optimization (GSO) algorithm was implemented in [16]. Improved performance was observed in comparison to other nature-inspired algorithms such as Particle Swarm Optimization (PSO) and Cuckoo Search for the network load balancing problem.

A job scheduling mechanism for the Ions Motion Optimization cloud algorithm is implemented in [17]. Initially agents are populated as ions. Each ion holds attributes, such as position, charge, and distance. Position represents the feasible solution. The algorithm divides the population randomly into two groups: anions and cations. The fitness of each ion is calculated using an objective function. With the liquid and crystal phases of execution, the algorithm finds the convergence of the optimal solution by avoiding local minima. The result demonstrates better performance in comparison with other nature-inspired algorithms such as Cat Swarm Optimization (CSO) and GSP.

In a cloud environment deploying virtualization technology, cloud service providers are providing services to users. The users are charged for accessing the services. One of the challenges faced by the service provider is reducing power consumption. DL-based resource provisioning and task scheduling are proposed in [18, 19] to minimize energy consumption. The proposed method generates the decision based on the user's requests and the present energy consumption. This method has resulted in 320% energy cost efficiency. The dynamic flexible job shop scheduling problem in [20] considers new job arrivals, machine status failures, job rejections or cancellations, and changes in the operation's processing time. The proposed method applies the Monte Carlo Tree Search (MCTS) algorithm, where the goal is to minimize the makespan. The findings reveal that the suggested approach to dynamic scheduling is effective and promising in terms of computation efficiency and solution quality.

To handle the dynamic changes in the cloud environment, actor-critic deep reinforcement learning is proposed in [21-23]. The resource allocation based on improved energy efficiency and QoS is designed for the dynamic environment of a cloud data center. MDP is incorporated to define the resource allocation in the cloud data center. The resource allocator is embedded with a DRL-based resource controller. The DRL-based controller incorporates the asynchronous advantage actor critic model for the training process. Allocating the resources and selecting the job for scheduling are performed by the actor network. The critic network evaluates the actions of the actor network. The result displays improvements in latency, job dismissal rate, and energy efficiency. The results are also compared with other classic cloud resource allocation methods.

In DRL, rewards are typically unavailable until all jobs have been scheduled due to the extended trajectory of jobs in the cloud. This often results in suboptimal scheduling policies. To address this, a framework called deep Adversarial Imitation reinforcement learning (AIRL) is proposed for job scheduling [24]. AIRL caches the high rewards as experts and gives immediate input during the training. The Dueling-DQN for the state-action value function is implemented, which is an aggregation of the state value function and the action advantage function.

After studying multiple RL algorithms for job shop scheduling and analyzing their performance, it has been decided to implement the Actor-Critic algorithm for job scheduling.

II. METHODOLOGY

The RL environment was created to mimic the job shop scheduling problem. In this environment, each VM resource is considered a state, the placement of a job on a VM is considered an action, and the job duration on a VM is termed a reward. When a job is placed on a VM, the state of that VM changes from free to occupied. When all jobs are placed or all VMs are occupied, the process terminates, and the total accumulated reward is calculated. When the environment is prepared, it is passed to the actor-critic model. The actor-critic model calculates the policy gradient, value function, and advantage function to decide which action to select in which state. The model is trained for about 1000 iterations. The best set of actions that result in maximum reward and minimum makespan is generated as the policy of the solution.

The RL environment is constantly updated after every action is performed. The states are changed, the number of available states and actions is computed, the reward value is updated, and the list of non-suitable actions is calculated. This helps in tracking which VM resources have already been allocated, which are left to be allocated, which actions are forbidden in a given state, which actions can be performed, what is the reward achieved so far, etc. A deep neural network is the backbone for updating the policy and value function parameters. These functions are optimized using gradient descent, where the weights are updated after each iteration. The architecture of the neural network is the backbone for the value and the policy function. A CNN with an LSTM layer towards the end is employed. After experimenting with different neural network architectures, based on the results in terms of model accuracy and convergence, CNN and LSTM-based architectures were constructed. The CNN model has four 32×32 layers, a filter size of 3×3, a stride length of 2, and a padding of 1. An LSTM layer is utilized as the final layer to convert the architecture into a sequential format. ReLU is deployed as the activation function in each CNN layer. The Softmax activation function is used in the final layer for continuous prediction in the output. Each state in the proposed model represents a VM, with a status indicated by a binary value (0 for unavailable, 1 for available). Actions in this context involve selecting tasks and assigning them to appropriate VMs. As tasks are assigned, the task space decreases and the action space increases. The Actor-Critic framework employs policy functions for the actor to make decisions and value functions for the critic to evaluate those decisions. This approach allows for dynamic and efficient task scheduling, ensuring optimal utilization of VMs and minimizing idle time. After being assigned, tasks are allowed to be reassigned to other service technical experts. However, each task may not be assigned more than five times. The entire scheduling process uses minutes as the minimum time unit, and a task can only be assigned once in a minute; that is, the task must stay with the assigned service expert for at least one time unit after each assignment.

III. THE ACTOR CRITIC ALGORITHM

The components of the Actor-Critic algorithm are: The agent is the decision-making and environment-interacting entity. The outside scope of the system the agent communicates with is referred to as the environment. The state should provide illustrations of the circumstances as they are right now. Action is the choice or action taken by the representative. A reward is considered to evaluate the agent's performance. The plan or collection of guidelines directing the agent's next action is referred to as policy. Guided by the existing policy, the actor selects actions aimed to optimize the expected cumulative rewards. The components used to policy gradient are given in (1). Its primary task is to explore the action space efficiently while adapting to the dynamic environment through continuous policy refinement.

$$\nabla_\theta J(\theta) \approx 1/N \sum_{i=0}^{N} \nabla_\theta \log \pi_\theta(a_i|s_i) \cdot A(s_i, a_i) \qquad (1)$$

where $J(\theta)$ represents the expected return over several iterations under the policy parameterized by θ, $\pi_\theta(a_i|s_i)$ is the policy function of the actor, N is the number of experiences sampled, $A(s_i, a_i)$ is the advantage function, which represents the advantage of taking action a_i in state s_i over other actions available in the same state, and i represents the index of the sample currently being chosen.

By evaluating the actor's performance, the critic provides feedback on the effectiveness of the chosen actions. This assessment serves to guide the actor towards actions with higher expected returns, thereby enriching the overall learning process. The critic's role is pivotal in directing the actor's actions towards greater success. Equation (2), provides the value function applied for the critic. Starting from state s, the value function, represented by V(s), calculates the predicted cumulative reward. The parameters of the value function are represented by w, which is modeled by the critic network.

$$\nabla_w J(w) \approx 1/N \sum_{i=1}^{N} \nabla_w \big(V_w(s_i) - Q_w(s_i, a_i)\big)^2 \qquad (2)$$

where $\nabla_w J(w)$ is the gradient of the loss function for the critic's parameters w, N is the number of samples, $V_w(s_i)$ is the critic's estimate of the value of state s with parameter w, $Q_w(s_i, a_i)$ is the critic's estimation of the action-value of taking action a, and i represents the index of the sample.

The actor and critic update rules entail employing gradient descent for the critic and gradient ascent for the actor to modify their respective parameters. The key calculations in the actor-critic approach are shown by these mathematical equations. The update on the critic is done to minimize the discrepancy between the value estimation and the action parameters. Better actor selection is promoted by updating the actor's policy gradient. The model is trained for 1000 iterations, and reward values are scaled between 0-200 for better visualization. The model converges after about 600 iterations. The considered model parameters are: The learning rates of actor and critic are kept constant at 0.01. The discount for future rewards (γ) is set to 0.9. It is used to make the model focus on optimizing current rewards rather than expected future rewards. The lambda value of the Generalized Advantage Estimator (GAE) λ is fixed at 1, indicating that relative actions are more important than absolute actions for policy gradient calculation.

Algorithm 1 depicts the steps followed to train the model. The process begins with initializing state s0, policy parameters θ, value function V(s), and learning rate α. For each state s_i the next state s_{i+1} is predicted and the reward r_i is calculated. The next action a_i is determined using the policy πθ. The policy parameters θ and the value function parameters w are updated based on the sampled reward and value function corrections.

```
Algorithm 1: Model Training
1: Input: πθ, V(s)
2: Initialization: Initial state s0, $\theta \leftarrow \theta_0$, $V \leftarrow V(s_0)$, $\alpha \leftarrow \alpha_0$
3: for i = 1 to N do
4:     Find next state $s_{i+1} \approx P(s_{i+1}|s_i, a_i)$
5:     Find sample reward $r_i \approx R(s_i, a)$
6:     Find the next action $a_{i+1} \approx$ πθ ($a_{i+1}, s_{i+1}$)
7:     Update policy parameter $\theta \leftarrow \theta + \alpha_\theta J(\theta) \log$ πθ ($a_i|s_i$)
8:     Compute correction error $\delta_i \leftarrow r_i + J(\theta)_{i+1} - J(\theta)_i$
9:     Compute value function parameter $w \leftarrow w + \alpha_w \delta_i \nabla_w Q_w(s_i, a_i)$
10:    Update $s \leftarrow s_{i+1}$, a$\leftarrow a_{i+1}$
11: end for
```

IV. RESULTS AND DISCUSSION

Alibaba cloud usage dataset [25] was utilized in this study. To manage the number of jobs and complexity of the environment, an optimization scheduling method was designed. A task scheduling system is needed to maximize customer problem transfer efficiency, balance scheduling costs and efficiency, and ensure customer satisfaction. There are two sets of data: the work order set and the process time matrix. The work order set contains task ID, task generation time given in minutes, problem classification ID, and maximum task response time in minutes. Problem classification ID is based on the classification of the problem as selected by the customer. The process time matrix indicates the service technical expert's ideal processing time for the task in the category. The work_order dataset has 8840 tasks across 107 jobs, and the process time has 107 jobs associated with 133 machines. Some of these jobs have many more tasks associated with them compared to others. The number of tasks that each scheduler handles at any given time does not exceed 3. That is, when the number of tasks handled by a scheduler at a certain time is 3, no new tasks are allowed to be assigned to it at this time. By default, the scheduler will start immediately as soon as a task is submitted to it. The scheduler is implemented using the Actor-Critic algorithm to dynamically assign tasks. This optimizes the overall task scheduling process and ensures efficient utilization of resources. After every iteration, the reward is calculated for the state-action pair as shown in Figure 1. At the initial stage of learning, high instability is observed in the reward parameter. Once the model starts learning, the reward value keeps increasing, with some occasional drops due to the high exploration vs. exploitation trade-offs. The later part of the graph portrays the steady nature of the reward value. Once the model understands the environment better, it reduces the exploration and tries to exploit those actions that give higher rewards. The reward value reaches its peak and remains stable. Figure 2 depicts the total loss (comprising policy and value losses) over 20 episodes for VM task scheduling. An initial loss is observed at the beginning, followed by a notable spike in the second episode. After this initial fluctuation, the values generally remain negative. The subsequent episodes exhibit varying degrees of negative total loss, indicating periods of learning and adjustment in the model's policy and value estimations. The trend of negative total loss towards the latter episodes suggests that the model is potentially stable.

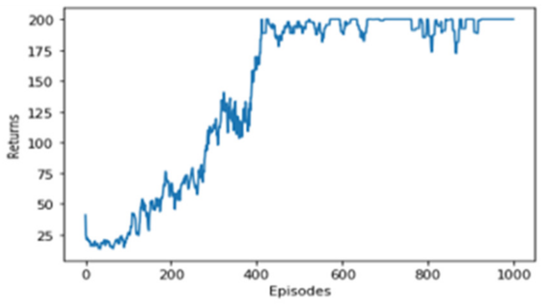

Fig. 1. Reward.

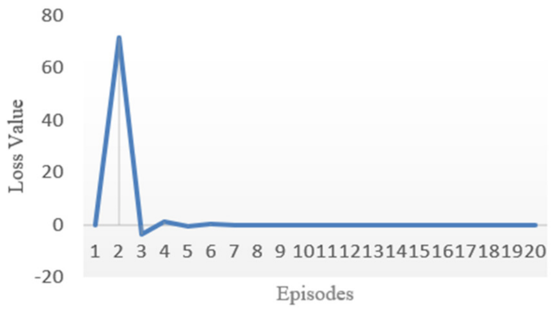

Fig. 2. Total loss.

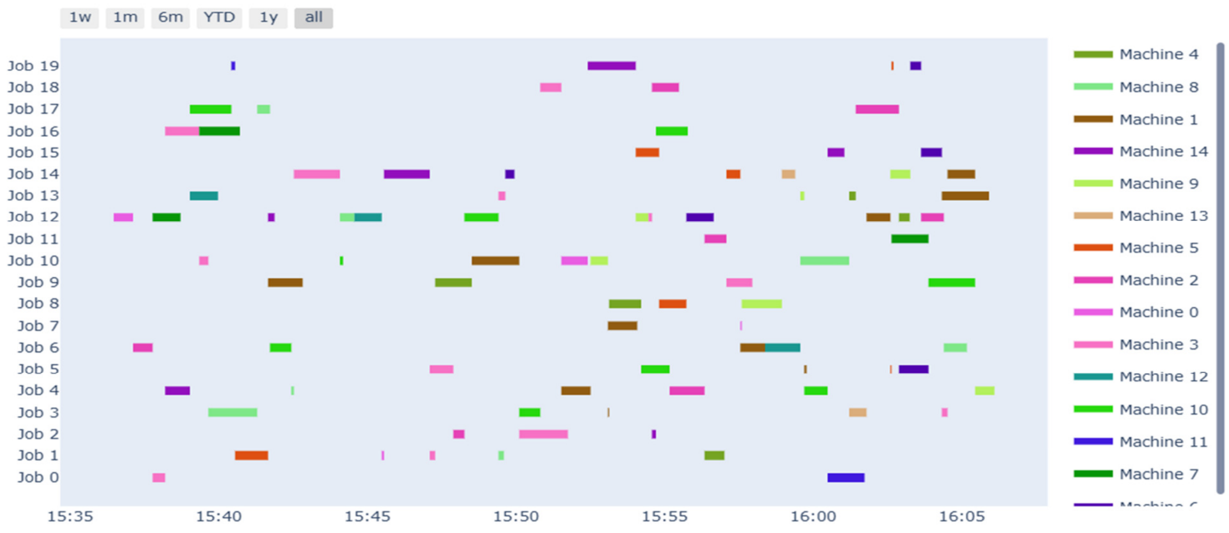

Fig. 3. Gantt chart.

Figure 3 displays the Gantt chart for a sample of jobs and machines from the test set. The x-axis shows the time on which jobs should be allocated to machines, the y-axis demonstrates the job and machine IDs, with the description provided on the right-hand side of the y-axis. Figure 4 manifests the makespan versus the episodes for a sample of jobs and machines from the test set.

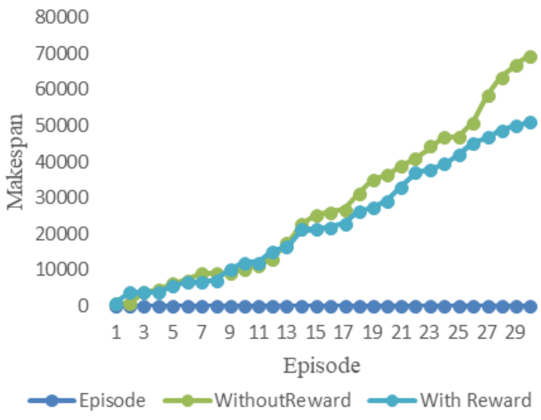

Fig. 4. Makespan comparison.

The x-axis exhibits the episodes over which the jobs are allocated to the VM. The y-axis represents the makespan for the jobs in minutes. It presents the makespan results for both with and without rewards. It is observed that the makespan is higher at the beginning when learning through rewards is being implemented. However, as learning continues and becomes more effective, the makespan is reduced. 5% of makespan is reduced for learning with rewards.

V. CONCLUSION

This study addresses the critical need for efficient job scheduling on large-scale computing systems, a gap highlighted by existing limitations in current methodologies. The proposed method demonstrates robust performance, efficiently scheduling thousands of jobs across hundreds of machines and achieving a reduced makespan of 5% within a reasonable time frame. It notably outperforms the state-of-the-art results on the same dataset, highlighting its effectiveness. A key novelty of this work lies in its dual approach to enhancing the convergence rate of CNN. The contribution of this study is twofold: first, it provides an advanced scheduling method that surpasses current benchmarks, and second, it introduces innovative techniques to enhance CNN training efficiency. This contribution is significant as it represents substantial advancements in both job scheduling and machine learning, offering practical solutions that can be widely applied across various tasks and applications. The integration of skip connections and pre-trained graphical models marks a significant step forward, paving the way for more efficient and effective CNN deployment in future machine learning endeavours.

REFERENCES

[1] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under Industry 4.0," *Journal of Intelligent Manufacturing*, vol. 30, no. 4, pp. 1809–1830, Apr. 2019, https://doi.org/10.1007/s10845-017-1350-2.

[2] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, Feb. 2020, https://doi.org/10.1016/j.jksuci.2018.01.003.

[3] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latre, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *IEEE Network Operations and Management Symposium*, Krakow, Poland, Dec. 2014, pp. 1–9, https://doi.org/10.1109/NOMS.2014.6838258.

[4] R. Eswaraprasad and L. Raja, "A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 703–711, Jul. 2017, https://doi.org/10.1080/09720510.2017.1395190.

[5] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, Feb. 2019, https://doi.org/10.1016/j.future.2018.09.014.

[6] T. Gabel and M. Riedmiller, "Adaptive reactive job-shop scheduling with reinforcement learning agents," *International Journal of Information Technology and Intelligent Computing*, pp. 1–30, 2008.

[7] T. Zhang, S. Xie, and O. Rose, "Real-time job shop scheduling based on simulation and Markov decision processes," in *Winter Simulation Conference*, Las Vegas, NV, USA, Dec. 2017, pp. 3899–3907, https://doi.org/10.1109/WSC.2017.8248100.

[8] A. Naik and K. R. Kavitha Sooda, "A study on Optimal Resource Allocation Policy in Cloud Environment," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 14, pp. 5438–5446, 2021.

[9] P. Tassel, M. Gebser, and K. Schekotihin, "A Reinforcement Learning Environment For Job-Shop Scheduling." arXiv, Apr. 08, 2021, https://doi.org/10.48550/arXiv.2104.03760.

[10] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Computers & Industrial Engineering*, vol. 180, Jun. 2023, Art. no. 109255, https://doi.org/10.1016/j.cie.2023.109255.

[11] W. Guo, W. Tian, Y. Ye, L. Xu, and K. Wu, "Cloud Resource Scheduling With Deep Reinforcement Learning and Imitation Learning," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3576–3586, Mar. 2021, https://doi.org/10.1109/JIOT.2020.3025015.

[12] A. Jayanetti, S. Halgamuge, and R. Buyya, "Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments," *Future Generation Computer Systems*, vol. 137, pp. 14–30, Dec. 2022, https://doi.org/10.1016/j.future.2022.06.012.

[13] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems," *IEEE Access*, vol. 8, pp. 71752–71762, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.2987820.

[14] Y. Garí, D. A. Monge, and C. Mateos, "A Q-learning approach for the autoscaling of scientific workflows in the Cloud," *Future Generation Computer Systems*, vol. 127, pp. 168–180, Feb. 2022, https://doi.org/10.1016/j.future.2021.09.007.

[15] M. R. Maganti and K. R. Rao, "Enhancing 5G Core Network Performance through Optimal Network Fragmentation and Resource Allocation," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14588–14593, Jun. 2024, https://doi.org/10.48084/etasr.7235.

[16] T. Akhtar, N. G. Haider, and S. M. Khan, "A Comparative Study of the Application of Glowworm Swarm Optimization Algorithm with other Nature-Inspired Algorithms in the Network Load Balancing Problem," *Engineering, Technology & Applied Science Research*, vol. 12, no. 4, pp. 8777–8784, Aug. 2022, https://doi.org/10.48084/etasr.4999.

[17] M. E. Hassan and A. Yousif, "Cloud Job Scheduling with Ions Motion Optimization Algorithm," *Engineering, Technology & Applied Science Research*, vol. 10, no. 2, pp. 5459–5465, Apr. 2020, https://doi.org/10.48084/etasr.3408.

[18] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jeju, Korea (South), Jan. 2018, pp. 129–134, https://doi.org/10.1109/ASPDAC.2018.8297294.

[19] Y. Huang *et al.*, "Deep Adversarial Imitation Reinforcement Learning for QoS-Aware Cloud Job Scheduling," *IEEE Systems Journal*, vol. 16, pp. 4232–4242, Sep. 2022, https://doi.org/10.1109/JSYST.2021.3122126.

[20] K. Li, Q. Deng, L. Zhang, Q. Fan, G. Gong, and S. Ding, "An effective MCTS-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 155, May 2021, Art. no. 107211, https://doi.org/10.1016/j.cie.2021.107211.

[21] V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems*, Denver, CO, USA, Dec. 1999.

[22] F. Cheng, Y. Huang, B. Tanpure, P. Sawalani, L. Cheng, and C. Liu, "Cost-aware job scheduling for cloud instances using deep reinforcement learning," *Cluster Computing*, vol. 25, no. 1, pp. 619–631, Feb. 2022, https://doi.org/10.1007/s10586-021-03436-8.

[23] S. Bhatnagar, V. S. Borkar, and S. Guin, "Actor–Critic or Critic–Actor? A Tale of Two Time Scales," *IEEE Control Systems Letters*, vol. 7, pp. 2671–2676, 2023, https://doi.org/10.1109/LCSYS.2023.3288931.

[24] J. Yan *et al.*, "Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach," *Computers and Electrical Engineering*, vol. 99, Apr. 2022, Art. no. 107688, https://doi.org/10.1016/j.compeleceng.2022.107688.

[25] https://tianchi.aliyun.com/competition/entrance/531831/information.