Application of a Deep Reinforcement Learning Algorithm to Virtual Machine Migration Control in Multi-Stage Information Processing Systems

Yukinobu Fukushima
Okayama University
Okayama, Japan
e-mail: fukusima@okayama-u.ac.jp

Yuki Koujitani
Okayama University
Okayama, Japan

Kazutoshi Nakane
Nagoya University
Nagoya, Japan

Yuya Tarutani
Okayama University
Okayama, Japan

Celimuge Wu
The Univ. of Electro-Commun.
Tokyo, Japan

Yusheng Ji
National Institute of Informatics
Tokyo, Japan

Tokumi Yokohira
Okayama University
Okayama, Japan

Tutomu Murase
Nagoya University
Nagoya, Japan

***Abstract*—This paper tackles a Virtual Machine (VM) migration control problem to maximize the progress (accuracy) of information processing tasks in multi-stage information processing systems. The conventional methods for this problem (e.g., VM sweeping method and VM number averaging method) are effective only for specific situations, such as when the system load is high. In this paper, in order to achieve high accuracy in various situations, we propose a VM migration method using a Deep Reinforcement Learning (DRL) algorithm. It is difficult to directly apply a DRL algorithm to the VM migration control problem because the size of the solution space of the problem dynamically changes according to the number of VMs staying in the system while the size of the agent's action space is fixed in DRL algorithms. Therefore, the proposed method divides the VM migration control problem into two problems: the problem of determining only the VM distribution (i.e., the proportion of the number of VMs deployed on each edge server) and the problem of determining the locations of all the VMs so that it follows the determined VM distribution. The former problem is solved by a DRL algorithm, and the latter problem is solved by a heuristic method. The simulation results confirm that our proposed method can select quasi-optimal VM locations in various situations with different link delays.**

Keywords-Multi-stage information processing system, VM migration control, Deep reinforcement learning, Deep Deterministic Policy Gradient (DDPG)

I. INTRODUCTION

In recent years, ultra-real-time services, such as Cross Reality (XR) and automated driving, are expected to appear. In these services, information processing tasks requested by clients need to be executed immediately (e.g., on the order of milliseconds) and the processing results should be as accurate as possible.

A multi-stage information processing system [1] [2] is one of the promising candidates for the edge computing infrastructures for ultra-real-time services. In the system, information processing tasks requested by clients are executed in parallel by an edge server and a data center. The edge server prioritizes responsiveness over accuracy; it returns the highly responsive but low accurate processing results to the clients while the data center prioritizes accuracy over responsiveness; it return the highly accurate but low responsive processing results to the clients. When operating ultra-real-time services in a multi-stage information processing system, it is important to maximize the accuracy of information processing tasks executed by the edge servers that satisfy the responsiveness requested by clients.

Previous researches on multi-stage information processing systems focused on improving the accuracy of information processing tasks executed by edge servers through Virtual Machine (VM) migration control [1] [2]. VM migration control dynamically migrates VMs, which execute the information processing tasks requested by clients on edge servers, among multiple edge servers, which leads to effective use of CPU resources on edge servers and reducing the communication delay between clients and VMs, thereby improving the accuracy of the information processing tasks. In the previous researches, as heuristic methods for VM migration control, VM sweeping method [2], VM number averaging method [2], early-blooming type priority processing method [1], and late-blooming type priority processing method [1] were proposed and their effectiveness were confirmed. These methods are, however, effective only in specific situations, such as when the system load is high and the type of information processing tasks is late-blooming type. Since the system load and the type of information processing tasks change dynamically, VM migration control that can achieve high accuracy in a wide variety of situations is needed.

In this paper, in order to achieve high accuracy in a variety of situations, we propose a VM migration method using a Deep Reinforcement Learning (DRL) algorithm. DRL algorithms are expected to achieve a quasi-optimal performance in a variety of situations through interactions between a learning agent and a dynamically changing environment. On the other hand, it is difficult to directly apply a DRL algorithm to the VM migration control problem because, in the problem, the size of the solution space dynamically changes according to the dynamic changes in the number of VMs staying in the system while the size of the agent's action space is fixed in DRL algorithms. Therefore, in this paper, we divide the VM migration control problem into two problems: the problem of determining only the VM distribution (i.e., the proportion of the number of VMs deployed on each edge server) and the

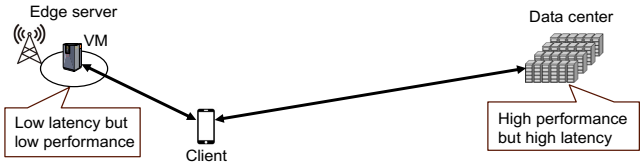

Figure 1. Multi-stage information processing systems.

problem of determining the locations of all the VMs so that it follows the determined VM distribution. The former problem is solved by a DRL algorithm, and the latter problem is solved by a heuristic method. This approach makes it possible to apply a DRL algorithm with a fixed action space size to the VM migration control problem.

The rest of this paper is organized as follows. Section 2 introduces related work on VM migration control. Section 3 describes the multi-stage information processing system and the VM migration control problem. In Section 4, we propose a VM migration method using a DRL algorithm. In Section 5, we evaluate the effectiveness of our proposed method with computer simulations. In Section 6, we summarize the paper and describe our future works.

II. Related Work

The work in [3]–[10] tackle VM migration control problems in server migration services and propose heuristic methods [3] [5], mathematical programming methods [4], [6]–[8], [10], and Q-learning methods [9]. These methods, however, aim at improving the communication quality between clients and VMs and reducing network power consumption, and do not consider the accuracy of information processing tasks.

The research in [1] [2] tackle VM migration control problems in multi-stage information processing systems, and propose the heuristic methods; VM sweeping method [2], VM number averaging method [2], early-blooming type priority processing method [1], and late-blooming type priority processing method [1]. These methods are, however, effective only in specific situations. For example, the VM sweeping method is shown to be effective only in situations where the system load is high and the type of information processing tasks is late-blooming type. Since the system load and the type of information processing tasks change dynamically, VM migration control that can achieve high accuracy in a wide variety of situations is needed.

The work in [11] [12] tackle VM migration control problems in mobile edge computing, and propose VM migration methods using Deep Q-Network (DQN) [13], which is a kind of DRL algorithms. These methods, however, can only be applied to VM migration control problems with a single VM because the size of an agent's action space is fixed in DQN, and cannot be applied to VM migration control problems with multiple VMs.

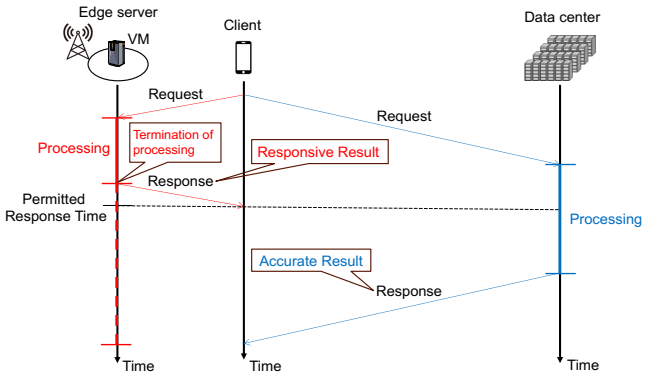

Figure 2. Flow of information processing in a multi-stage information processing system.

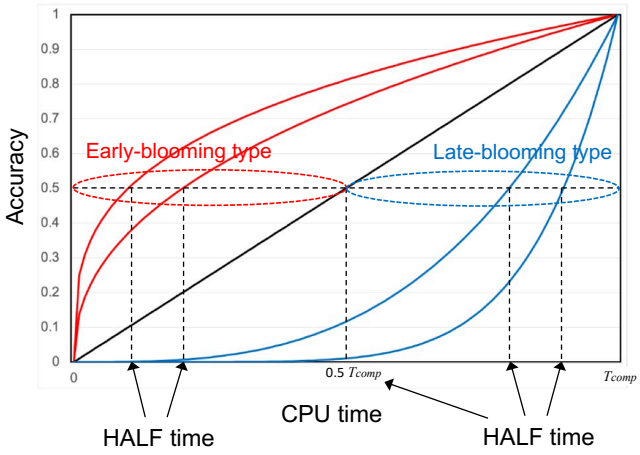

Figure 3. Relationship between CPU time allocated to a task and accuracy of the task.

III. Multi-Stage Information Processing Systems

As shown in Figure 1, a multi-stage information processing system consists of edge servers located proximate (e.g., base stations) to clients and data centers located distant from them. The system provides clients with both highly responsive and highly accurate processing results by executing information processing tasks in parallel at the edge servers and the data centers.

Figure 2 shows the flow of information processing in a multi-stage information processing system. A client requests both an edge server and a data center to process its task in parallel. When the response time permitted by the client approaches, the edge server terminates its processing to meet the permitted response time and returns the highly responsive processing result to the client. The data center, on the other hand, accomplishes its processing and returns the highly accurate processing result to the client.

In this paper, we assume the accuracy model (i.e., the relationship between the CPU time (t_{CPU}) allocated to a task

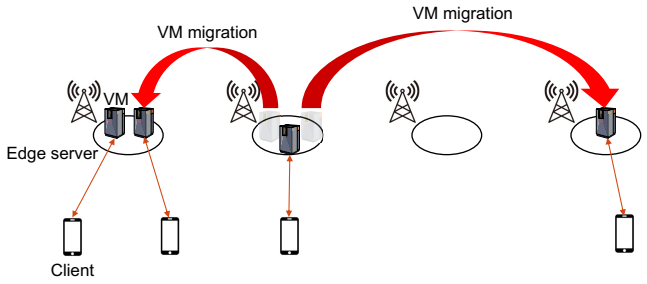

Figure 4. VM migration control in a multi-stage information processing system.

and the accuracy ($f(t_{CPU})$) of the task) that is adopted in [2]. Figure 3 shows the accuracy model. In the accuracy model, the accuracy of the task is calculated as follows.

$$f(t_{CPU}) = (\frac{t_{CPU}}{T_{comp}})^{\frac{log(0.5)}{log(\frac{HALFtime}{T_{comp}})}} \quad (1)$$

where T_{comp} represents the time for the task to be completed (i.e., accuracy reaches 1.0) and HALF time represents the time for the task to reach accuracy of 0.5. Tasks are classified based on their HALF time. The tasks with HALF time shorter than 0.5 T_{comp} are classified into early-blooming type while those with HALF time longer than 0.5 T_{comp} are classified into late-blooming type.

In this paper, we tackle a VM migration control problem among multiple edge servers for maximizing the accuracy of information processing tasks returned by edge servers within the permitted response times (Figure 4). VM migration control enables effective use of CPU resources on edge servers and reducing the communication delay between clients and VMs, thereby improving the accuracy of information processing tasks.

IV. Proposed Method

In this paper, in order to achieve high accuracy in a variety of situations, we propose a VM migration method using a DRL algorithm. With regard to applying a DRL algorithm to a VM migration control problem, it should be noted that the size of the solution space (i.e., the total number of all possible solutions) of the problem dynamically changes according to the dynamic changes in the number of VMs staying in the system. As shown in Figure 5, the size of the solution space is E^K where E is the number of edge servers and K is the number of VMs, and the size of the solution space E^K dynamically changes according to the number of VMs K. On the other hand, the size of the agent's action space in DRL algorithms is fixed. For example, an agent in Deep Deterministic Policy Gradient (DDPG) [14] outputs a vector with a fixed number of dimensions. Therefore, It is difficult to directly apply a DRL algorithm to the VM migration control problem.

To cope with the dynamic change in the size of solution space, we divide the VM migration control problem into two

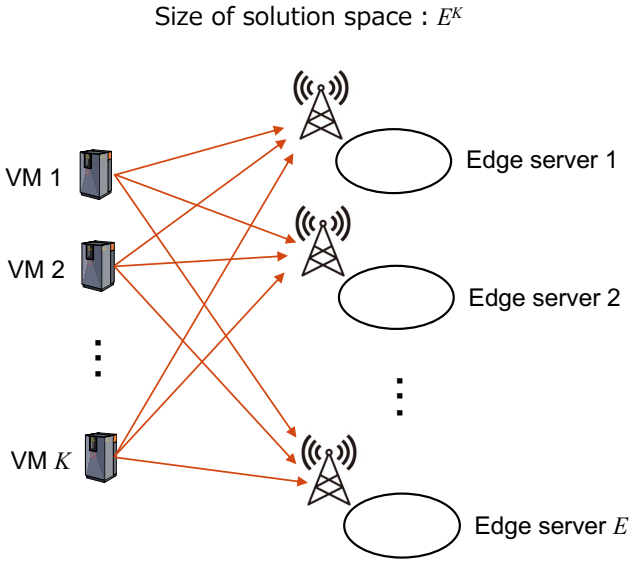

Figure 5. Size of solution space in VM migration control problem.

problems (Figure 6): the problem of determining only the VM distribution (i.e., the proportion of the number of VMs deployed on each edge server) and the problem of determining the locations of all the VMs so that it follows the determined VM distribution. The former problem is solved by a DRL algorithm, and the latter problem is solved by a heuristic method. This approach makes it possible to apply a DRL algorithm with a fixed action space size to the VM migration control problem because the VM distribution can be expressed by a vector with a fixed number of dimensions.

We adopt DDPG [14] as a DRL algorithm. DDPG approximates both a policy function $\mu(s|\theta)$ (Actor), which maps a given state to an action to be taken, and an action-value function $Q(s, a|\phi)$ (Critic) with deep neural networks. In DDPG, the Actor can output the VM distribution (i.e., the proportion of the number of VMs deployed on each edge server) as an action because it can operate over continuous action space. As well as DQN [13], DDPG adopts experience replay and target network techniques in order to learn Actor and Critic in a stable and robust way.

Figure 7 depicts an interaction between a DDPG agent and an environment, which corresponds to the VM migration control problem. When applying a DRL algorithm to the VM migration control problem, we need to define action, state, and reward in accordance with the problem. Action a_t of the agent is defined as the VM distribution (i.e., the proportion of the number of VMs deployed on each edge server), and is expressed with the following equation.

$$a_t = (p_1, p_2, \ldots, p_E) \quad (2)$$

where p_i is the proportion of the number of VMs deployed on edge server i. State s_t of the environment is defined as the

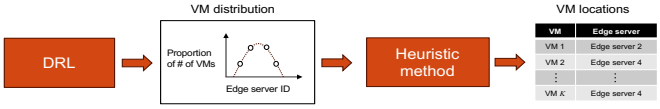

Figure 6. Outline of our proposed method.

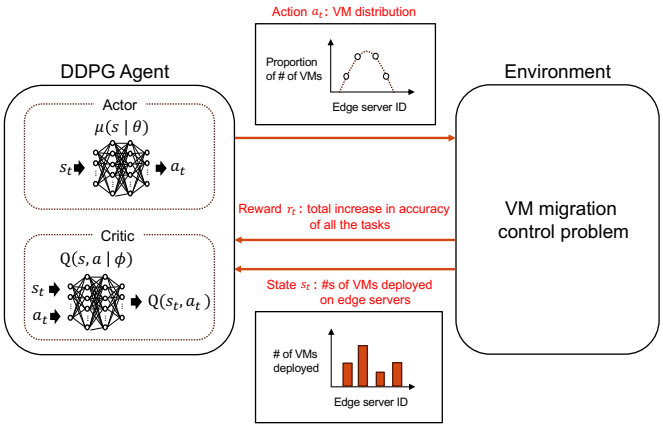

Figure 7. Interaction between the DDPG agent and the environment.

numbers of VMs deployed on edge servers, and is expressed with the following equation.

$$s_t = (d_1, d_2, \ldots, d_E) \tag{3}$$

where d_i is the number of VMs deployed on edge server i. Reward r_t is defined as the total increase in accuracy of all the tasks during the period from the last VM migration control to the current one. Algorithm 1 in Figure 8 shows the procedure of our proposed method.

After determining the VM distribution, we determine the locations of all the VMs by a heuristic method so that it follows the determined VM distribution. In this paper, we adopt a minimum client-VM delay method as the heuristic method. The minimum client-VM delay method selects the VM location with the minimum sum of the delays between clients and VMs in a brute force manner among the VM locations that follow the VM distribution determined by the DDPG agent.

V. Performance Evaluation

In this section, we evaluate our proposed method with computer simulations. Section V.A explains the simulation model. Section V.B shows the evaluation results.

A. Simulation Model

We developed the VM migration control simulator and the DDPG agent with OpenAI Gym [15] and Keras-rl [16], respectively. Table I summarizes the parameter settings as to the DDPG agent. We adopt the same parameter values as those used by the DDPG agent in Keras-rl [16] because the

Algorithm 1 Procedure of our proposed method

1: Randomly initialize weights θ of Actor $\mu(s|\theta)$ and weights ϕ of Critic $Q(s, a|\phi)$
2: Initialize weights of Actor's target network $\mu'(s|\theta')$ and Critic's target network $Q'(s, a|\phi')$: $\theta' \leftarrow \theta$, $\phi' \leftarrow \phi$
3: Initialize replay buffer R
4: **for** episode = 1, M **do**
5: Initialize a random noise $\mathcal{N}$ for action exploration
6: Observe initial state s_1 from the environment
7: **for** t = 1, T **do**
8: Select VM distribution $a_t = \mu(s_t|\theta) + \mathcal{N}_t$ as action
9: Determine locations of all the VMs by the heuristic method among the VM locations that follow the determined VM distribution a_t, and migrates the VMs
10: Observe reward r_t and the next state s_{t+1}
11: Store experience (s_t, a_t, r_t, s_{t+1}) in R
12: Sample a random minibatch of N experiences (s_i, a_i, r_i, s_{i+1}) from R
13: Learning of Critic:
Calculate target value y_i:
$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta')|\phi')$
Update weights ϕ with a gradient descent method so that loss $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\phi))^2$ is minimized
14: Learning of Actor:
Calculate policy gradient $\nabla_\theta J$:
$\nabla_\theta J \propto \frac{1}{N}\sum_i \nabla_a Q(s_i, \mu(s_i)|\phi)\nabla_\theta \mu(s_i|\theta)$
Update weights θ with a gradient ascent method so that performance of Actor J is maximized
15: Update weights of target networks:
$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
16: **end for**
17: **end for**

Figure 8. Procedure of our proposed method.

work [14] reports that a DDPG agent with the same parameter setting successfully solved various physics tasks.

The left side of Figure 9 shows the network model. This paper tackles the early stage of the performance evaluation of our proposed method; we focus on the case where four clients join and leave the multi-stage information processing system in a specific pattern on the small-scale network. The network consist of four edge servers, which are connected in a full mesh manner. We assume that the delays of all the links are identical. In order to evaluate whether our proposed method can cope with various situations with different link delay, we set the delay of each link to one of the following values: 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 [ms]. An edge server equally allocates its CPU time to all the VMs located on it. A VM is individually generated for each client. We set the response time permitted by a client to 110 [ms], the completion time

TABLE I
PARAMETER SETTINGS

Parameter	Value
Number of training episodes (M)	10,000
Discount rate (γ)	0.99
Number of hidden layers	Actor : 2, Critic : 5
Number of neurons in a hidden layer	Actor : 256, 256, Critic : 16, 32, 32, 256, 256
Activation function of hidden layers	Actor : relu, Critic : relu
Learning rate (α)	Actor : 0.001, Critic : 0.002
Noise process for action exploration ($\mathcal{N}$)	Ornstein-Uhlenbeck process
Size of replay buffer	10,000
Minibatch size (N)	64
Weights of updated parameters when updating the weights of target networks (τ)	0.005

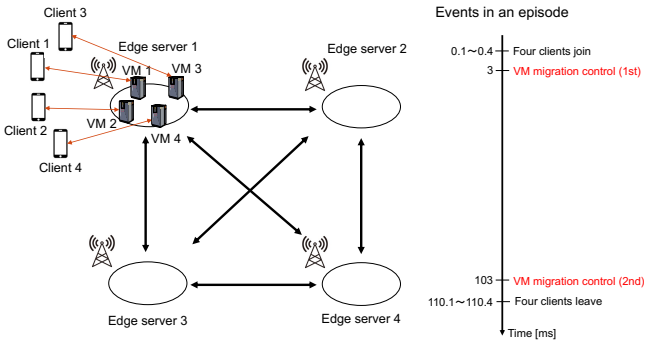

Figure 9. Network model and events in an episode.

of an information processing task (T_{comp}) to 110 [ms], and HALF time to 11 [ms] (= 0.1 $\times T_{comp}$) assuming the task type is the early-blooming type.

During an episode of the simulation, the following events occur (right side of Figure 9). When an episode starts, four clients join the system in turn every 0.1 [ms] from time 0.1 [ms] to time 0.4 [ms]. The locations of all the clients are fixed at edge server 1 during the episode. The initial locations of all the VMs are set to edge server 1. At time 3 [ms], we perform the first VM migration control. Then, at time 103 [ms], we perform the second VM migration control. Lastly, the four clients leave the system in turn every 0.1 [ms] from time 110.1 [ms] to time 110.4 [ms]. The first VM migration control aims at determining the locations of the VMs during the episode and the second VM migration control aims at obtaining the reward and the experience for learning the DDPG agent.

We compare our proposed method with the following methods.

- VM sweeping method [2]
  It migrates a VM with higher accuracy increase rate to an idle edge server so that the VM occupies the CPU time on it.
- VM number averaging method [2]
  It equally distributes all the VMs to all the edge servers for load balancing.
- Non-migration method
  It fixes all the VMs at their initial locations.
- Minimum client-VM delay method
  It locates each of the VMs to the location most proximate to its client.

B. Evaluation Results

Figure 10 shows the average accuracy among the information processing tasks executed by the four VMs for all the VM migration methods. The accuracy of our proposed method (DDPG + Minimum client-VM delay method) is plotted with 95% confidence interval because it varies depending on the initial weights of Actor and Critic, and the noises for action exploration.

Both non-migration method and minimum client-VM delay method show the constant accuracy of about 0.65 regardless

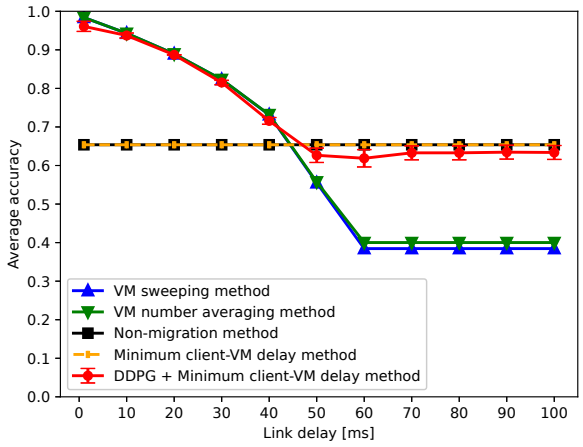

Figure 10. Average accuracy as a function of link delay.

of the link delay. This is because these methods always fix all the VMs at their initial locations (edge server 1) regardless of the link delay.

Both VM sweeping method and VM number averaging method achieve the maximum accuracy of about 0.98 when the link delay is 1 [ms], and the accuracy decreases as the link delay increases. This is explained as follows. These methods always distribute the VMs to all edge servers so that a VM is located at an edge server regardless of the link delay. As the link delay increases, the VM migration time and the communication delay between the client and the VM increases, and consequently the CPU time allocated to the task at the VM decreases after VM migration.

We compare the performances of non-migration method, minimum client-VM delay method, VM sweeping method, and VM number averaging method. When the link delay is lower than or equal to 40 [ms], VM sweeping method and VM number averaging method achieve 12 to 50% higher accuracy than non-migration method and minimum client-VM delay method. Therefore, in this case, it is desirable to distribute all the VMs to different edge servers. When the link delay

is higher than or equal to 50 [ms], non-migration method and minimum client-VM delay method achieve 17 to 70 % higher accuracy than VM sweeping method and VM number averaging method. Therefore, in this case, it is desirable to fix all the VMs at their initial locations.

Lastly, we focus on the performance of our proposed method. When the link delay is lower than or equal to 40 [ms], our proposed method 1) achieves 9 to 47% higher accuracy than non-migration method and minimum client-VM delay method, and 2) achieves almost as high accuracy (at most 2% lower accuracy) as VM sweeping method and VM number averaging method. When the link delay is higher than or equal to 50 [ms], our proposed method 1) achieves 12 to 65% higher accuracy than VM sweeping method and VM number averaging method, and 2) achieves almost as high accuracy (at most 5% lower accuracy) as non-migration method and minimum client-VM delay method. Therefore, our proposed method can select quasi-optimal VM locations in various situations with different link delays.

VI. Conclusions

In this paper, we proposed a VM migration method using a DRL algorithm in order to achieve high accuracy of information processing tasks in various situations for multi-stage information processing systems. Our proposed method divides the VM migration control problem into two problems: the problem of determining only the VM distribution and the problem of determining the locations of all the VMs so that it follows the determined VM distribution. Our proposed method solves the former problem by a DRL algorithm and the latter problem by the minimum client-VM delay method. The simulation results confirm that our proposed method can select quasi-optimal VM locations in various situations with different link delays.

In our future work, we plan to evaluate the performance of our proposed method 1) when the number of clients and VMs, and the type of information processing tasks dynamically change and 2) when different heuristic methods are adopted for the VM location decision problem in our proposed method.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP23K11065.

References

[1] K. Nakane, T. Anjiki, J. Xie, Y. Fukushima, and T. Murase, "VM Migration Considering Downtime for Accuracy Improvement in Multi-Stage Information Processing System," in *Proc. of IEEE ICCE*, Jan. 2022, pp. 335–336.

[2] T. Anjiki, K. Nakane, and T. Murase, "Performance Improvement by Controlling VM Migration between Edge Nodes in a Multi-Stage Information Processing System," in *Proc. of WSCE*, Sept. 2022, pp. 53–58.

[3] A. Yamanaka, Y. Fukushima, T. Murase, T. Yokohira, and T. Suda, "Destination selection algorithm in a server migration service," in *Proc. of CFI*, Sept. 2012, pp. 15–20.

[4] Y. Fukushima, T. Murase, T. Yokohira, and T. Suda, "Optimization of server locations in server migration service," in *Proc. of ICNS*, March 2013, pp. 200–206.

[5] Y. Hoshino, Y. Fukushima, T. Murase, T. Yokohira, and T. Suda, "An online algorithm to determine the location of the server in a server migration service," in *Proc. of IEEE CCNC*, Jan. 2015, pp. 740–745.

[6] Y. Fukushima, T. Murase, T. Yokohira, and T. Suda, "Power-Aware Server Location Decision in Server Migration Service," in *Proc of ICTC*, pp. 150–155, Oct. 2016.

[7] Y. Fukushima, T. Murase, G. Motoyoshi, T. Yokohira, and T. Suda, "Determining Server Locations in Server Migration Service to Minimize Monetary Penalty of Dynamic Server Migration," Springer Journal of Network and Systems Management, Vol. 26, Iss. 4, pp. 993–1033, Oct. 2018.

[8] Y. Fukushima, T. Murase, and T. Yokohira, "Link Capacity Provisioning and Server Location Decision in Server Migration Service," in *Proc of IEEE CloudNet*, pp. 1–3, Oct. 2018.

[9] R. Urimoto, Y. Fukushima, Y. Tarutani, T. Murase, and T. Yokohira, "A Server Migration Method Using Q-Learning with Dimension Reduction in Edge Computing," in *Proc of ICOIN*, pp. 301–304, Jan. 2021.

[10] Y. Fukushima, T. Suda, T. Murase, Y. Tarutani, and T. Yokohira, "Minimizing the Monetary Penalty and Energy Cost of Server Migration Service," Wiley Transactions on Emerging Telecommunications Technologies, Vol. 33, Iss. 9, pp. 1–34, Sept. 2022.

[11] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource Management at the Network Edge: A Deep Reinforcement Learning Approach," IEEE Network, Vol. 33, Iss. 3, pp. 26–33, May/June 2019.

[12] C. Zhang and Z. Zheng, "Task Migration for Mobile Edge Computing Using Deep Reinforcement Learning," Future Generation Computer Systems, Vol. 96, pp. 111–118, 2019.

[13] M. Volodymyr, et al. "Playing Atari with Deep Reinforcement Learning," arXiv preprint arXiv:1312.5602, 2013.

[14] T. P. Lillicrap, et al. "Continuous Control with Deep Reinforcement Learning," arXiv preprint arXiv:1509.02971, 2015.

[15] G. Brockman, et al. "OpenAI Gym," arXiv preprint arXiv:1606.01540, 2016.

[16] M. Plappert, "Keras-rl," GitHub repository, 2016.