Multi-task Deep Reinforcement Learning for IoT Service Selection

Hiroki Matsuoka and Ahmed Moustafa
Nagoya Institute of Technology, Nagoya, Japan

Keywords: Deep Reinforcement Learning, Service Selection.

Abstract: Reinforcement learning has emerged as a powerful paradigm for sequential decision making. By using reinforcement learning, intelligent agents can learn to adapt to the dynamics of uncertain environments. In recent years, several approaches using the RL decision-making paradigm have been proposed for IoT service selection in smart city environments. However, most of these approaches rely only on one criterion to select among the available services. These approaches fail in environments where services need to be selected based on multiple decision-making criteria. The vision of this research is to apply multi-task deep reinforcement learning, specifically (IMPALA architecture), to facilitate multi-criteria IoT service selection in smart city environments. We will also conduct its experiments to evaluate and discuss its performance.

1 INTRODUCTION

The Internet has played an important role in people's daily lives. In recent years, the development of IT has promoted the creation of smart cities, which aim to improve the quality of life and the efficiency of city operations and services by using IoT for energy and life infrastructure management. IoT is a paradigm in which real-world realities are connected to the Internet (Singh, 2014), and services can be provided by devices attached to them. With the development of IoT technology, the number of devices and their services deployed around the world is rapidly increasing. Therefore, In a smart city environment, a large number of IoT services are provided, and it is a challenge to select the most optimal IoT service (Xiongnan, 2014). In addition, the complexity and dynamics of the network environment makes it more difficult to select the optimal IoT service.

To solve the above challenges, we propose an approach using reinforcement learning in this research. Reinforcement learning is a powerful paradigm for sequential decision making. By using reinforcement learning, intelligent agents are able to adapt to dynamic environments. Therefore, several approaches have been proposed to use the decision-making paradigm of reinforcement learning for IoT service selection in smart city environments. However, many of these approaches select services from among the available services according to a single decision criterion. For example, quality of service (QoS) is an important criterion in many of the service selections; QoS includes a number of factors (e.g., convenience, response time, cost, etc.), which should be considered separately because they are completely different types of data. However, most of the previous studies have calculated the average value of each of these QoS factors and used that value as the QoS value. This poses the problem that it becomes difficult to perform more optimal service selection.

Therefore, in this study, we use multi-task deep reinforcement learning with IMPALA architecture to facilitate multi-criteria IoT service selection in smart city environments. By using this method, each element of QoS can be considered separately and trained for each element to consider more accurate QoS values, which will enable more optimal service selection.

The remainder of this paper is organized as follows: In Section 2, we describe the related works. Section 3 describes the proposed approach, which uses IMPALA, a multi-task deep reinforcement learning, to enable dynamic service selection with multiple criteria. In Section 4, we present experimental evaluations and results to validate the approach proposed in this paper. In Section 5, we conclude.

2 RELATED WORKS

In this section, we introduce several techniques related to dynamic service selection with multiple criteria, including reinforcement learning and multi-task deep reinforcement learning.

Matsuoka, H. and Moustafa, A.
Multi-task Deep Reinforcement Learning for IoT Service Selection.
DOI: 10.5220/0010857800003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 548-554
ISBN: 978-989-758-547-0; ISSN: 2184-433X

2.1 Reinforcement Learning

In recent years, machine learning has become a hotspot of research in the field of information technology. Reinforcement learning has emerged as a powerful paradigm for sequential decision making. With reinforcement learning, intelligent agents can learn to respond to the dynamic nature of uncertain environments. Several approaches have been proposed to use the reinforcement learning decision-making paradigm for service selection, such as in smart city environments (Hongbing, 2020). Reinforcement learning is also being used in the same way for ”service composition”. Here, service composition is mainly a study of how to combine existing services to create a system that can satisfy complex requirements, and there are many studies that apply this to service selection. (Wang, 2010), Markov decision process models and reinforcement learning are used to perform adaptive service composition. (Wang, 2016), for the problem of efficiency of reinforcement learning, a hierarchical reinforcement learning algorithm is used to improve the efficiency of service composition. Also, in (Wang, 2014), (lei, 2016), and(Hongbing, 2019), multi-agent techniques are applied to service composition. These techniques improve the efficiency of service composition and the quality of service composition results. However, most of these approaches select services based on a single decision criterion from among the available services, which leads to the loss of accuracy in a smart city environment where services need to be selected based on multiple decision criteria. In recent years, algorithms for multi-task learning have also been studied (Volodymyr, 2016). Early work proposed a multi-task deep reinforcement learning algorithm that combines reinforcement learning with deep learning (Volodymyr, 2013) and parallel computing. One type of multi-task deep reinforcement learning is the IMPALA architecture. IMPALA is described below.

2.2 IMPALA

One of the algorithms for multi-task deep reinforcement learning is the IMPALA architecture, which is also used in this research. In this section, we briefly describe IMPALA, a distributed reinforcement learning technique that allows learning to be performed on thousands of machines without compromising the learning stability or efficiency (Lasse, 2018). IMPALA uses the Actor-Critic setting to learn the measures π and the number of values V_{π}. As shown in Figure1, it consists of multiple processes (Actors) that only collect data and one or more processes (Learners) that learn with off-policy.

An Actor collects data every n -steps. It first updates its own policy μ to Learner’s policy π and collects data for n -steps. Then, it sends the collected empirical data (states, actions, rewards), the distribution of the measures $\mu(a_t|x_t)$, and the initial state of the LSTM to the Learner. The Learner, on the other hand, learns iteratively using the data sent by multiple Actors. However, in this configuration, a problem arises that the measure μ at the time of data collection does not necessarily match the measure π being learned. Therefore, by using an algorithm called V-trace, we can compensate for this misalignment of the measures and obtain a high throughput without losing sample efficiency.

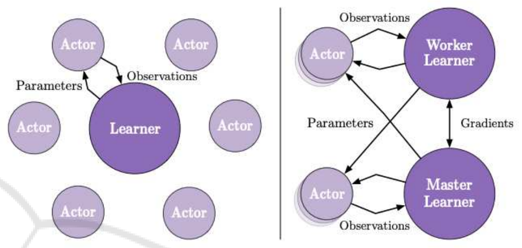

Figure 1: Left: Single Learner. Each actor generates trajectories and sends them via a queue to the learner. Before starting the next trajectory, actor retrieves the latest policy parameters from learner. Right: Multiple Synchronous Learners. Policy parameters are distributed across multiple learners that work synchronously.

2.3 V-trace

We will briefly describe V-trace here. Off-policy learning is important in decoupled distributed actor-learner architectures because there is a lag between the time an actor generates an action and the time a learner estimates the gradient. To solve these problems, we introduce an algorithm called V-trace.

In the following, we consider finding a measure π that maximizes the expected discounted reward sum (value function)$V^{\pi}(x) := E_{\pi}[\sum_{t\geq 0} \Upsilon^{t} r_t]$ in an infinite-time MDP. Here, $\gamma \in [0,1)$ is the discount factor and the measure is a stochastic measure $at \sim \pi(\cdot|xt)$. In the following, we consider learning the value function V_{π} of the learning measure π using the data collected by the behavioral measure μ.

2.3.1 V-trace Operator

The n-step V-trace operator R_n is defined as follows:

$$R^{n}V(x_s) := V(x_s) + E_{\mu}[\sum_{t=s}^{s+n-1} \Upsilon^{t-s}(C_s \cdots c_{t-1})\delta_t V] \tag{1}$$

where $\delta_t V := \rho_t(r_t + \Upsilon V(x_{t+1}) - V(x_t))$ is the TD error weighted by Importance Sampling (IS). Let $\rho_i = min(\bar{\rho}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)})$ and $c_i = min(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)})$ represent the weight coefficients of the clipped IS, and the clipping threshold satisfies $\rho^- \geq c^-$.

Here, for the on-policy ($\mu = \pi$) case, we have

$$R^n V(x_s) = V(x_s) + E_\mu[\sum_{t=s}^{s+n-1} \Upsilon^{t-s}(\tau_t + \Upsilon V(x_{t+1}) - V(X_t))] \quad (2)$$

$$R^n V(x_s) = E_\mu[\sum_{t=s}^{s+n-1} \Upsilon^{t-s}\tau_t + \Upsilon^n V(x_{s+n})] \quad (3)$$

and the V-trace in the on-policy case corresponds to the n-step Bellman operator in online learning.

In V-trace, the two different IS weight thresholds $\bar{\rho}$ and $\bar{c}$ play different roles. First, the threshold $\bar{\rho}$ for the weight factor ρ_i can be thought of as defining the only immovable point of the V-trace operator. In the tabular case, where there is no error in function approximation, the V-trace operator has as its only immovable point the value function $V^{\pi\bar{\rho}}$ of the measure $\pi^{\bar{\rho}}$, which is expressed by the following equation.

$$\pi_{\bar{\rho}}(a|x) := \frac{min(\bar{\rho}\mu(a|x), \pi(a|x))}{\sum_b min(\bar{\rho}\mu(b|x), \pi(b|x))} \quad (4)$$

Thus, when $\bar{\rho}$ is infinite, the V-trace operator has the value function V^π of the measure π as its only immovable point; when $\bar{\rho} < \infty$, it has the value function of the measure between π and μ as its immovable point. Therefore, we suppress the variance by clipping at $\bar{\rho}$. Therefore, the larger $\bar{\rho}$ is, the larger the variance in learning the off-policy (while the bias is small), and the smaller $\bar{\rho}$ is, the smaller the variance (and the larger the bias) becomes. In addition, unlike c_i, ρ_i is not multiplied by the time series, so it does not show divergent behavior depending on the time series.

Next, the threshold $\bar{c}$ of the weighting factor c_i can be thought of as controlling the speed of convergence of the V-trace operator. The multiplication of the weighting factors $(c_s \cdots c_{t-1})$ gives the TD error at time t $\delta_t V = \rho_t(r_t + \Upsilon V(x_{t+1}) - V(x_t))$. Since c_i involves a multiplication operation in the time series, it is prone to divergence, and it is important to clip the weight coefficient c_i to suppress variance. Since the size of $\bar{c}$ does not affect the immovable point of the V-trace operator (the point at which learning converges), it is desirable to set it to a value smaller than $\bar{\rho}$ in order to suppress variance.

In practice, the V-trace can be calculated recursively as follows

$$R^n V(x_s) = V(x_t) + E_\mu[\delta_t V + \Upsilon c_t(R^n V(x_{t+1}) - V(x_{t+1}))] \quad (5)$$

2.3.2 V-trace Actor-critic

We approximate the value function $V\theta$ as a function of the parameter θ and the parameter $\pi\omega$. The empirical data is assumed to have been collected with the action measure μ.

To learn the value function, we use the TD error in the V-trace operator as the loss function.

$$L\theta = (R^n V(x_s) - V_\theta(x_s))^2 \quad (6)$$

The gradient is easily computable and can be expressed by the following equation.

$$\nabla_\theta V_\theta = (R^n V(x_s) - V_\theta(x_s))\nabla_\theta V_\theta(x_s) \quad (7)$$

Also, by the measure gradient theorem and IS, the gradient of the measure $\pi_{\bar{\rho}}$ can be expressed by the following equation.

$$E_{a_s\ \mu}[\frac{\pi_{\bar{\rho}}(a_s|x_s)}{\mu(a_s|x_s)}\nabla_\omega \log \pi_{\bar{\rho}}(a_s|x_s)(q_s - b(x_s))|x_s] \quad (8)$$

where $q_s = r_s + \gamma R^n V(x_{s+1})$ represents the estimate of the action value function $Q^{\pi\omega}(x_s, a_s)$ under the V-trace operator, and $b(x_s)$ represents the state-dependent baseline function for suppressing variance. When the bias due to clipping is very small ($\bar{\rho}$ is sufficiently large), the above gradient is considered to be a good estimate of the measure gradient of π_ω. Therefore, by using $V_\theta(x_s)$ as the baseline function, we obtain the following measure gradient.

$$\nabla_\omega L_\omega = \rho_s \nabla_\omega \log \pi_\omega(a_s|x_s)(r_s + \gamma R^n V(x_{s+1}) - V_\theta(x_s)) \quad (9)$$

We can also add an entropy loss to prevent the measure π_ω from converging to a local solution.

$$L_{ent} = -\sum_a \pi_\omega(a|x_s)\log \pi_\omega(a|x_s) \quad (10)$$

In IMPALA, these three loss functions are used for training.

3 PROPOSED APPROACH

In this research, we propose an approach to adapt IMPALA architecture, a multi-task deep reinforcement learning method, to service selection based on multiple criteria in smart city environments.

As mentioned earlier in the service selection process, QoS (Quality of Service) is an important criterion in service selection. Essentially, QoS includes a number of factors (e.g., convenience, response time, cost, etc.). Since these are completely different data types, they should be considered separately. However, most of the previous studies calculate the average value of these QoS factors as follows and use that value in their calculations.

$$R = \frac{\sum_i^m (w_i * r_i)}{m} \tag{11}$$

where R represents the reward value, m is the number of QoS attributes to be considered, r_i is the value of the i-th QoS attribute, and w_i is the weight of the i-th attribute. The weights reflect the importance of the different attributes.

It is impossible to calculate the correct value of QoS with this method, and research using more accurate values is necessary. In this study, we use IMPALA, a distributed reinforcement learning method. This makes it possible to learn for each element of QoS. By judging them in a combined manner, we can consider more accurate QoS values, which will lead to optimal service selection. In addition, the main scenarios for service selection are described below.

3.1 Main Scenarios for Service Selection

A typical service selection workflow is shown in Figure 2. For an abstract service, it is necessary to determine its concrete services and finally form a concrete service selection workflow. For example, the results of possible service configurations based on the service selection workflow in Figure 2 are shown in Figure 3 and Figure 4. It can be clearly seen that as the number of candidate services increases and the service composition workflow becomes more complex, the number of possible service composition results increases dramatically.

By trying various patterns of service selection scenarios through repeated trial and error with reinforcement learning, it is possible to learn the best pattern of service selection among them.

4 EXPERIMENTS AND RESULTS

In this section, we describe the experiments we conducted to demonstrate the usefulness of our proposed method and their results.

4.1 Data Set to Be Used

In this study, we conduct experiments using IMPALA, a distributed reinforcement learning method, to enable service selection based on multiple decision criteria. The experiments focus on QoS, which is considered to be important in service selection. The data set used in this research is the QWS data set (Al-Masri, 2007). The QWS data set includes a set of 2,507 web services and their Quality of Web Service (QWS) measurements that were conducted using our Web Service Broker (WSB) framework. Each row

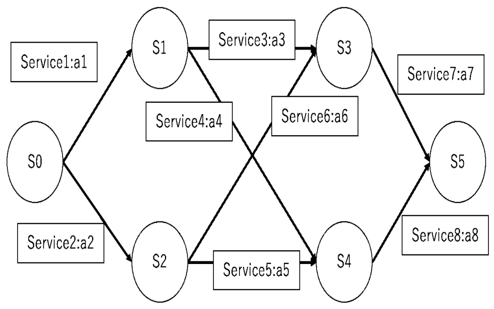

Figure 2: A simple service selection workflow.

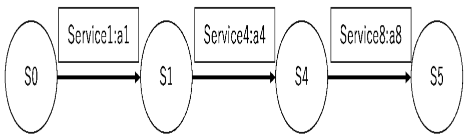

Figure 3: A possible service selection result.

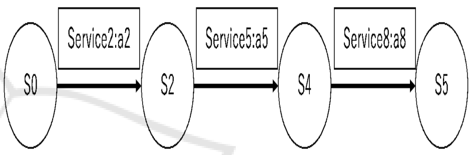

Figure 4: Another possible service selection result.

in this dataset represents a web service and its corresponding nine QWS measurements .

In this experiment, we will focus on three of the nine data (response time, availability, and throughput). By adapting the IMPALA architecture, which is a distributed reinforcement learning, to each of these three data, we will verify whether we can select the best service considering the three values.

4.2 Service Selection Model

We propose a service selection model based on the service selection workflow in the previous section. The workflow of service selection can be regarded as a Markov decision process. Based on the definition of Markov decision process, we define the Markov decision process model of service selection in dynamic environment as follows:

Definition 1. The MDP defined in this study consists of 6-tuples in total: $MDP =< S_i, S_0, S_r, A(), P, R >$.

- S_i is a finite set of states.
- S_0 is the initial state. The workflow of service selection is executed from here.
- S_r is the end state. When the end state is reached, the workflow terminates.
- A($\cdot$) is a finite set of services. where $A(s)$

represents the set of services that can be selected in states $s \in S$.

- P is a probability distribution function. When a service A is selected, it transitions from the current state s to the subsequent state *s*'. The probability of the transition is denoted by $P(s'|s,a)$.
- R is the immediate reward function. When the current state is *s* and a service is invoked, the immediate reward $R(s'|s,a,t)$ is obtained. In service selection, the value of the reward is generally determined by the QoS attributes of the service.

The workflow of service selection can be constructed based on the definition of the Markov decision process. A simple service selection workflow, as shown in Figure 2, can be described in MDP. The state set is $S = S_0, S_1, S_2, S_3, S_4, S_5$. The initial state is S_0 and the end state is S_5. The services that can be selected in different states are $A(S_0) = a_1, a_2$, $A(S_1) = a_3, a_4$, etc. Essentially, there are multiple services that can be selected in each state S. The transition probabilities include $P(S_1|S_0,a_1) = 1$. The reward value is calculated by the QoS obtained when selecting a service.

Once the workflow is determined, the service selection starts from the initial state and transitions to a new state by selecting a specific service for each state. Then, when the end state is reached, the service selection workflow is complete. This workflow consisting of multiple selected services is the result of service selection. The optimal service selection result is capable of maximizing the total reward.

In service selection using IMPALA, when selecting a service in state *S*, each agent selects one service for each element of QoS, and the total value of each reward value is the total reward value obtained when the service is selected. In this way, it is possible to select a service with higher accuracy than the conventional QoS value for service selection, and also to increase the efficiency of service selection.

4.3 Reward Function in Service Selection

Reinforcement learning algorithms are suitable for service selection problems because they use a Markov decision process model to output the optimal action selection. In order to use a reinforcement learning algorithm, we need to set up a reward function suitable for the task.

In service selection, satisfaction in choosing a service is often judged by its QoS. Therefore, the reward function is defined by the QoS attributes of the service. Since the QoS attributes may have different ranges of values, the attributes must first be normalized and mapped to [0,1]. Considering that some QoS attributes are positively correlated (e.g. throughput) and some are negatively correlated (e.g. response time), we define the following two equations.

$$r = \frac{QoS - \min}{\max - \min} \tag{12}$$

$$r = \frac{\max - QoS}{\max - \min} \tag{13}$$

Equation (12) is used for the normalization of positively correlated QoS attributes, and Equation (13) is used for the normalization of negatively correlated QoS attributes. Let *r* denote the resulting normalized value of this attribute, QoS denotes the QoS value of the attribute after selecting the service, and max and min denote the maximum and minimum QoS values of the attribute. In this study, we assume that multiple QoS values are considered individually by IMPALA. Therefore, in a state *S*, *r* is calculated for the number of QoS elements. For example, if three QoS elements (throughput, response time, and availability) are to be considered, then three *r*'s will be calculated.So the QoS value of a state is the sum of those *r*'s as the total reward value. The formula to be used for this is defined as follows.

$$R = \sum_{i=1}^{m} w_i * r_i \tag{14}$$

Here, *r* represents the total reward value, m is the number of QoS attributes to be considered, r_i is the normalized value of the *i*-th QoS attribute, and w_i is the weight of the *i*-th attribute. The weights reflect the importance of the different attributes. Typically, they are set to $\sum_{i=1}^{m} w_i = 1$ according to the user's preference for different attributes.

4.4 Details of the Experiment

In this section, we describe the experiments conducted in this study. In this experiment, we create a Markov decision process as shown in Figure 2 and conduct the experiment. Figure 5 shows the flow of the Markov decision process created in this experiment.

Here, *m* is the number of times to select a service, and we set $m = 50$. The number of services that can be selected when transitioning from state *S* to state S' is five. When a service is selected, the next group of services is determined deterministically.

In each state, we will select a service, and the reward for doing so will be calculated using Equation (12) and Equation (13). In this experiment, we

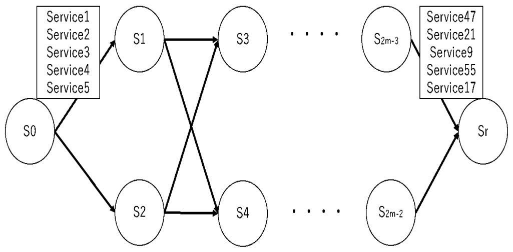

Figure 5: Experoent's MDP model.

learn to multi-task for the three QoS factors (throughput, reliability, and availability). The sum of the reward values calculated for each state will be the final cumulative reward, and the system will learn to maximize it. As for the experiments conducted, we will conduct comparison experiments between our method and a single-task service selection method using DQN. Here, DQN can only learn to single-task for each QoS element. We thought it would be possible to demonstrate the usefulness of our method by comparing the results of learning each QoS by DQN with the results of multi-task learning by IMPALA.

4.5 Experiment's Results

In this section, we present the results of the experiment and a discussion of the results. The results obtained from the experiments are shown in the Figure 6, 7, and 8. On the left is a graph of the results of single-task learning using DQN for each element of QoS. The right graph shows the results of multi-task learning using IMPALA for each element of QoS. In order to determine whether the service selected at each step is the optimal service selection, the maximum cumulative reward for each step was obtained, and the accuracy of the optimal service selection was calculated by dividing the obtained cumulative reward by the maximum cumulative reward. The results are shown in the Table 1. As a result, it can be confirmed that for all elements of QoS, multi-task learning using our method is more effective in selecting the optimal service than single-task learning using DQN. In addition, the graph shows that the cumulative reward is higher with each episode, indicating that the learning is well done. In summary, the usefulness of our method was sufficiently demonstrated.

Table 1: Comparison of service selection accuracy.

Method	available	reliability	throughput
DQN	79.1295	71.6792	80.1672
IMPALA	**81.3412**	**74.7325**	**83.0479**

As a discussion, we believe that our method was able to maintain a high level of accuracy even during multi-task learning because it uses techniques such as V-trace that are not available in DQN. In addition, learning to multi-task has made it possible to learn for multiple elements of QoS, which has made it possible to select services more flexibly according to user needs. However, although it is now possible to select one criterion from multiple elements of QoS to meet the user's needs, it would be desirable to be able to select services considering multiple criteria from multiple elements of QoS in consideration of real-world applications. For example, it would be desirable to select a service with high throughput and reliability. In the future, we would like to make this system applicable to the real world.

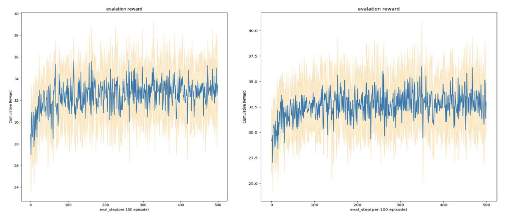

Figure 6: Results of experiments about Reliability.

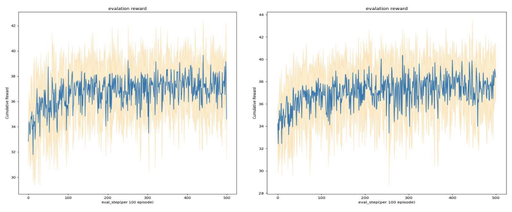

Figure 7: Results of experiments about Available.

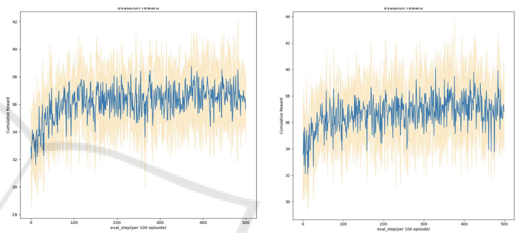

Figure 8: Results of experiments about Throughput.

5 CONCLUSION AND FUTURE CHALLENGES

With the development of IoT technology, the number of devices and their services deployed around the world is rapidly increasing, and it is important to select the best service that meets the user's needs from among them. In order to select the most appropriate service, many researches have been conducted incorporating the paradigm of reinforcement learning.

However, in conventional research on service selection, the QoS values to be considered for service selection are calculated by converting either a single criterion or multiple criteria into a single criterion. In this study, we believe that using distributed reinforcement learning (IMPALA), each element of QoS can be learned separately, enabling service selection that is more accurate and tailored to the user's needs. To demonstrate the usefulness of our method, we conducted an experiment to compare the method of learning to a single task by DQN with the method of learning to multiple tasks by our method. As a result of the experiment, it was confirmed that for all the elements of QoS, the best service was selected by learning to multi-task with our method rather than learning to single-task with DQN. Therefore, our method is more accurate and can select services that meet the individual needs of users.

This made it possible to select services more flexibly according to users' needs. However, although it is now possible to select one criterion from multiple QoS factors to suit the user's needs, it would be desirable to be able to select services considering multiple criteria from multiple QoS factors when considering real-world applications. For example, it would be desirable to select a service with high throughput and reliability. In the future, we would like to make this system applicable to the real world. Specifically, we believe that by adapting multi-objective genetic algorithms, we will be able to optimize for multiple criteria.

ACKNOWLEDGEMENT

This work has been supported by Grant-in-Aid for Scientific Research [KAKENHI Young Researcher] Grant No. 20K19931.

REFERENCES

Xiongnan Jin, Sejin Chun, Jooik Jung, and Kyong-Ho Lee. IoT Service Selection based on Physical Service Model and Absolute Dominance Relationship. 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications.

Singh, D., Tripathi, G., and Jara, A. J. A survey of Internet-of-Things: Future vision, architecture, challenges and services. in Proc. of the IEEE World Forum on Internet of Things (WF-IoT), pp. 287-292, IEEE, 2014.

Hongbing Wang , Jiajie Li , Qi Yu , Tianjing Hong , Jia Yan , Wei Zhao. Integrating recurrent neural networks and reinforcement learning for dynamic service composition. Future Generation Computer Systems Volume 107, June 2020, Pages 551-563.

H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, A. Bouguettaya. Adaptive Service composition based on reinforcement learning. International Conference on Service-Oriented Computing, Springer, 2010, pp. 92–107.

H. Wang, G. Huang, Q. Yu. Automatic hierarchical reinforcement learning for efficient large-scale service composition. 2016 IEEE International Conference on Web Services, ICWS, 2016, pp. 57–64.

H. Wang, X. Chen, Q. Wu, Q. Yu, Z. Zheng, A.Bouguettaya. Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition. Service-Oriented Computing, Springer, 2014, pp. 154–168.

Y. Lei, P.S. Yu. Multi-agent reinforcement learning for service composition. 2016 IEEE International Conference on Services Computing, SCC, 2016, pp. 790–793.

P. Kendrick, T. Baker, Z. Maamar, A. Hussain, R. Buyya, D. Al-Jumeily. An efficient multi-cloud service composition using a distributed multiagent- based, memory-driven approach. IEEE Trans. Sustain. Comput. (2019) 1–13.

Volodymyr Mnih,Adrià Puigdomènech Badia, Mehdi Mirza , Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, Koray Kavukcuoglu Asynchronous Methods for Deep Reinforcement Learning. Proceedings of The 33rd International Conference on Machine Learning, PMLR 48:1928-1937, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop 2013

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. Proceedings of the International Conference on Machine Learning (ICML) 2018.

Al-Masri, E., and Mahmoud Q. H. Investigating web services on the world wide web. 17th international conference on World Wide Web (WWW '08), pp.795-804.