# Accepted Manuscript

Task migration for mobile edge computing using deep reinforcement learning

Cheng Zhang, Zixuan Zheng

PII: S0167-739X(18)32967-4
DOI: https://doi.org/10.1016/j.future.2019.01.059
Reference: FUTURE 4752

To appear in: *Future Generation Computer Systems*

Received date : 26 November 2018
Revised date : 14 January 2019
Accepted date : 27 January 2019

Please cite this article as: C. Zhang and Z. Zheng, Task migration for mobile edge computing using deep reinforcement learning, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.01.059

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

# Task Migration for Mobile Edge Computing Using Deep Reinforcement Learning

Cheng Zhang[a,*], Zixuan Zheng[1]

[a]Department of Computer Science and Communications Engineering, Waseda University, Tokyo, Japan

**Abstract**

Mobile edge computing (MEC) is a new network architecture that puts computing capabilities and storage resource at the edges of the network in a distributed manner, instead of a kind of centralized cloud computing architecture. The computation tasks of the users can be offloaded to the nearby MEC servers to achieve high quality of computation experience. As many applications' users have high mobility, such as applications of autonomous driving, the original MEC server with the offloaded tasks may become far from the users. Therefore, the key challenge of the MEC is to make decisions on where and when the tasks had better be migrated according to users' mobility. Existing works formulated this problem as a sequential decision making model and using Markov decision process (MDP) to solve, with assumption that mobility pattern of the users is known ahead. However, it is difficult to get users' mobility pattern in advance. In this paper, we propose a deep Q-network (DQN) based technique for task migration in MEC system. It can learn the optimal task migration policy from previous experience without necessarily acquiring the information about users' mobility pattern in advance. Our proposed task migration algorithm is validated by conducting extensive simulations in the MEC system.

*Keywords:* service migration, mobile edge computing, deep reinforcement learning
*2018 MSC:* 00-01, 99-00

## 1. Introduction

With recent years' proliferation of various wireless computing applications such as wearable devices, virtual reality and smart-phones, it leads to an explosion of data. Therefore, cloud computing [1] was proposed to provide high quality of service (QoS) for the applications [2]. Meanwhile, some applications also require low latency and the central cloud computing servers need store and transmit tremendous data. These raised drawbacks of traditional cloud computing can be overcome by the Mobile Edge Computing (MEC) [3][4]. The basic idea of MEC is performing the related tasks of the application to the nearby edges of radio access network. The autonomous driving which is currently most concerned topic usually involving diverse kinds of tasking dealing (e.g. localization, road visualization and route planning). By using the MEC technology, cellular operators (the autonomous vehicles) can fulfill multiple tasks efficiently.

Still we are confronting many challenges regarding MEC technology [5]: (1) computation offloading; (2) allocation of computing resource and (3) mobility management. In this paper, we focus on the last one. Mobility management is the problem about how to guarantee the service continuity for the applications, if the user equipments (UEs, we also use "user" to point UE) roams from one network region to another. Several methods are provided to tackle with UEs' mobility problem. The most direct method is adapting the evolved node B (eNB) or small-cell eNB (SCeNB) transmission power for the offloaded applications. However, it can only be feasible for the UEs with a low mobility. As for the autonomous driving applications, the virtual machine (VM) migration (or equivalently, task migration) model should be introduced to guarantee the services. Even though task migration migration may spend more time and backhaul resources for the transmission between MEC nodes, it can bring us the benefits when the autonomous vehicles experience lower latency in the vehicles' neighbor and backhaul need not to be assigned for the transport. Therefore, it proves

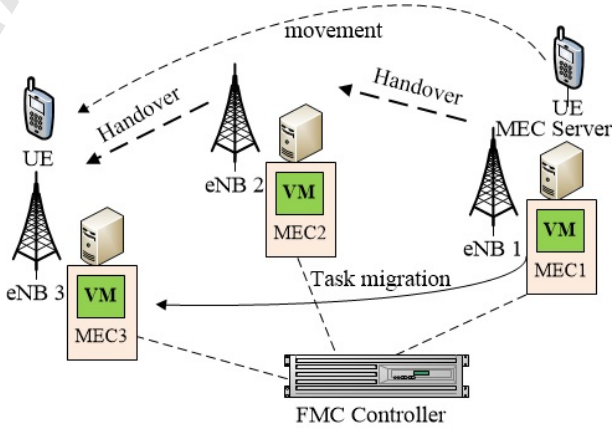


Figure 1: UE's movement makes its connection to base station from eNB 1 to eNB 3. The network should consider the problem whether to migrate UE's offloaded tasks from MEC 1 to MEC 3.

*Corresponding author
*Email address:* cheng.zhang@akane.waseda.jp (Cheng Zhang)

[1]Zixuan Zheng has done this research work independently with Dr. Cheng Zhang when she was a master course student of Department of Computer Science, The State University of New Jersey, United States

ACCEPTED MANUSCRIPT

to be a tough problem to how to maximize the migration gain under the restriction of migration cost.

The paper [6] formulated the problem about VM migration as a continuous time Markov Decision Processing (MDP) and design a policy to determine whether initiating the VM migration. The [7] also use MDP to deal with the VM migration but it always initiates the VM migration as long as the UE is bounded by the thresholds. The main drawback for [6] and [7] is that their proposed methods all are based on one-dimension models. Actually, the general setting for the VM migration involves 2D mobility. The paper [8] improve their VM migration process by a mobility prediction: estimate an advanced throughput when roaming throughout the network. However, to make satisfying prediction, it requires to acquire large amount of information. In [9], they enhance the VM migration decision processing in the field of minimizing the total cost during a period time by predicting future migration cost under a specific predicting error bound. But it will be hard for this offline algorithm to be applied in the real world because of its high complexity. In [10], it focuses on the load of individual MEC serves that may have impact on the VM migration. The authors propose online control algorithm to minimizing the overall transmission and reconfiguration costing during the VM migration. The paper [11] introduce a protocol architecture in dealing with the mobility management but for optimizing cloud access problem.

In this paper, considering the practical challenges faced by the autonomous vehicles with MEC technology: vehicles are with high level mobility and changing mobility pattern. We propose a method based on the deep reinforcement learning (DQN) to tackle the problem. Reinforced learning problems can be described as an agent learning continuously from interactions with the environment to achieve specific goals such as achieving maximum reward values. And at the initial state, the agent will rarely have ideas about decision making or the task. It receives a reward from its performance about the task. Theoretically, it will become easy for agent to make an optimal decision if agent explore whole states and acquire corresponding values. However, in the real word especially on the autonomous vehicles application, it seems impossible for the agent to experience all situations let alone the situations are not constant. Therefore, we propose method based on the DQN to generalize the experience rather than know all situations. DQN is a neural network version Q-learning. And the Q-function could be executed by the neural network, but it is not simply replaced. DQN provides a stable solution for deep value-based reinforcement learning problems.

There are main three contributions about this paper:

- We formulated the sequential decision problem that deal with where and when to migrate the tasks using reinforcement learning framework.
- We propose an algorithm based on a deep Q-learning neural network to solve task migration decision problem without knowing users' mobility pattern.
- Substantial simulations have been done to validated the proposed deep reinforcement learning based algorithm, and the results prove our method's usefulness and effectiveness.

We organize the rest sections of the paper as the following. The section 2 , we describe recent years' related study in the field of MEC and its application in the automatic driving. The section 3, defines our system model. Section 4 formulates the problem which can be solved by our proposed reinforcement learning based technique. Section 5 explicitly illustrates our proposed algorithm. Section 6 shows the simulation results to valid our algorithm Section 7 draws out the conclusion.

## 2. Related Work

Previous works enable the mobility of users via handover procedure once users choose to change the serving eNB/SCeNB when roaming through the networks in order to insure the QoS. A user offloads the computation tasks to a MEC server in one location, and when the user moves far away from the MEC server, the distance between the user and the MEC server becomes long. Then, the delay is large for the user to use the original MEC server. Therefore, task migration comes to be a novel approach to be utilized in the MEC service.

Task migration mainly mediates about two parts: the cost and gain for migration. On the one side, the cost represents the time consumption for the task migration, and the resources consumed in network backhaul in transmission of the tasks between the MEC servers. One the other side, the system can obtain gain from the view of lower delay between the the user and MEC server.

Taleb et al. in [12] proved the influence performance to the UE in VM migration. they employed the Markov chains to describe the mobility of UEs. In the case of without VM migration, the probability decreases as the distance between the eNB and the user increases, as well as higher detention. In contrast, one UE connects to the optimum mobile edge computing server with the lowest time delay while may pay more cost on the process of migration.

Taleb et al. extends their work to [13], in which applied a MDP based algorithm for optimizing task migration decisions, and studied their implementation on Software Defined Networking (SND) technologies, or the Locator/Identifier Separation Protocol.

Ksentini et al. in [6] proposed to use follow me cloud (FMC) to implement mobile edge computing, and they considered about an optimal threshold policy to decide migration or not. Each time the handover performed by the UE, the decision policy tries to perfume the execution judgment. If the cost is higher than the gain, the computation task is kept in the present MEC server instead of migration and vice versa.

Similarly, Sun et al. in [14] also researched the topic to obtain benefits from VM migration while considering the extra cost. They proposed a strategy to decide if the task migration is supposed to be executed or not. They proposed an mixed-integer quadratic programming based heuristic algorithm to solve the formulated problem since it is proven that the target problem is

ACCEPTED MANUSCRIPT

NP-hard. Moreover, the paper also gives the impact of parameter weighting on cost and gain.

Wang et al. in [7] formulated the task migration problem by Markov decision process, and then proposed a low time complexity threshold policy for task migration . The purpose is to reduce the whole cost during the VM migration as much as possible, where the cost is defined as the time delay for task migration. The results show that the optimal threshold is always better than never migrate or always migrate.

Not like the one dimension mobility model in [6][7], Wang et al. in [15] considered two dimension mobility model and real trace mobility trace. A sequential decision making problem is formulated and a optimal algorithm with time complexity $O(N^3)$ (N is the hops between the user and the MEC server that has user's task) is proposed.

Nadembega et al. in [8] improves the migration process via a mobility prediction. The scheme makes it possible to 1) evaluate the throughput between each MEC server and the user as the user roams throughout the network in advance; 2) evaluate the duration windows for the user performing handover, and 3) select the optimal MEC servers for task migration management scheme according to offered throughput. The problem is that it needs much information and does not consider the cost of migration, which is not generally used in reality.

Authors in [9] and [15] demonstrated the improvement on the mobility prediction and found the upper bound on the error of mobility prediction. Similar to [7][15], the objective is to minimize the sum cost of task migration. The paper proposed an offline algorithm to get the optimal sequence decision for a certain look-ahead window with size $K$, which stands for the time to which the cost forecasting is done. The task migration's performance strongly depends on the size of $K$. In the proposed scheme, only single-UE problem is formulated. Furthermore, [15] extended the paper to consider multiple UEs offloading multiple tasks to the MEC server. Since previous offline algorithm is too complex to be used in reality. They proposed an online approximation algorithm, with outperforming no task migration and keep migrating policy by about 32% and 50%.

Until now, all of the researches on task migration omitted the influence on scheduling of workload. Urgaonkar et al. in [10] firstly mentioned the issue of risk being affected by a load from independent MEC servers. It may suffer from some defects such as 1) a broad understanding of user mobility and the statistics of request arrival process is rather difficult, 2) the time complexity of the problem is very high, and 3) the optimal solution is needed to be recalculated due to any changes in mobility and arrival statistics. Authors in [10] evidenced a new methodology surmounting these disadvantages of the Lyapunov optimization framework. An online control algorithm that decides where the task should be migrated to minimize total transmission and reconfiguration costs is proposed.

The paper [16] studies the topic about how to minimize the overall time of migration, when the VM migration will be executed. The paper gives out an algorithm to reduce the sum of data transmission in migration. More data with low compression will be transmitted. In the opposite, high compression rate can result in a significant reduction in data transfers during migration, but with increases of time for data compression. Hence, the paper proposes that the compression ratio is dynamically adjusted according to the available bandwidth of the return trip and the workload of the MEC server. Simulation shows the superiority of the scheme with the dynamic adaptation to deal with the variations of available bandwidth in the process of task migration.

Authors in [11] demonstrate that proper task migration may not only reduce latency for task execution, but also increase system throughput. [12] proposed a locator/identifier separation protocol (LISP) based protocol for cloud access optimization. Once the user encounters a delay that exceeds the maximum tolerance threshold, the task starts to migrate to a new MEC server.

Qiu et al. in [13] proposed one optimal and one near-optimal heuristic algorithms to solve the real-time heterogeneous task assignment problem when different tasks have different execution time. Zhu et al. in [19] considered QoS requirements of tasks of distributed systems, and proposed a fault-tolerant scheduling algorithm with QoS needs on heterogeneous clusters. Qiu et al. in [20] studied green cloud. They proposed a genetic-based algorithm that can not only schedule and assign tasks to cores in the chip multiprocessor system, but also provides a phase-change memory multi-level cell configuration that balances the phase-change memory performance as well as the efficiency.

Different from aforementioned papers, our proposed DQN [21] based algorithm is aimed to solve the task migration problem for MEC without the knowledge of user's mobility pattern. Its effectiveness is shown in two aspects: (1) fast convergence rate (2) without depending on knowing user mobility in advance.

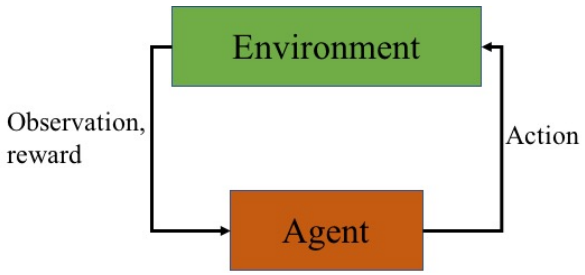


Figure 2: Illustration for reinforcement learning.

## 3. System Model

In this section, we describe the system by defining the necessary notations. We consider that there are $M$ distributed MEC servers equipped with cloud computing resource attached to base station eNB, and there is only one user moving from one place to another. The location set of eNBs is defined as $\mathcal{L}=\{1, ..., L\}$. It is assumed a time-slotted model as $t\in\mathcal{T}=\{1, 2, ..., T\}$. At time $t$, the user should connect to one of eNBs at some location index $l_t\in\mathcal{L}$. The user offloads its task to MEC server at location index $l_s\in\mathcal{L}$. User's current

ACCEPTED MANUSCRIPT

Table 1: Notations.

| Notation | Description |
|---|---|
| $\mathcal{M}$ | $\mathcal{M}$={1, ..., $M$}, MEC server set. |
| $\mathcal{T}$ | the time slot set. |
| $t$ | $t \in \mathcal{T}$, the time slot. |
| $\mathcal{L}$ | $\mathcal{L}$={1, ..., $L$}, location set. |
| $l_t$ | $l_t \in \mathcal{L}$, user's location index at time $t$. |
| $l_s$ | $l_s \in \mathcal{L}$, user's location index where he migrate his task to the MEC sever. |
| $u_t$ | $u_t = \|l_t - l_s\|$ is the distance between user's current location ($l_t$) and user's location ($l_s$) where he offloaded his task to MEC server. |
| $\boldsymbol{s}_t$ | $\boldsymbol{s}_t = l_t$, state of user. |
| $a_t$ | FMC controller's action at time $t$. |
| $\mathcal{A}$ | FMC controller's action set. |
| $a^0$ | $a^0$ means the action of no task migration. |
| $a^1$ | $a^1$ means the action of task migration. |
| $\beta$ | The parameter that measure the cost of one hop between eNB. |
| $c_t(s_t, a_t)$ | cost at state $s_t$ when choose action $a_t$ at $t$. |
| $C(s_t)$ | cost of task migration. |
| $H$ | replay memory. |
| $Q$ | state-action function. |
| $\pi$ | FMC controller's *policy*. |
| $\Pi$ | FMC controller's *policy* set. |
| $r_t(s_t, a_t)$ | FMC controller's reward at time $t$. |
| $\phi_t(s_t)$ | function at time $t$ about mapping from one state $s_t$ to an action decision. |
| $\phi_t^*$ | optimal action decision at time $t$. an action decision. |
| $\gamma$ | the discount rate |
| $\alpha$ | the learning rate |
| $\epsilon$ | the trade-off between *exploitation* and *exploration* |

location index $l_t$ is not necessary equal to the location $l_s$ where he/she offloaded hisher task due to user's movement. We define $u_t = |l_t - l_s|$ as the distance between user's current location and user's location where he offloaded his task to MEC server. Whenever user moves to a new location, FMC controller have to make a decision whether to migrate the task from MEC server at location $l_s$ to the MEC server at current location $l_t$.

We utilize the reinforcement learning technique for FMC controller to make the task migration decision.

The general idea of reinforcement learning is that by gradually interaction with the *environment* (shown as Fig. 2), agent will be able to make better decisions using the past *experience*. At the initial state, the agent rarely has the idea about how to take *action* or even what the task is. Then according to agent performance of this task it will obtain a *reward* as the feedback. Theoretically, agent could make the optimal decisions if it can explore whole *states* and obtain corresponding values of its action. Nevertheless, it seems impractical to experience the total situations. When some unknown situations occur, the agent could have difficulties making good decision because it

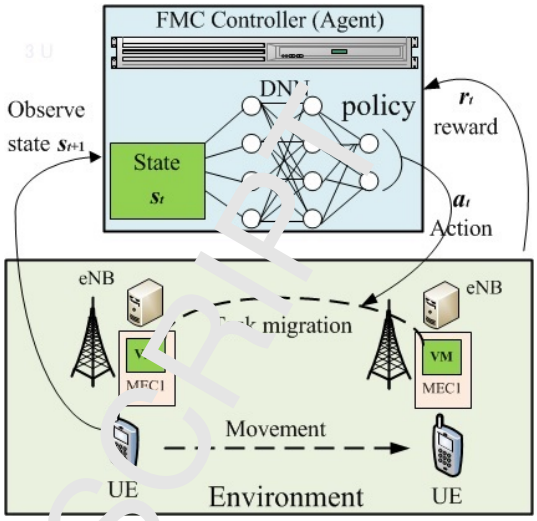


Figure 3: An deep Q-network based modeling.

lacks the abilities of generalizing the past experiences. Because of this, DQN [21] was introduced to deal with generalizing agent experience problem. DQN can help agent predict Q-value based on general Q-learning [22] by utilizing deep neural networks (DNN) [23]. DQN will establish maps between the agent different states actions to its corresponding reward values shown as Fig. 3. Therefore, agent could make optimum task offloading decision by directly choosing the action that achieve the highest Q-value.

We formally define the elements in reinforcement learning model for our system as follows.

- *agent*: The agent is FMC controller. The FMC controller has all the information of eNBs and corresponding MEC servers. The FMC controller also knows which eNB the user is associated with and the MEC server that the user has offloaded the tasks. The FMC controllers determine whether migrated UE's task on MEC servers.

- *state*: The state at time slot $t \in \mathcal{T}$ is defined as $s_t = u_t$. Please note that $u_t$ is absolute difference between user's current location index and the location index where he offloaded his task to MEC server. User's movement changes the state $s_t$. If $s_t = 0$, it means that user is still in the same location as the MEC server with the user's offloaded tasks.

- *action*: The agent, FMC controller, takes action to migrate user's tasks to the current MEC server from original MEC server, or do not migrate. The action at time $t$ is denoted as $a_t \in \mathcal{A} = \{a^0, a^1\}$, where $\mathcal{A}$ is the action set, $a_0$ means the action of no task migration, and $a_1$ means the action of task migrations.

- *reward*: At each time $t$, FMC controller gets reward based on its action, and current state. The reward is defined as the difference between the quality of service and the migration cost. The quality of service is determined by $u_t$,

ACCEPTED MANUSCRIPT

the difference between user's current location index and the location index where he offloaded his task to MEC server.

$$q(s_t) = D - \lambda u_t \tag{1}$$

where $D$ is the maximal quality the user can enjoy, and it is a decreasing function of $u_t$, which means that the user's quality decreases when the user gets far away from the offloaded MEC server. Then, task migration become necessary. As for the cost needed to migrate a task from one MEC server to another one: (i) the cost of converting a task to be executed on a new MEC server; (ii) the cost needed for transferring the task itself over the network. Here, we assume that different MEC server has the same VM environment, and task can be migrated among MEC servers without conversion. Formally, the cost of FMC controller is defined as follows.

$$c_t(s_t, a_t) = \begin{cases} 0 & \text{if } q_t = a^0 \\ C(s_t) & \text{if } a_t = a^1 \end{cases} \tag{2}$$

where the function $C(s_t)$ determines the time delay of task migration. It dependents on $u_t$, one simple implementation of this function is $C(s_t) = \beta u_t$, where $\beta$ is a parameter that measure the cost of one hop task migration between eNB. Therefore, the reward $r_t(s_t, a_t)$ is defined as follows in Eq. (3)

$$r_t(s_t, a_t) = q(s_t) - c_t(s_t, a_t) = \begin{cases} q(s_t) & \text{if } a_t = a^0 \\ D - C(s_t) & \text{if } a_t = a^1 \end{cases} \tag{3}$$

## 4. Problem Formulation

FMC controller aims to maximize the total sum reward of these time epochs through deciding the actions of every time epoch from the first to the last one. *policy* is defined as the action sequences, the details shown as the following.

**Definition 1.** *The FMC controller's policy is about taking actions from the time $t = 1$ to $t = \infty$ defined as the following Eq. (4)*

$$\pi = \left\{ \phi_t(s_t),\ \forall\ t \in \{1, ..., \infty\} \right\} \tag{4}$$

*here $\phi_t(s_t)$ is a function at time t about mapping from one state $s_t$ to an action decision.*

The $\Pi$ is to represent the set of $\pi$. If the task migration policy $\pi$ is employed, the corresponding state is expressed as $s_t^\pi$.

The goal of the FMC controller is to reduce total expected reward from the time $t = 1$ to $t = \infty$ with an optimal policy $\pi^*$ ( shown as Eq. (5))

$$\max_{\pi \in \Pi} E_{s_1}^{\pi} \left[ \sum_{t=1}^{\infty} \gamma r_t(s_t^\pi, a_t) \right] \tag{5}$$

where $\gamma$ is the discount-rate and $\gamma \in [0, 1]$.

The *optimal policy* represents the optimum solution to the defined problem in the Eq. (5). Note that in order to achieve the global optimum, the action at every particular time $t$ should consider about the both current and future expected cost. In other words, at time $t$ the optimal action for the problem shown as Eq. (5) may not be the best action for its current time. The objective function aims at minimizing the total cost including the problem Eq.(5) and delay cost caused by the migration.

## 5. DQN Based Task Migration Algorithm

The reinforcement learning model is that an agent can make optimal actions by gradually interact with the unknown environment to obtain information. In this paper, the agent is FMC controller. The state is the distance between the optimal MEC server and original MEC server that have user's offloaded task. The action is whether to choose task migration from the original MEC server to the optimal MEC server or not. There is a reward related to FMC controller's action selection for every time epoch. The FMC controller's object is to get the maximum total reward of the whole time epochs.

There is one crucial part in the reinforcement learning: it has to make tradeoff between the between *exploration* and *exploitation*.

*exploitation*: Based on current known information, the best decision is made by FMC controller. *exploration*: Explore unknown areas, such as actions that are not executed in this state before a state is executed. So the purpose of the FMC controller using exploitation and exploration is to obtain a strategy with the highest long-term benefit, which may have a loss to the short-term reward. If there are too many exploitations, then the model is more likely to fall into local optimum, but there are too many explorations and the model converges too slowly.

At the beginning, the FMC controller has no experience. Hence, it need *explore* to obtain the information of reward caused by taking some actions in the states. As long as the FMC controller has the experiences, it will exploit the known information of the states while keep *exploration*. In **Algorithm 1**, the parameter $\epsilon$ is set as the trade-off between *exploitation* and *exploration*.

Our proposed reinforcement learning is model-free. This strategy evaluation is to calculate each state value of a strategy without knowing the transition probability and reward function of the Markov decision process. There are two main algorithms to fulfill the goal, one is Monte Carlo algorithm and the other is temporal difference (TD) learning algorithm [22]. Sometimes the reinforcement learning problems are continuous. Therefore, this Monte Carlo algorithm does not apply. In order to solve this problem, TD has been proposed which utilizes the Markov property and only utilizes the next information. The TD allows the system to explore according to the policy guidelines, and updates the state value at each step of the exploration. The update formula is using the *Bellman equation* in Eq. (6) [24]

$$Q_t^*(s_t, a_t) = \mathbb{E}_{s_{t+1}} \left[ r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) | s_t, a_t \right] \tag{6}$$

here $\gamma$ represents a factor between (0,1). It is usually called discount factor. Q-value is a function of state and action. In the

ACCEPTED MANUSCRIPT

**Algorithm 1**: Task Migration Algorithm for FMC

1: Replay memory $H$ initialization.
2: $Q$ function initialization with random parameters $\theta$
3: $\bar{Q}$ function initialization with parameters $\theta^-$
4: $t \leftarrow 1$; $l_1 \leftarrow$ random value from $\mathcal{L}$.
5: Set $s_1 = l_1$
6: **while** $t \leq T$:
7: $l_t$ is fetched by FMC controller from eNB
8: generate a random number $rnd$ in [0,1]
9: **if** $rnd > \epsilon$ :
10: Select action $a$ according to Eq. (8)
11: **else**:
12: Select action $a$ randomly
13: **end if**
14: $s_{t+1} \leftarrow l_t$
15: Calculate $r_t(s_t, a_t)$ by Eq. (3)
16: Store experience $(s_t, a_t, r_t, s_{t+1})$ in $H$
17: Sample minibatch of $(s_j, a_j, r_j, s_{j+1})$ randomly from $H$
18: **if** $j + 1$ is the final one:
19: Set $z_j = r_j$
20: **else:**
21: Set $z_j = r_j + \gamma \max_{a_{j+1}} Q_t(s_{j+1}, a_{j+1}; \theta_j^-)$
22: **end if**
23: Gradient descent step is executed on $(z_j - Q_t(s_t, a_t; \theta)^2$ by $\theta$.
24: Reset $\bar{Q}$=$Q$ in every C steps
25: $t \leftarrow t + 1$
26: **end while**

basic Q-learning, optimum policy could simple be got[illegible] from the best Q-value, $Q_t^*(s_t, a_t)$, that is expressed in th[illegible] following Eq. (7)

$$\phi_t^* = \arg\max_{a_t \in \mathcal{A}} Q_t^*(s_t, a_t) \tag{7}$$

The mobility of the user is the transition probability in our proposed Q-learning model which assumes unknown. The pre-requirement in [25] will not affect the offloading algorithm based on the Q-learning. Still two problems remain as in the paper [26] .

- (i) Great amount of states makes it tough to directly implement the Q-learning. One common solution is to established a two-dimension table which can reserve the Q-value. The row and column of the table indicates actions and states. However, this method also becomes infeasible when the sizes of states and actions increases. As shown in [13], the defined Markovian model also faced a state space explosion problem, especially when there are a large number of locations. The solution in [13] is to this problem is to reduce the state space by aggregating states that show the same behavior.
- (ii) The algorithm converges at decreasing rate if the user experience more and more states and finally proves to be very slow. Besides, the agent lacks generalization ability from the past experiences to some unknown states.

Hence, in this paper, using algorithm based on DQN [21] to deal with the problem defined in Eq.(5). DNN [23] can help FMC controller generalize its past experience to predict Q value about unexplored states. Hence, great amount of states is not a problem for our DQN based algorithm.

In DQN, an approximator $Q_t(s, a; \theta)$ in the action-value function is designed for estimation with parameters $\theta$. Then FMC controller's policy can be achieved by the following Eq.(8)

$$\phi_t^* = \arg\max_{a_t \in \mathcal{A}} Q_t(s_t, a_t, \theta) \tag{8}$$

Q-network contains an approximator with weights $\theta$. We train the Q-network model during each iteration $i$ so that lessen the mean-squared error(MSE) in mentioned Bellman equation by changing the value of parameters $\theta_i$. Here, the optimal objective values in Eq.(6), $r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1})$, are substituted by the expected object values

$$z = r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}; \theta_i^-) \tag{9}$$

here the parameter $\theta_i^-$ represents the previous iterations. The MSE or the loss function is shown as in Eq.(10).

$$L_i(\theta_i) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\left[(z - Q_t(s_t, a_t; \theta_i))^2\right] \tag{10}$$

Using differentiating to get the loss function's gradients.

$$\begin{aligned} &\nabla_{\theta_i} L_i(\theta_i) = \\ &\mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\left[\left(z - Q_t(s_t, a_t; \theta_i)\right) \nabla_{\theta_i} Q_t(s_t, a_t; \theta_i)\right] \end{aligned} \tag{11}$$

The gradient $\nabla_{\theta_i} Q_t(s_t, a_t; \theta_i)$ guides loss function in Eq.(10) to be reduced at a feasible direction. Therefore, update the paremeter as the following the Eq.(12)

$$\theta_{i+1} = \theta_{i+1} + \alpha \nabla_{\theta_i} L_i(\theta_i) \tag{12}$$

here $\alpha$ denotes the learning rate with values among (0,1). Algorithm 1 shows our proposed DQN based task migration algorithm. As indicated from line 16 in Algorithm 1, the experience of the FMC controller's $(s_t, a_t, r_t, s_{t+1})$ is reserved in replay memory, thus without needing transition probabilities.

## 6. Performance Evaluation

In this section, numeric analysis is conducted to validate the effectiveness of our our *proposed DQN* based task migration algorithm. We compare the *proposed DQN* with the following two other cases: (i) the case when dynamic programming based algorithm (*DP*) is applied [6]; (ii) the case when no task migration is applied (*No Migration*).

Please note that the dynamic programming based algorithm requires transition probability. The dynamic programming based algorithm can obtain optimal solutions if the transition probability is correct. However, it is difficult to get the ground truth transition probability. We would use several "incorrect" transition probability because user's ground truth transition probability usually contains noise. On the other hand, our deep

ACCEPTED MANUSCRIPT

Table 2: Parameters in the simulation.

| Parameters | Value |
|---|---|
| $L$ | 36 |
| $\mathcal{T}$ | 80 |
| $M$ | 18 |
| time slot | 1 seconds |
| $\Pr(l\|l)$ | 0.5 |
| $\Pr(l_{t+1}\|l_t)$ | $\frac{1-\Pr(l\|l)}{\#\text{neighbour locations}}$ |
| D | 36 |
| $\beta$ | 10 |
| $\lambda$ | 18 |

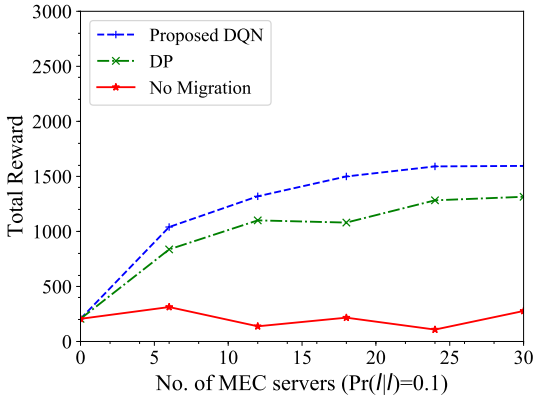


Figure 4: Total reward vs. No. of MEC servers when $\Pr(l|l) = 0.1$.

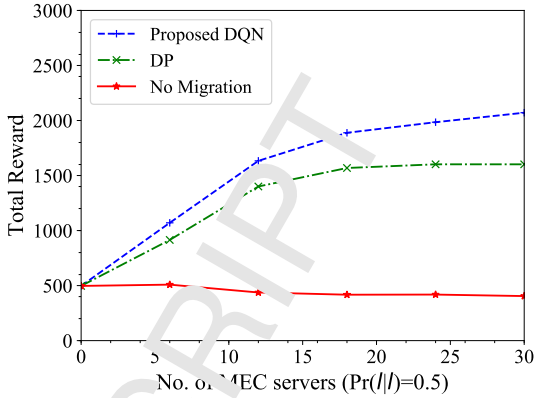


Figure 5: Total reward vs. No. of MEC servers when $\Pr(l|l) = 0.5$.

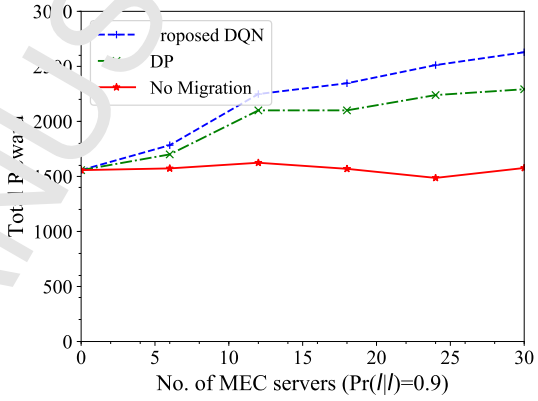


Figure 6: Total reward vs. No. of MEC servers when $\Pr(l|l) = 0.9$.

Q-learning based algorithm has no requirement for transition probability.

We use a four by four grid area in the simulation and the number of locations is 16. At each locations, there are eNB and corresponding MEC server. The user randomly walks in the area with assumption that the length of time slot is 5 seconds. Also set the parameter $\epsilon$ to 0.05. The transition probability from location $l$ to $l'$ is denoted as $\Pr(l'|l)$. For instance, $\Pr(l|l) = 0.5$ represents that the user will have 0.5 probability at the same place during the time period $t$ to $t'$.

Also assume that user has the same intention moving to the nearby areas. In other word, the probability of nearest locations for user to choose next is the same. Therefore, use $\Pr(l_{t+1}|l_t) = \frac{1-\Pr(l|l)}{\#\text{ of neighbour locations}}$. This transition probability is used for DP algorithm, while our proposed method does not require it.

We perform the simulation by the Python 2.7 version.

Figure 4 shows how total rewards alter with the different number of MEC servers among our *Proposed DQN* algorithm, *DP* algorithm, and *Non Migration* algorithm when $\Pr(l|l) = 0.1$. Total reward of both *DP* algorithm and *Proposed DQN* algorithm increase with the number of MEC servers deployed. The reason behind the phenomena is that it is much easier for user to migrate his/her tasks to a new optimal MEC server when there are many more MEC servers, and the reward for a new optimal MEC server is much higher. The total reward for our *Proposed DQN* algorithm is highest and *Non Migration* algorithm' reward is lowest, while the total reward of *DP* algorithm is between that of *Non Migration* algorithm and *Proposed DQN* algorithm. The reason that *Non Migration* algorithm's total reward is lowest is that user does not always connect to an optimal MEC server if there is no task migration. While for our *Proposed DQN* algorithm, optimal task migration decision is held based on the long term reward for the user. An interesting phenomena is that the total reward of three algorithms is zero when the number of MEC server is zero. The reason is that both *Proposed DQN* algorithm and *DP* algorithm degenerate to *Non Migration* algorithm when there is no MEC server deployed.

Figure 5 and Figure 6 show how total rewards alter with the different number of MEC servers among *Proposed DQN* algorithm, *DP* algorithm, and *Non Migration* algorithm, when $\Pr(l|l)$ is 0.5 and 0.9, respectively. Both of Figure 5 and Figure 6 share the same characteristics with Figure 4. The overall total reward in Figure 6 is much higher than that of Figure 5, and the overall total reward in Figure 5 is much higher than that of Figure 4. The reason that when the user is much more dynamic (when $\Pr(l|l)$ is low), the probability for task migration is higher. Then, the reward decreases since there is cost for task migration.

Figure 7 shows how total reward alters with the different $\Pr(l|l)$ when the number of MEC servers is 6. Please note that $\Pr(l|l)$ measures the dynamicity of a user. When $\Pr(l|l)$ is low, it means that the user is dynamic and tend to move. When

ACCEPTED MANUSCRIPT

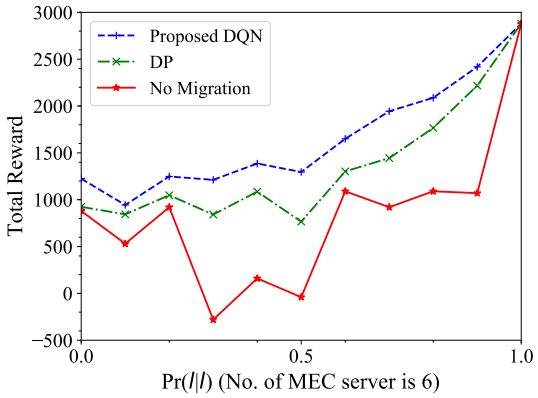


Figure 7: Total reward vs. Pr($l|l$) when the number of MEC servers is 6.

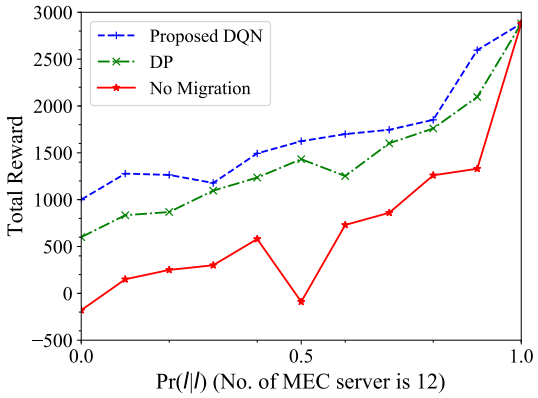


Figure 8: Total reward vs. Pr($l|l$) when the number of MEC servers is 12.

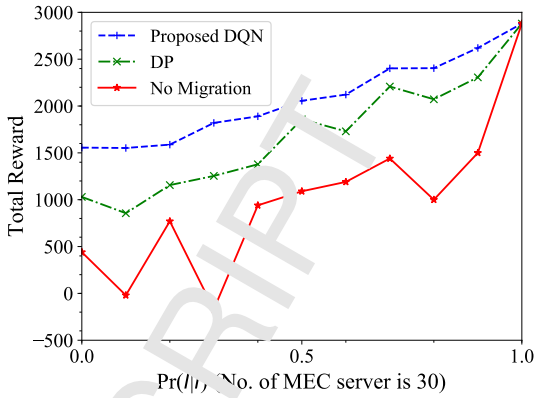


Figure 9: Total reward vs. Pr($l|l$) when the number of MEC servers is 30.

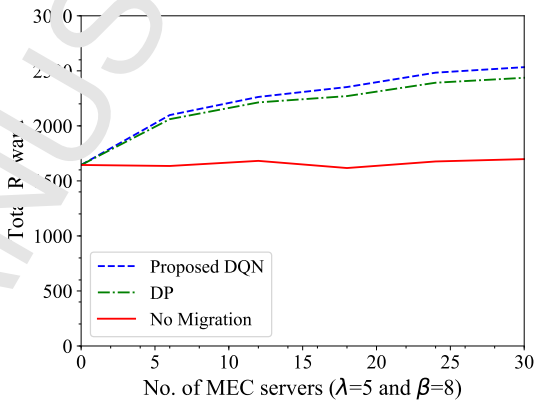


Figure 10: Total reward vs. No. of MEC servers when $\lambda < \beta$.

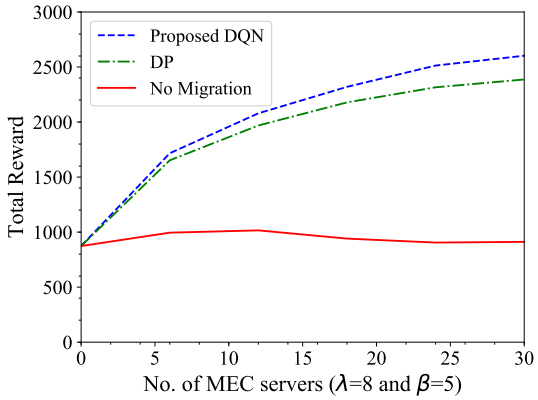


Figure 11: Total reward vs. No. of MEC servers when $\lambda > \beta$.

Pr($l|l$) is high, it means that the user tends to stay in the same place, and less dynamic. Total reward of both *DP* algorithm and *Proposed DQN* algorithm increase with Pr($l|l$). The reason is that less task migration is needed when user tends to stay in the same place (when Pr($l|l$) is high). Less task migration means less cost for task migration, then much higher reward can be achieved. An interesting phenomena is that the total reward of three algorithms is the same when Pr($l|l$) is 1. The reason is that both when user stay in the same place all the time (when Pr($l|l$) is 1), task migration is no longer needed for the user. Then there is no cost for task migration no matter what the task migration algorithm is.

Figure 8 and Figure 9 show how total rewards alter with the different Pr($l|l$) when the number of MEC servers is 12 and 30, respectively. Both of Figure 8 and Figure 9 share the same characteristics with Figure 7. The overall total reward in Figure 9 is much higher than that of Figure 8, and the overall total reward in Figure 8 is much higher than that of Figure 7. The reason that when the there are more MEC servers, it is easier for user to migrate to a MEC server to increase the quality of service. Then, the reward increase when there are more MEC servers deployed.

The quality decrease rate parameter $\lambda$ and the cost migration parameter for task migration $\beta$ are also very important for the algorithms. We evaluate how the total reward of *Proposed DQN* algorithm, *DP* algorithm, and *Non Migration* algorithm changes with different number of MEC servers in the case of $\lambda < \beta$ in Figure 10, and $\lambda > \beta$ in Figure 11. $\lambda < \beta$ means that the quality of service drops slower than the cost of task migration, and $\lambda > \beta$ means that the quality of service drops faster than the cost of task migration. Then in the case of no task migration in *Non Migration* algorithm, $\lambda < \beta$ can achieve higher reward compared to $\lambda > \beta$, which is the exactly the results showed in 10 and Figure 11. When the number of MEC server is relative

ACCEPTED MANUSCRIPT

low, total rewards of *Proposed DQN* algorithm, *DP* algorithm are much higher in the case of $\lambda < \beta$ than in the case $\lambda > \beta$. The reason is that less MEC servers means that less task migration is held, then the high cost for task migration can be saved.

## 7. Conclusion

In this paper, we studied task migration problem in which a FMC controller has to determine where and when to migrate its task from one MEC server to another. We proposed to use a deep reinforcement learning based algorithm with the goal that maximizing the user's total reward. It can work even without knowing the users' mobilities in advance. The simulation results of the MEC system validate our proposed method. In the process of task migration, the related data should be encrypted to protect users' data as security problem has become a severe problem nowadays [27]. In the future work, we may adopt data encryption for UEs' tasks.

## References


[1] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1 (1) (2010) 7–18. doi:10.1007/s13174-010-0007-6.

[2] Y. Zhang, M. Qiu, C. Tsai, M. M. Hassan, A. Alamri, Health-CPS: Healthcare cyber-physical system assisted by cloud and big data, IEEE Systems Journal 11 (1) (2017) 88–95. doi:10.1109/JSYST.2015.2460747.

[3] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (1) (2018) 450–465. doi:10.1109/JIOT.2017.2750180.

[4] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing 13 (18) (2011) 1587–1611. doi:10.1002/wcm.1203.

[5] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys Tutorials 19 (3) (2017) 1628–1656. doi:10.1109/COMST.2017.2682318.

[6] A. Ksentini, T. Taleb, M. Chen, A markov decision process-based service migration procedure for follow me cloud, in: 2014 IEEE International Conference on Communications (ICC), 2014, pp. 1350–1354. doi:10.1109/ICC.2014.6883509.

[7] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, K. K. Leung, Mobility-induced service migration in mobile micro-clouds, in: 2014 IEEE Military Communications Conference, 2014, pp. 835–840. doi:10.1109/MILCOM.2014.145.

[8] A. Nadembega, A. S. Hafid, R. Brisebois, Mobility prediction model-based service migration procedure for follow me cloud to support qos and qoe, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6. doi:10.1109/ICC.2016.7511148.

[9] S. Wang, R. Urgaonkar, K. Chan, T. He, M. Zafer, K. K. Leung, Dynamic service placement for mobile micro-clouds with predicted future costs, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 5504–5510. doi:10.1109/ICC.2015.7249199.

[10] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, K. K. Leung, Dynamic service migration and workload scheduling in edge-clouds, Performance Evaluation 91 (2015) 205 – 228, special Issue: Performance 2015. doi:10.1016/j.peva.2015.06.013.

[11] S. Secci, P. Raad, P. Gallard, Linking virtual machine mobility to user mobility, IEEE Transactions on Network and Service Management 13 (4) (2016) 927–940.

[12] T. Taleb, A. Ksentini, An analytical model for follow me cloud, in: 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 1291–1296. doi:10.1109/GLOCOM.2013.6831252.

[13] T. Taleb, A. Ksentini, P. Frangoudis, Follow-me cloud: When cloud services follow mobile users, IEEE Transactions on Cloud Computing (2018) 1–1doi:10.1109/TCC.2016.2525987.

[14] X. Sun, N. Ansari, PRIMAL: PRofit Maximization Avatar pLacement for mobile edge computing, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6. doi:10.1109/ICC.2016.7511131.

[15] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, K. K. Leung, Dynamic service placement for mobile micro-clouds with predicted future costs, IEEE Transactions on Parallel and Distributed Systems 28 (4) (2017) 1002–1016. doi:10.1109/TPDS.2016.2604814.

[16] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, M. Satyanarayanan, Adaptive vm handoff across cloudlets, 2015.

[17] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, The Locator/ID Separation Protocol (LISP), RFC 6830 (Jan. 2013). doi:10.17487/RFC6830. URL https://rfc-editor.org/rfc/rfc6830.txt

[18] M. Qiu, E. H. M. Sha, Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems, ACM Transactions on Design Automation of Electronic Systems 14 (2) (2009) 25:1–25:30. doi:10.1145/1497561.1497568.

[19] X. Zhu, C. Qin, M. Qiu, Qos-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters, IEEE Transactions on Computers 60 (6) (2011) 800–812. doi:10.1109/TC.2011.68.

[20] M. Qiu, Z. Ming, J. Li, K. Gai, Z. Zong, Phase-change memory optimization for green cloud with genetic algorithm, IEEE Transactions on Computers 64 (12) (2015) 3528–3540. doi:10.1109/TC.2015.2401257.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533. doi:10.1038/nature14236.

[22] P. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.

[24] R. Bellman, Dynamic Programming, Princeton University Press, 1957.

[25] C. Zhang, B. Gu, Z. Liu, K. Yamori, Y. Tanaka, Cost- and energy-aware multi-flow mobile data offloading using markov decision process, IEICE Trans. Commun. E101-B (3). doi:10.1587/transcom.2017NRP0014.

[26] C. Zhang, B. Gu, Z. Liu, K. Yamori, Y. Tanaka, A reinforcement learning approach for cost- and energy-aware mobile data offloading, Proc. 16th Asia-Pacific Network Operations and Management Symposium (APNOMS 2016), Kanazawa, Japan (2016) 1–6doi:10.1109/APNOMS.2016.7737203.

[27] K. Gai, M. Qiu, Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers, IEEE Transactions on Industrial Informatics 14 (8) (2018) 3590–3598. doi:10.1109/TII.2017.2780885.

ACCEPTED MANUSCRIPT

**Cheng Zhang** received his Ph.D. degree from Waseda University, Tokyo, Japan, in 2015. From 2008 to 2015, he was a research engineer at Sony Digital Network Applications, Japan and HGST Japan, Inc. (ex Hitachi Global Storage Technologies), where he researched and developed control algorithms for image stabilization module of Sony digital camera, and servo control algorithms for next generation high capacity HDD. He is currently an assistant professor of Graduate Program for Embodiment Informatics (Program for Leading Graduate School) at Graduate School of Fundamental Science and Engineering, Waseda University. His research interests include machine control algorithm, embedded software, game theory, network economics and machine learning. He received the IEICE Young Researcher's Award in 2013. He is a member of IEEE and IEICE.

**Zixuan Zheng** received her B.E degree in Information Communication Engineering from University of Electronic Science and Technology of China, UESTC in 2017, and now she is pursing master degree of computer science in the State University of New Jersey. Her research interests include machine learning (deep learning, reinforcement leaning), computer vision and related applications.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

# Task Migration for Mobile Edge Computing Using Deep Reinforcement Learning

- Mobile edging computing (MEC) is an effective way to reduce the computation time for users.
- Task migration is necessary for high mobility users.
- Deep reinforcement learning is effective for task migration in MEC.

ACCEPTED MANUSCRIPT