Learning Based Mobility Management under Uncertainties for Mobile Edge Computing

Jingrong Wang[†], Kaiyang Liu[†*], Minming Ni[‡], Jianping Pan[†]
[†]Department of Computer Science, University of Victoria, Victoria, Canada
[*]School of Information Science and Engineering, Central South University, Changsha, China
[‡]State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University
Email: {jingrongwang, liukaiyang, pan}@uvic.ca, mmni@bjtu.edu.cn

***Abstract*—Mobile edge computing (MEC) offloads computation-intensive applications and overcomes the long latency by pushing data traffic towards the network edges. With base stations (BSs) densely deployed in a hot-spot area to improve user experience, mobile user equipments (UEs) have multiple choices to offload tasks to edge servers by jointly considering both the channel condition and the computing capacity. However, precise full system information is hard to be synchronized between BSs and UEs for mobility management decision making. In this paper, a Q-learning based mobility management scheme is proposed to handle the system information uncertainties. Each UE observes the task delay as an experience and automatically learns the optimal mobility management strategy through trial and error. Simulations show that the proposed scheme manifests the superiority in dealing with the uncertainties. Compared with the traditional received signal strength-based handover scheme, the proposed scheme reduces the task delay by about 30%.**

***Index Terms*—Mobility management, mobile edge computing, handover decision, Q-learning**

I. Introduction

Mobile edge computing (MEC) has been considered as a promising solution for next-generation mobile networks where various kinds of resource-hungry and computation-intensive applications emerge, such as face recognition and virtual reality [1]. MEC pushes the data traffic towards the network edges by distributing the computing capabilities closer to the mobile user equipments (UEs), such as the base stations (BSs) equipped with the edge servers. As the transmission of data between UEs and BSs does not need to go through the core networks, it overcomes long service latency [2].

With BSs densely deployed in a hot-spot area to increase the system capacity and improve user experience, mobile UEs have multiple choices to offload tasks to edge servers. Driven by the dense deployment, handover decision has gradually evolved from the traditionally cell-centric to a more flexible user-centric mobility management [3], [4]. In the traditional handover scheme, when more and more UEs move toward the region of a certain BS, UEs keep connecting to the BS based on the received signal strength (RSS), making certain BSs overloaded. However, UEs could have the opportunities to offload the tasks to other neighboring BSs whose resources remain unused. Thus, not only the channel condition but also the computing capacity should be considered when associating a UE with the appropriate BS. Since UEs connect to different BSs due to the location alteration, user mobility aggravates the rapid alteration of the system, making it hard to obtain an accurate full network information for an appropriate UE-BS association. Therefore, faced with the performance deterioration and system uncertainties, mobility management is still of great importance and should be redesigned in MEC scenarios.

The majority of the existing work addressed mobility issues such as whether and where to hand over. Demarchou *et al.* [5] proposed a user-centric handover scheme where a handover occurs according to UE's future location predicted by the trajectory and velocity. Hasan *et al.* [6] adjusted handover parameters depending on the overloaded BSs and adjacent BSs by employing an adaptive threshold to determine the overloaded BSs. The current literature is based on full system information, such as predictable UE's mobility pattern and BS side information. To handle the system uncertainties where the network conditions, BS workloads and the future information are unknown, pioneering studies worked on reinforcement learning where each UE interacts with the environment and identifies the optimal action-selection policy [7]–[10]. However, how to adjust the handover criteria in MEC scenario to achieve a better quality of service (QoS) is not well studied in the literature. This motivated us to propose a generic solution for mobility management to handle the system information uncertainties in MEC.

In this paper, a Q-learning based mobility management scheme is proposed to handle the uncertainties of the MEC-enabled networks. In order to shorten the service latency, each UE makes handover decisions based on the observed task delay from the environment in the past and selects the action with the highest Q-values in the current state. Each UE keeps exploring different BSs with a probability ϵ as well as trying to connect to the optimal BS in the current experience. When UEs move to another BS, Q-values are updated to keep up with the dynamic system.

The rest of the paper is organized as follows. The communication model and the computation model are introduced in Section II. The Q-learning based mobility management scheme is proposed in Section III. The performance of the proposed algorithm and the impact of key parameters are evaluated in Section IV. Finally, the conclusion and future work are summarized in Section V.

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

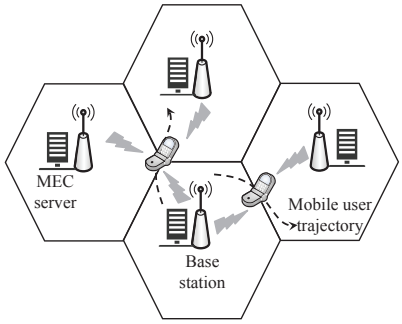


Figure 1. The architecture of MEC with UEs.

II. System Model

In this section, the user mobility model and the task generation model are first introduced. Then, the communication model and the computation model ensue.

A. User Mobility and Task Generation

As shown in Fig. 1, BSs $\mathcal{N} = \{1, 2, ..., N\}$ equipped with computing capabilities are densely deployed in a hexagonal grid. UE $i \in \mathcal{I} = \{1, 2, ..., I\}$ can access the edge server through a cellular BS and then a wired connection [11]. UEs are moving around randomly.

As shown in Fig. 2, each UE alternates between the idle state and the execution state. The task intensity p per hour per UE is the rate of the transition from the idle state to the execution state, and μ otherwise. During the execution state, the task can be executed locally or offloaded to the edge server. After executing the task, the UE will return to the idle state.

B. Communication Model

The path loss (in dB) between BS n and UE i at time t can be expressed as

$$PL_{t,n,i} = PL(d_0) + 10\theta \log\left(\frac{d_{t,n,i}}{d_0}\right), i \in \mathcal{I}, n \in \mathcal{N}, \quad (1)$$

where $d_{t,n,i} \geq d_0$ is the distance between BS n and UE i at time t, θ is the path loss exponent, and d_0 is the reference distance.

Based on the path loss model, the RSS (in dBm) from BS n for UE i at time t is calculated as

$$\begin{aligned} P^{\mathsf{U}}_{t,n,i} =& P^{\mathsf{U}} - PL_{t,n,i} \\ & - X_\sigma - 10\log_{10} h, i \in \mathcal{I}, n \in \mathcal{N}, \end{aligned} \quad (2)$$

where X_σ denotes the shadowing loss (in dB) and obeys a Gaussian distribution with zero mean and standard deviation σ. h denotes the multipath fast fading, which can be modeled as Rician distribution with K-factor, and P^{U} (in dBm) is the transmitted power of UE i. For simplicity, we assume that all UEs have the same transmission power.

On the other hand, according to Shannon's Theory, the maximum uplink transmission rate for UE i at time t can be calculated as

$$r_{t,n,i} = W \log_2\left(1 + \frac{10^{P^{\mathsf{U}}_{t,n,i}/10}}{I_{t,n,i} + N_0}\right), \quad (3)$$

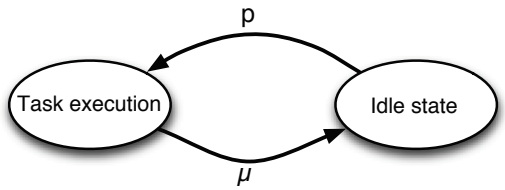


Figure 2. Transition diagram of task generation.

where W is the channel bandwidth, $I_{t,n,i}$ is the mutual interference caused by other UEs [12] and N_0 is the noise power. $P^{\mathsf{U}}_{t,n,i}$ is the RSS (in dBm) from UE i for BS n, which can be calculated as in (2) with UE transmission power P^{U}.

C. Computation Model

*1) **Local task execution**:* With the dynamic voltage and frequency scaling (DVFS) technology, each UE adjusts its computing capacity for different tasks [12]. Therefore, the local task execution time is denoted as

$$D^{\mathsf{L}}_{t,i,m} = \frac{\xi_m \lambda_m}{f_{t,i,m}}, \quad (4)$$

where ξ_m and λ_m is the computation intensity and the data size of task m executed in UE i, and $f_{t,i,m}$ is the allocated CPU frequency for task m.

The energy consumption with the local task execution [13] is calculated as

$$E^{\mathsf{L}}_{t,i,m} = (\eta_i (f_{t,i,m})^{\mathcal{X}_i} + \beta_i) D^{\mathsf{L}}_{t,i,m}, \quad (5)$$

where $\mathcal{X}_i$, η_i and β_i are the parameters determined by the CPU processing model [13].

*2) **Task execution on the edge server**:* The computation delay is computed as

$$D^{\mathsf{C}}_{t,i,m} = \frac{\xi_m \lambda_m}{F}, \quad (6)$$

where ξ_m is the computation intensity of task m, and λ_m is the data size of task m offloaded to BS n. The first come first serve (FCFS) principle is considered as the scheduling strategy for edge users where requests are served in the order of their arrival [14]. For simplicity, we assume all BSs are equipped with the computing capacity F.

For UE i, the transmission delay of task m at time t is given by

$$D^{\mathsf{T}}_{t,i,m} = \frac{\lambda_m}{r_{t,n,i}}, \quad (7)$$

As multiple UEs are competing for the limited computation resources at the edge server, the task queuing delay of BS n at time t is considered as follows

$$D^{\mathsf{Q}}_{t,i,m} = \frac{B_{t,n}}{F}, \quad (8)$$

where $B_{t,n}$ is the current workload of BS n at time t, which is changing with time.

As shown in Fig. 3, the overall time delay of the edge service $D^{\mathsf{E}}_{t,i,m}$ consists of the transmission, queuing and computation delay

$$D^{\mathsf{E}}_{t,i,m} = D^{\mathsf{T}}_{t,i,m} + D^{\mathsf{Q}}_{t,i,m} + D^{\mathsf{C}}_{t,i,m}. \quad (9)$$

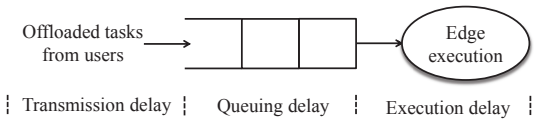


Figure 3. Delay model of task offloading.

Similar to [12], [15], the data transmission also consumes the tail energy in practice. Thus, the energy consumption for the transmission of task m is

$$E^{\mathsf{T}}_{t,i,m} = P^{\mathsf{T}}_i \cdot D^{\mathsf{T}}_{t,i,m} + P^{\mathsf{tail}}_i \cdot D^{\mathsf{tail}}_i, \tag{10}$$

where P^{T}_i is the transmission power of UE i, P^{tail}_i is the cellular tail power consumption of UE i, and D^{tail}_i is the tail time.

For real-time applications offloaded to the edge server, the energy consumption during the queuing and execution time cannot be ignored. Thus, the energy consumption during this time is

$$E^{\mathsf{Q}}_{t,i,m} = P^{\mathsf{idle}}_i (D^{\mathsf{C}}_{t,i,m} + D^{\mathsf{Q}}_{t,i,m}), \tag{11}$$

where P^{idle}_i is the idle power of UE i.

Thus, the total energy consumption of offloading a task to the edge server consists of the transmission energy consumption and idle energy consumption, which is calculated as

$$E^{\mathsf{E}}_{t,i,m} = E^{\mathsf{T}}_{t,i,m} + E^{\mathsf{Q}}_{t,i,m}, \tag{12}$$

Considering that the result of a task is relatively small compared with the transmitted data, similar to the previous studies [12], [16], the downlink transmission delay is not considered in this work.

III. Mobility Management Scheme

Unmanaged mobility in the wireless environment causes communication disruption while the UE moves around [17]. In the mobility management, handover allows the UE to link to the target BS from the serving BS in a connected mode. In the existing network, a handover procedure is triggered when the RSS of the serving BS drops below a certain threshold than the best available BS. However, in the MEC-enabled networks, as BSs provide both communication and computation services, using the same set of handover criteria may degrade the QoS. In this section, we adapt the handover criteria to the MEC scenario. As the whole system changes rapidly, an online learning based mobility management scheme is proposed to handle the uncertainties.

A. Mobility Management with Full Information

As BSs are densely deployed in a hot-spot area, UEs have multiple choices to offload their tasks. If the UE can connect to a BS whose edge server can complete the task rapidly when compared with other available BSs, the task delay can be reduced, as long as it is under the constraint of the total energy budget. Thus, to fully utilize the advantages of MEC, the handover trigger criteria should be adapted to MEC-enabled networks. This motivated us to propose a simple but efficient mobility management scheme if the accurate full network information is available and can be synchronized in real time as a benchmark.

Algorithm 1 Delay-Based Moblity Management Scheme

```
Initialize the UE connection state by associating the UE to the BS with the strongest RSS;
for Time t = 1, ... T do
    if ∃ Task m then
        Calculate the expected delay for each available BS as in (9);
        Select the BS with the shortest expected delay;
    else
        Select the BS with the strongest RSS in excess of a certain threshold when compared with the serving BS;
    end if
end for
```

The delay-based mobility management scheme with full information (DFI) is summarized in Algorithm 1, which can be considered as a greedy optimization approach. Each UE is initially connected to the BS with the strongest RSS. In each time slot, each UE calculates the expected task delay based on the task information, channel condition and BS workloads, and then greedily chooses the BS with the shortest task delay. If there is no task to offload, UE will choose the target BS with the strongest RSS.

B. Mobility Management with Partial Information

The main challenges in the mobility management decision making are the uncertainties of the whole network information such as the BS workload and the future channel condition. Even if the whole information is available, the rapid changes are hard to be synchronized with UEs for decision making. Our objective is to create an intelligent scheme that automatically learns the handover strategy through trials and feedbacks to shorten the latency only based on the local observations. Learning-based algorithms, especially the reinforcement learning, can help users learn from the previous experience and make decisions without the complete information. Q-learning, a typical algorithm to identify the optimal action-selection policy, is introduced to solve the cell selection problem with partial information. Each UE interacts with the environment, which conversely provides a reward, to learn how to take actions in the specific state to maximize its reward.

The parameters of the Q-learning algorithm are defined as follows:

*1) **Agent**:* The agent is UE $i \in \mathcal{I}$ who decides to select the most appropriate BS to achieve the shortest task delay in MEC.

*2) **State**:* The state is defined as $\mathcal{S} = \{s = n \times l \mid n \in \mathcal{N}, l \in \mathcal{L}\}$, jointly considering the serving BS n and the current channel state l.

*3) **Action**:* The action, which is the target BS, is the decision made by the agent. The set of actions per state is defined as $\mathcal{A} = \{a = n' | n' \in \mathcal{N}\}$. By taking action $a \in \mathcal{A}$, the agent transitions from the current state to the next state.

*4) **Reward**:* The reward for executing an action a in state s is the task execution speed, which is defined as $R(s,a) = \lambda_m / D^{\mathsf{E}}_{t,i,m}$.

At time t, UE i is in state s_t and takes an action a_t to connect to BS n' for task offloading. UE i thus observes the throughput as the reward of taking action a_t in the current state s_t and then updates the action-value accordingly. Therefore, at time t, when visiting state-action pair (s_t, a_t), the action-value function Q is then updated after observing the corrseponding reward, which can be denoted as

$$\begin{aligned} Q(s_t,a_t) =& (1-\alpha) Q(s_t,a_t) \\ &+ \alpha \left\{ R(s_t,a_t) + \gamma \left[\max_{a'} Q(s_{t+1},a') \right] \right\}, \end{aligned} \tag{13}$$

where $\alpha \in (0,1]$ is the learning rate which determines to what extent the recent observation overrides the experience, and $\gamma \in [0,1]$ is the discount factor which denotes the effect of the future reward on the current state value.

Algorithm 2 Q-Learning-Based Mobility Management Scheme

1: Initialize action-value function $Q(s,a) \leftarrow 0$.
2: Initialize state s by associating the UE to the BS with the strongest RSS;
3: **for** Time $t = 1, \dots T$ **do**
4: **if** $\exists$ Task m **then**
5: With probability ϵ select a random action a_t, otherwise select action $a_t = \max_a Q(s_t,a)$;
6: Execute action a_t, observe reward $R(s_t,a_t)$ and obtain the next state s_{t+1};
7: Update action-value Q according to (13);
8: **else**
9: Select the BS with the strongest RSS in excess of a certain threshold when compared with the serving BS;
10: **end if**
11: **end for**

The proposed Q-learning based mobility management scheme with partial information (QPI) is summarized in Algorithm 2. The action-value function and each UE's starting state are first initialized. At the beginning, each UE is connected to the BS with the strongest RSS. In each time slot, ϵ-greedy policy is introduced to solve the trade-off between exploitation and exploration where the UE keeps exploring different BSs as well as trying to connect to the optimal BS according to the experience. Each UE selects an action between exploration (select a random action with a small probability) and exploitation (select the best action in the current state). Thus, UEs could not only adapt to the changes of the system but also behave in an optimal way. Then, the reward of the action is observed and the action-value function is updated accordingly. If there is no task to offload, UEs will choose the target BS with the strongest RSS.

In theory, Q-learning algorithm has been proven to converge towards the optimum when the state-action pair is infinitely visited and the learning rate is decreased to zero [18]. However, when UEs move to another BS, the environment varies and Q-values need to be updated to keep up with the dynamic system. In this case, the UE's adaption to the system also changes the system itself, which triggers other UEs to readjust their strategies. This indicates that the estimation is never completely convergent but continues to change in response to the dynamic system, which is desirable in the nonstationary system [19]. On the other hand, the exploration step encourages the UE to select an action independently of the state-action estimates, making it more flexible if an alternative strategy appears.

IV. Performance Evaluation

In this section, extensive simulations are conducted to evaluate the performance of the proposed scheme.

A. Simulation Setup

Considering an exhibition event scenario in Fig. 4, 4 BSs are deployed in the hexagonal grid with radius 80 m and 350 UEs are moving around in between the BSs. Random waypoint model is used as the UE mobility model with speed $v \in$ [0.2, 2.2] m/s, walk interval $w \in$ [2,6] s, and pause interval $p \in$ [0,1] s. A typical computation-intensive application, face recognition, is considered in this scenario, which requires the comparison with a large database stored on the edge server [20], [21]. Other simulation parameters are selected based on [12], [16].

For comparisons, two other benchmarks are introduced to evaluate the performance of the two proposed algorithms: 1) RSS-based mobility management scheme with full information (RFI): as shown in Fig. 5, when the UE moves from BS 1 to BS 2, the RSS from the serving BS (BS 1) decreases and that from the target BS (BS 2) increases. Here, UEs always connect to the best available BS based on the channel condition; 2) Local task execution (LOC): UEs always execute tasks locally and select BSs based on RSS.

B. BS-Side Performance Evaluation

From the BS's point of view, the decision making of different schemes has an impact on the workloads of each BS. As shown in Fig. 6, the variance of the BS workload is evaluated. A higher variance indicates the unbalanced workload among different BSs. When more and more UEs move towards a certain BS, the BS may be overload with RFI. However, DFI shows the ability to balance the BS workload because the UE with full system information always chooses the BS which can complete the task as soon as possible. Compared with DFI, QPI manifests the capability of handling the system uncertainties with some learning loss. As executing task locally does not affect the BS workload, LOC is not evaluated in this performance metric.

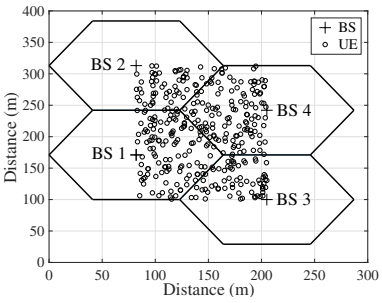


Figure 4. Simulation topology.

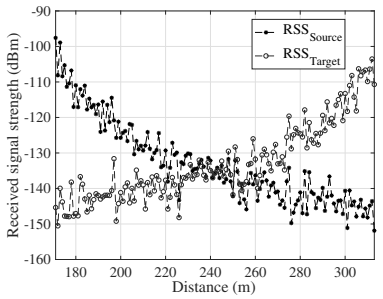


Figure 5. RSS from the adjacent BSs.

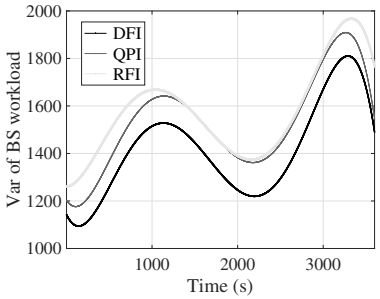
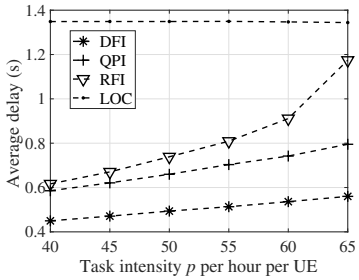
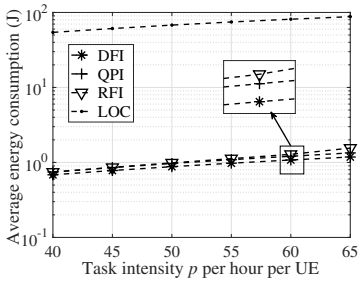


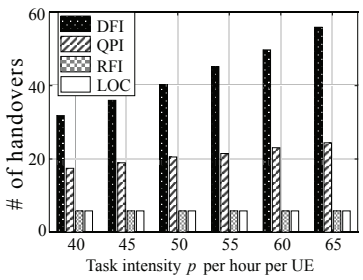
Figure 6. Variance of the BS workload.



(a) Average per-task delay

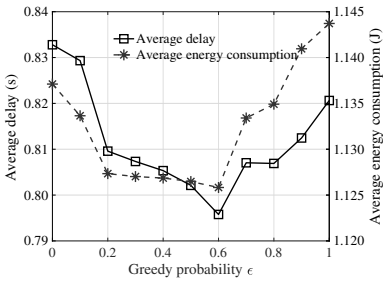


(b) Average per-UE energy consumption

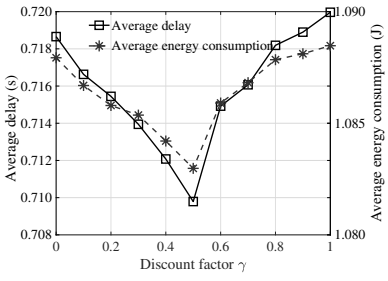


(c) Per-UE handover frequency

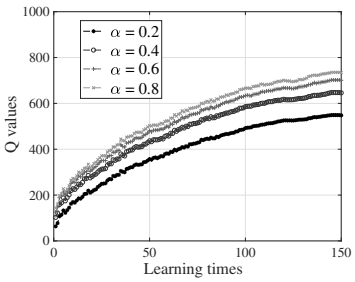
Figure 7. Impact of the task intensity.



(a) Impact of ϵ



(b) Impact of γ



(c) Impact of α

Figure 8. Impact of the prameters on the learning algorithm.

C. UE-Side Performance Evaluation

The impact of the task intensity on the average per-task delay is shown in Fig. 7(a). The larger the task intensity is, the more tasks generated by UEs, which indicates that the edge server who helps offload tasks receives more task requests. The average per-task delay in LOC remains constant, since the UEs executing tasks locally are not affected by the edge server and are limited by their local capacity which is assumed to be stable in this case. Although a higher BS workload increases the average per-task delay when more and more tasks are offloaded to the edge server, in DFI, QPI and RFI, offloading tasks to the edge server helps reduce the average per-task delay when compared with LOC. On the other hand, both the QPI algorithm and DFI algorithm perform better than RFI by jointly considering the channel condition and computing capacity. Moreover, the RFI algorithm is more sensitive to the task intensity while QPI and DFI can balance the BS workload effectively. That is, if all users select a BS which provides the best wireless channel condition, the BS may be overloaded and the tasks need to wait for a relatively long time to be processed at the edge server. In addition, the channel conditions become worse when more UEs get involved in the network, thus increasing the data transmission time in task offloading. What excels the QPI algorithm is that it does not need to know the full system information and each UE makes a decision based on their experience while observing the reward as well as updating the experience.

The impact of the task intensity on the per-UE energy consumption is shown in Fig. 7(b). With the increase of the task intensity, the average per-UE energy consumption increases. As the face recognition task is offloaded to the edge server, the UE only needs to transmit the task to the BS and waits until the task is accomplished. A high delay aggravates the energy consumption. However, considering the

sharp difference when compared with LOC, the other three schemes all maintain a relatively low energy consumption level, where the proposed QFI scheme does not require any prior knowledge of the system.

The impact of the task intensity on the handover frequency is shown in Fig. 7(c). DFI and QPI are sensitive to the disturbance of the system as the workload of the edge server changes over time. With the increase of the task intensity, handover frequency increases dramatically in DFI and QPI because the UE always tries to connect with the BS that completes the tasks earlier. RFI is quite stable because the decision making is triggered based on the RSS, which mainly depends on the relative location between the UE and the BS, thus ignoring the BS workload. As LOC and RFI both make a handover decision based on the RSS, their handover performance stays the same.

The impact of three parameters, i.e., ϵ, γ and α, on Algorithm 2 is shown in Fig. 8. The change of the energy consumption is consistent with that of the task delay. As shown in Fig. 8(a), the larger ϵ is, the higher the probability that a UE randomly chooses a BS to fully explore the state space is. When ϵ is 0 in QPI, the UE makes the decision only based on the experience and takes the action with the largest Q-values. However, excessive ϵ introduces a stronger randomness. Thus there is a trade-off between the exploitation and the exploration. Choosing an appropriate ϵ encourages the UE to both explore the state space and try to connect to the highest ranked BS according to the experience. As shown in Fig. 8(b), a small discount factor γ makes the UE near-sighted by only considering the current rewards, while with the increase of γ, the UE values future rewards more than the current rewards and strives for a long-term reward. Fig. 8(c) evaluates the impact of learning times and learning rate. The Q-values keep increasing and then become stable when the UE interacts with the environment with more and more tasks. Moreover, a higher learning rate α accelerates the learning procedure as the UE learns more about the difference between the recent observation and the experience.

V. Conclusion and Future Work

In this paper, a Q-learning-based mobility management scheme is proposed to tackle the challenges of mobility management in the MEC as the full system information changes rapidly due to the user mobility. To handle the uncertainties of the system information, UEs make decisions by interacting with the environment and keep updating their experience based on the observation of the task delay. Simulations show that the proposed scheme is superior to the traditional RSS-based mobility management scheme in reducing the task delay while maintaining a relatively low energy consumption. For the future work, experience replay and neural network can be introduced to speed up the convergence and avoid bad feedback loops. Moreover, this problem can also be addressed in a more complex offloading scenario where multiple servers help offload a task simultaneously when given a certain task execution deadline constraint.

Acknowledgement

This work is supported in part by NSERC, CFI and BCKDF.

References

[1] W. Junior, A. França, K. Dias, and J. N. de Souza, "Supporting mobility-aware computational offloading in mobile cloud environment," *Journal of Network and Computer Applications*, vol. 94, pp. 93–108, 2017.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[4] J. Liu, K. Au, A. Maaref, J. Luo, H. Baligh, H. Tong, A. Chassaigne, and J. Lorca, "Initial access, mobility, and user-centric multi-beam operation in 5G new radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 35–41, 2018.

[5] E. Demarchou, C. Psomas, and I. Krikidis, "Intelligent user-centric handover scheme in ultra-dense cellular networks," in *Proc. of IEEE GLOBECOM*, 2017, pp. 1–6.

[6] M. M. Hasan, S. Kwon, and J. H. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2205–2217, 2018.

[7] C. Dhahri and T. Ohtsuki, "Q-learning cell selection for femtocell networks: Single- and multi-user case," in *Proc. IEEE GLOBECOM*, 2012, pp. 4975–4980.

[8] M. E. Helou, M. Ibrahim, S. Lahoud, K. Khawam, D. Mezher, and B. Cousin, "A network-assisted approach for RAT selection in heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1055–1067, 2015.

[9] M. Yan, G. Feng, and S. Qin, "Multi-RAT access based on multi-agent reinforcement learning," in *Proc. IEEE GLOBECOM*, 2017, pp. 1–6.

[10] M. Yan, G. Feng, J. Zhou, and S. Qin, "Smart multi-RAT access based on multi-agent reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2018.

[11] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, 2018.

[12] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing," *Future Generation Computer Systems*, vol. 64, pp. 1–14, 2016.

[13] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[14] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[15] D. Zhang, Y. Zhang, Y. Zhou, and H. Liu, "Leveraging the tail time for saving energy in cellular networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1536–1549, 2014.

[16] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[17] M. Zekri, B. Jouaber, and D. Zeghlache, "A review on mobility management and vertical handover solutions over heterogeneous wireless networks," *Computer Communications*, vol. 35, no. 17, pp. 2055–2068, 2012.

[18] C. J. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[20] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM MobiSys*, 2010, pp. 49–62.

[21] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, 2012, pp. 945–953.