# Reinforcement Learning Based Computation-aware Mobility Management in Ultra Dense Networks

Ziyue Zhang[1], Jie Gong[2], Xiang Chen[1], Terng-Yin Hsu[3]

[1] School of Electronics and Information Technology, Sun Yat-sen University, China
[2] School of Data and Computer Science, Sun Yat-sen University, China
[3] Department of Computer Science, National Chiao Tung University, Taiwan
zhangzy29@mail2.sysu.edu.cn, gongj26@mail.sysu.edu.cn, chenxiang@mail.sysu.edu.cn, tyhsu@cs.nctu.edu.tw

## Abstract

Computation-aware delay optimal mobility management (MM) is an important problem in ultra-dense network (UDN) with mobile edge computing (MEC). Since the additional time delay caused by task computation is not taken into consideration, traditional radio access-oriented mobility management scheme cannot guarantee the overall delay performance of delay-sensitive user equipment (UE). In this paper, we propose a novel dynamic programming-based mobility management (DPMM) scheme to minimize the average delay under an energy consumption constraint. DPMM makes MM decisions using statistic information to handle the inaccurate state information. Cooperative data transmission is adopted to improve the delay performance. Simulation shows that the proposed DPMM scheme can achieve delay performance close to optimal and reduce the frequency of handover. However, the wireless link, computation resources and UE's location in UDN environment is dynamic, which leads to information uncertainties. We further propose an MM scheme based on deep Q-network (DQN) to learn the system information from the environment. In this scheme, UE takes the current and past observed delay as experience, learning the optimal mobility management strategy through DQN training. Simulation shows that DQN-based MM can learn from experience and reduce the handover frequency to a certain degree.

**Keywords:** Mobility management, Dynamic programming, Deep Q-network, Cooperative transmission

## 1 Introduction

With the increase of mobile content service [1], mobile traffic has shown explosive growth, and there have been various applications that require wireless and computing resources, such as intelligent identification, virtual reality and unmanned aerial vehicles [2-4]. Ultra-dense network (UDN) and mobile edge computing (MEC) are effective technologies to meet this challenge [5-8]. However, since the base stations equipped with MEC function are deployed denser and closer to each other, high-mobility users will trigger a large number of handover processes, resulting in additional signaling overhead [9]. At the same time, frequent handover can cause handover failures, low task offload efficiency and other issues. With the aim of further enhancing user experience and the quality of services, novel mobility management (MM) schemes are needed.

In current 3rd Generation Partnership Project (3GPP) standards, handover mechanism is based on the user equipment (UE) measuring the signal quality of the candidate base stations (BSs), selecting the BSs with the best signal quality. When its performance drops to a certain threshold, handover is triggered [10]. Merging UDN and MEC, mobility management considers not only the the wireless channel quality, but also the MEC computing capacity. Simply applying existing solutions will lead to poor performance due to the overlapped coverage of multiple BSs and the effect of wireless access and computing. Thus, computation-aware MM need to be designed for task computation. More importantly, how to deal with the information uncertainty is another key issue [11]. In particluar, MM in UDN environment equipped with MEC faces the following challenges:

(1) Multiple BSs equipped with MEC are densely deployed in UDN environment with different radio channel qualities and computation capabilities. Both wireless transmission and computing capability need to be considered. However, it's difficult to make MM decisions to access which BSs for a better performance.

(2) UDN environment is a dynamic and complex network due to the densely deployment of BSs and the dynamic of computation capabilities of MEC [12]. The MM decisions need to make without accurate information and future information.

(3) The high mobility of UE may cause frequent handover in UDN environment, new MM scheme should balance handover frequency and short-term

*Corresponding Author: Xiang Chen; E-mail: chenxiang@mail.sysu.edu.cn
DOI: 10.3966/160792642020112106019

delay.

Mobility management has received a lot of attentions as mobile network develops. There are a few mobility management methods based on game theory. Ref. [13] proposed a coordination-based and context-aware MM procedure for small cell networks using multi armed bandit (MAB) theory, considering a UE's throughput history and velocity to enable fair scheduling and enhanced cell association. However, it only considered the wireless access. Ref. [14] used LSTM scheme to learn the low-speed and medium-speed users' mobility patterns from their historical trajectories for prediction. However, none of these works consider the dynamic computation capacity of MEC. It is important to consider the computation capacity of MEC in computation-aware mobility management.

MM in UDN environment with MEC functions requires new mobility management solutions. There are also a few works considering computation offloading. In [15], Lyapunov optimization and MAB theory are used to handle the imperfect state information in UDN and MEC environment. It takes radio handover and computation offloading cost into consideration. A Q-learning based mobility management scheme is proposed in [16] to handle the system information uncertainties. A context-aware QoS prediction method is proposed in [17] to determine the suitability of a service to a user. However, the additional delay caused by handover was not considered in these studies. As BSs are densely deployed in UDN environment, the handover will cause much time delay due to UE's high mobility.

In this paper, we propose two computation-aware delay optimal mobility management schemes in UDN and MEC scenarios to guide UE to connect to the appropriate base stations and process handover at the appropriate time. Moreover, cooperative data transmission is used to improve task offloading. Firstly, by using Markov decision process (MDP) and dynamic programming (DP), we aim to solve the problem of minimizing the average delay. The proposed scheme uses the statistic information of UE's movement and base station to make MM decisions. The numerical experiment shows the proposed scheme can effectively improve the delay performance and reduce handover frequency.

Further, we proposed a training scheme using deep Q-network (DQN) based on the historical information obtained by interacting with the environment. The MM decisions are made according to the historical information other than future information. When UE gets new information, DQN is further trained to keep up with the dynamic system. Simulations show that

The rest of the paper is organized as follows. The system model and the problem formulation are introduced in Section 2. The DP based mobility management scheme and the DQN based mobility management are proposed in Section 3. The performance of the proposed algorithms is evaluated in Section 4. Finally, the conclusion and future works are summarized in Section 5.

## 2 Problem Formulation

In this section, system overview is firstly introduced including user mobility, communication model and computation model. Then, the problem is formulated according to the system model.

### 2.1 User Equipment Mobility and Task Generation

As shown in Figure 1, BSs $\mathcal{S}_N = \{1, 2, \ldots, N\}$ equipped with MEC server are densely deployed in a UDN environment. A UE can offload computation tasks to MEC server through one or two BSs. The UE randomly moves in the UDN environment, generating totally $M$ tasks that need to offload to MEC server. The trajectory of UE is modeled as the two-dimensional random waypoint model [18].

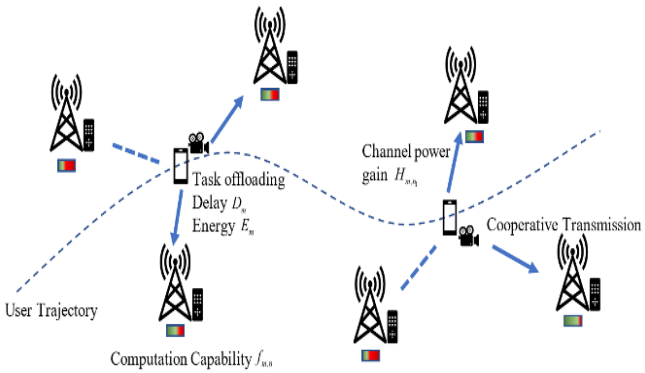


**Figure 1.** System Model

The location of UE is denoted as $L_m \in \mathcal{S}_L$ where task $m$ is generated. There are always several candidate BSs $\mathcal{S}_A(L_m) \subseteq \mathcal{S}_N$, where $\mathcal{S}_A(L_m)$ denotes the set of candidate BSs in location $L_m$. We assume that the task can be completed before the user moves to the next location.

### 2.2 Computation Model

The computation task $m$ is characterized using a two-parameter model [19]: input data of size $\lambda_m$ bits that needs to be o'oaded and computation intensity $\mu$ indicating how many CPU cycles are required to compute one bit input data. Without loss of generality, we assume that tasks are all of equal size $\lambda$. The following analysis can be extended to the cases where the data sizes are not equal. Moreover, the CPU frequency distribution of BSs follows uniform distribution.

Each BS $n \in \mathcal{S}_N$ is equipped with an MEC server of

CPU frequency $f \in \mathcal{S}_f$ and supports cooperative data reception with other BSs. A UE who needs task o'oading can choose one BS or two BSs according to the candidate BSs and time delay. Once the task is o'oaded to two BSs via cooperation, it can be computed jointly by the two BSs. The equivalent CPU frequency is modeled as the sum of the two serving BSs' CPU frequency for simplicity. If BS $n_1$ and BS $n_2$ are selected for task computation, the computation delay is

$$d^c_{m,n_1,n_2} = \frac{\mu\lambda}{f_{m,n_1} + f_{m,n_2}}, \tag{1}$$

Where $f_{m,n_1}$ and $f_{m,n_2}$ are the CPU frequencies that BS $n_1$ and BS $n_2$ can provide to task $m$ respectively.

If the UE selects one BS to compute tasks, the computation delay is

$$d^c_{m,n_1} = \frac{\mu\lambda}{f_{m,n_1}}. \tag{2}$$

Considering the high mobility of UE, different tasks may be offloaded to different BSs for a lower time delay. As UE moves, handover is executed, causing additional time delay. Let $c_m$ be the handover cost for task $m$ and $a_m = (n_i, n_j)$ be the set of the selection of serving BSs for task $m$, where $n_i \in \mathcal{S}_N$ and $n_j \in \mathcal{S}_N \cup \{0\}$. The overall handover cost is

$$D_h = c_m \sum_{m=1}^{M-1} d^h_m = c_m \sum_{m=1}^{M-1} f_I\{a_m \neq a_{m+1}\}, \tag{3}$$

where $d^h_m$ is the handover delay of task $m$. We assume that the handover delay is constant. Hence, $d^h_m$ can be expressed as an indicator function $f_I\{x\}$. $f_I\{x\} = 1$ if $x$ is true and $f_I\{x\} = 0$ otherwise.

## 2.3 Communication Model

According to the quality of wireless communication, UE can choose cooperative transmission method or non-cooperative transmission method to offload tasks.

### 2.3.1 Cooperative Data Transmission

Maximal ratio combining (MRC) is one of diversity merging technologies [20]. Compared with selection combining (SC) and equal gain combining (EGC), MRC can get the best performance by getting a higher signal noise ratio (SNR).

According to Ref. [21], the SNR can be calculated as follows:

$$\mathrm{SNR}_{m,n_1,n_2} = \frac{P_t(H_{m,n_1} + H_{m,n_2})}{\sigma^2}, \tag{4}$$

where $H_{m,n_1}$ and $H_{m,n_2}$ represent the channel power gain at task $m$ between the UE and the serving BSs $n_1$ and $n_2$ respectively. $\sigma^2$ denotes the noise power and $P_t$ denotes the transmit power.

The task is o'oaded to the serving BSs through the wireless uplink channel. Assume that the SNR of uplink channel is constant during task transmission. The uplink transmission rate is represented as follows:

$$r_{m,n_1,n_2} = W\log_2\left(1 + \mathrm{SNR}_{m,n_1,n_2}\right), \tag{5}$$

where $W$ is the channel bandwidth.

The transmission delay offloading the task data of size $\lambda$ to BS $n_1$ and BS $n_2$ is

$$d^t_{m,n_1,n_2} = \frac{\lambda}{r_{m,n_1,n_2}}. \tag{6}$$

The energy consumption of uplink transmission is

$$e^t_{m,n_1,n_2} = \frac{P_t\lambda}{r_{m,n_1,n_2}}. \tag{7}$$

Meanwhile, cooperative data transmission also causes extra energy consumption:

$$e^{cop}_{m,n_1,n_2} = \frac{P_{cop}\lambda}{r_{m,n_1,n_2}}, \tag{8}$$

where $P_{cop}$ is the energy consumption per second.

### 2.3.2 Non-Cooperative Data Transmission

The UE can access to a single BS if necessary. In this case, the uplink transmission rate of task $m$ is

$$r_{m,n_1} = W\log_2\left(1 + \mathrm{SNR}_{m,n_1}\right). \tag{9}$$

The transmission delay offloading the task data of size $\lambda$ to BS $n_1$ is

$$d^t_{m,n_1} = \frac{\lambda}{r_{m,n_1}}. \tag{10}$$

And the energy consumption of uplink transmission is

$$e^t_{m,n_1} = \frac{P_t\lambda}{r_{m,n_1}}. \tag{11}$$

## 2.4 Problem Formulation

In this paper, we consider the problem of minimizing the average delay under the constraint of average energy consumption, to determine which BSs serve the user and calculate the task. For task $m$, $a_m \in \mathcal{S}_A(L_m)$ is the set of the selection of serving BSs. The overall delay is

$$D_{m,a_m} = d^c_{m,a_m} + d^t_{m,a_m} + d^h_{m,a_m}, \quad (12)$$

consisting of computing delay $d^c_{m,a_m}$, transmission delay $d^t_{m,a_m}$ and handover delay $d^h_{m,a_m}$.

The overall energy consumption for task m is

$$E_{m,a_m} = e^t_{m,a_m} + e^{cop}_{m,a_m}. \quad (13)$$

We formulate the problem as an infinite horizon average cost minimization problem. The problem is expressed as follows:

$$\begin{aligned} \min_{a_1,a_2,\cdots} \quad & \lim_{M\to+\infty} \frac{1}{M}\sum_{m=1}^{M} D_{m,a_m} \\ s.t. \quad & \lim_{M\to+\infty} \frac{1}{M}\sum_{m=1}^{M} E_{m,a_m} \le \alpha B \\ & a_m \in \mathcal{S}_A(L_m), \qquad \forall m \end{aligned} \quad (14)$$

where $B$ is the battery capacity and $\alpha \in (0,1]$ indicates the desired energy consumption for all tasks.

## 3 Mobility Management Scheme

### 3.1 Mobility Management with Statistic Information

The main challenges in mobility management are the rapid change of information in the whole UDN environment caused by the high mobility of UE [22]. It is hard to be synchronized with UEs for decision making. We need to create a scheme that can learn the handover decisions through the statistic information of UE and network. Dynamic programming (DP) can solve the handover decision making problem with statistic information. The problem can be formulated as a Markov decision process which can be solved by dynamic programming.

The parameters of the DP-based mobility management are defined as follows:

(1) ***Agent***: The agent is a UE who makes decisions to select candidate BSs to ensure the shortest delay of task execution.

(2) ***State***: The state is defined as $s = ((n_i, n_j), l, (f_i, f_j))$, where $n_i$ and $n_j$ are the service BSs of the UE and $a_m = (n_i, n_j)$. $l$ represents the location of UE. The available computing power of service BSs is denoted by $f_i$ and $f_j$ respectively. If non-cooperative data transmission is used, $n_j$ and $f_j$ equal to zero. The state space can be expressed as

$$\begin{aligned} \mathcal{S} = \{ s \mid & n_i \in \mathcal{S}_N, n_j \in \mathcal{S}_N \cup \{0\}, l \in \mathcal{S}_L, \\ & f_i \in \mathcal{S}_F, f_j \in \mathcal{S}_F \cup \{0\} \} \end{aligned}. \quad (15)$$

(3) ***Action***: The action is to select service BSs to access. The set of action per state is defined as

$$\mathcal{A} = \left\{ a = (n_i, n_j) \mid n_i \in \mathcal{S}_N, n_j \in \mathcal{S}_N \cup \{0\} \right\}. \quad (16)$$

By taking action $a \in \mathcal{A}$, UE can execute handover to access to target BSs for a better time delay performance.

(4) ***Reward***: The reward for executing an action $a$ at state $s$ is the negative time delay of each task, which is defined as

$$R(s,a) = -D_m. \quad (17)$$

(5) ***State Transition Probability***: The state transition probability $P \in \mathbb{R}^{s\times s\times a}$ is generated according to the statistic information of the random walk trajectory model and BSs' CPU frequency distribution.

When the state $s$ and action $a$ are determined, the state transition probability $P$ can be calculated according to the random walk trajectory model and the frequency distribution of base stations. Meanwhile, each state-action pair $(s,a)$ has an action-value function. The action-value function can be defined as follows:

$$q_\pi(s,a) = R^a_s + \sum_{s'\in\mathcal{S}} P^a_{ss'} v_\pi(s'), \quad (18)$$

where $\pi$ is the action selection policy. $v(s)$ is the value function of state $s$ defined as follows:

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s) q_\pi(s,a), \quad (19)$$

Value iteration algorithm is used to calculate the optimal value function to get optimal decisions. Firstly, we define the $i$ stage value function as

$$v_{i+1}(s) = \max_a R^a_s + \sum_{s'\in\mathcal{S}} P^a_{ss'} v_i(s'). \quad (20)$$

The differential utility is defined as $v_i(s) = v_i(s) - v_i(s_0)$, where $s_0$ is a fixed state. Then, Bellman equation [23] can be given as follows:

$$\gamma^* + v^*(s) = \max_a \left[ R^a_s + \sum_{s'\in\mathcal{S}} P^a_{ss'} v^*(s') \right], \quad (21)$$

where $\gamma^*$ is the optimal average utility. The differential utility $v_i(s)$ represents the maximum difference between the expected utility that from state $s$ to a given state $s_0$ and the utility that if the utility of each stage is $\gamma^*$.

The differential utility can be calculated by

$$\gamma_{i+1}(s_0) = \max_a \left[ R^a_{s_0} + \tau \sum_{s'\in\mathcal{S}} P^a_{s_0 s'} v_i(s') \right], \quad (22)$$

$$\begin{aligned} v_{i+1}(s) = & (1-\tau) v_i(s) \\ & + \max_a \left[ R^a_s + \tau \sum_{s'\in\mathcal{S}} P^a_{ss'} v_i(s') \right] - \gamma_{i+1}(s_0) \end{aligned}. \quad (23)$$

In value iteration algorithm, the optimal policy

$\pi^*(s)$ can be found by calculating the optimal $v^*(s)$. Note that the parameter $\tau \in (0,1)$ is used to guarantee the convergence of relative value iteration and does not change the optimal value [24]. Since the optimal average utility is irrelative with the initial state, $\gamma_{i+1}(s_0)$ converges to $\gamma^*$.

The proposed dynamic programming-based mobility management (DPMM) is summarized in Algorithm 1. Firstly, the state transition probability $P$ is generated according to the known statistic information. Then the differential utility, the policy and a random state are initialized. Line 4-8 is value iteration algorithm to iterate to get the best policy. The iteration will stop when the maximal differential utility is reached.

**Algorithm 1.** DP-based Mobility Management

1. Initialize state set $\mathcal{S}$, action set $\mathcal{A}$, the reward of state-action pair $R(s,a)$ and state transition probability $P$ according to statistic information.
2. Initialize the differential utility $v_0(s)$=0, the initial policy $\pi_0$ and parameter $\tau$.
3. Randomly choose a fixed state $s_0$.
4. **repeat**
5. **for** every $s$ **do**
6. $$v_{i+1}(s) = (1-\tau)v_i(s) + \max_a \left[ R_s^a + \tau \sum_{s' \in S} P_{ss'}^a v_i(s') \right] - \gamma_{i+1}(s_0)$$
7. **end for**
8. **until** $v_{i+1} = v_i$
9. **output** $\pi^*(s)$

## 3.2 Mobility Management with DQN

In DPMM algorithm, the policy is obtained using the statistic information of UE and UDN environment. In many cases, however, the known statistical model does not fully comply with the rapidly changing UDN environment, which will cause the performance of the scheme to be greatly degraded. In addition, a look-up table created by a discrete state space is used, which becomes intractable as the number of possible states becomes very large. Therefore, discretization will affect the accuracy of the algorithm. Considering above two problems, we further proposed DQN based mobility management scheme. It is a model-free learning. Different from DPMM scheme, UE only interact with UDN environment in DQNMM scheme, getting the reward of time delay. Thus, the reward is defined as $R(s,a) = -D_m$.

DQN algorithm is proposed based on Q-learning. It combines deep neural networks with reinforcement learning, and performs non-linear fitting of Q value through deep neural networks. Similar to DP algorithm, The Q value is defined as expectation of reward $R$:

$$Q(s,a) = E[R(s,a)]. \quad \textbf{(24)}$$

According to Bellman equation, the Q value can be expressed as

$$Q(s,a) = R + \beta \max(Q(s',a')). \quad \textbf{(25)}$$

As shown in Fig. 2, the DQN algorithm consists of two deep neural networks as a non-linear function approximation, one is evaluate network and the other is target network. In DQN algorithm, the action-value function $Q$ [25] is updated iteratively by

$$Q(s,a) = Q(s,a) + \alpha[R + \beta \max_{a'} Q(s',a') - Q(s,a)], \quad \textbf{(26)}$$

where $\alpha \in [0,1]$ is the learning rate and $\beta$ is the discount factor. To train the DQN algorithm, there are two important part [26]:

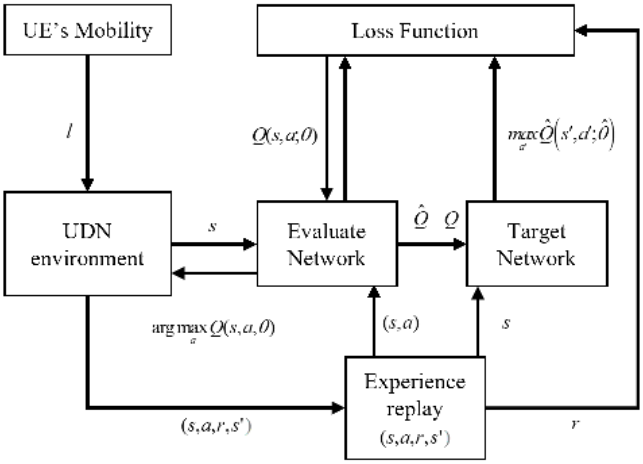


**Figure 2.** Workflow of DQN

(1) ***Experience Replay:*** In DQN, samples are obtained by interacting with the environment, which means that the non-correlation and static distribution between samples cannot be guaranteed. It stores the samples obtained by interacting with the environment and randomly selected some minibatch to train DQN when needed, thereby reducing the correlation between the samples.

(2) ***Target Network:*** Although the structure of target network and evaluate network is same, the parameters of target network is not up to date. The parameters $\hat{\theta}$ of the target network will be periodically replaced by the parameters of the evaluate network. Therefore, the target Q value remains unchanged, which reduces the correlation between the current Q value and the target Q value, and improves the stability of the algorithm.

The update of the Q value function in the evaluate network is basically the same as Q-learning. The parameters $\theta$ are updated in real time every time UE interact with the environment.

$$\theta = \theta + \alpha\left[R + \beta \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right], \nabla_\theta Q(s, a; \theta) \quad (27)$$

where $\nabla_\theta Q(s, a; \theta)$ means using gradient descent method to find the gradient of the Q value. The Eq. (27) can be rewrite as follows:

$$\theta = \theta + \alpha\left[R + \gamma \max_{a'} Q_T(s', a'; \theta_T) - Q(s, a; \theta)\right], \nabla_\theta Q(s, a; \theta) \quad (28)$$

where $Q_T(s', a'; \theta_T)$ means the parameters in the target network. The goal is to make the estimate Q value approach the target Q value. Thus, the loss function is defined as follows:

$$L(\theta) = \mathbb{E}[(R + \beta \max_{a'} Q(s', a'; \theta) - Q(s, a, \theta))^2]. \quad (29)$$

As shown in Figure 2, during the DQN training process, the user interacts with the environment to put the current state $s$, action $a$, reward $R$, and the next state $s'$ into the experience replay. The mobility of user will change the location $l$. Meanwhile, we input the current state $s$, action $a$ and reward $R$ into DQN for training the Q value. After training a certain number of times, the evaluate network randomly samples from the experience replay. During the training process, after a certain number of trainings, the parameters of the target network are replaced with the parameters of the evaluate network. By DQN, we can approximate the Q value so that the action taken can be selected by the maximum Q value.

The proposed DQN-based mobility management (DQNMM) is summarized in Algorithm 2. First, we initialize the parameters of DQN. Line 6 to line 12 is a DQN training process. In a training process, we use $\varepsilon$-greedy strategy to select an action $a_t$. Then, the user can get the experience $(s_t, a_t, r_t, s_{t+1})$ by interacting with the environment. The experirence replay is used to store the experience. Then, we use samples randomly sampled from experience replay to calculate the Q value of evaluate network and the weights $\theta$ of it are updated by calculating the loss function. At last, after every C training, the parameters in the target network are updated with the parameters in the evaluate network.

### 3.3 Mobility Management with Full Information

Consider the ideal situation where all the information of the network and UE is known. A mobility management with full information is simulated as a benchmark. In the scheme, the UE simply access to candidate BSs with optimal time delay. The proposed delay optimal greedy strategy (DOGS) is summarized in Algorithm 3.

**Algorithm 2.** DQN-based Mobility Management

1. Initialize experience replay $D$ to capacity $N$.
2. Initialize action-value function $Q$ with random weights $\theta$ and target action-value function $\hat{Q}$ with weights $\hat{\theta}$=0
3. **for** episode =1: M **do**
4. Initialize initial state sequence $s$
5. **for** t=1: T **do**
6. With probability $\varepsilon$ to select a random action $a_t$ otherwise select $a_t = \arg\max_a Q(s, a, \theta)$
7. Execute action $a_t$ and observe the experience $(s_t, a_t, r_t, s_{t+1})$
8. Set $s_{t+1} = s_t$
8. Store $(s_t, a_t, r_t, s_{t+1})$ into experience replay $D$
9. Sample random minibatch of $(s_j, a_j, r_j, s_{j+1})$ from experience replay $D$
10. Calculate the $Q$ value using

$$y_j = \begin{cases} R_j & \text{if terminates at } j+1 \\ R_j + \beta \max_{a'} \hat{Q}(s_{j+1}, a'; \hat{\theta}) & \text{otherwise} \end{cases}$$

11. Calculate the loss value by the loss function
12. Every $C$ steps reset $\hat{Q} = Q$
13. **end for**
14. **end for**

**Algorithm 3.** Delay Optimal Greedy Strategy

1. Initialize the UE by randomly accessing to the BSs
2. **for** task 1: $M$ **do**
3. Calculate the time delay for each available BS
4. Select BSs with the shortest time delay
5. **end for**

## 4 Simulations

In this section, we verified the proposed MM schemes through simulation and analyzed the proposed MM algorithms according to the experimental results. The average delay performance, the average energy consumption and the handover frequency of the proposed schemes are evaluated.

### 4.1 Simulation Setup

We simulate a typical UDN environment with BSs densely deployed. Each BS equips a MEC server which can provide computation capability. As shown in Figure 3, four BSs is densely deployed in a square area with a side length of 300 $m$ in which BSs are much more dense compare with current network. The coverage of a base station is 100 $m$. The trajectory of UE is generated by the classic random walk model with speed $v \in [5,10]$ $m/s$. The wireless channel gain is modeled as

$$H_{m,n} = 127 + 30\log_{10} d \quad \textbf{(30)}$$

as suggested in [27]. The cooperative power is $P_{cop} = 0.2W$.

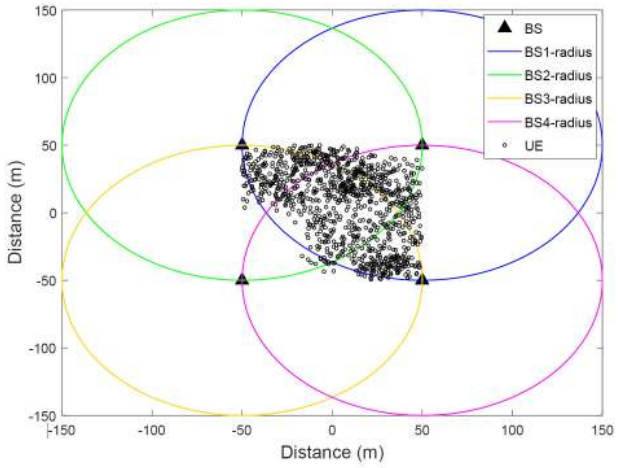


**Figure 3.** Network topology

Other simulation parameters are based on [9, 13], including channel bandwidth $W = 20Mhz$, noise power $\sigma^2 = 2\times10^{-13}W$ and transmit power $P_t = 0.5W$. Consider a video stream analysis with some tasks generated during the whole process. The size of every task is $\lambda = 100Mbits$ and the computation intensity is $\mu = 20\ cycles/bit$. The available CPU computation frequency is $f_{m,n} \in [0.5F, F]$, where $F = 25Ghz$.

## 4.2 Evaluation of DPMM

The proposed scheme is compared with DOSG scheme, which is delay optimal. From UE's perspective, time delay and energy consumption are two important performance indicators. As shown in Figure 4 and Figure 5, the average delay and the average energy consumption is evaluated. It can be seen that the delay performance of DPMM can get performance close to the delay performance of DOGS, sacrificing about 10% of the delay performance. Meanwhile, the energy consumption can be reduced through a more reasonable mobility management scheme.

When UE moves at high speed in a UDN environment, the handover frequency will seriously affect its service quality. As shown in Figure 6, the DPMM scheme can greatly reduce handover frequency compared with DOGS. The DPMM algorithm can make handover decisions based on the long-term state, instead of only considering the optimal at a certain state. Since the handover frequency of DPMM is lower, it can save more energy used by handover.

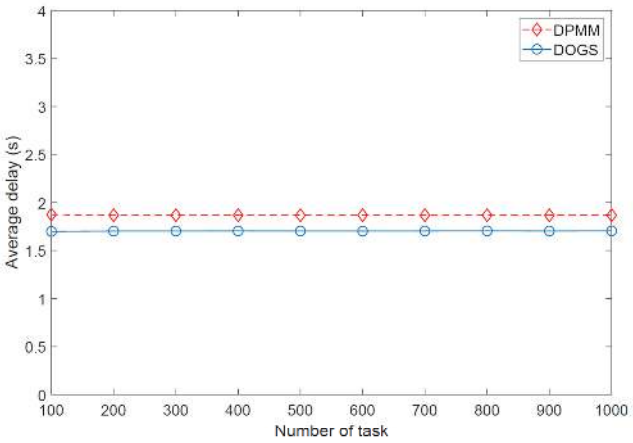


**Figure 4.** Average delay of DPMM

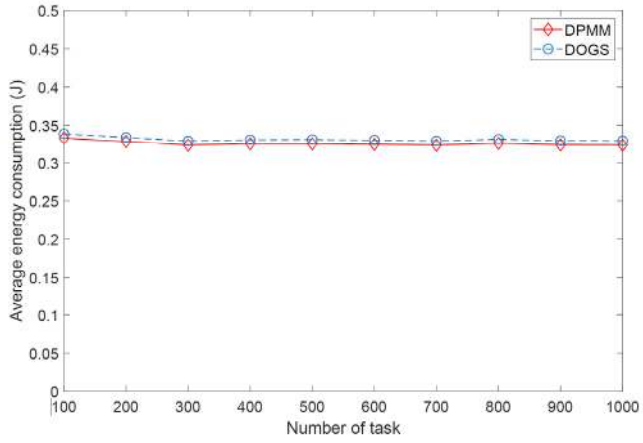


**Figure 5.** Average energy consumption of DPMM

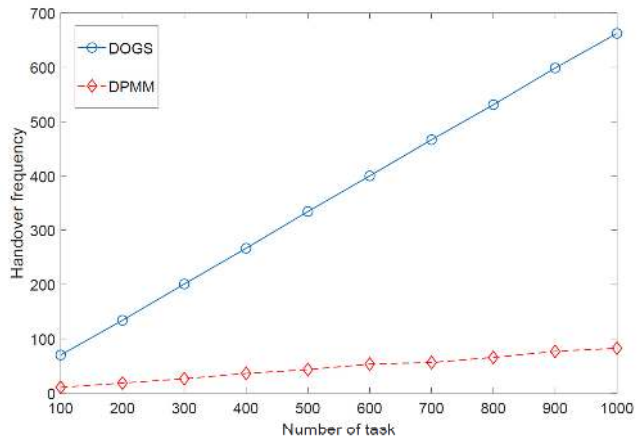


**Figure 6.** Handover performance of DPMM

In order to further reduce the delay of UE's tasks, we have added cooperative transmission MRC to the consideration of mobility management. As shown in Figure 7, the average delay is reduced by 50% by using MRC transmission. Figure 8 shows that the energy consumption of DPMM with MRC is 20% higher than its without MEC. The additional energy consumption is due to cooperative transmission. Therefore, by sacrificing a certain amount of energy, the delay performance can be greatly improved.

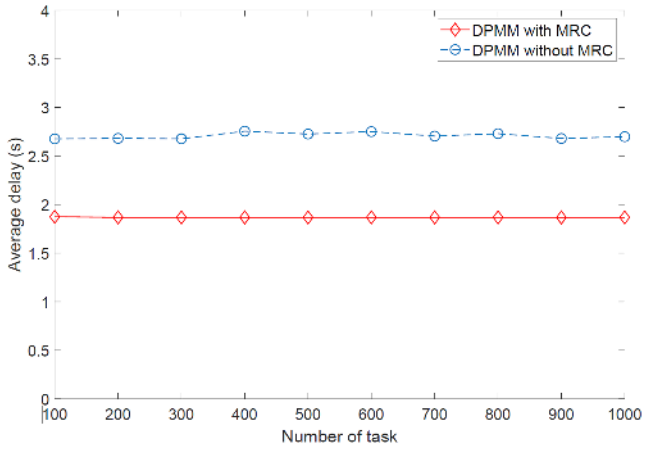


**Figure 7.** Average delay with MRC

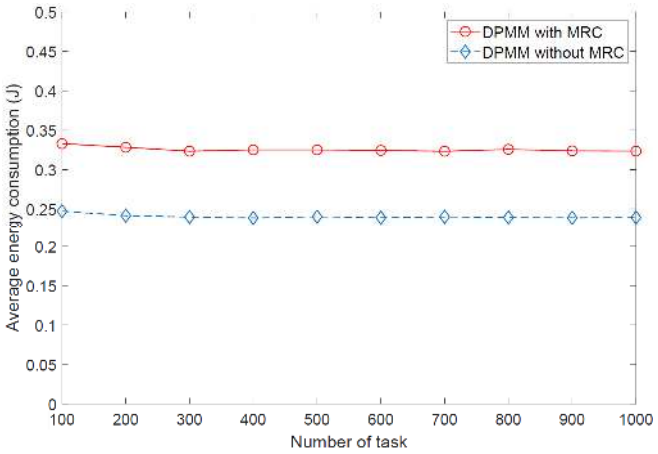


**Figure 8.** Average energy consumption with MRC

## 4.3 Evaluation of DQN

Simulation parameters used in training DQN are shown in Table 1. In particular, we gradually reduce the exploration rate $\varepsilon$ from 1 to 0.1 to adapt to the continuous changes in the environment during training. The number of tasks per episode is 1000.

**Table 1.** Parameters used in training DQN

| Parameters | Value |
|---|---|
| Memory size | 2000 |
| Batch size | 32 |
| Discount factor | 0.9 |
| Learning rate | 0.01 |
| Exploration rate | $1 \rightarrow 0.1$ |
| Tasks per episode | 1000 |

The time delay performance of DQN is shown in Figure 9 and Table 2. The performance of DQNMM is worse than DPMM and DOSG schemes. DQN algorithm gets corresponding rewards by interacting with the UDN environment, instead of knowing all the state information or statistic information. This leads to its time delay higher than that of DOGS and DPMM.

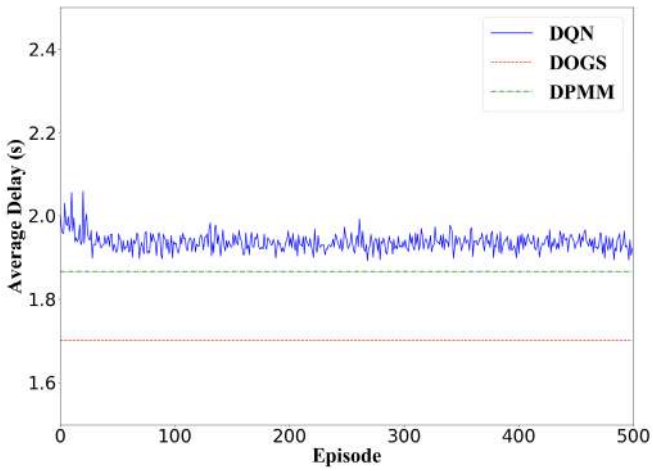


**Figure 9.** Average delay of DQN

**Table 2.** Average performance of DQN at different episode

| Episode | 50 | 100 | 300 | 400 |
|---|---|---|---|---|
| Delay (s) | 1.99 | 1.98 | 1.93 | 1.93 |
| Energy (J) | 0.340 | 0.344 | 0.357 | 0.357 |
| Handover | 682 | 423 | 255 | 253 |

Meanwhile, the delay performance converges after 100 episodes as shown in Figure 9. It means DQN can get a convergent mobility management scheme. Due to the continuous changes in the environment, DQN is still constantly learning, which causes the performance to fluctuate at the convergence.

As shown in Figure 10, the energy consumption of DQN is higher than DOGS and DPMM. Table 2 shows that the energy consumption of DQN increases with training DQN. The additional energy consumption compared with DPMM is caused by more frequent handover as shown in Figure 11. To get a delay optimal scheme, it sacrifices a certain amount of energy to access BSs with better computing performance.

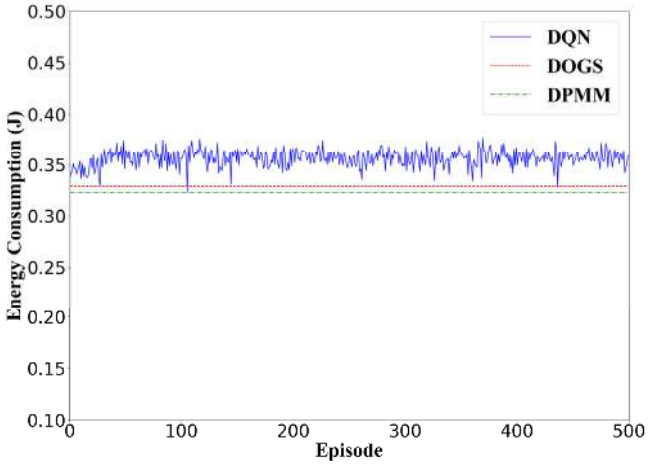


**Figure 10.** Average Energy Consumption of DQN

Finally, Figure 11 shows the handover performance of DQN. When handover is frequent, it can catch the changes of UDN environment, but the frequent handover increases the handover regret and degrades the delay performance. The handover frequency of DQN converge to about 353 per 1000 tasks, which is much lower than DOGS and higher than DPMM. This is because DQN needs to balance the effects of the handover frequency and UDN environment changes while DPMM uses system statistics to make decisions.

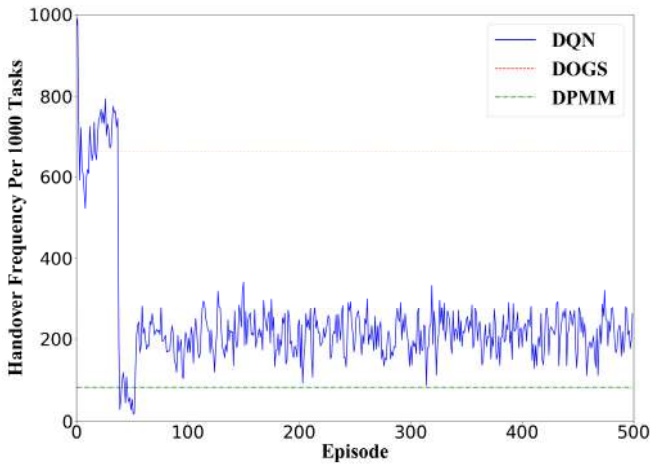


**Figure 11.** Handover Performance of DQN

## 5 Conclusion

In this paper, we propose a universal mobility management scheme called DPMM, taking both statistical information of user's high mobility and UDN environment into consideration. Our scheme learns offline mobility management decisions through user movement and the distribution of base station computing resources. Taking handover and cooperative data transmission into consideration, we prove that DPMM has greatly reduced handover times, and the delay performance is close to the optimal. Furthermore, a DQN-based mobility management scheme is proposed to learn online policy when the information of the network and the UE is not known in advance. Simulations shows that our proposed DQN-based algorithm can balance the handover frequency to a certain degree and get close to optimal results. Future research includes design mobility management schemes for multi-user scenarios and cooperative computing among BSs.

## Acknowledgements

This work was supported in part by the State's Key Project of Research and Development Plan under Grants 2019YFE0196400, in part by the NSFC under Grant 61771495, in part by the Guangdong R&D Project in Key Areas under Grant 2019B010158001, and in part by the Guangdong Provincial Special Fund For Modern Agriculture Industry Technology Innovation Teams under Grant 2020KJ122.

## References

[1] H. Gao, L. Kuang, Y. Yin, B. Guo, K. Dou, Mining Consuming Behaviors with Temporal Evolution for Personalized Recommendation in Mobile Marketing Apps, *Mobile Networks and Applications*, Vol. 25, No. 4, pp. 1233-1248, August, 2020.

[2] W. Junior, A. França, K. Dias, J. N. de Souza, Supporting Mobility-aware Computational Offloading in Mobile Cloud Environment, *Journal of Network and Computer Applications*, Vol. 94, pp. 93-108, September, 2017.

[3] B. Li, Z. Fei, Y. Zhang, UAV Communications for 5G and Beyond: Recent Advances and Future Trends, *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 2241-2263, April, 2019.

[4] B. Li, Z. Fei, Y. Zhang, M. Guizani, Secure UAV Communication Networks over 5G, *IEEE Wireless Communications*, Vol. 26, No. 5, pp. 114-120, October, 2019.

[5] D. López-Pérez, M. Ding, H. Claussen, A. H. Jafari, Towards 1 Gbps/UE in Cellular Systems: Understanding Ultra-Dense Small Cell Deployments, *IEEE Communications Surveys & Tutorials*, Vol. 17, No. 4, pp. 2078-2101, Fourth Quarter, 2015.

[6] M. Kamel, W. Hamouda, A. Youssef, Ultra-Dense Networks: A Survey, *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 4, pp. 2522-2545, Fourth Quarter, 2016.

[7] X. Ma, H. Gao, H. Xu, M. Bian, An IoT-based Task Scheduling Optimization Scheme Considering the Deadline and Cost-aware Scientific Workflow for Cloud Computing, *EURASIP Journal on Wireless Communications and Networking*, Vol. 2019, Article No. 249, November, 2019.

[8] I. Hwang, B. Song, S. S. Soliman, A Holistic View on Hyper-dense Heterogeneous and Small Cell Networks, *IEEE Communications Magazine*, Vol. 51, No. 6, pp. 20-27, June, 2013.

[9] Y. Sun, Y. Chang, M. Hu, T. Zeng, A Universal Predictive Mobility Management Scheme for Urban Ultra-Dense Networks with Control/Data Plane Separation, *IEEE Access*, Vol. 5, pp. 6015-6026, April, 2017.

[10] C. Shen, C. Tekin, M. van der Schaar, A Non-Stochastic Learning Approach to Energy Efficient Mobility Management, *IEEE Journal on Selected Areas in Communications*, Vol. 34, No. 12, pp. 3854-3868, December, 2016.

[11] J. Xu, Y. Sun, L. Chen, S. Zhou, E2M2: Energy Efficient Mobility Management in Dense Small Cells with Mobile Edge Computing, *2017 IEEE International Conference on Communications (ICC)*, Paris, France, 2017, pp. 1-6.

[12] H. Gao, W. Huang, Y. Duan, The Cloud-edge Based Dynamic Reconfiguration to Service Workflow for Mobile Ecommerce Environments: A QoS Prediction Perspective, *ACM Transactions on Internet Technology*, 2020, DOI: 10.1145/3391198.

[13] M. Simsek, M. Bennis, İ. Güvenç, Context-aware Mobility

Management in HetNets: A Reinforcement Learning Approach, *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, 2015, pp. 1536-1541.

[14] C. Wang, Z. Zhao, Q. Sun, H. Zhang, Deep Learning-Based Intelligent Dual Connectivity for Mobility Management in Dense Network, *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, 2018, pp. 1-5.

[15] Y. Sun, S. Zhou, 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, November, 2017.

[16] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, X. Wang, Context-Aware QoS Prediction with Neural Collaborative Filtering for Internet-of-Things Services, *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp. 4532-4542, May, 2020.

[17] J. Wang, K. Liu, M. Ni, J. Pan, Learning Based Mobility Management Under Uncertainties for Mobile Edge Computing, *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

[18] X. Ge, J. Ye, Y. Yang, Q. Li, User Mobility Evaluation for 5G Small Cell Networks Based on Individual Mobility Model, *IEEE Journal on Selected Areas in Communications*, Vol. 34, No. 3, pp. 528-541, March, 2016.

[19] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, A Survey on Mobile Edge Computing: The Communication Perspective, *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 4, pp. 2322-2358, Fourth Quarter, 2017.

[20] H. Gao, C. Liu, Y. Li, X. Yang, V2VR: Reliable Hybrid-Network-Oriented V2V Data Transmission and Routing Considering RSUs and Connectivity Probability, *IEEE Transactions on Intelligent Transportation Systems*, pp. 1-14, April, 2020.

[21] X. Zhang, N. C. Beaulieu, SER of Threshold-based Hybrid Selection/Maximal-ratio Combining in Correlated Nakagami Fading, *IEEE Transactions on Communications*, Vol. 53, No. 9, pp. 1423-1426, September, 2005.

[22] D. Tse, P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2015.

[23] J. Gong, S. Zhou, Z. Zhou, Networked MIMO with Fractional Joint Transmission in Energy Harvesting Systems, *IEEE Transactions on Communications*, Vol. 64, No. 8, pp. 3323-3336, August, 2016.

[24] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2005.

[25] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[26] Y. Ye, D. Qiu, D. Papadaskalopoulos, G. Strbac, A Deep Q Network Approach for Optimizing Offering Strategies in Electricity Markets, *2019 International Conference on Smart Energy Systems and Technologies (SEST)*, Porto, Portugal, 2019, pp. 1-6.

[27] C. Niu, Y. Li, R. Q. Hu, F. Ye, Fast and Efficient Radio Resource Allocation in Dynamic Ultra-Dense Heterogeneous Networks, *IEEE Access*, Vol. 5, pp. 1911-1924, February, 2017.

## Biographies

**Ziyue Zhang** received the B.E. and M.E. degrees in communication engineering from the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China, in 2018 and 2020, respectively. His research interests include deep learning algorithms, mobile management in 5G communications.

**Jie Gong** received his B.S. and Ph.D. degrees in Department of Electronic Engineering in Tsinghua University, Beijing, China, in 2008 and 2013, respectively. He is currently an associate professor in School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include optimization and dynamic programming.

**Xiang Chen** received the B.E. and Ph.D. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2002 and 2008, respectively. He is currently an associate professor at School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China. His research interests are mainly 5G wireless communications.

**Terng-Yin Hsu** received the B.S. and M.S. degrees from Feng Chia University, Taichung, Taiwan, in 1993 and 1995, respectively, and the Ph.D. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1999. He is currently a Professor at Department of Computer Science, National Chiao Tung University.