# Dynamic Service Migration in Mobile Edge-Clouds

Shiqiang Wang*, Rahul Urgaonkar†, Murtaza Zafer‡¶, Ting He†, Kevin Chan§, and Kin K. Leung*
*Imperial College London, United Kingdom, Email: {shiqiang.wang11, kin.leung}@imperial.ac.uk
†IBM T. J. Watson Research Center, Yorktown Heights, NY, United States, Email: {rurgaon, the}@us.ibm.com
‡Nyansa Inc., Palo Alto, CA, United States, Email: murtaza.zafer.us@ieee.org
§Army Research Laboratory, Adelphi, MD, United States, Email: kevin.s.chan.civ@mail.mil

***Abstract*—We study the dynamic service migration problem in mobile edge-clouds that host cloud-based services at the network edge. This offers the benefits of reduction in network overhead and latency but requires service migrations as user locations change over time. It is challenging to make these decisions in an optimal manner because of the uncertainty in node mobility as well as possible non-linearity of the migration and transmission costs. In this paper, we formulate a sequential decision making problem for service migration using the framework of Markov Decision Process (MDP). Our formulation captures general cost models and provides a mathematical framework to design optimal service migration policies. In order to overcome the complexity associated with computing the optimal policy, we approximate the underlying state space by the *distance* between the user and service locations. We show that the resulting MDP is exact for uniform one-dimensional mobility while it provides a close approximation for uniform two-dimensional mobility with a constant additive error term. We also propose a new algorithm and a numerical technique for computing the optimal solution which is significantly faster in computation than traditional methods based on value or policy iteration. We illustrate the effectiveness of our approach by simulation using real-world mobility traces of taxis in San Francisco.**

***Index Terms*—Cloud technologies, edge computing, Markov decision process (MDP), mobility, optimization, wireless networks**

## I. Introduction

Mobile applications that utilize cloud computing technologies have become increasingly popular over the recent years, with examples including data streaming, real-time video processing, etc. Such applications generally consist of a front-end component running on the mobile device and a back-end component running on the cloud [1], [2], where the cloud provides additional data processing and computational capabilities. With this architecture, it is possible to run complicated applications on handheld devices that have limited processing power. However, it also introduces new challenges to communication networks due to increased network overhead. The concept of *mobile edge-cloud (MEC)* has recently emerged as a promising technique to address these challenges. The core idea of MEC is to move computation closer to users, where small servers or data-centers that can host cloud applications are distributed across the network and connected directly to entities (such as cellular basestations) at the network edge, as shown in Fig. 1. This idea received notable commercial interest recently [3], and is expected to develop rapidly with the growth of new mobile applications and more advanced smartphones. MECs are also more robust than traditional centralized cloud computing systems [4], because they are distributed and are thus less impacted by failures at a centralized point. The idea of distributing cloud servers at the network edge is also known as cloudlets [4], edge computing [5], fog computing [6], follow me cloud [7], etc. In all these techniques, each server is responsible for a small geographical area, although some servers may not be directly connected to the basestation. In this paper, we focus on the case where MECs are colocated with the basestation, while noting that our proposed solution can be easily extended to the more general scenario.

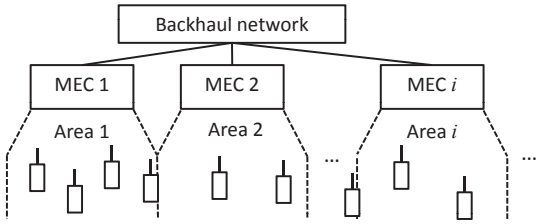



Figure 1. Application scenario of mobile edge-clouds (MECs).

One of the new challenges that MEC brings in is dynamic service placement/migration. As a user moves across different geographical areas, should its service be migrated out of the original MEC that hosts the service? If so, where should it be migrated? There is a tradeoff between the service migration cost and the transmission cost (such as communication delay and network overhead) between the user and the MEC. It is challenging to find the optimal decision also because of the uncertainty in user mobility as well as possible non-linearity of the migration and transmission costs.

Most existing work on service migration focuses on traditional cloud environments without explicitly considering the impact of user mobility. However, user mobility becomes a key factor in MECs. The performance of MECs with the presence of user mobility is studied in [8], but decisions on whether and where to migrate the service is not considered. A preliminary work on mobility-driven service migration based on Markov Decision Processes (MDPs) is given in [9], which mainly considers one-dimensional (1-D) mobility patterns with a specifically defined cost function. Standard solution procedures are used to solve this MDP, which can be time-consuming especially when the MDP has a large number of states. Due to real-time dynamics, the cost functions and transition probabilities of the MDP may change rapidly over time, thus it is desirable to solve the MDP in an effective manner. With this motivation, a more effective solution to the 1-D mobility case was proposed in [10], where the transmission and migration costs are assumed to be constants whenever transmission/migration occurs. To the best of our knowledge,

¶ Contributions of the author to this work are not related to his current employment at Nyansa Inc.

two-dimensional (2-D) mobility has not been considered in the literature, which is a much more realistic case compared to 1-D mobility. The service migration problem is also different from handover policies in cellular networks, because users can connect to MECs that are located at remote basestations (see [11] for more discussions).

In this paper, we use the MDP framework to study service migration in MECs. We provide novel contributions compared to [9] and [10], by considering general cost models, 2-D user mobility, and application to real-world traces. The details are summarized as follows.

*1)* Our formulation captures general cost models and provides a mathematical framework to design optimal service migration policies. We note that the resulting problem becomes difficult to solve due to the large state space. In order to overcome this challenge, we propose an approximation of the underlying state space by defining the states as the *distance* between the user and the service locations[1]. This approximation becomes exact for uniform 1-D mobility[2]. We prove several structural properties of the distance-based MDP, which includes a closed-form solution to the discounted sum cost. We leverage these properties to develop an algorithm for computing the optimal policy, which reduces the complexity from $O(N^3)$ (by policy iteration [12, Section 6]) to $O(N^2)$, where the number of states in the distance-based MDP is $N+1$.

*2)* We show how to use the distance-based MDP to approximate the solution for 2-D mobility models, which allows us to efficiently compute a service migration policy for 2-D mobility. For uniform 2-D mobility, the approximation error is bounded by a constant. Simulation results comparing our approximation solution to the optimal solution (where the optimal solution is obtained from a 2-D MDP) suggest that it performs very close to optimal, and the proposed approximation approach obtains the solution significantly faster.

*3)* We demonstrate how to apply our algorithms in a practical scenario driven by real mobility traces of taxis in San Francisco which consist of multiple users. We compare the proposed policy with several baseline strategies that include myopic, never-migrate, and always-migrate policies. It is shown that the proposed approach offers significant gains over those baseline approaches.

## II. Problem Formulation

Consider a mobile user that accesses a cloud-based service hosted on the MECs. The set of possible locations is given by $\mathcal{L}$, where $\mathcal{L}$ is assumed to be finite (but arbitrarily large). We consider a time-slotted model where the user's location remains fixed for the duration of one slot and changes from one slot to the next according to a Markovian mobility model. The time-slotted model can be regarded as a sampled version of a continuous-time model, and the sampling can be performed either at equal intervals over time or occur right after a cellular handover instance. In addition, we assume that each location

[1]Throughout this paper, we mean by *user location* the location of the basestation that the user is associated to.

[2]The 1-D mobility is an important practical scenario often encountered in transportation networks, such as vehicles moving along a road.

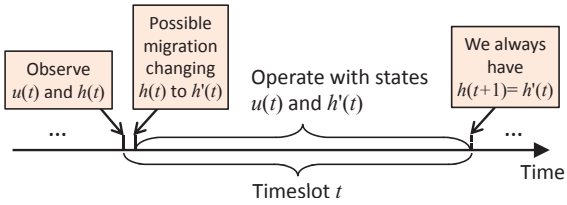



Figure 2. Timing of the proposed service migration mechanism.

$l \in \mathcal{L}$ is associated with an MEC that can host the service for the user. The locations in $\mathcal{L}$ are represented as 2-D vectors and there exists a distance metric $\|l_1 - l_2\|$ that can be used to calculate the distance between locations $l_1$ and $l_2$. Note that the distance metric may not be Euclidean distance. An example of this model is a cellular network in which the user's location is taken as the location of its current basestation and the MECs are co-located with the basestations. As shown in Section IV, these locations can be represented as 2-D vectors $(i, j)$ with respect to a reference location (represented by $(0, 0)$) and the distance between any two locations can be calculated in terms of the number of hops to reach from one cell to another cell. We denote the user and service locations at timeslot $t$ as $u(t)$ and $h(t)$ respectively.

*Remark:* Although we formulate the problem for the case of a single user accessing a single service, our solution can be applied to manage services for multiple users, as long as each user accesses a separate copy of the service. We will illustrate such an application in Section V.

The main notations in this paper are summarized in [11].

### A. Control Decisions and Costs

At the beginning of each slot, the MEC controller can choose from one of the following control options:

1) Migrate the service from location $h(t)$ to some other location $h'(t) \in \mathcal{L}$. This incurs a cost $c_m(x)$ that is assumed to be a non-decreasing function of $x$, where $x$ is the distance between $h(t)$ and $h'(t)$, i.e., $x = \|h(t) - h'(t)\|$. Once the migration is completed, the system operates under state $(u(t), h'(t))$. We assume that the time to perform migration is negligible compared to the time-scale of node mobility (as shown in Fig. 2).
2) Do not migrate the service. In this case, we have $h'(t) = h(t)$ and the migration cost is $c_m(0) = 0$.

In addition to the migration cost, there is a transmission cost incurred by the user for connecting to the currently active service instance. The transmission cost is related to the distance between the service and the user after possible migration, and it is defined as a general non-decreasing function $c_d(x)$, where $x = \|u(t) - h'(t)\|$. We set $c_d(0) = 0$.

### B. Performance Objective

Let us denote the overall system state at the beginning of each timeslot (before possible migration) by $s(t) = (u(t), h(t))$. The state $s(t)$ is named as the *initial state* of slot $t$. Consider any policy $\pi$ that makes control decisions based on the state $s(t)$ of the system, and we use $a_\pi(s(t))$ to represent the control action taken when the system is in state $s(t)$. This action causes the system to transition to a new

*intermediate state* $s'(t) = (u(t), h'(t)) = a_\pi(s(t))$. We also use $C_{a_\pi}(s(t))$ to denote the sum of migration and transmission costs incurred by a control $a_\pi(s(t))$ in slot $t$, and we have $C_{a_\pi}(s(t)) = c_m(\|h(t) - h'(t)\|) + c_d(\|u(t) - h'(t)\|)$. Starting from any initial state $s(0) = s_0$, the long-term expected discounted sum cost incurred by policy $\pi$ is given by

$$V_\pi(s_0) = \lim_{t\to\infty} \mathbb{E}\left\{ \sum_{\tau=0}^{t} \gamma^\tau C_{a_\pi}(s(\tau)) \middle| s(0) = s_0 \right\} \quad (1)$$

where $0 < \gamma < 1$ is a discount factor.

Our objective is to design a control policy that minimizes the long-term expected discounted sum total cost starting from any initial state, i.e.,

$$V^*(s_0) = \min_\pi V_\pi(s_0) \quad \forall s_0. \quad (2)$$

This problem falls within the class of MDPs with infinite horizon discounted cost. It is well known that the optimal solution is given by a stationary policy and can be obtained as the unique solution to the Bellman's equation:

$$V^*(s_0) = \min_a \Big\{ C_a(s_0) + \gamma \sum_{s_1 \in \mathcal{L}\times\mathcal{L}} P_{a(s_0),s_1} V^*(s_1) \Big\} \quad (3)$$

where $P_{a(s_0),s_1}$ denotes the probability of transitioning from state $s'(0) = s'_0 = a(s_0)$ to $s(1) = s_1$. Note that the intermediate state $s'(t)$ has no randomness when $s(t)$ and $a(\cdot)$ are given, thus we only consider the transition probability from $s'(t)$ to the next state $s(t+1)$ in (3). Also note that we always have $h(t+1) = h'(t)$.

### C. Characteristics of Optimal Policy

We next characterize some structural properties of the optimal solution. The following lemma states that it is not optimal to migrate the service to a location that is farther away from the user, as one would intuitively expect.

**Lemma 1.** *Let $a^*(s) = (u, h')$ denote the optimal action at any state $s = (u, h)$. Then, we have $\|u - h'\| \le \|u - h\|$. (If the optimal action is not unique, then there exists at least one such optimal action.)*

**Corollary 1.** *If $c_m(x)$ and $c_d(x)$ are both constants (possibly of different values) for $x > 0$, and $c_m(0) < c_m(x)$ and $c_d(0) < c_d(x)$ for $x > 0$, then migrating to locations other than the current location of the mobile user is not optimal.*

See [11] for the proofs of Lemma 1 and Corollary 1.

### D. Simplifying the Search Space

Lemma 1 simplifies the search space for the optimal policy considerably. However, it is still very challenging to derive the optimal control policy for the general model presented above, particularly when the state space $\{s(t)\}$ is large. One possible approach to address this challenge is to re-define the state space to represent only the *distance* between the user and service locations $d(t) = \|u(t) - h(t)\|$. The motivation for this comes from the observation that the cost functions in our model depend only on the distance. Note that in general, the optimal control actions can be different for two states $s_0$ and $s_1$ that have the same user-service distance. However, it is reasonable to use the distance as an approximation of the state space for many practical scenarios of interest, and this simplification allows us to formulate a far more tractable MDP. We discuss the distance-based MDP in the next section, and show how the results on the distance-based MDP can be applied to 2-D mobility models and real-world mobility traces in Sections IV and V.

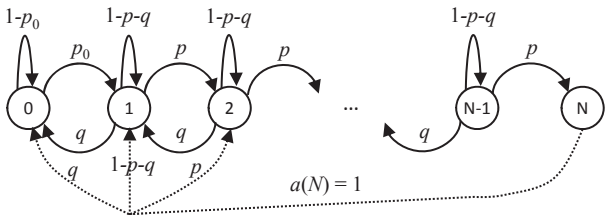



Figure 3. An example of distance-based MDP with the distances $\{d(t)\}$ (before possible migration) as states. In this example, migration is only performed at state $N$, and only the possible action of $a(N) = 1$ is shown for compactness. The solid lines denote state transitions without migration.

In the remainder of this paper, where there is no ambiguity, we reuse the notations $P$, $C_a(\cdot)$, $V(\cdot)$, and $a(\cdot)$ to respectively represent transition probabilities, one-timeslot costs, discounted sum costs, and actions of different MDPs.

## III. Optimal Policy for Distance-Based MDP

In this section, we consider a distance-based[3] MDP where the states $\{d(t)\}$ represent the distances between the user and the service before possible migration (an example is shown in Fig. 3), i.e., $d(t) = \|u(t) - h(t)\|$. We define the parameter $N$ as an application-specific maximum allowed distance, and we always perform migration when $d(t) \ge N$. We set the actions $a(d(t)) = a(N)$ for $d(t) > N$, so that we only need to focus on the states $d(t) \in [0, N]$. After taking action $a(d(t))$, the system operates in the intermediate state $d'(t) = a(d(t))$, and the value of the next state $d(t+1)$ follows the transition probability $P_{d'(t),d(t+1)}$ which is related to the mobility model of the user. To simplify the solution, we restrict the transition probabilities $P_{d'(t),d(t+1)}$ according to the parameters $p_0$, $p$, and $q$ as shown in Fig. 3. Such a restriction is sufficient when the underlying mobility model is uniform 1-D random walk where the user moves one step to the left or right with equal probability $r_1$ and stays in the same location with probability $1 - 2r_1$, in which case we can set $p = q = r_1$ and $p_0 = 2r_1$. It is also sufficient to approximate the uniform 2-D random walk model, as will be discussed in Section IV-B.

For an action of $d'(t) = a(d(t))$, the new service location $h'(t)$ is chosen such that $\|h(t) - h'(t)\| = |d(t) - d'(t)|$ and $\|u(t) - h'(t)\| = d'(t)$. This means that migration happens along the shortest path that connects $u(t)$ and $h(t)$, and $h'(t)$ is on this shortest path (also note that $d'(t) \le d(t)$ according to Lemma 1). Such a migration is possible for the 1-D case where $u(t)$, $h(t)$, and $h'(t)$ are all scalar values. It is also possible for the 2-D case if the distance metric is properly defined (see Section IV-B for details). The one-timeslot cost is then $C_a(d(t)) = c_m(|d(t) - d'(t)|) + c_d(d'(t))$. We define the cost functions $c_m(x)$ and $c_d(x)$ in a constant-plus-exponential

[3]We assume that the distance is quantized, as it will be the case with the 2-D model discussed in later sections.

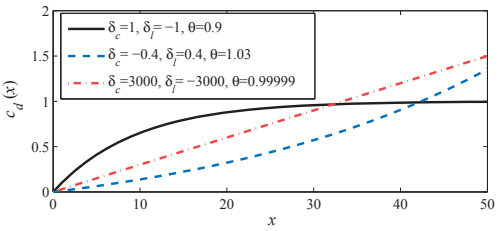



Figure 4. Example of exponential cost function $c_d(x)$.

form (shown in (4) and (5) below). Such a cost function can have different shapes and are thus applicable to many realistic scenarios. It also has nice properties allowing obtain a closed-form solution to the discounted sum cost, based on which we design an efficient algorithm for finding the optimal policy.

### A. Constant-Plus-Exponential Cost Functions

The constant-plus-exponential cost functions are defined as

$$c_m(x) = \begin{cases} 0, & \text{if } x = 0 \\ \beta_c + \beta_l \mu^x, & \text{if } x > 0 \end{cases} \tag{4}$$

$$c_d(x) = \begin{cases} 0, & \text{if } x = 0 \\ \delta_c + \delta_l \theta^x, & \text{if } x > 0 \end{cases} \tag{5}$$

where $\beta_c$, $\beta_l$, $\delta_c$, $\delta_l$, $\mu$, and $\theta$ are real-valued parameters. Note that $x = |d(t) - d'(t)|$ in $c_m(x)$, and $x = d'(t)$ in $c_d(x)$. The values of these parameters are selected such that $c_m(x) \geq 0$, $c_d(x) \geq 0$, and both $c_m(x)$ and $c_d(x)$ are non-decreasing in $x$ for $x \geq 0$. Explicitly, we have $\mu \geq 0$; $\beta_l \leq 0$ when $\mu \leq 1$; $\beta_l \geq 0$ when $\mu \geq 1$; $\beta_c \geq -\beta_l$; $\theta \geq 0$; $\delta_l \leq 0$ when $\theta \leq 1$; $\delta_l \geq 0$ when $\theta \geq 1$; and $\delta_c \geq -\delta_l$. We set $c_m(0) = c_d(0) = 0$ for convenience, because a non-zero cost for $x = 0$ can be offset by the values of $\beta_c$ and $\delta_c$, thus setting $c_m(0) = c_d(0) = 0$ does not affect the optimal decision.

With this definition, the values of $\beta_c + \beta_l$ and $\delta_c + \delta_l$ can be regarded as constant terms of the costs, at least such an amount of cost is incurred when $x > 0$. The parameters $\mu$ and $\theta$ specify the impact of the distance $x$ to the costs, and their values can be related to the network topology and routing mechanism of the network. The parameters $\beta_l$ and $\delta_l$ further adjust the costs proportionally.

An example of the cost function is shown in Fig. 4. This exponential cost function can be used to approximate an arbitrary cost function as discussed in [11].

### B. Closed-Form Solution to the Discounted Sum Cost

*1) Problem Formulation with Difference Equations:* From (1), we get the following balance equation on the discounted sum cost for a given policy $\pi$:

$$V_\pi(d(0)) = C_{a_\pi}(d(0)) + \gamma \sum_{d(1)=a_\pi(d(0))-1}^{a_\pi(d(0))+1} P_{a_\pi(d(0)),d(1)} V_\pi(d(1)). \tag{6}$$

In the following, we will omit the subscript $\pi$ and write $d(0)$ as $d$ for short.

**Proposition 1.** *For a given policy $\pi$, let $\{n_k : k \geq 0\}$ denote the series of migration states (i.e. $a(n_k) \neq n_k$) as specified by policy $\pi$, where $0 \leq n_k \leq N$. The discounted sum cost $V(d)$ for policy $\pi$ can be expressed as*

$$V(d) = A_k m_1^d + B_k m_2^d + D + \begin{cases} H \cdot \theta^d & \text{if } 1 - \frac{\phi_1}{\theta} - \phi_2\theta \neq 0 \\ Hd \cdot \theta^d & \text{if } 1 - \frac{\phi_1}{\theta} - \phi_2\theta = 0 \end{cases} \tag{7}$$

*where the coefficients $m_1$, $m_2$, $D$, and $H$ are expressed as follows:*

$$m_1 = \frac{1 + \sqrt{1 - 4\phi_1\phi_2}}{2\phi_2}, m_2 = \frac{1 - \sqrt{1 - 4\phi_1\phi_2}}{2\phi_2} \tag{8}$$

$$D = \frac{\phi_3}{1 - \phi_1 - \phi_2} \tag{9}$$

$$H = \begin{cases} \frac{\phi_4}{1 - \frac{\phi_1}{\theta} - \phi_2\theta} & \text{if } 1 - \frac{\phi_1}{\theta} - \phi_2\theta \neq 0 \\ \frac{\phi_4}{\frac{\phi_1}{\theta} - \phi_2\theta} & \text{if } 1 - \frac{\phi_1}{\theta} - \phi_2\theta = 0 \end{cases} \tag{10}$$

*where we define $\phi_1 \triangleq \frac{\gamma q}{1-\gamma(1-p-q)}$, $\phi_2 \triangleq \frac{\gamma p}{1-\gamma(1-p-q)}$, $\phi_3 \triangleq \frac{\delta_c}{1-\gamma(1-p-q)}$, and $\phi_4 \triangleq \frac{\delta_l}{1-\gamma(1-p-q)}$.*

*The constants $A_k$ and $B_k$ are related to the interval $[0, n_0]$ ($k = 0$) or $[n_{k-1}, n_k]$ ($k > 0$).*

*Proof.* The proof is based on solving a difference equation according to (6), see [11] for details. □

We also note that for two different states $d_1$ and $d_2$, if the policy $\pi$ has actions $a_\pi(d_1) = d_2$ and $a_\pi(d_2) = d_2$, then

$$V_\pi(d_1) = c_m(|d_1 - d_2|) + V_\pi(d_2). \tag{11}$$

*2) Finding the Coefficients:* The coefficients $A_k$ and $B_k$ are unknowns in the solution (7) that need to be evaluated using additional constraints. These coefficients may be different for states within different intervals of $[n_{k-1}, n_k]$. After these coefficients are determined, (7) holds for all $d \in [n_{k-1}, n_k]$.

We assume $1 - \frac{\phi_1}{\theta} - \phi_2\theta \neq 0$ and $1 - \frac{\phi_2}{\theta} - \phi_1\theta \neq 0$ from now on, the other cases can be derived in a similar way.

*Coefficients for $0 \leq d \leq n_0$:* One constraint is from the balance equation (6) for $d = 0$, which is

$$V(0) = \gamma p_0 V(1) + \gamma(1 - p_0)V(0). \tag{12}$$

By substituting (7) into (12), we get

$$A_0(1 - \phi_0 m_1) + B_0(1 - \phi_0 m_2) = D(\phi_0 - 1) + H(\phi_0\theta - 1) \tag{13}$$

where $\phi_0 \triangleq \frac{\gamma p_0}{1-\gamma(1-p_0)}$. Another constraint is obtained by substituting (7) into (11), which gives

$$\begin{aligned} &A_0\left(m_1^{n_0} - m_1^{a(n_0)}\right) + B_0\left(m_2^{n_0} - m_2^{a(n_0)}\right) \\ &= \beta_c + \beta_l \mu^{n_0 - a(n_0)} - H\left(\theta^{n_0} - \theta^{a(n_0)}\right) \end{aligned} \tag{14}$$

The values of $A_0$ and $B_0$ can be solved from (13) and (14).

*Coefficients for $n_{k-1} \leq d \leq n_k$:* Assume that we have found $V(d)$ for all $d \leq n_{k-1}$. By letting $d = n_{k-1}$ in (7), we have the first constraint given by

$$A_k m_1^{n_{k-1}} + B_k m_2^{n_{k-1}} = V(n_{k-1}) - D - H \cdot \theta^{n_{k-1}}. \tag{15}$$

For the second constraint, we consider two cases. If $a(n_k) \leq$

$n_{k-1}$, then

$$A_k m_1^{n_k} + B_k m_2^{n_k} = \beta_c + \beta_l \mu^{n_k - a(n_k)} + V(a(n_k)) - D - H \cdot \theta^{n_k}. \quad (16)$$

If $n_{k-1} < a(n_k) \leq n_k - 1$, then

$$A_k \left( m_1^{n_k} - m_1^{a(n_k)} \right) + B_k \left( m_2^{n_k} - m_2^{a(n_k)} \right) = \beta_c + \beta_l \mu^{n_k - a(n_k)} - H \left( \theta^{n_k} - \theta^{a(n_k)} \right). \quad (17)$$

The values of $A_k$ and $B_k$ can be solved from (15) together with either (16) or (17).

*3) Solution is in Closed-Form:* We note that $A_0$ and $B_0$ can be expressed in closed-form, and $A_k$ and $B_k$ for all $k$ can also be expressed in closed-form by substituting (7) into (15) and (16) where necessary. Therefore, (7) is a *closed-form solution* for all $d \in [0, N]$. Numerically, we can find $V(d)$ for all $d \in [0, N]$ in $O(N)$ time.

### C. Algorithm for Finding the Optimal Policy

Note that standard approaches of solving for the optimal policy of an MDP include value iteration and policy iteration [12, Section 6]. Value iteration finds the optimal policy from the Bellman's equation (3) iteratively, which may require a large number of iterations before converging to the optimal result. Policy iteration generally requires a smaller number of iterations, because, in each iteration, it finds the exact values of the discounted sum cost $V(d)$ for the policy resulting from the previous iteration, and performs the iteration based on the exact $V(d)$ values. However, in general, the exact $V(d)$ values are found by solving a system of linear equations, which has a complexity of $O(N^3)$ when using Gaussian-elimination.

We propose a modified policy-iteration approach for finding the optimal policy, which uses the above result instead of Gaussian-elimination to compute $V(d)$, and also only checks for migrating to lower states or not migrating (according to Lemma 1). The algorithm is shown in Algorithm 1, where Lines 4–7 find the values of $n_k$, Lines 8–17 find the discounted sum cost values, and Lines 18–20 update the optimal policy. The overall complexity for each iteration is $O\left(N^2\right)$ in Algorithm 1, which reduces complexity because standard[4] policy iteration has complexity $O(N^3)$, and the standard value iteration approach does not compute the exact value function in each iteration and generally has long convergence time.

**Algorithm 1** Modified policy-iteration algorithm based on difference equations

1: Initialize $a(d) = 0$ for all $d = 0, 1, 2, ..., N$
2: Find constants $\phi_0$, $\phi_1$, $\phi_2$, $\phi_3$, $\phi_4$, $m_1$, $m_2$, $D$, and $H$
3: **repeat**
4: $k \leftarrow 0$
5: **for** $d = 1...N$ **do**
6: **if** $a(d) \neq d$ **then**
7: $n_k \leftarrow d$, $k \leftarrow k + 1$
8: **for** all $n_k$ **do**
9: **if** $k = 0$ **then**
10: Solve for $A_0$ and $B_0$ from (13) and (14)
11: Find $V(d)$ with $0 \leq d \leq n_k$ from (7) with $A_0$ and $B_0$ found above
12: **else if** $k > 0$ **then**
13: **if** $a(n_k) \leq n_{k-1}$ **then**
14: Solve for $A_k$ and $B_k$ from (15) and (16)
15: **else**
16: Solve for $A_k$ and $B_k$ from (15) and (17)
17: Find $V(d)$ with $n_{k-1} < d \leq n_k$ from (7) with $A_k$ and $B_k$ found above
18: **for** $d = 1...N$ **do**
19: $a_{\text{prev}}(d) = a(d)$
20: $a(d) = \arg\min_{a \leq d} \left\{ C_a(d) + \gamma \sum_{j=a-1}^{a+1} P_{aj} V(j) \right\}$
21: **until** $a_{\text{prev}}(d) = a(d)$ for all $d$
22: **return** $a(d)$ for all $d$

## IV. Approximate Solution for 2-D Mobility Model

In this section, we show that the distance-based MDP can be used to find a near-optimal service migration policy, where the user conforms to a uniform 2-D random walk mobility model on infinite space. This mobility model can be used to approximate real-world mobility traces (see Section V). We consider a hexagon cell structure, but the approximation procedure can also be used for other 2-D mobility models (such as Manhattan grid) with some parameter changes. The user is assumed to transition to one of its six neighboring cells at the beginning of each timeslot with probability $r$, and stay in the same cell with probability $1 - 6r$.

[4]We use the term "standard" here to distinguish with the modified policy iteration mechanism proposed in Algorithm 1.

### A. Offset-Based MDP

Define the *offset* of the user from the service as a 2-D vector $e(t) = u(t) - h(t)$ (recall that $u(t)$ and $h(t)$ are also 2-D vectors). Due to the space-homogeneity of the mobility model, it is sufficient to model the state of the MDP by $e(t)$ rather than $s(t)$. The distance metric $\|l_1 - l_2\|$ is defined as the minimum number of hops that are needed to reach from cell $l_1$ to cell $l_2$ on the hexagon model.

We name the states with the same value of $\|e(t)\|$ as a *ring*, and express the states $\{e(t)\}$ with polar indexes $(i, j)$, where the first index $i$ refers to the ring index, and the second index $j$ refers to each of the states within the ring, as shown in Fig. 5. For $e(t) = (i, j)$, we have $\|e(t)\| = i$. If $u(t) = h(t)$ (i.e., the actual user and service locations (cells) are the same), then we have $e(t) = (0, 0)$ and $\|e(t)\| = 0$.

Similarly as in the distance-based MDP, we assume in the 2-D MDP that we always migrate when $\|e(t)\| \geq N$, where $N$ is a design parameter, and we only consider the state space $\{e(t)\}$ with $\|e(t)\| \leq N$. The system operates in the intermediate state $e'(t) = u(t) - h'(t) = a(e(t))$ after taking action $a(e(t))$. The next state $e(t+1)$ is determined probabilistically according to the transition probability $P_{e'(t),e(t+1)}$. We have $P_{e'(t),e(t+1)} = 1 - 6r$ when $e(t+1) = e'(t)$; $P_{e'(t),e(t+1)} = r$ when $e(t+1)$ is a neighbor of $e'(t)$; and $P_{e'(t),e(t+1)} = 0$ otherwise. Note that we always have $e(t) - e'(t) = h'(t) - h(t)$, so the one-timeslot cost is $C_a(e(t)) = c_m(\|e(t) - e'(t)\|) + c_d(\|e'(t)\|)$.

We note that, even after simplification with the offset model, the 2-D offset-based MDP has a significantly larger number of

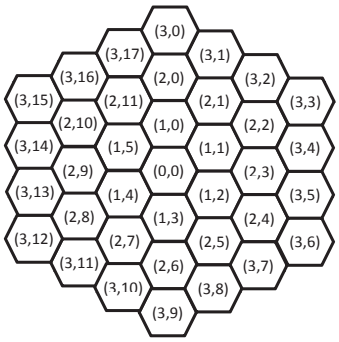



Figure 5. Example of 2-D offset model on hexagon cells, where $N = 3$.

states compared with the distance-based MDP, because for a distance-based model with $N$ states (excluding state zero), the 2-D offset model has $M = 3N^2 + 3N$ states (excluding state $(0, 0)$). Therefore, we use the distance-based MDP proposed in Section III to approximate the 2-D offset-based MDP, which significantly reduces the computational time as shown in Section IV-D.

### *B. Approximation by Distance-based MDP*

In the approximation, the parameters of the distance-based MDP are chosen as $p_0 = 6r$, $p = 2.5r$, and $q = 1.5r$. The intuition of the parameter choice is that, at state $(i'_0, j'_0) = (0, 0)$ in the 2-D MDP, the aggregate probability of transitioning to any state in ring $i_1 = 1$ is $6r$, so we set $p_0 = 6r$; at any other state $(i'_0, j'_0) \neq (0, 0)$, the aggregate probability of transitioning to any state in the higher ring $i_1 = i'_0 + 1$ is either $2r$ or $3r$, and the aggregate probability of transitioning to any state in the lower ring $i_1 = i'_0 - 1$ is either $r$ or $2r$, so we set $p$ and $q$ to the median value of these transition probabilities.

To find the optimal policy for the 2-D MDP, we first find the optimal policy for the distance-based MDP with the parameters defined above. Then, we map the optimal policy from the distance-based MDP to a policy for the 2-D MDP. To explain this mapping, we note that, in the 2-D hexagon offset model, there always exists at least one shortest path from any state $(i, j)$ to ring $i'$, the length of this shortest path is $|i - i'|$, and each ring between $i$ and $i'$ is traversed once on the shortest path. For example, one shortest path from state $(3, 2)$ to ring $i' = 1$ is $(3, 2), (2, 1), (1, 0)$. When the system is in state $(i, j)$ and the optimal action from the distance-based MDP is $a^*(i) = i'$, we perform migration on the shortest path from $(i, j)$ to ring $i'$. If there exist multiple shortest paths, one path is arbitrarily chosen. For example, if $a(3) = 2$ in the distance-based MDP, then either $a(3, 2) = (2, 1)$ or $a(3, 2) = (2, 2)$ in the 2-D MDP. With this mapping, the one-timeslot cost $C_a(d(t))$ for the distance-based MDP and the one-timeslot cost $C_a(e(t))$ for the 2-D MDP are the same.

### *C. Bound on Approximation Error*

Error in the approximation arises because the transition probabilities in the distance-based MDP are not exactly the same as that in the 2-D MDP (there is at most a difference of $0.5r$). In this subsection, we study the difference in the discounted sum costs when using the policy obtained from the distance-based MDP and the true optimal policy for the 2-D MDP. The result is summarized as Proposition 2.

**Proposition 2.** *Let $V_{dist}(e)$ denote the discounted sum cost when using the policy from the distance-based MDP, and let $V^*(e)$ denote the discounted sum cost when using true optimal policy of the 2-D MDP, then we have $V_{dist}(e) - V^*(e) \leq \frac{\gamma r k}{1-\gamma}$ for all $e$, where $k \triangleq \max_x \{c_m(x + 2) - c_m(x)\}$.*

*Proof.* (Outline) The proof is completed in three steps. First, we modify the states of the 2-D MDP in such a way that the aggregate transition probability from any state $(i'_0, j'_0) \neq (0, 0)$ to ring $i_1 = i'_0 + 1$ (correspondingly, $i_1 = i'_0 - 1$) is $2.5r$ (correspondingly, $1.5r$). We assume that we use a given policy on both the original and modified 2-D MDPs, and show a bound on the difference in the discounted sum costs for these two MDPs. In the second step, we show that the modified 2-D MDP is equivalent to the distance-based MDP. This can be intuitively explained by the reason that the modified 2-D MDP has the same transition probabilities as the distance-based MDP when only considering the ring index $i$, and also, the one-timeslot cost $C_a(e(t))$ only depends on $\|e(t) - a(e(t))\|$ and $\|a(e(t))\|$, both of which can be determined from the ring indexes of $e(t)$ and $a(e(t))$. The third step uses the fact that the optimal policy for the distance-based MDP cannot bring higher discounted sum cost for the distance-based MDP (and hence the modified 2-D MDP) than any other policy. By utilizing the error bound found in the first step twice, we prove the result. For details of the proof, see [11]. □

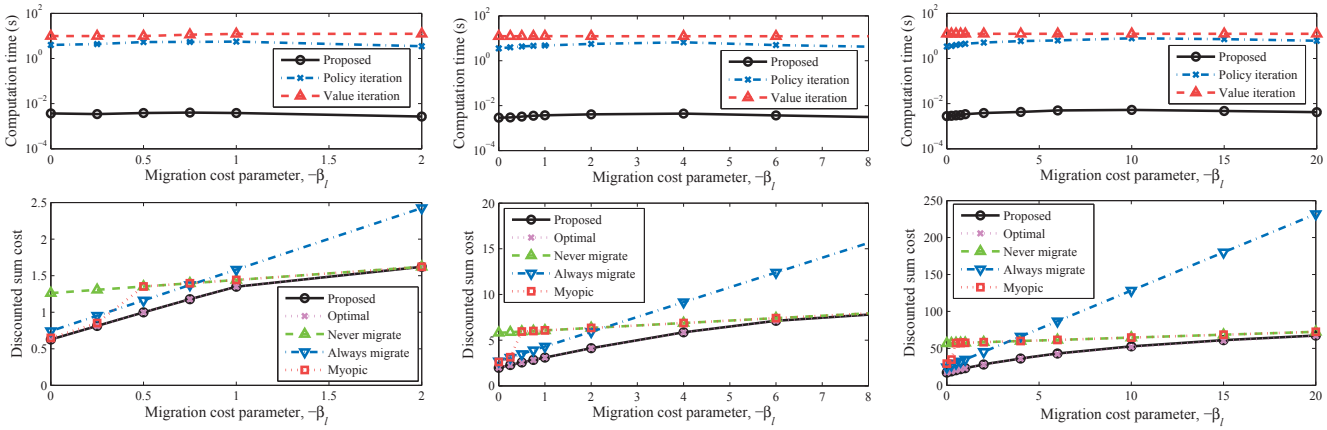
The error bound is a constant value when the parameters are given. It increases with $\gamma$. However, note that the absolute value of the discounted sum cost also increases with $\gamma$, so the relative error can remain low.

### *D. Numerical Evaluation*

The error bound derived in Section IV-C is a worst-case upper bound of the error. In this subsection, we evaluate the performance of the proposed approximation method numerically, and focus on the average performance of the approximation.

We consider 2-D random walk mobility with randomly chosen parameter $r$. The maximum user-service distance is set as $N = 10$. The transmission cost function parameters are selected as $\theta = 0.8$, $\delta_c = 1$, and $\delta_l = -1$. With these parameters, we have $\delta_c + \delta_l = 0$, which means that there is no constant portion in the cost function. For the migration cost, we choose $\mu = 0.8$ and fix $\beta_c + \beta_l = 1$ to represent a constant server processing cost for migration. The parameter $\beta_l \leq 0$ takes different values in the simulations, to represent different sizes of data to be migrated.

The simulations are performed in MATLAB on a computer with Intel Core i7-2600 CPU, 8GB memory, and 64-bit Windows 7. We study the computation time (i.e., the time used to run the algorithm) and the discounted sum cost of the proposed approach that is based on approximating the original 2-D MDP with the distance-based MDP. For the computation time comparison, standard value and policy iteration approaches [12, Section 6] are used to solve the original 2-D MDP. The discounted sum cost from the proposed approach is compared with the costs from alternative policies, including the true optimal policy from standard policy iteration on the 2-D model, the never-migrate policy which never migrates except




(a) (b) (c)

Figure 6. Simulation result with 2-D random walk: (a) $\gamma = 0.5$, (b) $\gamma = 0.9$, (c) $\gamma = 0.99$.

when at states in the outer ring $i = N$, the always-migrate policy which always migrates when the user and the service are at different locations, and the myopic policy that chooses actions to minimize the one-timeslot cost.

The simulations are run with 50 different random seeds, and the overall results are shown in Fig. 6 with different values of the discount factor $\gamma$. We can see that the proposed method brings discounted sum costs that are very close to the optimum. Meanwhile, the computation time when using the proposed method is only about $0.1\%$ of the computation time of standard value and policy iteration approaches. For further discussion on the simulation results, see [11].

## V. Application to Real-World Scenarios

In this section, we discuss how the aforementioned approaches can be applied to service migration in the real world, where multiple users and services are involved.

### A. Operating Procedure

In the real-world scenario, we assume that each user follows a sample path of the 2-D hexagon mobility model. The parameter $r$ is estimated from the sample paths of multiple users (discussed in details in Section V-A2). The cost parameters $\beta_c$, $\beta_l$, $\mu$, $\delta_c$, $\delta_l$, and $\theta$ are selected based on the actual application scenario, and their values vary with the background traffic load of the network and MEC servers. By appropriately adjusting the cost functions according to the load, the proposed approach can be used for load balancing; we will demonstrate how to achieve load balancing by a proper selection of cost parameters in Section V-B. The discount factor $\gamma$ is selected based on the duration of services, and a larger $\gamma$ is selected for a longer service duration. Such a selection is because the discount factor $\gamma$ determines the amount of time to look ahead, if a user only requires the service for a short time, then there is no need to consider the cost for the long-term future.

*1) Initial Service Placement:* Upon initialization, the service is placed onto the MEC that is connected to the same basestation as the user is currently connected to. In other words, when a service starts, the user and the service are in the same location, i.e., state $(0, 0)$ of the 2-D MDP. This is due to the consideration that the initialization cost and the cost of further operation can be minimized by initially placing the service closest to the user.

*2) Dynamic Service Migration:* After the initial service placement, the subsequent actions of possible service migration is performed according to the optimal policy found from the mechanisms proposed earlier in this paper.

To facilitate the mapping between the real-world and the MDP model, we define a timeslot-length $T$ which is the same for all MECs. The parameter $T$ can be regarded as a protocol parameter, and different MECs do *not* need to synchronize on the timeslots. We also define a window length $T_w > T$, which specifies the amount of time to look back to estimate the parameter $r$. We consider the case where the parameter $r$ is the same across the whole geographical area, which is a reasonable assumption when different locations within the geographical area under consideration have similarities (for example, they all belong to an urban area). More sophisticated cases can be studied in the future. The optimal policy is computed by a MEC controller, and a policy update interval $T_u$ is defined.

The operating procedure is described as follows. Each MEC obtains the identity of associated users of the cell connected to the MEC, at the beginning of each timeslot with length $T$. Based on this information, the MEC computes the number of users that have left the cell and the total number of users in the cell, and stores this information for each timeslot. At interval $T_u$, the MEC controller sends a request to all MECs to collect the current statistics. After receiving the request, each MEC $i_{\text{MEC}}$ computes the empirical probability $f_{i_{\text{MEC}}}$ of users moving outside of the cell, based on the statistics on the departed and total users within the duration of $T_w$. These empirical probabilities are sent together with other monitored information (such as current network and server load) to the controller. The controller then computes the average of the empirical probabilities $f_{i_{\text{MEC}}}$, denoted by $\overline{f}$, and updates parameter $r = \overline{f}/6$. It also computes the transmission and migration cost parameters based on the current system condition. Based on these updated parameters, the controller computes the optimal policy under the current system condition and sends the result to the MECs.

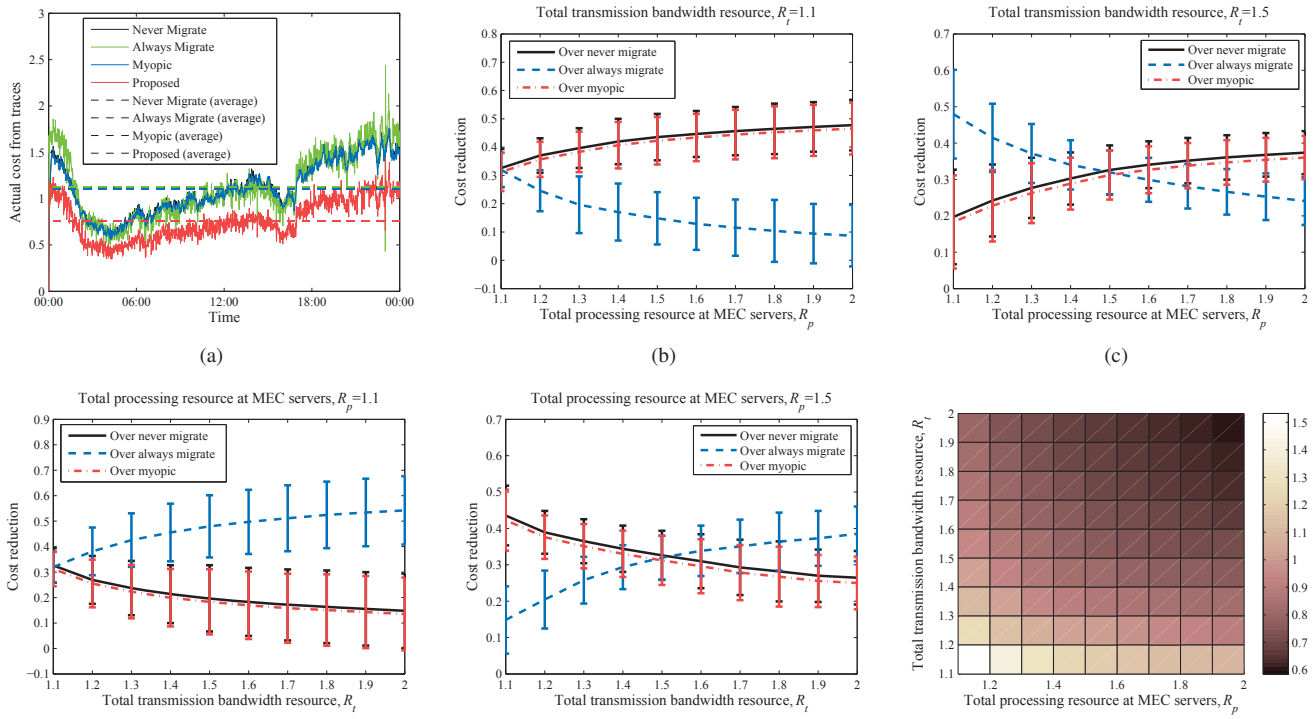



Figure 7. Simulation result with real-world traces: (a) average cost per user in each timeslot over a day, where $R_t = R_p = 1.5$; (b)–(e) average cost reduction compared with alternative policies (error bars denote the standard deviation); (f) average cost.

### B. Trace-Driven Simulation

We perform simulation with real-world mobility traces of 536 taxis in San Francisco, collected on the day of May 31, 2008 [13], [14]. A hexagon cell structure with 500 m cell separation is assumed and each taxi is assumed to be connected to the nearest basestation. The parameters for data collection and policy update are set as $T = T_u = 60$ s, and $T_w = 3600$ s. We choose $N = 10$, $\mu = \theta = 0.8$, and $\gamma = 0.9$.

*1) Cost definition:* It is assumed that the network load is proportional to the number of taxis in operation, and we denote the normalized total amount of transmission bandwidth resource (correspondingly, processing resource at MEC servers) with a factor $R_t$ (correspondingly, $R_p$). Then, we define a quantity $G_t$ (correspondingly, $G_p$) as $G_t = 1 \Big/ \left(1 - \frac{m_{\text{cur}}}{R_t m_{\max}}\right)$ (correspondingly, $G_p = 1 \Big/ \left(1 - \frac{m_{\text{cur}}}{R_p m_{\max}}\right)$), where $m_{\text{cur}}$ denotes the number of taxis in operation at the time when the optimal policy is being computed, and $m_{\max}$ denotes the maximum number of taxis that may simultaneously operate at any time instant in the considered dataset. The cost parameters are then defined as $\beta_c = G_p + G_t$, $\beta_l = -G_t$, $\delta_c = G_t$, and $\delta_l = -G_t$. With such a definition, we have $\beta_c + \beta_l = G_p$ and $\delta_c + \delta_l = 0$, which means that the constant part of the migration cost is $G_p$ (to represent the processing cost for migration) and there is no constant part in the cost for data transmission. We set $\beta_l = \delta_l = G_t$ because this part of cost can be regarded as related to data transmission in both $c_m(x)$ and $c_d(x)$. The choice of $G_t$ and $G_p$ can serve for the purpose of load balancing [15] and can also represent the delay of data transmission ($G_t$) and processing ($G_p$).

*2) Results:* The average cost per user in each timeslot (i.e., the $C_a(s(t))$ values) is collected and shown in Fig. 7. Denote the cost of the proposed method as $C$ and the cost of the method under comparison as $C_0$, then the cost reduction is defined as $(C_0 - C)/C_0$. The results show that the proposed approach is beneficial with cost reductions ranging from 9% to 54% compared with the never/always migrate or myopic policy. The results also show that the costs fluctuate (due to different system load) over the day, and they vary with different amount of total available resources, which implies that it is necessary to compute the optimal policy in real-time, based on recent observations on the system condition.

## VI. Discussions

We have made some assumptions to make the problem theoretically tractable. In this section, we justify these assumptions from a practical point of view.

*Cost Functions:* To ease our discussion, we have limited our attention to transmission and migration costs in this paper. This can be extended to include more sophisticated cost models. For example, the transmission cost can be extended to include the computational cost of hosting the service at an MEC, by adding a constant value to the transmission cost expression. As in Section V-B, the cost values can also be time-varying and related to the background system load. Furthermore, the cost definition can be practically regarded as the average cost over all locations, which means that when seen from a single location, the monotonicity of cost values with distances does not need to apply. This makes the proposed approach less restrictive in terms of practical applicability.

We also note that it is generally possible to formulate an MDP with additional dimensions in cost modeling, such as one that includes the state of the network, load at each specific MEC hosting the service, state of the service to avoid service interruption when in critical state, etc. However, this requires a significantly larger state space compared to our formulation in this paper, as we need to include those network/MEC/service states in the state space of the MDP. There is a tradeoff between the complexity of solving the problem and accuracy of cost modeling. Such issues can be studied in the future.

*Single/Multiple Users:* As pointed out in Section II, although we focused on a single user in our problem modeling, practical cases involving multiple users running independent services can be considered by setting cost functions related to the background traffic generated by other users, as in Section V-B. For more complicated cases such as multiple users sharing the same service, or where the placement of different services is strongly coupled and reflected in the cost value, we can formulate the problem as an MDP with larger state space. The details are left for future work where we envision similar approximation techniques as in this paper can be used to approximately solve the resulting MDP.

*Transition Probability in MDP:* In the theoretical modeling, the transition probabilities in the MDP are assumed to be known. In practice, these can be estimated from the cell association history of users, as discussed in Section V-A.

*Uniform Random Walk:* The uniform random walk mobility model is used as a modeling assumption, which not only simplifies the theoretical analysis, but also makes the practical implementation of the proposed method fairly simple in the sense that only the empirical probability of users moving outside of the cell needs to be recorded (see Section V-A2). This model can capture the average mobility of a large number of users. The simulation results in Section V-B confirm that this model provides good performance, even though individual users do not necessarily follow a uniform random walk.

*MEC Controller:* The MEC controller does not need to be a separate cloud entity. Rather, it can be a service running at one of the MECs. The additional overhead incurred by the proposed approach is low, because it has low computational complexity and the interactions between each MEC and the MEC controller is infrequent (the interval is specified by $T_u$).

## VII. Conclusions

In this paper, we have studied service migration in MECs. The problem is formulated as an MDP, but its state space can be arbitrarily large. To make the problem tractable, we have reduced the general problem into an MDP that only considers a meaningful parameter, namely the distance between the user and the service. The distance-based MDP has several structural properties that allow us to develop an efficient algorithm to find its optimal policy. We have then shown that the distance-based MDP is a good approximation to scenarios where the users move in a 2-D space, which is confirmed by analytical and numerical evaluations and also by simulations with real-world traces of taxis in San Francisco.

For ease of presentation, in this paper we have assumed that MECs are colocated with basestations. However, our proposed approach is not restricted to such cases and can easily incorporate scenarios where MECs are not colocated with basestations as long as the costs are geographically dependent.

The results in this paper provide an efficient solution to service migration in MECs. Further, we envision that the approaches used in this paper can be extended to a range of other problems that share similar properties. The highlights of our approaches include: a closed-form solution to the discounted sum cost of a particular class MDPs, which can be used to simplify the procedure of finding the optimal policy; a method to approximate an MDP (in a particular class) with one that has smaller state space, where the approximation error can be shown analytically; and a method to collect statistics from the real-world to serve as parameters of the MDP.

## Acknowledgment

This research was sponsored in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

## References

[1] Y. Abe, R. Geambasu, K. Joshi, H. A. Lagar-Cavilla, and M. Satyanarayanan, "vTube: efficient streaming of virtual appliances over last-mile networks," in *Proc. of the 4th annual Symposium on Cloud Computing*. ACM, 2013.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. of ACM MobiSys*, 2014.

[3] "Smarter wireless networks," *IBM Whitepaper No. WSW14201USEN*, Feb. 2013. [Online]. Available: www.ibm.com/services/multimedia/Smarter_wireless_networks.pdf

[4] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49, Oct. 2013.

[5] S. Davy, J. Famaey, J. Serrat-Fernandez, J. Gorricho, A. Miron, M. Dramitinos, P. Neves, S. Latre, and E. Goshen, "Challenges to support edge-as-a-service," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 132–139, Jan. 2014.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[7] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, Sept. 2013.

[8] ——, "An analytical model for follow me cloud," in *Proc. of IEEE GLOBECOM 2013*, Dec. 2013.

[9] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. of IEEE ICC 2014*, June 2014.

[10] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. of IEEE MILCOM 2014*, Oct. 2014.

[11] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Supplementary materials for mobility-driven service migration in mobile micro-clouds." [Online]. Available: www.commsp.ee.ic.ac.uk/%7esw4410/papers/MigrationSup.pdf

[12] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2009, vol. 414.

[13] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Proc. of COMSNETS*, Jan. 2009.

[14] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.org/epfl/mobility/, Feb. 2009.

[15] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. on Networking*, vol. 20, no. 1, pp. 206–219, 2012.