Mobility-Induced Service Migration in Mobile Micro-Clouds

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

***Abstract*—Mobile micro-cloud is an emerging technology in distributed computing, which is aimed at providing seamless computing/data access to the edge of the network when a centralized service may suffer from poor connectivity and long latency. Different from the traditional cloud, a mobile micro-cloud is smaller and deployed closer to users, typically attached to a cellular basestation or wireless network access point. Due to the relatively small coverage area of each basestation or access point, when a user moves across areas covered by different basestations or access points which are attached to different micro-clouds, issues of service performance and service migration become important. In this paper, we consider such migration issues. We model the general problem as a Markov decision process (MDP), and show that, in the special case where the mobile user follows a one-dimensional asymmetric random walk mobility model, the optimal policy for service migration is a threshold policy. We obtain the analytical solution for the cost resulting from arbitrary thresholds, and then propose an algorithm for finding the optimal thresholds. The proposed algorithm is more efficient than standard mechanisms for solving MDPs.**

***Index Terms*—Cloud computing, Markov decision process (MDP), mobile micro-cloud, mobility, service migration, wireless networks**

I. Introduction

Cloud technologies have been developing successfully in the past decade, which enable the centralization of computing and data resources so that they can be accessed in an on-demand basis by different end users. Traditionally, clouds are centralized, in the sense that services are provided by large data-centers that may be located far away from the user. A user may suffer from poor connectivity and long latency when it connects to such a centralized service. In recent years, efforts have been made to distribute the cloud closer to users, to provide faster access and higher reliability to end users in a particular geographical area. A notable concept in this regard is the mobile micro-cloud, where a small cloud consisting of a small set of servers is attached directly to the wireless communication infrastructure (e.g., a cellular basestation or wireless access point) to provide service to users within its coverage. Applications of the mobile micro-cloud include data and computation offloading for mobile devices [1], [2], which is a complement for the relatively low computational and data storage capacity of mobile users. It is also beneficial for scenarios requiring high robustness or high data-processing capability closer to the user, such as in hostile environments [3] or for vehicular networks [4]. There are a few other concepts which are similar to that of the mobile micro-cloud, including edge computing [5], Cloudlet [3], and Follow Me Cloud [6]. We use the term mobile micro-cloud throughout this paper.

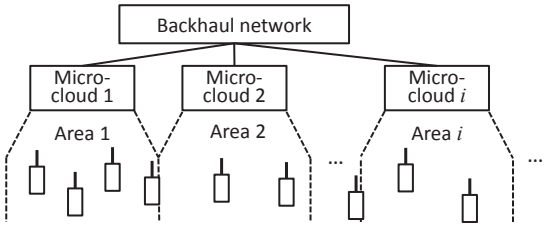

Figure 1. Application scenario with mobile micro-cloud.

A significant issue in the mobile micro-cloud is service migration caused by the mobility of users. Because different micro-clouds are attached to different basestations or access points, a decision needs to be made on whether and where to migrate the service, when a user moves outside the service area of a micro-cloud that is providing its service. Consider the scenario as shown in Fig. 1, which resembles the case where a micro-cloud is connected to a basestation that covers a particular area, and these micro-clouds are also interconnected with each other via a backhaul network. When a mobile user moves from one area to another area, we can either continue to run the service on the micro-cloud for the previous area, and transmit data to/from the user via the backhaul network, or we can migrate the service to the micro-cloud responsible for the new area. In both cases, a cost is incurred; there is a data transmission cost for the first case, and a migration cost for the second case.

In the literature, only a few papers have studied the impact of mobility and its relationship to service migration for mobile micro-clouds. In [7], analytical results on various performance factors of the mobile micro-cloud are studied, by assuming a symmetric 2-dimensional (2-D) random walk mobility model. A service migration procedure based on Markov decision

¶ Contributions of the author to this work are not related to his current employment at Samsung Research America.

process (MDP) for 1-D random walk is studied in [8]. It mainly focuses on formulating the problem with MDP, which can then be solved with standard techniques for solving MDPs.

In this paper, similarly to [8], we consider an MDP formulation of the migration problem. In contrast to [8], we propose an optimal threshold policy to solve for the optimal action of the MDP, which is more efficient than standard solution techniques. A threshold policy means that we always migrate the service for a user from one micro-cloud to another when the user is in states bounded by a particular set of thresholds, and not migrate otherwise. We first prove the existence of an optimal threshold policy and then propose an algorithm with polynomial time-complexity for finding the optimal thresholds. The analysis in this paper can also help us gain new insights into the migration problem, which set the foundation for more complicated scenarios for further study in the future.

The remainder of this paper is organized as follows. In Section II, we describe the problem formulation. Section III shows that an optimal threshold policy exists and proposes an algorithm to obtain the optimal thresholds. Simulation results are shown in Section IV. Section V draws conclusions.

II. Problem Formulation

We consider a 1-D region partitioned into a discrete set of areas, each of which is served by a micro-cloud, as shown in Fig. 1. Such a scenario models user mobility on roads, for instance. A time-slotted system (Fig. 2) is considered, which can be viewed as a sampled version of a continuous-time system, where the sampling can either be equally spaced over time or occur right after a handoff instance.

Mobile users are assumed to follow a 1-D asymmetric random walk mobility model. In every new timeslot, a node moves with probability p (or q) to the area that is on the right (or left) of its previous area, it stays in the same area with probability $1-p-q$. If the system is sampled at handoff instances, then $1-p-q=0$, but we consider the general case with $0 \leq 1-p-q \leq 1$. Obviously, this mobility model can be described as a Markov chain. We only focus on a single mobile user in our analysis; equivalently, we assume that there is no correlation in the service or mobility among different users.

The state of the user is defined as the *offset* between the mobile user location and the location of the micro-cloud running the service at the beginning of a slot, before possible service migration, i.e., the state in slot t is defined as $s_t = u_t - h_t$, where u_t is the location (index of area) of the mobile user, and h_t the location of the micro-cloud hosting the service. Note that s_t can be zero, positive, or negative. At the beginning of each timeslot, the current state is observed, and the decision on whether to migrate the service is made. If migration is necessary, it happens right after the state observation, i.e., at the beginning of the timeslot. We assume that the time taken for migration is negligible compared with the length of a slot.

We study whether and where to migrate the service when the mobile user has moved from one area to another. The cost in a single timeslot $C_a(s)$ is defined as the cost under state s when

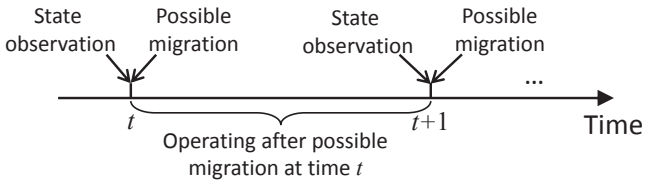

Figure 2. Timing of the proposed mechanism.

performing action a, where a represents a migration decision for the service such as no migration or migration to a specified micro-cloud. The goal is to minimize the discounted sum cost over time. Specifically, under the current state s_0, we would like to find a policy π that maps each possible state s to an action $a = \pi(s)$ such that the expected long-term discounted sum cost is minimized, i.e.,

$$V(s_0) = \min_{\pi} \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t C_{\pi(s_t)}(s_t) \middle| s_0\right] \tag{1}$$

where $\mathrm{E}[\cdot|\cdot]$ denotes the conditional expectation, and $0 < \gamma < 1$ is a discount factor.

Because we consider a scenario where all micro-clouds are connected via the backhaul (as shown in Fig. 1), and the backhaul is regarded as a central entity (which resembles the case for cellular networks, for example), we consider the following one-timeslot cost function for taking action a in state s in this paper:

$$C_a(s) = \begin{cases} 0, & \text{if no migration or data transmission} \\ \beta, & \text{if only data transmission} \\ 1, & \text{if only migration} \\ \beta+1, & \text{if both migration and data transmission} \end{cases} \tag{2}$$

Equation (2) is explained as follows. If the action a under state s causes no migration or data transmission (e.g., if the node and the micro-cloud hosting the service are in the same location, i.e., $s=0$, and we do not migrate the service to another location), we do not need to communicate via the backhaul network, and the cost is zero. A non-zero cost is incurred when the node and the micro-cloud hosting the service are in different locations. In this case, if we do not migrate to the current node location at the beginning of the timeslot, the data between the micro-cloud and mobile user need to be transmitted via the backhaul network. This data transmission incurs a cost of β. When we perform migration, we need resources to support migration. The migration cost is assumed to be 1, i.e., the cost $C_a(s)$ is normalized by the migration cost. Finally, if both migration and data transmission occur, in which case we migrate to a location that is different from the current node location, the total cost is $\beta+1$.

Lemma 1. *Migrating to locations other than the current location of the mobile user is not optimal.*

Proof: We consider an arbitrary trajectory of the mobile user. Denote t_u as the first timeslot (starting from the current timeslot) that the mobile user is in the location indexed u. Assume that the user is currently in location u_0, then the current

timeslot is t_{u_0}.

Case 1 – migrating to location $u \neq u_0$ at t_{u_0}: This incurs a cost of $\beta + 1$ at timeslot t_{u_0}, because $t_u > t_{u_0}$ as a node cannot be in two different locations at the same time. Define a variable $t_m \in [t_{u_0} + 1, t_u]$ being the largest timeslot index such that we do not perform further migration at timeslots within the interval $[t_{u_0} + 1, t_m - 1]$, which means that either we perform migration at t_m or we have $t_m = t_u$. Then, we have a cost of β at each of the timeslots $t \in [t_{u_0} + 1, t_m - 1]$.

Case 2 – no migration at t_{u_0}: In this case, the cost at each timeslot $t \in [t_{u_0}, t_m - 1]$ is either β (if $s_t = u_t - h_t \neq 0$) or zero (if $s_t = 0$). For the timeslot t_m, we construct the following policy. If $t_m < t_u$, we migrate to the same location as in Case 1, which means that the cost at t_m cannot be larger than that in Case 1. If $t_m = t_u$, we migrate to u, which can increase the cost at t_m by at most one unit compared with the cost in Case 1. With the above policy, the costs at timeslots $t > t_m$ in Cases 1 and 2 are the same.

The cost at t_{u_0} in Case 1 is one unit larger than that in Case 2, and the cost at t_m in Case 1 is at most one unit smaller than that in Case 2. Because $0 < \gamma < 1$, Case 2 brings lower discounted sum cost than Case 1. Therefore, there exists a better policy than migrating to $u \neq u_0$ at t_{u_0}. This holds for any movement pattern of the mobile user, and it ensures that the cost in any timeslot is either 0, 1, or β. ∎

From Lemma 1, we only have two candidate actions, which are migrating to the current user location or not migrating. This simplifies the action space to two actions: a migration action, denoted as $a = 1$; and a no-migration action, denoted as $a = 0$. In practice, there is usually a limit on the maximum allowable distance between the mobile user and the micro-cloud hosting its service for the service to remain usable. We model this limitation by a maximum negative offset M and a maximum positive offset N (where $M < 0, N > 0$), such that the service must be migrated ($a \equiv 1$) when the node enters state M or N. This implies that, although the node can move in an unbounded space, the state space of our MDP for service control is finite. The overall transition diagram of the resulting MDP is illustrated in Fig. 3. Note that because each state transition is the concatenated effect of (possible) migration and node movement, and the states are defined as the offset between node and host location, the next state after taking a migration action is either -1, 0, or 1.

With the above considerations, the cost function in (2) can be modified to the following:

$$C_a(s) = \begin{cases} 0, & \text{if } s = 0 \\ \beta, & \text{if } s \neq 0, M < s < N, a = 0 \\ 1, & \text{if } s \neq 0, M \leq s \leq N, a = 1 \end{cases} \tag{3}$$

With the one-timeslot cost defined as in (3), we obtain the following Bellman's equations for the discounted sum cost when respectively taking action $a = 0$ and $a = 1$:

$$V(s|a=0) = \begin{cases} \gamma \sum_{j=-1}^{1} p_{0j} V(j), & \text{if } s = 0 \\ \beta + \gamma \sum_{j=s-1}^{s+1} p_{sj} V(j), & \text{if } s \neq 0, M<s<N \end{cases} \tag{4}$$

$$V(s|a=1) = \begin{cases} \gamma \sum_{j=-1}^{1} p_{0j} V(j), & \text{if } s = 0 \\ 1 + \gamma \sum_{j=-1}^{1} p_{0j} V(j), & \text{if } s \neq 0, M \leq s \leq N \end{cases} \tag{5}$$

where p_{ij} denotes the (one-step) transition probability from state i to state j, their specific values are related to parameters p and q as defined earlier. The optimal cost $V(s)$ is

$$V(s) = \begin{cases} \min\{V(s|a=0), V(s|a=1)\}, & \text{if } M < s < N \\ V(s|a=1), & \text{if } s = M \text{ or } s = N \end{cases} \tag{6}$$

III. Optimal Threshold Policy

A. Existence of Optimal Threshold Policy

We first show that there exists a threshold policy which is optimal for the MDP in Fig. 3.

Proposition 1. *There exists a threshold policy (k_1, k_2), where $M < k_1 \leq 0$ and $0 \leq k_2 < N$, such that when $k_1 \leq s \leq k_2$, the optimal action for state s is $a^*(s) = 0$, and when $s < k_1$ or $s > k_2$, $a^*(s) = 1$.*

Proof: It is obvious that different actions for state zero $a(0) = 0$ and $a(0) = 1$ are essentially the same, because the mobile user and the micro-cloud hosting its service are in the same location under state zero, either action does not incur cost and we always have $C_{a(0)}(0) = 0$. Therefore, we can conveniently choose $a^*(0) = 0$.

In the following, we show that, if it is optimal to migrate at $s = k_1 - 1$ and $s = k_2 + 1$, then it is optimal to migrate at all states s with $M \leq s \leq k_1 - 1$ or $k_2 + 1 \leq s \leq N$. We relax the restriction that we always migrate at states M and N for now, and later discuss that the results also hold for the unrelaxed case. We only focus on $k_2 + 1 \leq s \leq N$, because the case for $M \leq s \leq k_1 - 1$ is similar.

If it is optimal to migrate at $s = k_2 + 1$, we have

$$V(k_2 + 1|a = 1) \leq \beta \sum_{t=0}^{\infty} \gamma^t = \frac{\beta}{1 - \gamma} \tag{7}$$

where the right hand-side of (7) is the discounted sum cost of a never-migrate policy supposing that the user never returns back to state zero when starting from state $s = k_2 + 1$. This cost is an upper bound of the cost incurred from any possible state-transition path without migration, and migration cannot bring higher cost than this because otherwise it contradicts with the presumption that it is optimal to migrate at $s = k_2 + 1$.

Suppose we do not migrate at state s' where $k_2 + 1 < s' \leq N$, then we have a (one-timeslot) cost of β in each timeslot until the user reaches a migration state (i.e., a state at which we perform migration). From (5), we know that $V(s|a = 1)$ is constant for $s \neq 0$. Therefore, any state-transition path L starting from state s' has a discounted sum cost of

$$V_L(s') = \beta \sum_{t=0}^{t_m - 1} \gamma^t + \gamma^{t_m} V(k_2 + 1|a = 1)$$

where $t_m > 0$ is a parameter representing the first timeslot that the user is in a migration state after reaching state s' (assuming

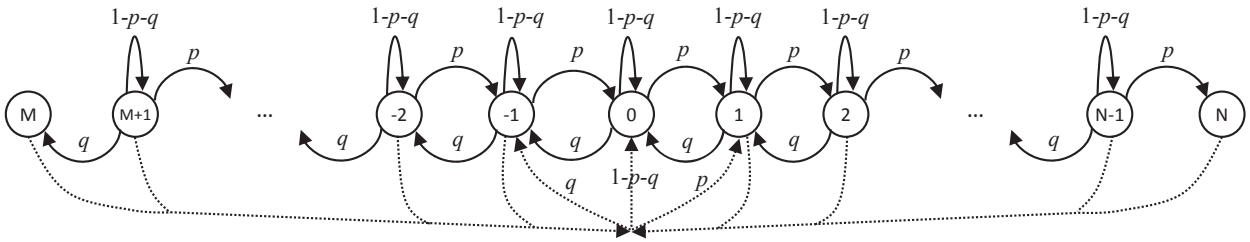

Figure 3. MDP model for service migration. The solid lines denote transition under action $a = 0$ and the dotted lines denote transition under action $a = 1$. When taking action $a = 1$ from any state, the next state is $s = -1$ with probability q, $s = 0$ with probability $1 - p - q$, or $s = 1$ with probability p.

that we reach state s' at $t = 0$), which is dependent on the state-transition path L. We have

$$\begin{aligned} &V_L(s') - V(k_2+1|a=1) \\ &= \beta\frac{(1-\gamma^{t_m})}{1-\gamma} - \left(1-\gamma^{t_m}\right)V(k_2+1|a=1) \\ &= \left(1-\gamma^{t_m}\right)\left(\frac{\beta}{1-\gamma} - V(k_2+1|a=1)\right) \geq 0 \end{aligned}$$

where the last inequality follows from (7). It follows that, for any possible state-transition path L, $V_L(s') \geq V(k_2+1|a=1)$. Hence, it is always optimal to migrate at state s', which brings cost $V(s'|a=1) = V(k_2+1|a=1)$.

The result also holds with the restriction that we always migrate at states M and N, because no matter what thresholds (k_1, k_2) we have for the relaxed problem, migrating at states M and N always yield a threshold policy. ∎

Proposition 1 shows the existence of an optimal threshold policy. The optimal threshold policy exists for arbitrary values of M, N, p, and q.

B. Simplifying the Cost Calculation

The existence of the optimal threshold policy allows us simplify the cost calculation, which helps us develop an algorithm that has lower complexity than standard MDP solution algorithms. When the thresholds are given as (k_1, k_2), the value updating function (6) is changed to the following:

$$V(s) = \begin{cases} V(s|a=0), & \text{if } k_1 \leq s \leq k_2 \\ V(s|a=1), & \text{otherwise} \end{cases} \tag{8}$$

From (4) and (5), we know that, for a given policy with thresholds (k_1, k_2), we only need to compute $V(s)$ with $k_1 - 1 \leq s \leq k_2 + 1$, because the values of $V(s)$ with $s \leq k_1 - 1$ are identical, and the values of $V(s)$ with $s \geq k_2 - 1$ are also identical. Note that we always have $k_1 - 1 \geq M$ and $k_2 + 1 \leq N$, because $k_1 > M$ and $k_2 < N$ as we always migrate when at states M and N.

Define

$$\mathbf{v}_{(k_1,k_2)} = \left[V(k_1-1)\ V(k_1)\cdots V(0)\cdots V(k_2)\ V(k_2+1)\right]^{\mathrm{T}} \tag{9}$$

$$\mathbf{c}_{(k_1,k_2)} = \left[\ 1\ \underbrace{\beta\ \cdots\ \beta}_{-k_1 \text{ elements}}\ 0\ \underbrace{\beta\ \cdots\ \beta}_{k_2 \text{ elements}}\ 1\ \right]^{\mathrm{T}} \tag{10}$$

$$\mathbf{P}'_{(k_1,k_2)} = \begin{bmatrix} p_{0,k_1-1} & \cdots & p_{00} & \cdots & p_{0,k_2+1} \\ p_{k_1,k_1-1} & \cdots & p_{k_1,0} & \cdots & p_{k_1,k_2+1} \\ \vdots & & \vdots & & \vdots \\ p_{0,k_1-1} & \cdots & p_{00} & \cdots & p_{0,k_2+1} \\ \vdots & & \vdots & & \vdots \\ p_{k_2,k_1-1} & \cdots & p_{k_2,0} & \cdots & p_{k_2,k_2+1} \\ p_{0,k_1-1} & \cdots & p_{00} & \cdots & p_{0,k_2+1} \end{bmatrix} \tag{11}$$

where superscript T denotes the transpose of the matrix.

Then, (4) and (5) can be rewritten as

$$\mathbf{v}_{(k_1,k_2)} = \mathbf{c}_{(k_1,k_2)} + \gamma\mathbf{P}'_{(k_1,k_2)}\mathbf{v}_{(k_1,k_2)} \tag{12}$$

The value vector $\mathbf{v}_{(k_1,k_2)}$ can be obtained by

$$\mathbf{v}_{(k_1,k_2)} = \left(\mathbf{I} - \gamma\mathbf{P}'_{(k_1,k_2)}\right)^{-1}\mathbf{c}_{(k_1,k_2)} \tag{13}$$

The matrix $\left(\mathbf{I} - \gamma\mathbf{P}'_{(k_1,k_2)}\right)$ is invertible for $0 < \gamma < 1$, because in this case there exists a unique solution for $\mathbf{v}_{(k_1,k_2)}$ from (12). Equation (13) can be computed using Gaussian elimination that has a complexity of $O\left(\left(|M|+N\right)^3\right)$. However, noticing that $\mathbf{P}'_{(k_1,k_2)}$ is a sparse matrix (because $p_{ij} = 0$ for $|j-i| > 1$), there can exist more efficient algorithms to compute (13).

C. Algorithm for Finding the Optimal Thresholds

To find the optimal thresholds, we can perform a search on values of (k_1, k_2). Further, because an increase/decrease in $V(s)$ for some s increases/decreases each element in the cost vector $\mathbf{v}$ due to cost propagation following balance equations (4) and (5), we only need to minimize $V(s)$ for a specific state s. We propose an algorithm for finding the optimal thresholds, as shown in Algorithm 1, which is a modified version of the standard policy iteration mechanism [9, Ch. 3].

Algorithm 1 is explained as follows. We keep iterating until the thresholds no longer change, which implies that the optimal thresholds have been found. The thresholds (k_1^*, k_2^*) are those obtained from each iteration.

Lines 4–6 compute $V(s)$ for all s under the given thresholds (k_1^*, k_2^*). Then, Lines 8–22 determine the search direction for k_1 and k_2. Because $V(s)$ in each iteration is computed using the current thresholds (k_1^*, k_2^*), we have actions $a(k_1^*) = a(k_2^*) = 0$, and (4) is automatically satisfied when replacing its left hand-side with $V(k_1^*)$ or $V(k_2^*)$. Lines 9 and 16 check whether

Algorithm 1 Modified policy iteration algorithm for finding the optimal thresholds

1: Initialize $k_1^* \leftarrow 0$, $k_2^* \leftarrow 0$
2: **repeat**
3: $k_1'^* \leftarrow k_1^*$, $k_2'^* \leftarrow k_2^*$ //record previous thresholds
4: Construct $\mathbf{c}_{(k_1^*,k_2^*)}$ and $\mathbf{P}'_{(k_1^*,k_2^*)}$ according to (10) and (11)
5: Evaluate $\mathbf{v}_{(k_1^*,k_2^*)}$ from (13)
6: Extend $\mathbf{v}_{(k_1^*,k_2^*)}$ to obtain $V(s)$ for all $M \leq s \leq N$
7: **for** $i = 1, 2$ **do**
8: **if** $i = 1$ **then**
9: **if** $1 + \gamma \sum_{j=-1}^{1} p_{0j} V(j) < V(k_1^*)$ **then**
10: dir $\leftarrow 1$, loopVec $\leftarrow [k_1^* + 1, k_1^* + 2, ..., 0]$
11: $k_1^* \leftarrow k_1^* + 1$
12: **else**
13: dir $\leftarrow 0$, loopVec $\leftarrow [k_1^* - 1, k_1^* - 2, ..., M + 1]$
14: **end if**
15: **else if** $i = 2$ **then**
16: **if** $1 + \gamma \sum_{j=-1}^{1} p_{0j} V(j) < V(k_2^*)$ **then**
17: dir $\leftarrow 1$, loopVec $\leftarrow [k_2^* - 1, k_2^* - 2, ..., 0]$
18: $k_2^* \leftarrow k_2^* - 1$
19: **else**
20: dir $\leftarrow 0$, loopVec $\leftarrow [k_2^* + 1, k_2^* + 2, ..., N - 1]$
21: **end if**
22: **end if**
23: **for** $k_i =$ each value in loopVec **do**
24: **if** dir $= 0$ **then**
25: **if** $\beta + \gamma \sum_{j=k_i-1}^{k_i+1} p_{k_i,j} V(j) < V(k_i)$ **then**
26: $k_i^* \leftarrow k_i$
27: **else if** $\beta + \gamma \sum_{j=k_i-1}^{k_i+1} p_{k_i,j} V(j) > V(k_i)$ **then**
28: **exit for**
29: **end if**
30: **else if** dir $= 1$ **then**
31: **if** $1 + \gamma \sum_{j=-1}^{1} p_{0j} V(j) < V(k_i)$ **then**
32: $k_i^* \leftarrow k_i - \text{sign}(k_i)$
33: **else if** $1 + \gamma \sum_{j=-1}^{1} p_{0j} V(j) > V(k_i)$ **then**
34: **exit for**
35: **end if**
36: **end if**
37: **end for**
38: **end for**
39: **until** $k_1^* = k_1'^*$ and $k_2^* = k_2'^*$
40: **return** k_1^*, k_2^*

iterating according to (5) can yield lower cost. If it does, it means that migrating is a better action at state k_1^* (or k_2^*), which also implies that we should migrate at states s with $M \leq s \leq k_1^*$ (or $k_2^* \leq s \leq N$) according to Proposition 1. In this case, k_1^* (or k_2^*) should be set closer to zero, and we search through those thresholds that are closer to zero than k_1^* (or k_2^*). If Line 9 (or Line 16) is not satisfied, according to Proposition 1, it is good not to migrate at states s with $k_1^* \leq s \leq 0$ (or $0 \leq s \leq k_2^*$), so we search k_1 (or k_2) to the direction approaching M (or N), to see whether it is good not to migrate under those states.

Lines 23–37 adjust the thresholds. If we are searching toward state M or N and Line 25 is satisfied, it means that it is better not to migrate under this state (k_i), and we update the threshold to k_i. When Line 27 is satisfied, it means that it is better to migrate at state k_i. According to Proposition 1, we should also migrate at any state closer to M or N than state k_i, thus we exit the loop. If we are searching toward state zero and Line 31 is satisfied, it is good to migrate under this state (k_i), therefore the threshold is set to one state closer to zero ($k_i - \text{sign}(k_i)$). When Line 33 is satisfied, we should not migrate at state k_i. According to Proposition 1, we should also not migrate at any state closer to zero than state k_i, and we exit the loop.

Proposition 2. *The threshold-pair (k_1^*, k_2^*) is different in every iteration of the loop starting at Line 2, otherwise the loop terminates.*

Proof: The loop starting at Line 2 changes k_1^* and k_2^* in every iteration so that $V(s)$ for all s become smaller. It is therefore impossible that k_1^* and k_2^* are the same as in one of the previous iterations and at the same time reduce the value of $V(s)$, because $V(s)$ computed from (13) is the stationary cost value for thresholds (k_1^*, k_2^*). The only case when (k_1^*, k_2^*) are the same as in the previous iteration (which does not change $V(s)$) terminates the loop. ■

Corollary 1. *The number of iterations in Algorithm 1 is $O(|M|N)$.*

Proof: According to Proposition 2, there can be at most $|M|N + 1$ iterations in the loop starting at Line 2. ■

If we use Gaussian elimination to compute (13), the time-complexity of Algorithm 1 is $O\left(|M|N\left(|M| + N\right)^3\right)$.

IV. Simulation Results

We compare the proposed threshold method with the standard value iteration and policy iteration methods [9, Ch. 3]. Simulations are run on MATLAB, on a computer with 64-bit Windows 7, Intel Core i7-2600 CPU, and 8GB memory. The value iteration terminates according to an error bound of $\epsilon = 0.1$ in the discounted sum cost. Note that the proposed method and the standard policy iteration method always produce the optimal cost. The number of states $|M| = N = 10$. The transition probabilities p and q are randomly generated. Simulations are run with 1000 different random seeds in each setting to obtain the average performance. The running time and the discounted sum costs under different values of β are shown in Fig. 4.

The results show that the proposed method always has lowest running time, and the running time of the standard policy iteration method is 2 to 5 times larger than that of the proposed algorithm, while the value iteration approach consumes longer time. This is because the proposed algorithm simplifies the solution search procedure compared with standard mechanisms. The results also show that the proposed method can provide the optimal cost compared with a never-migrate (except for states M and N) or always-migrate policy. It is also interesting to observe that the optimal cost approaches the cost of a never-migrate policy when β is small, and it approaches the cost of an

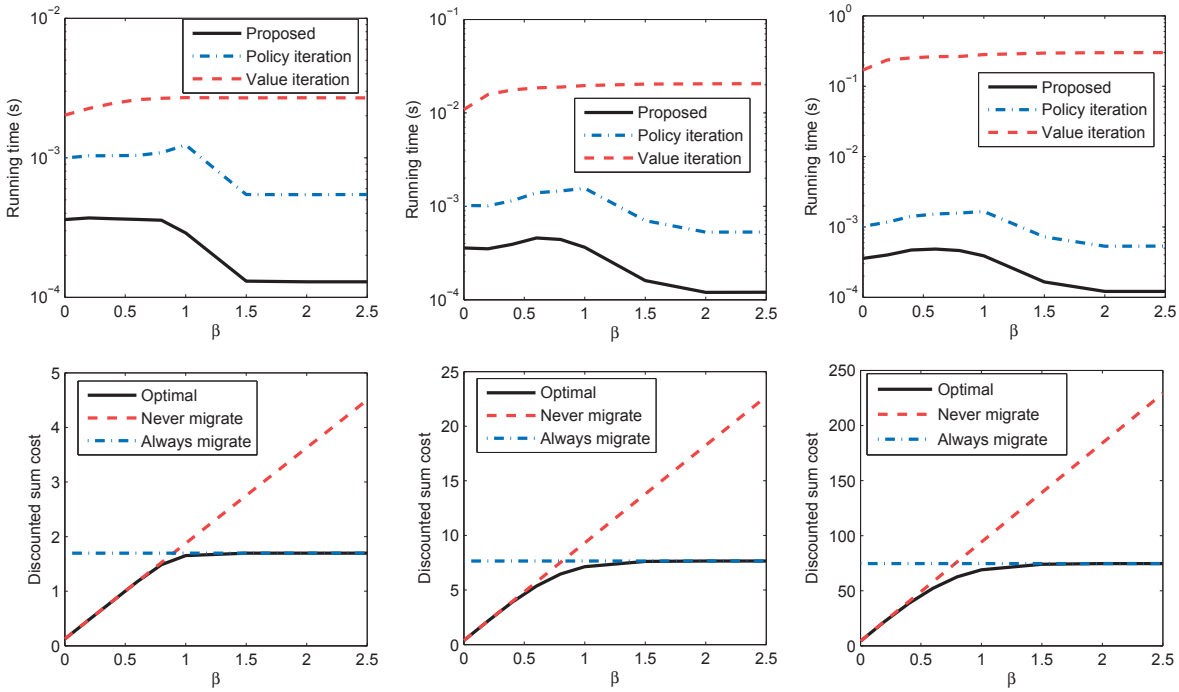

Figure 4. Performance under different β: (a) $\gamma = 0.5$, (b) $\gamma = 0.9$, (c) $\gamma = 0.99$.

always-migrate policy when β is large. Such a result is intuitive, because a small β implies a small data transmission cost, and when β is small enough, then it is not really necessary to migrate; when β is large, the data transmission cost is so large so that it is always good to migrate to avoid data communication via the backhaul network.

V. Conclusion and Future Work

In this paper, we have proposed a threshold policy-based mechanism for service migration in mobile micro-clouds. We have shown the existence of optimal threshold policy and proposed an algorithm for finding the optimal thresholds. The proposed algorithm has polynomial time-complexity which is independent of the discount factor γ. This is promising because the time-complexity of standard algorithms for solving MDPs, such as value iteration or policy iteration, are generally dependent on the discount factor, and they can only be shown to have polynomial time-complexity when the discount factor is regarded as a constant[1] [10]. Although the analysis in this paper is based on 1-D random walk of mobile users, it can serve as a theoretical basis for more complicated scenarios, such as 2-D user-mobility, in the future.

Acknowledgment

This research was partly sponsored 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.

[1]This is unless the MDP is deterministic, which is not the case in this paper.

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] 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.

[4] S. Wang, L. Le, N. Zahariev, and K. K. Leung, "Centralized rate control mechanism for cellular-based vehicular networks," in *Proc. of IEEE GLOBECOM 2013*, 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] 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.

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

[8] 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*, 2014.

[9] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

[10] I. Post and Y. Ye, "The simplex method is strongly polynomial for deterministic markov decision processes." in *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013, pp. 1465–1473.