Offloading in Mobile Cloudlet Systems with Intermittent Connectivity

Yang Zhang, Dusit Niyato, *Senior Member, IEEE*, and Ping Wang, *Senior Member, IEEE*

Abstract—The emergence of mobile cloud computing enables mobile users to offload applications to nearby mobile resource-rich devices (i.e., cloudlets) to reduce energy consumption and improve performance. However, due to mobility and cloudlet capacity, the connections between a mobile user and mobile cloudlets can be intermittent. As a result, offloading actions taken by the mobile user may fail (e.g., the user moves out of communication range of cloudlets). In this paper, we develop an optimal offloading algorithm for the mobile user in such an intermittently connected cloudlet system, considering the users' local load and availability of cloudlets. We examine users' mobility patterns and cloudlets' admission control, and derive the probability of successful offloading actions analytically. We formulate and solve a Markov decision process (MDP) model to obtain an optimal policy for the mobile user with the objective to minimize the computation and offloading costs. Furthermore, we prove that the optimal policy of the MDP has a threshold structure. Subsequently, we introduce a fast algorithm for energy-constrained users to make offloading decisions. The numerical results show that the analytical form of the successful offloading probability is a good estimation in various mobility cases. Furthermore, the proposed MDP offloading algorithm for mobile users outperforms conventional baseline schemes.

Index Terms—Mobile cloud, offloading, offloading failure, mobility, Markov decision process, threshold policy

1 Introduction

The concept of cloud computing has been extended to a mobile paradigm. Cloud providers are not necessarily to be business-level providers such as Amazon. Instead, resource-rich and trusted devices connected to the Internet, e.g., a cluster or a vehicular base station, namely a cloudlet, can also provide cloud-like services to nearby mobile devices via Wi-Fi and cellular connections. Mobile devices (i.e., users) and cloudlets can form a small scale cloudlet-based system. In such a system, a mobile user has an opportunity to access and offload computation jobs to nearby cloudlets to improve the performance and reduce local execution cost.

However, challenges in offloading in mobile cloud environments exist. First, offloading may not always achieve the lowest cost due to possible high communication and remote execution costs. Therefore, a mobile user has to make a decision whether to execute a job locally on the mobile device or offload to nearby mobile cloudlets. Thus, a dynamic decision making algorithm is necessary. Second, random unavailability of wireless connections in mobile cloud environments has not received adequate attention. The users and cloudlets may change their locations and become disconnected from each other. This will cause offloading failure. Moreover, admission policies adopted by cloudlets may also cause rejections of offloading requests from mobile users. This intermittent wireless connection has to be taken into account.

In this paper, we aim to address the above challenges. We propose a Markov decision process (MDP) based offloading algorithm for the mobile users in a cloudlet system. The mobile user has an application to be executed. As the application is divided into code sections (denoted as phases), during the execution, the mobile user can dynamically decide to execute application phases locally on the mobile device or offload to nearby cloudlets. We formulate and solve the MDP model to obtain an optimal policy to minimize computation and communication costs of the mobile user. The MDP model considers the random mobility feature of each mobile user as well as a priority-based cloudlet admission control policy to analyze the intermittent connections between the mobile user and cloudlets. We compare the MDP-based algorithm with conventional static baseline offloading schemes with fixed offloading decisions. Numerical results show that, the proposed MDP offloading algorithm outperforms conventional baseline schemes in terms of expected cost. To the best of our knowledge, there is no existing work on stochastic modeling and dynamic optimization of a fine-grained code-level offloading decision, which considers offloading failures caused by the mobility of mobile users and admission control policy of cloudlets in an intermittently connected cloudlet system.

The major contributions of this paper are as follows:

- We propose an MDP-based model for a mobile user in an intermittently connected cloudlet system to obtain an optimal offloading policy. The policy determines offloading/local execution actions based on the state of the mobile user to achieve the minimum cost.
- We model a mobile user distribution in an intermittent cloudlet system as a Poisson point process (PPP). We then obtain success probabilities of offloading actions with limited knowledge of network parameters. These probabilities are used in the MDP to obtain an optimal offloading policy.

• *The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore.*
E-mail: {yzhang28, dniyato, wangping}@ntu.edu.sg.

Manuscript received 7 Feb. 2014; revised 3 Nov. 2014; accepted 9 Feb. 2015. Date of publication 18 Feb. 2015; date of current version 30 Oct. 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2015.2405539

- We prove that the optimal solution of the MDP is a threshold policy. We introduce an algorithm with bounded errors for the mobile user to make offloading/local execution actions based on the threshold policy.

The rest of this paper is organized as follows. In Section 2, we review related work. Section 3 describes the system model for a mobile cloudlet environment. In Section 4, the MDP model is formulated and solved to minimize the computation and offloading costs from a user's perspective, considering the user's local load and the availability of cloudlets in the surrounding area. Furthermore, the existence of threshold policy is proved in MDP in Section 4. Section 5 studies the mobility feature of mobile users, and derives a set of analytical expressions for mobile users to estimate the probability of successful offloading actions. The numerical results are provided in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

2.1 Mobile Cloud System: Architectures

Mobile cloud computing is referred to as a cloud system which enables mobile devices (e.g., smartphones and tablets) to migrate their data storage and data processing to cloud providers. To fully exploit the potential of cloud computing, the concept of a cloudlet [2], [3] was proposed to employ various local resource-rich devices (e.g., a mobile base station) as cloud servers, instead of business cloud providers (e.g., Amazon) at the far end of the Internet.

Hoang et al. [4] presented a comprehensive survey of mobile cloud and its applications. Satyanarayanan et al. [2] introduced an architecture of a cloudlet system to seamlessly handle mobile users' computation requests. Furthermore, a general architecture for ad hoc mobile users to form an autonomous mobile cloud system was proposed in [5], where some mobile users may act as cloudlets to provide cloud services to others when they have redundant resources. The authors in [6], [7], [8] implemented prototypes of program code partitioning and offloading algorithms on the existing architectures. Additionally, Kovachev et al. [9] compared several popular existing mobile cloud system models and implementations.

2.2 Partitioning and Offloading in Mobile Cloud

As stated in [10], the modern resource-rich, energy-efficient mobile devices and high bandwidth wireless connections enable mobile users to offload jobs to the cloudlet as a practical paradigm. Li et al. [11] pointed out that offloading part of mobile game codes can reduce energy consumption from mobile users' perspective, such that the playing time of the mobile game can be extended. Kumar and Lu [12] showed that offloading can benefit mobile users by reducing energy consumption when the jobs to be executed require large amounts of computation resource and time (i.e., many instructions). However, Kumar and Lu [12] also showed that offloading is not always beneficial to mobile users in terms of energy saving, especially when the offloading process involves intensive communications. The offloading decisions of applications and parts of applications should be balanced optimally. Optimal offloading decisions could be made either in a static manner or a dynamic manner. Li et al. [11] employed a cost graph approach to quantify an immediate cost of each code section. Then the offloading scheme for each code section is decided statically in an offline manner, relying solely on the code section states. Giurgiu et al. [6] proposed both static and dynamic partitioning and offloading schemes. The schemes are based on a graph cutting algorithm to categorize code sections into different offloading decision set.

Chun and Maniatis [8] claimed that mobile users may work in different external environments and with changing workloads, such that static partitioning of jobs between mobile devices and cloud may not achieve optimal results with such externalities. As a result, a linear programming model for dynamically partitioning of an application was proposed. The authors in [7], [13] and [14] provided a grand view of the whole optimization process of a mobile user's offloading decisions. Huang et al. [14] optimized a mobile user's energy cost with Lyapunov optimization. Similar to [8], a linear optimization approach was adopted in [7] and [13] to find the optimal offloading decisions in various applications, such as face recognition and smartphone games. Furthermore, in [7] and [23], automatic partitioning of applications was developed with minimal programmer's involvement. As a result, this makes fine-grained code section offloading possible. Based on [7] and [13], ThinkAir [15], on the other hand, considered the scalability of a mobile cloud, i.e., the ability to handle multiple mobile users instead of one user at the cloud side.

2.3 Intermittent Connection Issues in Mobile Cloud Computing

Most of the aforementioned studies only examined the mobile cloud systems with persistent connectivity (i.e., the connection between any mobile user and cloud is guaranteed). However, as stated in [16], intermittent connectivity issues may exist due to mobility, heterogeneous network environment and smart devices' connection policies. According to [16], a non-persistent connection is a key feature distinguishing mobile cloud with conventional cloud systems. Only the control signals in mobile cloud systems have persistent connections, while regular mobile cloud requests are made via "on-demand" intermittent connections. The existence of intermittent connections may fail the offloading request from the user, which forces the user to execute or re-transmit the job again.

Intermittent connectivity caused by a cloudlet was studied in [3], [17] and [18]. A cloudlet may reject users' offloading requests, e.g., due to limited resources. Specifically, the authors in [17] and [18] jointly considered the cloud provider's revenue with mobile users' QoS requirements, i.e., admission control and resource allocation, respectively. Hoang et al. [18] modeled the admission and resource allocation as a semi-Markov decision process. Almeida et al. [17] solved the optimization analytically by employing a queuing-based approach. Xia et al. [3] introduced an admission control mechanism with the objective to maximize the system throughput. Shi et al. [19] studied the offloading problem in a mobile cloud system with such intermittent connections. The offloading points of applications were set considering the vulnerability of connection, but only the

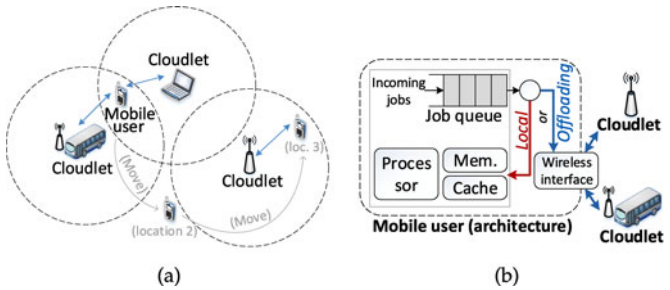

Fig. 1. (a) A cloudlet system with a mobile user offloading jobs, and (b) architecture for the moblie user.

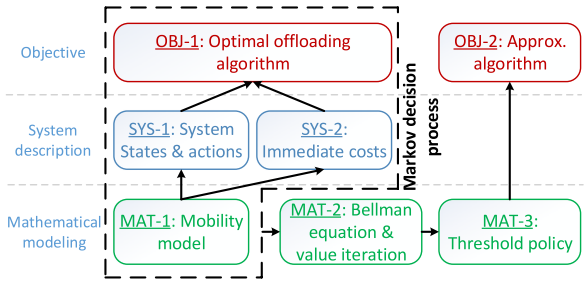

Fig. 3. Structured view of components and methodology used in the paper.

ideal case was discussed that the present and future intermittent connectivity patterns are known.

The works in the literature related to intermittent connections of cloudlets did not develop the optimal offloading algorithm of the mobile user, and this is the main focus of this paper.

3 System Model

In this section, we describe a general model of the cloudlet system under our consideration.

3.1 System Description

In a cloudlet system, we consider a particular mobile user which can be served by multiple mobile cloudlets, as shown in Fig. 1a. Fig. 1b shows the architecture of the mobile user, where the mobile device of the user contains a processor, data storage components such as memory and cache, a single-server FIFO queue to store arriving jobs pending for execution, and a wireless interface.

The job being retrieved from the queue and processed is divided into a sequence of fine-grained code sections [14], defined as application phases. The code sections can be either pre-determined or decided by the mobile user dynamically (e.g., setting break points). Processing a job means executing application phases in a certain sequence. The job processing is finished when the application transits to an exit phase, which is defined as the last phase of the job to process. Fig. 2 shows a typical diagram of application phases of a face recognition application (similar to [7], [20]), where phases 3 and 4 are exit phases.

Conventionally, without cloud or cloudlets, the user has to execute all the application phases on the mobile device, using local execution components, i.e., processor, memory and cache. However, in the mobile cloudlet system, the user has an option to offload parts of the application (i.e., an application phase) to the cloudlets nearby, which may potentially offer higher computing power, and thus, lower execution cost. In this work, we design a dynamic algorithm for the mobile user to decide whether to offload or not. The main procedures of the decision process are as follows:

1) During the application execution, we assume a time interval t which is long enough for the user to finish executing any application phase locally or remotely on cloudlets. Each of such time intervals is defined as a decision period. During a period, once an application phase is successfully executed or failed, the user starts the next decision period immediately. For simplicity of notation, $t-1$ and $t+1$ indicate the previous and next decision periods, respectively.
2) At the beginning of each decision period, the user observes the current system states, i.e., the number of jobs $\mathcal{Q}$ in the queue, the application phase $\mathcal{G}$ that the user is executing, and the number of accessible cloudlets $\mathcal{N}$.
3) Based on the observed composite state $\mathcal{S}=(\mathcal{G},\mathcal{Q},\mathcal{N})$ of the system, the user computes the immediate costs of local execution and offloading. Based on these immediate costs of the current state $\mathcal{S}$, the user takes an action decision $\mathcal{A}(\mathcal{S})$ (or $\mathcal{A}$ for short) of either executing the application phase locally on the mobile device (denoted as $\mathcal{A}=0$) or offloading to the cloudlets (denoted as $\mathcal{A}=1$).
4) However, an offloading action is not always successful, due to the user mobility as well as the admission control policy of the cloudlets. An offloaded application phase may fail. As a result, the user has to restore and execute the failed application phase again (either locally or remotely) in the next decision period.

Fig. 3 shows the flow and interaction among different parts of the paper.

In the system model, a Markov decision process is employed as an optimization method, with the aim to derive a cost-effective optimal offloading algorithm from the perspective of a mobile user (i.e., OBJ-1 in Fig. 3). To formulate the MDP, the system is required to have a set of states and actions, describing the operating statuses and offloading decisions of the mobile user (SYS-1 in Fig. 3), which is discussed in Sections 4.1 and 4.2. Moreover, an immediate cost (SYS-2 in Fig. 3) will be incurred to the mobile user in every system state. The immediate cost

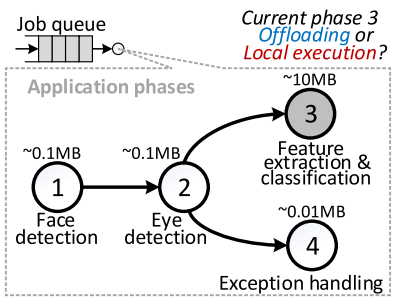

Fig. 2. Application phases: The code sections of a typical mobile phone program for a face recognition application.

TABLE 1
Notation Descriptions

Notation	Definition
$\mathcal{S} = (\mathcal{G}, \mathcal{Q}, \mathcal{N})$	Composite state of a mobile user, including application phase, queue state, and the number of cloudlets
$\mathcal{A}$	Offloading action of a mobile user
N, **G**, **Q**	Transition matrices for the number of cloudlets, application phase, and queue states
$\mathsf{P}^S(\mathcal{S}, \mathcal{S}')$	Transition probability from composite state $\mathcal{S}$ to state $\mathcal{S}'$
$\mathbf{G}_{EP}$	Set for exit phases
λ_c, λ_u	Spatial density of cloudlets and mobile users, respectively
λ_q	Job arrival rate to the queue
R	Bidirectional communication distance between a mobile user and a cloudlet
ε	Cloudlet's probability to accept a mobile user's offloading requests.
η_a	Probability of successful offloading to a single cloudlet
$p^{\eta}(\mathcal{N}, \eta_a \mid \mathcal{A})$	Probability that the current phase is successfully executed (locally or remotely)
$C(\mathcal{S}, \mathcal{A})$	Immediate cost of a mobile user taking action $\mathcal{A}$ at state $\mathcal{S}$

function is defined in Section 3.2.3 and applied in Section 4.3. The mobility feature (MAT-1 in Fig. 3) of each user has to be considered in the state transitions (SYS-1) and immediate cost function (SYS-2), which is modeled using a spatial Poisson process as defined in Section 3.2.2 and discussed in detail in Section 5.

The MDP optimization is expressed as a Bellman equation, and solved using a value iteration algorithm (MAT-2 in Fig. 3). The MDP solving process is shown in Section 4.3. Furthermore, we prove in Section 4.4 that the optimal policy of the MDP has a threshold structure (MAT-3 in Fig. 3). Based on the threshold structure, we propose a fast approximation algorithm (OBJ-2 in Fig. 3) to efficiently achieve offloading decisions with bounded performance. Although OBJ-2 is not optimal compared with OBJ-1, the algorithm complexity is significantly lower.

For convenience, Table 1 lists some important mathematical notations used in the paper.

3.2 Definitions and Assumptions for Optimization Model

To develop the optimal offloading algorithm for the mobile user in such a cloudlet system, we formulate and solve an MDP model. The optimization model has the following definitions and assumptions.

3.2.1 Jobs and Application Phases

The queue is used to store incoming jobs from the user. The queue length (i.e., the number of jobs in the queue) is denoted by $\mathcal{Q}$. Without loss of generality, we assume that the job arrival is a Poisson process. Other job arrival patterns could also be applied in the MDP model. Once a job is retrieved from the queue by the mobile user, the job will be divided into application phases, as shown in Fig. 2. Each phase is labeled as $\mathcal{G}$. For example, $\mathcal{G} = 1$ indicates the face detection phase in Fig. 2. In this model, a phase is an atomic unit processed by the mobile user.

An application phase $\mathcal{G}$ involves codes and data to be processed in the current decision period. The codes are generated by applications, while the data are produced by input, output, and temporary stored files. In this work, we consider the case that the codes and data are fully portable, which means that both the codes and data can be either executed locally or offloaded to the cloudlet. We denote $\delta(\mathcal{G})$ as the size of codes and data of an application phase $\mathcal{G}$ (i.e., size of $\mathcal{G}$ for short). For example, when the mobile user is at the feature extraction & classification phase, the size of the phase includes input data passed from the previous phase, a photo taken, and related codes to process the photo. The mobile user chooses either to locally execute the feature extraction and classification algorithms to process the photo or to offload the codes and data to the cloudlets to execute. Fig. 2 shows the data of each application phase of the face recognition application, obtained by experiments in [20].

3.2.2 Mobility and Cloudlet Availability

We assume that the geographical distributions of both cloudlets and mobile users follow an independent homogeneous poisson point process (HPPP) [21], [22].

At the beginning of a decision period, a mobile user may observe $\mathcal{N}$ nearby cloudlets. This number $\mathcal{N}$ is referred to as *number of cloudlets*, indicating the current number of cloudlets in the adjacent area that the user can offload application phases to. A cloudlet is only accessible to the mobile user if a wireless connection can be established. Let R denote the bidirectional communication distance between the cloudlet and the mobile user. The user can offload an application phase to and receive the result from the cloudlet, if the distance between them is less than or equal to R.

Based on the HPPP assumption for the geographical distribution of cloudlets, the probability that the user can access k cloudlets in the current decision period can be calculated by the probability mass function (PMF) of HPPP as follows:

$$\mathsf{Pr}\{K = k\} = e^{-\pi R^2 \lambda_c} \frac{\left(\pi R^2 \lambda_c\right)^k}{k!}, \quad k = 0, 1, 2, \ldots \tag{1}$$

where λ_c is the distribution density of cloudlets in the system.

The following conditions must hold for the user to offload and obtain the result from executing an application phase.

- At the beginning of a decision period, the user has at least one cloudlet in the communication range to offload an application phase to.
- The user is able to receive the result of the application phase or the failure signal before the end of the decision period.

However, due to the mobility of the mobile user or the cloudlets, offloading actions may not be always successful even if there are accessible cloudlets close to the mobile user. The following situations may happen. At the beginning of a decision period, the user is within the bidirectional communication distance R of a cloudlet. However,

before the cloudlet transmits back the result, the user moves out of the communication range of the cloudlet. Consequently, the result of the offloading request will fail to reach the user. Besides the mobility issue, the cloudlet availability also depends on the cloudlet's admission control decision [18]. That is, the cloudlet may refuse a request from a user with a certain probability. We define a probability η_a of successfully offloading to a cloudlet and receiving the results, namely *cloudlet* availability, as follows:

$$\eta_a = \mathbb{E}\left\{\varepsilon \cdot \frac{T-\tau}{T}\right\}, \tag{2}$$

where ε is the probability that the cloudlet accepts the user's offloading request (i.e., $1-\varepsilon$ is the cloudlet outage probability). T is the user's dwell time inside the communication range R of the cloudlet. τ is the execution time of the offloaded application phase on the cloudlet, plus round-trip time (RTT) of the wireless transmission. Therefore, the physical meaning of $\frac{T-\tau}{T}$ is the probability of successful offloading given that the cloudlet always accepts offloaded jobs. We derive an estimation of η_a of a mobile user in the proposed system in Section 5.

In practice, the execution time τ of a program without human-computer interaction is approximately $10^{-4} \sim 10^{-2}$ s [23]. The RTT of Wi-Fi networks is in the order of tens to hundreds of milliseconds [7], and the communication range of any cloudlet device is measured in meters [19]. Therefore, it is reasonable to presume $T-\tau > 0$ in the definition of η_a.

3.2.3 Immediate Cost

For the current decision period, an *immediate cost* $C(\mathcal{S},\mathcal{A})$ will be incurred to the mobile user given the current composite state $\mathcal{S}=(\mathcal{G},\mathcal{Q},\mathcal{N})$. Similar to [13], as the mobile user may execute jobs locally (i.e., $\mathcal{A}=0$) or remotely (i.e., $\mathcal{A}=1$), the immediate cost is denoted as follows:

$$C(\mathcal{S},\mathcal{A}) = \begin{cases} C_L(\mathcal{S}), & \mathcal{A}=0 \\ C_R(\mathcal{S}), & \mathcal{A}=1, \end{cases} \tag{3}$$

where $C_L(\mathcal{S})$ is the immediate local execution cost, and $C_R(\mathcal{S})$ is the immediate offloading cost. In the following, we present general definitions of $C_L(\mathcal{S})$ and $C_R(\mathcal{S})$.

The immediate local execution cost function $C_L(\mathcal{S})$ is the overall cost incurred when the mobile user executes the application phase locally, defined as follows:

$$C_L(\mathcal{S}) = w_{L1}E_{CPU}(\mathcal{G}) + w_{L2}C_{sch}(\mathcal{G},\mathcal{Q}) + w_{L3}D_{usr}(\mathcal{Q}), \tag{4}$$

where $E_{CPU}(\mathcal{G})$ denotes the processor energy consumption of the mobile user for processing the application phase $\mathcal{G}$. According to [13], $E_{CPU}(\mathcal{G}) = P_{CPU}\cdot T_{CPU}(\mathcal{G}) = P_{CPU}(\mathcal{G})\cdot \frac{\delta(\mathcal{G})}{s_{CPU}}$, where P_{CPU} is the processor power, $T_{CPU}(\mathcal{G})$ is the execution time of $\mathcal{G}$ on the processor, and s_{CPU} is the CPU processing speed. $C_{sch}(\mathcal{G},\mathcal{Q})$ denotes the scheduling overhead caused by the pending jobs of the mobile user for executing the phase $\mathcal{G}$ locally. The scheduling cost $C_{sch}(\mathcal{G},\mathcal{Q})$ is positively correlated to the queue size $\mathcal{Q}$, since the pending jobs have occupied local computational resources such as memory (e.g., memory stack occupation [24]), cache and flash storage, which affects the efficiency of employing those resources for local execution of the application phase $\mathcal{G}$. $D_{usr}(\mathcal{Q})$ denotes the current delay of all the $\mathcal{Q}$ jobs which are stored in the mobile user's queue. The delay of a job is defined as the period of time from when the job is generated to when the job is successfully completed (i.e., removed from the queue). Therefore, one unit of delay per job is incurred when the job remains in the queue for the current decision period. The immediate cost of delay $D_{usr}(\mathcal{Q})=\mathcal{Q}$ given there are $\mathcal{Q}$ jobs remained in the queue for the current decision period.

Since the functions $E_{CPU}(\mathcal{G})$, $C_{sch}(\mathcal{G},\mathcal{Q})$ and $D_{usr}(\mathcal{Q})$ represent different types of cost with different units. Therefore, we apply w_{L1}, w_{L2} and w_{L3} as the weight factors to combine different types of costs into a universal cost function $C_L(\mathcal{S})$. This cost function quantifies the immediate cost incurred when local execution happens. The weight factors can be set based on the mobile user's preference to different types of costs in practical systems.

By contrast, the immediate offloading cost $C_R(\mathcal{S})$ is incurred when the mobile user offloads the application phase to the cloudlet. This cost is expressed as follows:

$$\begin{aligned} C_R(\mathcal{S}) = {} & w_{R1}C_{pay}(\mathcal{G}) + w_{R2}E_{Tx}(\mathcal{G},r_{ch}) \\ & + w_{R3}C_{pen}(\mathcal{G},\mathcal{N}) + w_{R4}D_{usr}(\mathcal{Q}). \end{aligned} \tag{5}$$

Similarly, w_{R1} to w_{R4} are the weight factors. $C_{pay}(\mathcal{G}) = C_{bw} + C_{cloud}(\mathcal{G})$ is the payment for bandwidth usage C_{bw} and cloudlet resources usage $C_{cloud}(\mathcal{G})$. $E_{Tx}(\mathcal{G},r_{ch})$ is the energy consumption when offloading the application phase $\mathcal{G}$ via a wireless connection with the transmission rate r_{ch}. According to [25], $E_{Tx}(\mathcal{G},r_{ch}) = P_{cir}\cdot\frac{\delta(\mathcal{G})}{r_{ch}}$, where P_{cir} is the power consumption of the transmitter circuit. $C_{pen}(\mathcal{G},\mathcal{N})$ is the penalty cost (e.g., the dissatisfaction level of the user) if the offloaded phase $\mathcal{G}$ fails, i.e., no result returns to the mobile user. The exact form of $C_{pen}(\mathcal{G},\mathcal{N})$ depends on the pattern of offloading. For example, when $\mathcal{G}$ is sent to $\mathcal{N}$ cloudlets simultaneously, failure happens only when all the $\mathcal{N}$ cloudlets refuse or fail to execute the offloaded phase $\mathcal{G}$. In this case, the penalty cost can be defined as $C_{pen}(\mathcal{G},\mathcal{N}) = (1-\eta_a)^{\mathcal{N}}c_{pen}$, where $(1-\eta_a)^{\mathcal{N}}$ is the probability that offloading fails, and c_{pen} is a constant. Other forms of offloading patterns and penalty functions may also apply without altering the model. When the mobile user only has a single queue, the jobs will be blocked in the queue even when the current phase is offloaded to the remote cloudlets. In this case, the job delay $D_{usr}(\mathcal{Q})$ also exists.

4 Optimization Formulation

In this section, we propose an MDP model to describe the optimal offloading problem of a mobile user. First, we define the state and action spaces of the mobile user. State transition matrices are then derived. Afterwards, the MDP optimization problem is expressed by a Bellman equation, and solved by employing a value iteration algorithm. Additionally, a threshold policy is found in the solutions of the MDP, based on which an approximation offloading decision algorithm is proposed.

4.1 State Space and Action Space

The state space of the mobile user in the cloudlet system is defined as follows:

$$\Theta = \{\mathcal{S} = (\mathcal{G}, \mathcal{Q}, \mathcal{N}) \in \mathbb{S} | \mathcal{G} \in \mathbb{G}, \mathcal{Q} \in \mathbb{Q}, \mathcal{N} \in \mathbb{N}\}, \quad (6)$$

where $\mathcal{G}$ is the application phase to be executed (either locally or remotely) by the mobile user, $\mathcal{Q} \in \mathbb{Q} = \{0, 1, \ldots, |Q|\}$ is the queue state (i.e., the number of jobs in the queue), and $\mathcal{N} \in \mathbb{N} = \{0, 1, \ldots, |N|\}$ is the number of cloudlets. $\mathbb{G} = \{0\} \cup \{1, \ldots, |G|\}$ is the set of application phases, where $\mathcal{G} = 0$ if the user is idle. $\mathbb{G}_{EP} \subseteq \mathbb{G}$ is the set of exit phases (e.g., $\mathbb{G}_{EP} = \{3, 4\}$ in Fig. 2), which are the final phases of the application. $|G|$, $|Q|$ and $|N|$ are the maximum values of states $\mathcal{G}$, $\mathcal{Q}$ and $\mathcal{N}$, respectively.

The action space is $\mathbb{A} = \{\mathcal{A} = 0, \mathcal{A} = 1\}$, denoting that a mobile user can make a decision to execute an application phase locally on a mobile device (i.e., $\mathcal{A} = 0$), or offload to the accessible cloudlet(s) (i.e., $\mathcal{A} = 1$).

4.2 Transition Matrices

In the following, we derive the transition matrices for the states of the mobile user.

4.2.1 The Number of Cloudlet Transitions

The number of cloudlets $\mathcal{N}$ of the mobile user is distributed as an HPPP, independent of the application phase $\mathcal{G}$ and the queue state $\mathcal{Q}$. The transition probability of the number of cloudlets $\mathcal{N}$ is expressed as follows:

$$\begin{aligned} &\mathsf{P}^N(\mathcal{N}, \mathcal{N}') = \mathsf{Pr}\{K = \mathcal{N}'\} = e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^{\mathcal{N}'}}{\mathcal{N}'!}, \\ &\forall\, \mathcal{N} \in \{0, 1, \ldots, |N|\}, \forall\, \mathcal{N}' \in \{0, 1, \ldots, |N| - 1\}. \end{aligned} \quad (7)$$

We assume that the maximum number of cloudlets is finite and known (i.e., $|N| < \infty$). Therefore, the probability that the mobile user has $|N|$ accessible cloudlets within the communication rage R is truncated as follows:

$$\mathsf{P}^N(\mathcal{N}, |N|) = \sum_{k=|N|}^{\infty} \mathsf{Pr}\{K = k\} = \sum_{k=|N|}^{\infty} e^{-\pi R^2 \lambda_c} \frac{\left(\pi R^2 \lambda_c\right)^k}{k!}. \quad (8)$$

The transition matrix for the number of cloudlets $\mathcal{N}$ is denoted by $\mathbf{N}$. It is expressed in (9), where a row in the transition matrix (9) of the number of cloudlets $\mathcal{N}$ indicates the current number of cloudlets $\mathcal{N}$, while a column indicates the number of cloudlets $\mathcal{N}'$ of the next decision period. Each element is a probability of state transition.

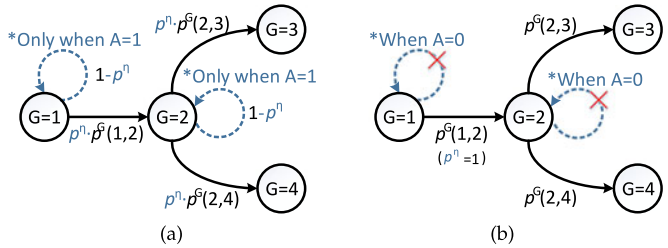

Fig. 4. (a) Transitions of application phase $\mathcal{G} = 1$ and $\mathcal{G} = 2$ given action $\mathcal{A} = 1$ is taken, and (b) transitions when action $\mathcal{A} = 0$.

period after the current phase i, where $i, j \in \{1, \ldots, |G|\}$. As in [6], we assume that there is no cyclic self-transition from phase i to i (i.e., $p_{ii} = 0$) in a mobile application. Note that typical applications suitable for offloading should have a finite loop. The finite loop can be divided into a sequence of phases without self-transition.

However, application phase transitions also depend on the action taken by the user (i.e., offloading or executing locally). If the action is to offload (i.e., $\mathcal{A} = 1$), offloading failures may happen with the probability $1 - \eta_a$. In this case, the user has to restore the failed phase. This is illustrated in Fig. 4a for phases $\mathcal{G} = 1$ and $\mathcal{G} = 2$. By contrast, if the action is to execute an application phase locally (i.e., $\mathcal{A} = 0$), the phase will deterministically transit to the next phase (i.e., without failure), as shown in Fig. 4b. We define $p^\eta(\mathcal{N}, \eta_a | \mathcal{A})$ as the probability that the current application phase is successfully executed locally or remotely by taking action $\mathcal{A}$, so that the mobile user is allowed to transit to the next phase. $p^\eta(\mathcal{N}, \eta_a | \mathcal{A})$ is defined as follows:

$$p^\eta(\mathcal{N}, \eta_a | \mathcal{A}) = \begin{cases} 1 - (1 - \eta_a)^{\mathcal{N}}, & \mathcal{A} = 1 \\ 1, & \mathcal{A} = 0, \end{cases} \quad (10)$$

where the term $1 - (1 - \eta_a)^{\mathcal{N}}$ indicates that all the $\mathcal{N}$ cloudlets fail to return the result of the offloaded application phase to the mobile user.

Given an action $\mathcal{A}$ taken by the mobile user as well as the probability $p^\eta(\mathcal{N}, \eta_a | \mathcal{A})$ that the current phase can be successfully executed locally or remotely, the transition probability of the application phase from $\mathcal{G}$ to $\mathcal{G}'$ is then expressed as follows:

$$\mathsf{P}^G(\mathcal{G}, \mathcal{G}' | \mathcal{A}) = \begin{cases} 1 - p^\eta(\mathcal{N}, \eta_a | \mathcal{A}), & \mathcal{G} = \mathcal{G}' \\ p^G(\mathcal{G}, \mathcal{G}') \cdot p^\eta(\mathcal{N}, \eta_a | \mathcal{A}), & \mathcal{G} \neq \mathcal{G}'. \end{cases} \quad (11)$$

In the following, we construct the overall transition matrix $\mathbf{G}$ for the application phase $\mathcal{G}$, given the transition probabilities derived.

$$\mathbf{N} = \begin{bmatrix} e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^0}{0!} & \cdots & e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^{|N|-1}}{(|N|-1)!} & \sum_{k=|N|}^{\infty} e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^k}{k!} \\ \vdots & \ddots & \vdots & \vdots \\ e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^0}{0!} & \cdots & e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^{|N|-1}}{(|N|-1)!} & \sum_{k=|N|}^{+\infty} e^{-\pi R^2 \lambda_c} \frac{(\pi R^2 \lambda_c)^k}{k!} \end{bmatrix}_{(|N|+1)\times(|N|+1)} \quad (9)$$

4.2.2 Application Phase Transitions

We first consider the application phase transitions without offloading failures. $p^G(i, j)$ is the transition probability that the application phase j will be executed in the next decision

First, $\mathbf{G}_0$ denotes the part of the transition matrix when the mobile user is currently idle, i.e., there is no job in the queue and the current application phase $\mathcal{G} = 0$. The matrix is expressed as follows:

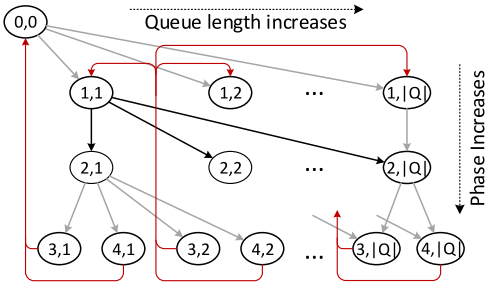

Fig. 5. Transition of the composite state $(\mathcal{G}, \mathcal{Q})$.

$$\mathbf{G}_0 = \left[\begin{array}{cccc} I_{\mathcal{Q}=0,\mathcal{Q}'=0}\cdot\mathbf{N} & I_{\mathcal{Q}=0,\mathcal{Q}'\geq 1}\cdot\mathbf{N} & 0 & \cdots \\ \hline \multicolumn{4}{c}{\mathbf{0}_{|G|\times(|G|+1)}} \end{array}\right]_{(|G|+1)\times(|G|+1)}. \quad (12)$$

The condition function $I_{\text{conditions}}$ in (12) returns 1 if the conditions hold. Otherwise, the function returns 0. For example, $I_{\mathcal{Q}=1,\mathcal{Q}'=0} = 1$ when the current queue state is $\mathcal{Q} = 1$ and the next queue state is $\mathcal{Q}' = 0$. Otherwise, $I_{\mathcal{Q}=1,\mathcal{Q}'=0} = 0$. $\mathbf{G}_0$ describes the transitions of the application phase in the following conditions: (1) $I_{\mathcal{Q}=0,\mathcal{Q}'\geq 1}\cdot\mathbf{N}$ is for the case that given the current queue state $\mathcal{Q} = 0$, which indicates the initial state of $(\mathcal{G}, \mathcal{Q}) = (0, 0)$ as shown in Fig. 5, the application phase $\mathcal{G}$ transits from the initial state $\mathcal{G} = 0$ to the first phase $\mathcal{G} = 1$ if new jobs arrive at the mobile user, i.e., $\mathcal{Q}' \geq 1$. One example of such a transition is shown as $(0, 0)$ to $(1, 1)$ in Fig. 5; (2) By contrast, $I_{\mathcal{Q}=0,\mathcal{Q}'=0}\cdot\mathbf{N}$ is for the case that, given the current queue state $\mathcal{Q} = 0$, the application phase $\mathcal{G}$ remains at the initial state $\mathcal{G} = 0$ if no new job arrives at the queue, i.e., $\mathcal{Q}' = 0$. The transition is thus $(0, 0)$ to $(0, 0)$.

$\mathbf{G}_{succ}$ is the part of the transition matrix for the cases when the mobile user is at non-exit application phases (e.g., $\mathcal{G} = 2$ in Fig. 2), and has a successful offloading. Without loss of generality, we denote the number of exit application phases to be $|\mathbb{G}_{EP}|$, and exit phases are labeled from $\mathcal{G} = |G| + 2 - |\mathbb{G}_{EP}|$ to $\mathcal{G} = |G| + 1$. The application phase $\mathcal{G}$ transits from the current state to the next state. The transition is defined as in (13), where $\mathcal{F}^G(i, j)$ is short for $p^G(i, j)\cdot p^\eta(\mathcal{N}, \eta_a|\mathcal{A})$, as in (11). The element $\mathcal{F}^G(i, j)$ in (11) is for the case that the application phase state $\mathcal{G}$ transits from $\mathcal{G} = i$ to $\mathcal{G} = j$. For example, as shown in Fig. 5, given $\mathcal{Q} = 2$, $\mathcal{F}^G(1, 2)$ indicates the transition from the composite state $(1, 2)$ to $(2, \mathcal{Q}')$, where the value $\mathcal{Q}' \in \{2, 3, \ldots, |Q|\}$ depends on the transition of the queue state.

$$\mathbf{G}_{succ} = \left[\begin{array}{ccccccc} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & F^G(1,2)\cdot\mathbf{N} & \mathcal{F}^G(1,3)\cdot\mathbf{N} & \cdots & \mathcal{F}^G(1,|G|+1)\cdot\mathbf{N} \\ 0 & \mathcal{F}^G(2,1)\cdot\mathbf{N} & 0 & \mathcal{F}^G(2,3)\cdot\mathbf{N} & \cdots & \mathcal{F}^G(2,|G|+1)\cdot\mathbf{N} \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & \mathcal{F}^G(k,1)\cdot\mathbf{N} & \mathcal{F}^G(k,2)\cdot\mathbf{N} & \cdots & \cdots & \mathcal{F}^G(k,|G|+1)\cdot\mathbf{N} \\ \hline \multicolumn{6}{c}{\mathbf{0}_{|\mathbb{G}_{EP}|\times(|G|+1)}} \end{array}\right]_{(|G|+1)\times(|G|+1)} \quad (13)$$

$\mathbf{G}_{exit}$ is a $(|G| + 1) \times (|G| + 1)$ matrix denoting the part of transition matrix for the case when the mobile user is at exit phases (e.g., $\mathcal{G} = 4$ in Fig. 2), and has a successful offloading action. It is defined as follows:

$$\mathbf{G}_{exit} = \left[\begin{array}{ccccc} \mathbf{0}_{(|G|+1-|\mathbb{G}_{EP}|)\times(|G|+1)} & & & & \\ p^\eta I_{\mathcal{Q}=1,\mathcal{Q}'=0}\cdot\mathbf{N} & p^\eta I_{\mathcal{Q}\geq 1,\mathcal{Q}'=\mathcal{Q}-1}\cdot\mathbf{N} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ p^\eta I_{\mathcal{Q}=1,\mathcal{Q}'=0}\cdot\mathbf{N} & p^\eta I_{\mathcal{Q}\geq 1,\mathcal{Q}'=\mathcal{Q}-1}\cdot\mathbf{N} & 0 & \cdots & 0 \end{array}\right]. \quad (14)$$

In the case shown in Fig. 5, $p^\eta I_{\mathcal{Q}=1,\mathcal{Q}'=0}\cdot\mathbf{N}$ describes the transition from $(3, 1)$ or $(4, 1)$ (i.e., with $\mathcal{G}$ to be exit phases) to the initial idle state $(0, 0)$. This happens when the last job in the queue is completed. $p^\eta I_{\mathcal{Q}\geq 1,\mathcal{Q}'=\mathcal{Q}-1}\cdot\mathbf{N}$ describes the transitions from exit phases to $\mathcal{G} = 1$ given there is still at least one job remained in or arrived at the queue, e.g., from $(3, 2)$ to $(1, 1)$, and from $(3, 2)$ to $(1, |Q|)$ as shown in Fig. 5.

$\mathbf{G}_{fail}$ is a $(|G| + 1) \times (|G| + 1)$ matrix denoting the part of the transition matrix for the unsuccessful offloading case. The application $\mathcal{G}$ has to be processed again with probability $1 - p^\eta(\mathcal{N}, \eta_a|\mathcal{A})$, e.g., the transition from $\mathcal{G} = 1$ to $\mathcal{G} = 1$ in Fig. 4a. $\mathbf{G}_{fail}$ is defined as follows:

$$\mathbf{G}_{fail} = \left[\begin{array}{ccc} 0 & & \\ & (1 - p^\eta(\mathcal{N}, \eta_a|\mathcal{A}))\cdot\mathbf{N} & \\ & & \ddots \\ & & (1 - p^\eta(\mathcal{N}, \eta_a|\mathcal{A}))\cdot\mathbf{N} \end{array}\right]. \quad (15)$$

The overall transition matrix for the application phase $\mathcal{G}$ denoted as $\mathbf{G} = \{\mathsf{P}^G(\mathcal{G}, \mathcal{G}'|\mathcal{N}, \mathcal{A})\}$, which includes the aforementioned parts of application phase transition matrices, shown as follows:

$$\mathbf{G} = \mathbf{G}_0 + \mathbf{G}_{succ} + \mathbf{G}_{exit} + \mathbf{G}_{fail}. \quad (16)$$

4.2.3 Queue State Transition

The job arrival follows the Poisson process, with arrival rate λ_q. In every decision period, the probability of k new arriving jobs at the local queue is $\mathsf{Po}(k) = P[N(t+1) - N(t) = k] = \frac{e^{-\lambda_q}\lambda_q^k}{k!}$ for $k = 0, 1, \ldots,$ where $N(t)$ is the number of jobs arrived up to time t.

Fig. 5 shows an example transition diagram of queue state and application phase, without showing offloading failure. In each decision period, the number of jobs in the queue can increase until the queue is full (i.e., $\mathcal{Q} = |Q|$). In particular, the queue state transition probability is obtained as follows, given $\mathcal{G} \notin \mathbb{G}_{EP}$:

$$\mathsf{P}^{Q}(\mathcal{Q},\mathcal{Q}')=\begin{cases}0, & \mathcal{Q}>\mathcal{Q}'\\ \mathsf{Po}(\mathcal{Q}'-\mathcal{Q}), & \mathcal{Q}\le\mathcal{Q}' \text{ and } \mathcal{Q}'<|Q|\\ 1-\sum_{k=0}^{\mathcal{Q}'-\mathcal{Q}-1}\mathsf{Po}(k), & \mathcal{Q}<|Q| \text{ and } \mathcal{Q}'=|Q|\\ 1, & \mathcal{Q}=\mathcal{Q}'=|Q|\\ 0, & \text{otherwise.}\end{cases} \tag{17}$$

For simplicity, the conditional parameters $\mathcal{G}$ (which is $\notin \mathbb{G}_{EP}$ or $\in \mathbb{G}_{EP}$) and $\mathcal{A}=0$ are omitted in the function of $\mathsf{P}^{Q}(\mathcal{Q},\mathcal{Q}'|\mathcal{G},\mathcal{A})$ in (17) and (18).

Only after the current application has finished exit phases (e.g., phases $\mathcal{G}=3$ and $\mathcal{G}=4$ in Fig. 2), the queue state may decrease by one, if there is no new job arrival, e.g., $(\mathcal{G}=4,\mathcal{Q}=2)$ to $(\mathcal{G}=1,\mathcal{Q}=1)$ in Fig. 5. The queue size may also increase, as shown by state $(\mathcal{G}=3,\mathcal{Q}=2)$ in Fig. 5. Given that the current phase is an exit phase, i.e., $\mathcal{G}\in\mathbb{G}_{EP}$, the queue state transition probability is

$$\mathsf{P}^{Q}(\mathcal{Q},\mathcal{Q}')=\begin{cases}\mathsf{Po}(0), & \mathcal{Q}'=\mathcal{Q}-1\\ \mathsf{Po}(\mathcal{Q}'-(\mathcal{Q}-1)), & \mathcal{Q}\le\mathcal{Q}'<|Q|\\ 1-\sum_{k=0}^{\mathcal{Q}'-\mathcal{Q}}\mathsf{Po}(k), & \mathcal{Q}'=|Q|\\ 0, & \text{otherwise.}\end{cases} \tag{18}$$

The transition matrix of a queue state $\mathcal{Q}$ is defined to consist of two parts, $\mathbf{Q}_{incr}$ and $\mathbf{Q}_{decr}$.

$\mathbf{Q}_{incr}$, defined as (19), denotes the part of transition matrix for the case that the queue size does not decrease, which is the expression for the case of $\mathcal{Q}\le\mathcal{Q}'<|Q|$ in (18). This case occurs when the current application phase is not an exit phase. In the definition of $\mathbf{Q}_{incr}$ in (19), $\mathscr{T}^{Q}(i,j)=\mathsf{Po}(j-i+1)\cdot\mathbf{G}_{exit}+\mathsf{Po}(j-i)\cdot(\mathbf{G}_{succ}+\mathbf{G}_{fail})$. The first term $\mathsf{Po}(j-i+1)\cdot\mathbf{G}_{exit}$ in $\mathscr{T}^{Q}(i,j)$ indicates the situation that the current application phase is an exit phase, i.e., queue length first decreases by one and then increases by $j-i+1$. The second term indicates the situation that the current application phase is not an exit phase, i.e., queue length does not decrease and then increases by $j-1$. Similarly, $\mathscr{T}^{Q}_{trunc}(i)=\mathbf{G}_{exit}\sum_{k=|Q|-i+1}^{\infty}\mathsf{Po}(k)+(\mathbf{G}_{succ}+\mathbf{G}_{fail})\sum_{k=|Q|-i}^{\infty}\mathsf{Po}(k)$.

$$\mathbf{Q}_{incr}=\begin{bmatrix}\mathsf{Po}(0)\cdot\mathbf{G}_0 & \mathsf{Po}(1)\cdot\mathbf{G}_0 & \cdots & \mathsf{Po}(|Q|-1)\cdot\mathbf{G}_0 & \sum_{k=|Q|}^{\infty}\mathsf{Po}(k)\cdot\mathbf{G}_0\\ & \mathscr{T}^{Q}(1,1) & \cdots & \mathscr{T}^{Q}(1,|Q|-1) & \mathscr{T}^{Q}_{trunc}(1)\\ & & \ddots & \ddots & \vdots\\ & & & \mathscr{T}^{Q}(|Q|-1,|Q|-1) & \mathscr{T}^{Q}_{trunc}(|Q|-1)\\ & & & & \mathscr{T}^{Q}_{trunc}(|Q|)\end{bmatrix}_{(|Q|+1)\times(|Q|+1)} \tag{19}$$

$\mathbf{Q}_{decr}$ is part of the transition matrix for the case that the queue size in the mobile user decreases by one in the next decision period, which describes the first expression for the case of $\mathcal{Q}'=\mathcal{Q}-1$ in (18). It is defined as follows:

$$\mathbf{Q}_{decr}=\begin{bmatrix}0 & & & \\ \mathsf{Po}(0)\cdot\mathbf{G}_{exit} & 0 & & \\ & \ddots & \ddots & \\ & & \mathsf{Po}(0)\cdot\mathbf{G}_{exit} & 0\end{bmatrix}_{(|Q|+1)\times(|Q|+1)}. \tag{20}$$

In this case, $\mathsf{Po}(0)\cdot\mathbf{G}_{exit}$ indicates that the current application phase is an exit phase, and no new job arrives at the queue.

The overall transition matrix for a queue state is defined as follows:

$$\mathbf{Q}=\mathbf{Q}_{incr}+\mathbf{Q}_{decr}. \tag{21}$$

The complexity of constructing the state transition matrices (9), (16) and (21) is $\mathcal{O}(|\mathbb{S}|^2)$, where $|\mathbb{S}|$ is the total number of all the states.

4.3 Solving the Optimal Policy for the MDP

Given the proposed MDP, a policy for the MDP is denoted as $\Phi(\mathcal{S},\mathcal{A})$, which is the probability of taking an action $\mathcal{A}$ at state $\mathcal{S}$. An optimal policy for the proposed MDP aims to optimize the offloading actions of the mobile user to minimize cost.

To obtain the optimal policy, the following Bellman equation [26] based on the defined state space and derived transition matrices has to be solved:

$$V(\mathcal{S})=\min_{\Phi(\mathcal{S},\mathcal{A})}H(\mathcal{S},\mathcal{A}) \tag{22}$$

$$\phi^{*}(\mathcal{S})=\arg\min_{\Phi(\mathcal{S},\mathcal{A})}H(\mathcal{S},\mathcal{A})$$

$$H(\mathcal{S},\mathcal{A})=C(\mathcal{S},\mathcal{A})+\gamma\sum_{\mathcal{S}'\in\mathbb{S}}\mathsf{P}^{S}(\mathcal{S},\mathcal{S}'|\mathcal{A})V(\mathcal{S}'),$$

where $\mathcal{S}=(\mathcal{G},\mathcal{Q},\mathcal{N})$. $V(\mathcal{S})$ is a *value function* of the mobile user. $\phi^{*}(\mathcal{S})$ denotes the optimal policy (i.e., the function that returns an optimal action for the user). $H(\cdot)$ is a *cost function* of MDP. $\gamma\in[0,1)$ is a discount factor of future states. $C(\mathcal{S},\mathcal{A})$ is the immediate cost with respect to the state $\mathcal{S}$ of the current decision period, as defined in Section 3.2.3. $\mathsf{P}^{S}(\mathcal{S},\mathcal{S}'|\mathcal{A})$ is the transition probability of the mobile user from the current state $\mathcal{S}$ to the next state $\mathcal{S}'$, which can be obtained from the transition matrices (9), (16) and (21). The Bellman equation can be solved numerically by employing the value iteration algorithm [27].

4.4 Threshold Policy in Offloading Decision Making

In this section, we introduce the concept of threshold policy and show its existence in the optimal policy obtained from the proposed MDP.

In an MDP model with binary choice of actions (i.e., $\mathcal{A}=a_1$ or a_2), a threshold policy is in the following form

$$\phi^{*}(\theta,\cdot)=\begin{cases}a_1, & \text{for } \theta\succ\theta_{thresh}\\ a_2, & \text{otherwise,}\end{cases} \tag{23}$$

where $\phi^*(\cdot)$ is the optimal policy function. It is defined that a state θ has a threshold policy, and the threshold is θ_{thresh}. The optimal action scheme $\phi^*(\theta,\cdot)$ is defined to have a threshold type.

Once the MDP is proved to have a threshold policy and the threshold θ_{thresh} is obtained, the user only needs to determine the threshold, and compare the current state θ with the threshold θ_{thresh}. Whenever the current state $\theta \succ \theta_{thresh}$, the action $\mathcal{A} = a_1$ should be taken. Otherwise, the action $\mathcal{A} = a_0$ should be taken.

To prove the existence of the threshold policy, we employ the method in [26], [28], i.e., to prove the supermodularity (submodularity) [28] feature of the cost function in the Bellman Equation (22). That is, the cost function $H(\mathcal{S},\mathcal{A})$ has to be supermodular (or submodular) with respect to $(\mathcal{Q},\mathcal{A})$ and $(\mathcal{N},\mathcal{A})$, so that the states $\mathcal{Q}$ and $\mathcal{N}$ have threshold structures, and the optimal policy solved by MDP is a threshold policy.

Definition 1. *A function $f(x,y) : (x \in \mathbb{X} \subseteq \mathbb{R}) \times (y \in \mathbb{Y} \subseteq \mathbb{R}) \in \mathbb{R}$ is supermodular in (x,y) if $f(x_1,y_1) - f(x_1,y_2) \geq f(x_2,y_1) - f(x_2,y_2)$, $\forall x_1,x_2 \in \mathbb{X}, \forall y_1,y_2 \in \mathbb{Y}, x_1 > x_2, y_1 > y_2$. Similarly, $f(x,y)$ is submodular in (x,y) if $f(x_1,y_1) - f(x_1,y_2) \leq f(x_2,y_1) - f(x_2,y_2), \forall\, y_1,y_2 \in \mathbb{Y}, x_1 > x_2, y_1 > y_2$.*

Theorem 1 (Threshold in the Number of Cloudlets). *Given any fixed queue state $\mathcal{Q}$ and application state $\mathcal{G}$, the optimal offloading decision is a threshold policy in the number of cloudlets $\mathcal{N}$, i.e.,*

$$\phi_t^*(\mathcal{S}) = \begin{cases} 1, & \text{for } \mathcal{N} > \mathcal{N}_{thresh} \\ 0, & \text{otherwise,} \end{cases} \tag{24}$$

given the immediate cost function $C(\mathcal{S},\mathcal{A})$ is monotonic in $\mathcal{N}$. The function $\phi_t^*(\mathcal{S})$ is the optimal policy of offloading actions, and $\mathcal{N}_{thresh}$ is the threshold in the number of cloudlets $\mathcal{N}$.

The proof of Theorem 1 is given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2015.2405539.

Theorem 2 (Threshold in queue state). *Given any number of cloudlets $\mathcal{N}$ and application state $\mathcal{G}$, the optimal offloading decision is a threshold policy in the queue state $\mathcal{Q}$, given any of the following conditions of immediate cost function $C(\cdot)$.*

- Condition 1. *Second order cross difference of the immediate cost function $C((\mathcal{G},\mathcal{Q},\mathcal{N}),\mathcal{A})$ in $\mathcal{G}$ and $\mathcal{Q}$ is greater than or equal to 0, i.e., $\frac{\Delta^2 C((\mathcal{G},\mathcal{Q},\mathcal{N}),\mathcal{A})}{\Delta\mathcal{G}\Delta\mathcal{Q}} \geq 0$, regardless of which action is taken, i.e., $\frac{\Delta^2 C((\mathcal{G},\mathcal{Q},\mathcal{N}),\mathcal{A})}{\Delta\mathcal{G}\Delta\mathcal{Q}}|_{\mathcal{A}_1} = \frac{\Delta^2 C((\mathcal{G},\mathcal{Q},\mathcal{N}),\mathcal{A})}{\Delta\mathcal{G}\Delta\mathcal{Q}}|_{\mathcal{A}_2}$, $\forall \mathcal{A}_1,\mathcal{A}_2 \in \mathbb{A}$, the cross difference is not a function of action $\mathcal{A}$.*
- Condition 2. *All the application phases executed by the user are with the same size, which cause the same base cost, i.e., $C((\mathcal{G}_1,\mathcal{Q},\mathcal{N}),\mathcal{A}) = C((\mathcal{G}_2,\mathcal{Q},\mathcal{N}),\mathcal{A})$, $\forall \mathcal{G}_1,\mathcal{G}_2 \in \mathbb{G}$, given the same $\mathcal{Q}$, $\mathcal{N}$ and action $\mathcal{A}$.*

The proof of Theorem 2 is given in Appendix B, available in the online supplemental material.

The threshold policy could assist a mobile user to make offloading decisions in the following aspects:

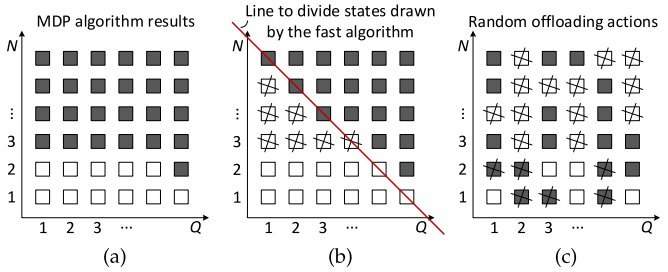

Fig. 6. (a) Optimal results calculated from an MDP algorithm for comparison, (b) fast equal-division algorithm, and (c) random action decision scheme.

- *Simplifying device's design.* For a user device executing a simple application (i.e., sensors), the control logic of optimal offloading decision making could be preset when manufacturing. That is, the threshold values of all the possible states of the device (i.e., potential mobile cloudlet user) are calculated in advance (e.g., employing the value iteration algorithm) and implanted into the device as sole discrete values or functions. As a result, the device simply compares its current state with the preset threshold and makes an offloading decision, instead of calculating or storing the action for every state. Thus, the hardware logic design will be simplified, since the decision logic is binary.
- *Fast threshold algorithm*. For resource-limited mobile users making dynamic offloading decisions, fast decision algorithms could also be designed to make use of the advantages of threshold policy without solving the MDP model. The idea is similar to the simple threshold activation policy in [29]. In the following, we propose a fast approximation algorithm, which has the complexity of $\mathcal{O}(1)$ compared with the complexity $\mathcal{O}(|\mathbb{A}| \cdot |\mathbb{S}|^2)$ [30] of the value iteration algorithm, where $|\mathbb{A}|$ and $|\mathbb{S}|$ are the number of actions and states, respectively. The proposed fast algorithm does not yield worst decisions under any circumstances compared with the baseline offloading schemes. In other words, the fast algorithm leads to acceptable error-bounded results without complicated and time-consuming calculation process.

We propose an equal-division decision making algorithm based on the threshold policy. The idea of the equal-division algorithm is simple, as follows:

1) Suppose the state space $\mathbb{S}_{thresh}$ contains all the states that have a threshold type. The user divides the state space $\mathbb{S}_{thresh}$ equally into two sides, e.g., drawing a line when the $\mathbb{S}_{thresh}$ is two-dimensional, as shown in the example in Fig. 6b. The higher dimensional state space partition is done in the same manner.
2) The user takes action $\mathcal{A} = 0$ when the current state is on one side, and takes action $\mathcal{A} = 1$ when the state is on the other side. To decide the action to be taken on each side, the supermoduality (or submodularity) feature of the immediate cost function $C(\cdot)$ has to be known, as discussed in Appendices A and B, available in the online supplemental material.

3) For the boundary states (e.g., the states on the line in Fig. 6b), they are divided equally in the similar manner.

For example, if only the state $\mathcal{N}$ is considered (i.e., one dimensional threshold), where $|N| = 5$, the user will take action $\mathcal{A} = 0$ when $\mathcal{N} \in \{0, 1, 2\}$, and action $\mathcal{A} = 1$ when $\mathcal{N} \in \{3, 4, 5\}$.

It is clear that when the equal-division algorithm is employed, the user will at most have $\frac{|\mathbb{S}|}{2}$ suboptimal decisions, compared with the optimal actions derived by the MDP algorithm, where $|\mathbb{S}|$ is the maximum number of composite states. That is, the upper bound of suboptimal decisions is $\frac{|\mathbb{S}|}{2}$ compared with the random action decision making scheme and all local/remote action schemes (introduced as baseline offloading schemes in Section 6.1), which have up to $|\mathbb{S}|$ suboptimal actions in the extreme cases. For example, with the parameter settings used in Section 6, Fig. 6a shows the optimal actions derived by the MDP algorithm, where the filled boxes in the figure indicate $\mathcal{A} = 1$ at the corresponding system states, and $\mathcal{A} = 0$ otherwise. Compared to Fig. 6a, the fast algorithm yields only seven suboptimal actions, indicated by the crossed boxes shown in Fig. 6b. A random action decision case is shown in Fig. 6c, where there are 19 suboptimal actions compared with that of the MDP optimal actions.

5 Mobility of Mobile Users: Cloudlet Availability Estimation

The optimization formulation in Section 4 requires the cloudlet availability η_a. In this section, we estimate the value of η_a based on some assumptions of mobile user's distribution and mobility.

5.1 Distance Priority-Based Cloudlet Admission Control

The cloudlet may refuse to accept the offloading requests from the mobile users. Consequently, cloudlet admission control is an important factor which affects the cloudlet availability. The admission control policy may vary. We design a priority-based admission control policy similar to that in [18]. That is, users in the same cloudlet coverage area have different priorities according to their distances to the cloudlet when they offload simultaneously. Nevertheless, other admission control policies can also be adopted into the proposed model.

We assume that the capacity of a cloudlet is limited. Rationally, the cloudlet has to execute its own jobs first, and shares the rest of resources to serve offloading users. We define a concept named External Service Ratio (ESR) metric, which is similar to Signal to Interference plus Noise Ratio (SINR), as follows. Only active users who are in the coverage area and currently offloading to the cloudlet are considered. Let us focus on user u. Note that there could be other users u_i, $i \in \{1, 2, \ldots\}$ in the coverage area of the same cloudlet. The definition of ESR is given as follows:

$$\beta_{ESR}(r) = \frac{g r^{-c}}{E_s + \sum g_i r_i^{-c}}, \tag{25}$$

where we further assume all $g_i \equiv g$ to be the energy consumption (e.g., transmission power) by the cloudlet to process a job offloaded by any user. E_s is the reserved energy to process the cloudlet's own jobs. r is the user u's distance to the cloudlet. Although the exact distance of a user to the cloudlet may not be known, the cloudlet can estimate it indirectly, e.g., by measuring the user's signal strength. c is a positive constant (e.g., $c = 1$) indicating the sensitivity of r (as well as r_i) to the function $\beta_{ESR}(r)$. By adopting ESR, a user located closer to the cloudlet has relatively larger ESR and will be possibly served with higher probability. We also assume that the cloudlet has a predetermined constant $\hat{\beta}$, only above of which the user's offloading request will be accepted by that cloudlet (i.e., $\beta_{ESR} \geq \hat{\beta}$).

In the following, we derive the probability that the cloudlet accepts an offloading request. With the help of guard zone concept [31], from the perspective of user u located at a distance r from the cloudlet, we define the corresponding virtual zone centered at the cloudlet with radius $R_g(r)$. Once an active user who also offloads appears in the virtual zone $R_g(r)$, the user u's ESR drops below $\hat{\beta}$ (i.e., $\beta_{ESR} < \hat{\beta}$). As a result, any offloading request from the user u will be rejected. The virtual zone radius $R_g(r)$ is calculated as follows [32]:

$$R_g(r) = \left(\frac{r^{-c}}{\hat{\beta}} - \frac{E_s}{g} \right)^{-\frac{1}{c}}, \tag{26}$$

where $\frac{E_s}{g}$ indicates the cloudlet's own jobs. When the cloudlet's own job is relatively less energy-consuming or negligible, $\frac{E_s}{g} = 0$, such that $R_g(r) = \hat{\beta}^{\frac{1}{c}} r$. Then the failure rate (i.e., ESR outage) $1 - \varepsilon(r)$ can be expressed as follows [32]:

$$1 - \varepsilon(r) = 1 - e^{-2\pi\lambda_{au}[R_g(r)]^2}, \tag{27}$$

which yields the admission probability $\varepsilon(r) = e^{-2\pi\lambda_{au}[R_g(r)]^2}$. In the expression, λ_{au} is the density of active users. In practice, $\lambda_{au} \leq \lambda_u$ since only some of existing users will be active. We denote $\lambda_{au} = c_a \lambda_u$, where c_a is a constant. The expression of $\varepsilon(r)$ indicates that the probability that the user can be accepted by a cloudlet depends on the density of active users in the system.

5.2 Dwell Time and Cloudlet Availability Estimation

In this section, we determine the dwell time of a user in a cloudlet coverage, and then derive the estimated cloudlet availability.

Assuming that the users are distributed in HPPP, the probability density that the user is located at a distance r from the cloudlet in the center of the circle is derived as follows:

$$\psi(r) = \frac{2\pi\lambda_u e^{-\pi\lambda_u r^2}}{1 - e^{-\pi\lambda_u R^2}} r, \tag{28}$$

where λ_u is mobile users' density, r is the user's distance to the cloudlet, and R is the coverage radius of the cloudlet, where $r \leq R$.

For example, we assume that the user moves straight in the cloudlet coverage. When the user moves in the centrifugal direction to leave the cloudlet coverage area (i.e., along

the fastest path) with radius R, the dwell time is $T(r) = \frac{R-r}{v}$, where v is the relative velocity of the user to the cloudlet. We define this to be *the worst case path*. By contrast, when the user moves in the centripetal direction, defined as *the best case path*, the dwell time is $T(r) = \frac{R+r}{v}$.

Considering the cloudlet admission control policy and the mobility of mobile users, the cloudlet availability is obtained as follows:

- *Worst case*:

$$\begin{aligned}\eta_{a,we} &= \mathop{\mathbb{E}}_{r}\left[\varepsilon(r)\frac{T(r)-\tau}{T(r)}\right] \\ &= \int_0^{R-v\tau} \psi(r)\cdot\varepsilon(r)\cdot\frac{T(r)-\tau}{T(r)}dr \\ &= \frac{2\pi\lambda_u}{1-e^{-\pi\lambda_u R^2}}\int_0^{R-v\tau}\Big(1-\frac{v\tau}{R-r}\Big)e^{\Lambda(r)}rdr,\end{aligned} \tag{29}$$

  where $\Lambda(r) = -\pi\lambda_u r^2 - 2\pi\lambda_{au}[R_g(r)]^2$. The upper bound $R - v\tau$ of an integral has the physical meaning that a user located at radius $R - v\tau$ to R cannot offload jobs, since the dwell time $T(r)$ of the user is less than τ. As a result, $\eta_{a,we} = 0$ when $R - v\tau < r \leq R$.
- *Best case*:

$$\eta_{a,be} = \frac{2\pi\lambda_u}{1-e^{-\pi\lambda_u R^2}}\int_0^{R}\left(1-\frac{v\tau}{R+r}\right)e^{\Lambda(r)}rdr \tag{30}$$

  which is derived similarly to (29) by using best case dwell time. Note that the upper bound of integral is R instead of $R - v\tau$.
- *Average case*: A conservative way for the user to estimate the cloudlet availability is to use a weighted average of best and worst case, i.e.,

$$\eta_{a,avg} = \alpha\eta_{a,we} + (1-\alpha)\eta_{a,be}, \tag{31}$$

  where $\alpha \in (0,1)$ is the weight, e.g., $\alpha = \frac{1}{2}$.

6 Numerical Results

6.1 Parameter Setting and Performance Metrics

We use the linear structured face recognition program [7], [20] as an example to illustrate the numerical results.

Unless otherwise stated, the following scenario and parameter settings are employed in the performance evaluation. The application phases of the face recognition in our discussed system are shown in Fig. 2. The queue can store six jobs (i.e., $|Q| = 6$). The rate of generating new job of the mobile user is $\lambda_q = 0.25$ jobs per minute. For the mobility parameters of the system: The density of mobile users is $\lambda_u = 0.0001$ per m^2. The mobile user has the velocity of $5.0\ \mathrm{m/s}$. The cloudlets have the coverage radius of $R = 10$ meters. The density of cloudlets is $\lambda_c = 0.0005$ per m^2. For the admission control parameters as discussed in Section 5: the density of active mobile users is $\lambda_{au} = 10\% \cdot \lambda_u$; the ESR constant is $\hat{\beta} = 0.5$ and the sensitivity constant of a cloudlet is $c = 2$. The execution time of the offloaded application phase on the cloudlet is $\tau = 0.5\,\mathrm{s}$. For the value iteration algorithm, the discount factor is $\gamma = 0.8$.

Various customized forms of cost function $C(\mathcal{S})$ can be adopted in the MDP model. As an example, the immediate cost function $C(\mathcal{S})$ defined in Section 3.2.3 is set as follows:

- In the immediate local execution cost $C_L(\mathcal{S})$ in (4), we set $\frac{P_{CPU}(\mathcal{G})}{s_{CPU}} = 1.0$ for all the application phases $\mathcal{G} \in \mathbb{G}$. The scheduling overhead is set to be $C_{sch}(\mathcal{G},\mathcal{Q}) = \frac{\mathcal{Q}}{|Q|}\cdot\delta(\mathcal{G})$. The delay is set as $D(\mathcal{Q}) = \mathcal{Q}$.
- For the immediate offloading cost $C_R(\mathcal{S})$ in (5), $C_{pay}(\mathcal{G})$ is set to be 0, i.e., bandwidth usage and cloudlet resources are free. Without loss of generality, $\frac{P_{cir}}{r_{ch}}$ is set to be 1.0. That is, the wireless transmission rate is fixed, and the transmission energy consumption $E_{CPU}(\mathcal{G})$ is only proportional to the size of phase $\delta(\mathcal{G})$. The penalty function is $C_{pen}(\mathcal{G},\mathcal{N}) = (1-\eta_a)^{\mathcal{N}}c_{pen}$, where the coefficient $c_{pen} = 2.5$ by default, and η_a is derived in Section 5. The delay of job is also set to be $D(\mathcal{Q}) = \mathcal{Q}$.
- All the weight factors in the immediate cost function are set to be $\frac{1}{3}$.

We evaluate the performance of the proposed MDP optimization model (Section 6.3) in terms of *expected cost* and *offloading rate*. We denote the probability that the mobile user starts to operate at the state $\mathcal{S}$ to be $p_{ini}(\mathcal{S})$. The expected cost $\mathscr{C}$ of the user is defined as $\mathscr{C} = \sum_{\mathcal{S}\in\mathbb{S}} p_{ini}(\mathcal{S})\cdot V(\mathcal{S})$. The offloading rate $\mathscr{R}$ is defined as the proportion of the user's offloading decisions made at all the states, i.e., $\mathscr{R} = \frac{1}{|\mathbb{S}|}\sum_{\mathcal{S}\in\mathbb{S}}\mathcal{A}(\mathcal{S})$, where $|\mathbb{S}|$ is the number of all the states.

We compare the performances of the proposed MDP offloading algorithm with other four baseline schemes:

- *Always executing locally scheme (LOC)*. The mobile user always executes application phases locally on the mobile device, i.e., $\mathcal{A} \equiv 0$;
- *Always offloading scheme (OFF)*. The mobile user always offloads to cloudlets, i.e., $\mathcal{A} \equiv 1$ (except when $\mathcal{N} = 0$);
- *Random scheme (RND)*. The user randomly chooses local execution or offloading action in each decision period;
- *Myopic scheme (MYO)*. The mobile user makes a short-sighted decision only based on immediate costs of the current decision period. The decision is defined as follows:

$$\mathcal{A} = \begin{cases} 0, & \text{for} \quad C_L(\mathcal{S},\mathcal{A}) < C_R(\mathcal{S},\mathcal{A}) \\ 1, & \text{for} \quad C_L(\mathcal{S},\mathcal{A}) \geq C_R(\mathcal{S},\mathcal{A}) \end{cases} \tag{32}$$

  which indicates that a local execution action will be taken if the immediate cost of local execution is lower than that of offloading, i.e., $C_L(\mathcal{S},\mathcal{A}) < C_R(\mathcal{S},\mathcal{A})$. Otherwise, the mobile user takes an offloading action.

6.2 Evaluation of Cloudlet Availability Model

In Section 3.2.2, we present the mobility model to estimate the cloudlet availability (Section 6.2) in an intermittently connected cloudlet system. To verify the proposed mobility model, we simulate three general cases with different user mobility patterns, as shown in Fig. 7a.

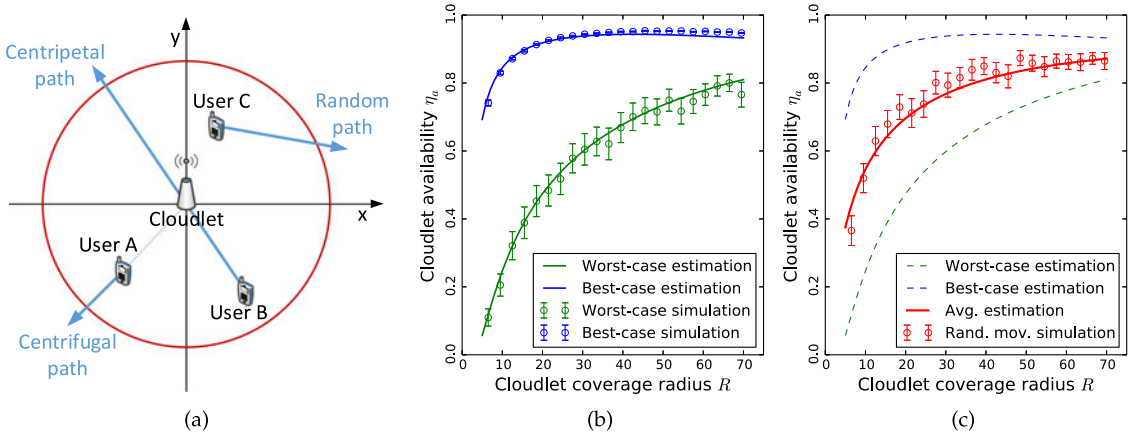

Fig. 7. Effect of mobility for (a) centrifugal, centripetal and random movement, (b) simulation results for centrifugal (worst-case) and centripetal (best-case) movements, and (c) simulation results for random direction movement.

In the simulation, the user is located at a random location in the cloudlet coverage area at the beginning of a decision period, following the spatial distribution of HPPP. As Fig. 7a shows, the user has different options when choosing a mobility path: centrifugal, centripetal and random direction.

At any time before the user moves out of the cloudlet coverage area, the user may start to offload a job and receive the result. We record the probability that the offloaded job is successfully executed. For each mobility pattern, $4{,}000$ sample users are tested independently, with the cloudlet coverage radius ranging from 5 to 70 m.

In Fig. 7b, the lines are the analytical results, while the points are the simulation results of user's average cloudlet availability, with 95 percent confidence interval. As the results shown from Fig. 7b, for both centrifugal (i.e., the worst case) and centripetal (i.e., the best case) mobility paths, the analytical results match closely with the simulation results. Therefore, the proposed estimations given in (29) and (30) are validated.

We also consider random mobility of the user. As Fig. 7c shows, compared with simulation results with 95 percent confidence interval, the analytical cloudlet availability obtained from (31) is acceptably accurate for when the user moves with random paths.

6.3 Impacts of Cloudlet to Optimization

In this section, we discuss how the parameters of mobile cloudlets affect the offloading decision and performance.

6.3.1 Number of Cloudlets-Coverage Radius and Density

We examine the impacts of cloudlets' coverage radius R and density λ_c to the offloading decision rate and cost. From Figs. 8a and 8c, as R or λ_c increases, the user's offloading rate increases. This is because with a larger cloudlet radius R or higher cloudlet density λ_c, the user has an access to more cloudlets. Consequently, as shown in Figs. 8b and 8d, the user's offloading cost decreases as R or λ_c increases. Furthermore, as shown in Figs. 8b and 8d, with the proposed MDP offloading algorithm, the user achieves lower expected cost than those of other schemes. Note that the expected cost of always executing locally remains constant, since the user in this case is not affected by cloudlets.

6.3.2 Cloudlet Availability

Fig. 9 shows the relationship between performances and cloudlet availability η_a, under different penalty coefficient c_{pen} of offloading failure. As expected, when there is no penalty in the cost function of offloading, the user just executes the application phase again if the offloaded phase fails. If the cost of local execution is more than that of offloading, the user still wants to offload even if η_a is small. This is observed for $c_{pen} = 0$ in Fig. 9a, where the offloading rate is higher than 0.3 for $\eta_a = 0.1$.

When the penalty in the cost function is significant, offloading is no longer an optimal action to the user. For

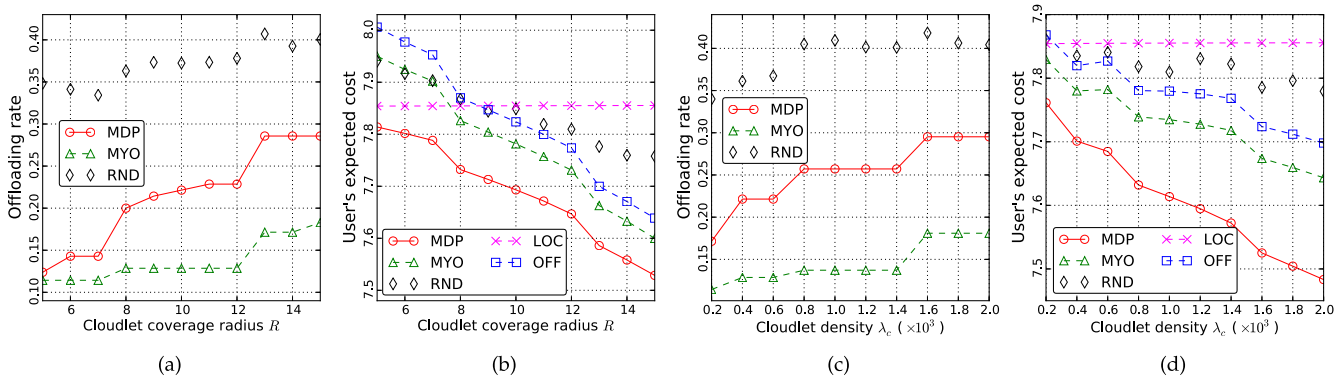

Fig. 8. (a) User's offloading rate to cloudlet radius R, (b) user's expected cost to R, (c) user's offloading rate to cloudlet density λ_c, and (d) user's expected cost to λ_c.

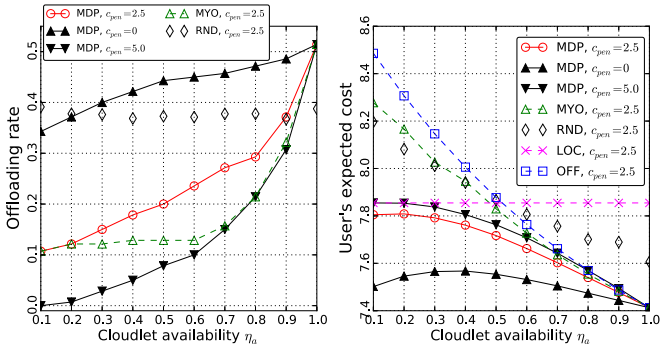

Fig. 9. (a) User's offloading rate to cloudlet availability η_a, and (b) user's expected cost to η_a.

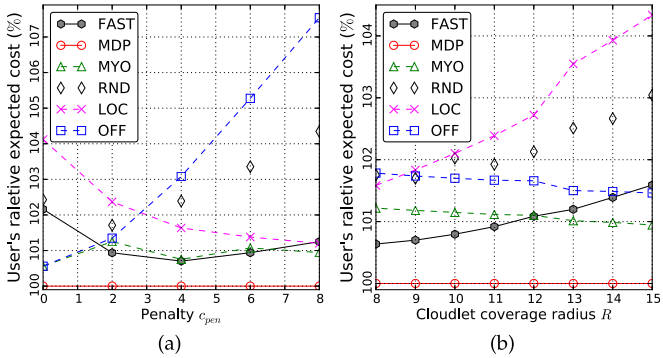

Fig. 11. User's relative expected cost (MDP cost is set as 100 percent): (a) with different cloudlet coverage radius R, and (b) with different offloading penalty coefficients.

$c_{pen} = 2.5$ and for $c_{pen} = 5.0$, the offloading rates are lower than that of $c_{pen} = 0$, as in Fig. 9a, since the possibility of offloading failure is high. As η_a increases, the penalty term has less effects to the decisions. Therefore, the immediate offloading cost $C_R(\mathcal{S})$ becomes low, as shown by the sharply increasing MYO curve in Fig. 9a when $\eta_a > 0.6$. The decreased effect of offloading failure penalty can also be observed from Fig. 9b, where we compare the MDP-based offloading algorithm with parameters $c_{pen} = 0$, 2.5 and 5.0. The higher penalty leads to higher user's expected cost. Furthermore, the cost gaps among MDP-based schemes with different penalties (solid lines in Fig. 9b) are smaller as η_a increases since the probability of offloading failure decreases. Additionally, Fig. 9b shows that, when η_a is low, the proposed MDP algorithm achieves the lower cost than those of myopic and always offloading schemes. When η_a is high (e.g., $\eta_a \geq 0.8$), the probability of successful offloading is high enough. As a result, the performances of MDP, OFF and MYO schemes become close to each other. Moreover, the costs for different values of c_{pen} tend to converge.

6.4 Threshold Policy

The threshold policy in the optimal solution of the proposed MDP algorithm is shown in Fig. 10. Given the application phase $\mathcal{G} = 1$ (in Fig. 10a) or $\mathcal{G} = 3$ (Fig. 10b) fixed, the two-dimensional threshold in $(\mathcal{Q}, \mathcal{N})$ can be observed, where the horizontal axes denote the queue state $\mathcal{Q}$ and the number of cloudlets state $\mathcal{N}$, respectively, and the vertical axis denotes the action (i.e., $\mathcal{A} = 0$ and $\mathcal{A} = 1$) taken by the mobile user. Given a certain $\mathcal{N}$ (or $\mathcal{Q}$), we increase $\mathcal{Q}$ (or $\mathcal{N}$) from 0 to $|Q|$ (or $|N|$), the action $\mathcal{A}$ taken by the user will monotonically change from 0 (i.e., local execution) to 1 (i.e., offloading). Consequently, a threshold policy in $(\mathcal{Q}, \mathcal{N})$ exists. For example, in Fig. 10a, the state $(\mathcal{Q} = 2, \mathcal{N} = 4)$ is a threshold in $\mathcal{N}$ such that $\mathcal{A} = 1$ when $\mathcal{N} \geq 4$, given the queue state $\mathcal{Q} = 2$.

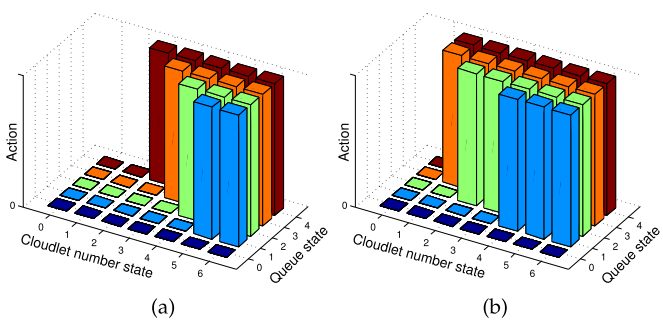

Fig. 10. A threshold policy in queue state $\mathcal{Q}$ and number of cloudlets state $\mathcal{N}$ when $R = 15$ m: (a) The application phase $\mathcal{G} = 1$, and (b) $\mathcal{G} = 3$.

Fig. 11 shows the suboptimality of the fast equal-division algorithm (labeled FAST) proposed in Section 4.4 in terms of user's expected cost. The optimal results obtained from MDP algorithm is set to be 1. Fig. 11 shows the relative expected cost compared with the MDP algorithm. In Fig. 11a, we vary the offloading penalty coefficient c_{pen} from 0 to 8. We also vary the cloudlet coverage radius R from 8 to 15 in Fig. 11b.

As shown in Fig. 11, when $R < 9m$ or $c_{pen} \geq 3$, the always offloading scheme (i.e., OFF) has the most unacceptable results (e.g., > 103 percent of optimal results when $c_{pen} \geq 4$). When $R \geq 9m$ or $c_{pen} < 3$, the always executing locally scheme (i.e., LOC) yields the most unsatisfactory results. Similarly, the random scheme (i.e., RND) could result in highly unsatisfactory cost if the offloading decisions in all or most of the states are made wrongly, e.g., the random result when $R = 10$ in Fig. 11a almost yields the worst cost. In summary, as shown in Fig. 11, compared with the always executing locally, always offloading and random schemes, the equal-division algorithm achieves acceptable performance in terms of the expected cost.

7 Conclusion

We have proposed a Markov decision process based dynamic offloading algorithm for a mobile user in a mobile cloudlet system. Consider that a mobile application can be divided into multiple phases, the user can make a decision to run each phase locally or offload to nearby cloudlets. Moreover, offloading failures caused by both user mobility and cloudlet admission control have been considered. The decision is optimized to achieve the lowest cost (e.g., computation and communication costs) by solving the MDP model using the value iteration algorithm. For the proposed MDP model, we have shown the existence of a threshold policy and also introduced a fast decision algorithm for the mobile user to make effective and acceptable performance. The numerical results have shown that the analytical estimations of cloudlet availability is relatively accurate for various user mobility. Furthermore, the proposed MDP-based dynamic offloading algorithm outperforms other baseline schemes, which do not take the state of the system into account.

ACKNOWLEDGMENTS

This work was supported by Singapore Ministry of Education (MOE) Tier 1 (RG18/13 and RG33/12). This paper is extended from the short version [1]. Dusit Niyato is the corresponding author.

REFERENCES

[1] Y. Zhang, D. Niyato, P. Wang, and C.-K. Tham, "Dynamic offloading algorithm in intermittently connected mobile cloudlet systems," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2014, pp. 4201–4206.
[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.
[3] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlet," in *Proc. IEEE 38th Conf. Local Comput. Netw.*, Oct. 2013, pp. 589–596.
[4] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.
[5] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Serv.: Soc. Netw. Beyond*, 2010, p. 6.
[6] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in *Proc. ACM/IFIP/USENIX 10th Int. Conf. Middleware*, 2009, pp. 83–102.
[7] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and V. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, Jun. 2010, pp. 49–62.
[8] B.-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Serv.: Soc. Netw. Beyond*, 2010, p. 7.
[9] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: A comparison of application models," CoRR, vol. abs/1107.4940, 2011.
[10] M. Satyanarayanan, "Mobile computing: The next decade," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Serv.: Soc. Netw. Beyond*, Jun. 2010, p. 5.
[11] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proc. Int. Conf. Compilers, Archit., Synthesis Embedded Syst.*, 2001, pp. 238–246.
[12] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy," *IEEE Comput.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[13] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–304.
[14] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.
[15] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE Conf. Comput. Commun.*, Mar. 2012, pp. 945–953.
[16] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, "Access schemes for mobile cloud computing," in *Proc. 11th Int. Conf. Mobile Data Manag.*, May 2010, pp. 387–392.
[17] J. Almeida, V. Almeida, D. Ardagna, Í. Cunha, C. Francalanci, and M. Trubian, "Joint admission control and resource allocation in virtualized servers," *J. Parallel Distrib. Comput.*, vol. 70, no. 4, pp. 344–362, Apr. 2010.
[18] D. T. Hoang, D. Niyato, and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2012, pp. 3145–3149.
[19] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: The challenge of intermittent connectivity," in *Proc. 1st Edition MCC Workshop Mobile Cloud Comput.*, 2012, pp. 23–28.
[20] Y.-C. Wang and K.-T. Cheng, "Energy-optimized mapping of application to smartphone platform? A case study of mobile face recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2011, pp. 84–89.
[21] D. Moltchanov, "Distance distributions in random networks," *Ad Hoc Netw.*, vol. 10, pp. 1146–1166, Aug. 2012.
[22] F. Baccelli, B. Blaszczyszyn, and P. Muhlethaler, "An Aloha protocol for multihop mobile wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 421–436, Feb. 2006.
[23] R. Bryant and D. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 2nd ed. Reading, MA, USA: Addison-Wesley, Feb. 2010, p. 632.
[24] T. P. Baker, "A stack-based resource allocation policy for realtime processes," in *Proc. 11th Real-Time Syst. Symp.*, Dec. 5–7, 1990, pp. 191–200.
[25] J. Vazifehdan, R. V. Prasad, M. Jacobsson, and I. Niemegeers, "An analytical energy consumption model for packet transfer over wireless links," *IEEE Commun. Lett.*, vol. 16, no. 1, pp. 30–33, Jan. 2012.
[26] M. H. Ngo and V. Krishnamurthy, "Optimality of threshold policies for transmission scheduling in correlated fading channels," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2474–2483, Aug. 2009.
[27] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, pp. 679–684, 1957.
[28] D. M. Topkis, *Supermodularity and Complementarity*. Princeton, NJ, USA: Princeton Univ. Press, 1998.
[29] K. Kar, A. Krishnamurthy, N. Jaggi, "Dynamic node activation in networks of rechargeable sensors," *IEEE/ACM Trans. Netw.*, vol. 14, no. 1, pp. 15–26, Feb. 2006.
[30] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. 11th Conf. Uncertainty Artif. Intell.*, 1995, pp. 394–402.
[31] A. Hasan and J. G. Andrews, "The guard zone in wireless ad hoc networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 3, pp. 897–906, Mar. 2007.
[32] M. Kaynia and N. Jindal, "Performance of ALOHA and CSMA in spatially distributed wireless networks," in *Proc. IEEE Int. Conf. Commun.*, May 19–23, 2008, pp. 1108–1112.

Yang Zhang received the BEng and MEng degrees from Beihang University (BUAA), Beijing, China, in 2008 and 2011, respectively. He is a research associate and a PhD candidate in the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include optimization approaches for resource allocation in wireless and mobile cloud computing systems.

Dusit Niyato (M'09-SM'15) received the PhD degree in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2008. He is currently an associate professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include optimization of wireless communication and mobile cloud computing, smart grid systems, and green radio communications. He is a senior member of the IEEE.

Ping Wang (M'08-SM'15) received the PhD degree in electrical engineering from the University of Waterloo, Canada, in 2008. She is currently an assistant professor in the School of Computer Engineering, Nanyang Technological University, Singapore. Her current research interests include resource allocation in multimedia wireless networks, cloud computing, and smart grid. She coreceived the Best Paper Award from IEEE Wireless Communications and Networking Conference (WCNC) 2012 and IEEE International Conference on Communications (ICC) 2007. She is an editor of *IEEE Transactions on Wireless Communications*, *EURASIP Journal on Wireless Communications and Networking*, and *International Journal of Ultra Wideband Communications and Systems*. She is a senior member of the IEEE.