Hindawi Publishing Corporation
International Journal of Antennas and Propagation
Volume 2015, Article ID 106053, 12 pages
http://dx.doi.org/10.1155/2015/106053

Research Article

A Novel Execution Mode Selection Scheme for Wireless Computing

Jiadi Chen and Wenbo Wang

Wireless Signal Processing and Network Lab, Key Laboratory of Universal Wireless Communication, Ministry of Education, Beijing University of Posts & Telecommunications, Beijing 100876, China

Correspondence should be addressed to Jiadi Chen; chenjd@bupt.edu.cn

Received 11 September 2014; Accepted 25 December 2014

Academic Editor: Feifei Gao

Computation offloading is an effective way to alleviate the resource limited problem of mobile devices. However, the offloading is not an always advantageous strategy for under some circumstances the overhead in time and energy may turn out to be greater than the offloading savings. Therefore, an offloading decision scheme is in demand for mobile devices to decide whether to offload a computation task to the server or to execute it in a local processor. In this paper, the offloading decision problem is translated into a dynamic execution mode selection problem, the objective of which is to minimize the task execution delay and reduce the energy consumption of mobile devices. A novel execution mode adjustment mechanism is introduced to make the execution process more flexible for real-time environment variation. Numerical results indicate that the proposed scheme can significantly reduce the task execution delay in an energy-efficient way.

1. Introduction

Nowadays, advancements in computing technology have made mobile devices more and more smart and powerful, for example, smart phones, handheld computers, ambient sensors, and autonomous robots. The increasing processing and storage capacity have made it possible for them to run complex applications. However, mobile devices are inherently resource limited and energy constrained [1], for the more complex and energy-intensive programs are emerging in an explosive speed [2].

Advanced computing architectures, for example, wireless/mobile computing [3] and cloud computing [4], can provide attractive solutions to alleviate the resource limitation problem. These solutions all introduce a new strategy, that is, the task offloading, also known as task migration, where the mobile terminals (MTs) can make use of the ample computing resources of the wireline domain, sending their computation tasks to remote computation servers and receiving the computation results afterwards.

Is the offloading really beneficial? From the perspective of saving time, the relatively faster server processor can accelerate the execution speed, however, overheads are introduced by wireless communication between MTs and the server. On the other hand, it is hard to conclude whether the offloading can reduce the MT's energy consumption or not. Most energy is consumed by the MT's CPU when the task is executed locally. If offloaded, this part of energy can be saved but the MT's transceiver will consume additional energy to accomplish the uplink and downlink data transmission.

Therefore, the offloading decision problem needs to be considered from many aspects, for example, task properties, wireless bandwidth, and MT's CPU capacity. As [5] concludes, offloading is beneficial when large amounts of computation are needed with relatively small amounts of communication. Except typical applications which are suitable for offloading (chess games, face recognition applications, etc.), there are many computation tasks that are hard to be classified as "tasks suitable for offloading" or "tasks suitable for local processing," for example, the online file editing or the relatively simple image processing, the local computation volume, and offloading communication volume which are comparative and the wireless bandwidth plays an important role in the offloading decision process. A weak wireless link can slow down the communication and raise power consumption at the transceiver; the overhead in time and energy may turn out to be greater than the offloading savings.

In this paper, we consider the computing architecture which has been introduced in [6]. The MT can execute the computation task in one of the following three modes: (1) locally: executing the task at the MT's local processor, (2) remotely: offloading the task to a remote server by wireless transmission, and (3) combinedly: executing the task in the former two ways simultaneously. Before task execution, the MT has to decide which mode to adopt. Therefore, the offloading decision problem is equivalent to an execution mode selection problem.

Based on the previous description, an adaptive execution mode selection scheme (AEMSS) is proposed to help MTs make offloading decisions. Main contributions of this paper are described as follows.

(i) For the existing offloading schemes, decisions are made in advance and cannot be changed once the execution has begun. The proposed AEMSS allows the task to change execution mode during processing, which can make the task execution more flexible for the changing wireless condition.

(ii) The execution mode selection problem is formulated as a finite-horizon Markov decision process (MDP) and the user's tolerance for execution delay is subtly formulated as the last decision epoch of the model. A heavy penalty factor for timeout can guarantee the delay performance of the proposed scheme.

(iii) The MDP is solved by the backward induction algorithm (BIA) and numerical results indicate that the proposed AEMSS can achieve a good performance on execution delay and energy efficiency.

The remainder of this paper is organized as follows. The related work is summarized in Section 2. System model and the problem formulation of AEMSS are presented in Section 3. The detailed finite-horizon MDP model and the solution are described in Section 4. In Section 5, some implementation issues are discussed. Simulation assumptions and the numerical results are presented in Section 6. Finally the conclusions are drawn in Section 7.

2. Related Work

In this section, we provide a discussion on the existing algorithms for offloading decision, which can be partitioned into two categories, that is, the static decisions and the dynamic decisions [7].

Static decision is that the program is partitioned during development—which part to execute locally and which part to offload. Algorithms to make static offloading decisions mostly appear in the work of earlier years [8–10]. In [8], an algorithm is proposed to divide the program into server tasks and client tasks such that the energy consumed at the client is minimized. Reference [9] presents a task partition and allocation scheme to divide the distributed multimedia processing between the server and a handheld device. Authors in [10] introduce a strategy that executes the program initially on the mobile system with a timeout. If the computation is not completed after the timeout, it is offloaded. These static offloading decisions have the advantage of low overhead during execution. However, this kind of approach is valid only when the parameters, for example, wireless transmission capacity, can be accurately predicted in advance.

In contrast, dynamic decisions can adapt to various runtime conditions, for example, fluctuating network bandwidths and varying server load. Prediction mechanisms are usually used in dynamic approaches for decision making. For example, the bandwidth is monitored and predicted using a Bayesian scheme in [11]. Offloading decision algorithms in recent work are most dynamic ones [12–16]. In [12], a theoretical framework of energy-optimal mobile cloud computing under stochastic wireless channel is provided. In [13], a study on the feasibility of applying machine learning techniques is presented to address the adaptive scheduling problem in mobile offloading framework. Reference [14] presents a collaborative WiFi-based mobile data offloading architecture targeted at improving the energy efficiency for smartphones. Reference [15] proposes a fine grained application model and a fast optimal offloading decision algorithm where multiple offloading decisions are made per module based on the execution paths leading to the module. Authors in [16] also formulate the offloading decision problem based on MDP and solve it by using a linear programming approach. They address the problem of extending the lifetime of a battery powered mobile host in a client-server wireless network by using task migration and remote processing.

The work in this paper is inspired by [6], where a task migration jointly with the terminal power management mechanism is formulated in the framework of dynamic programming. The solution is a policy specifying when the terminal should initiate task migration versus executing the task locally, in conjunction with the power management. Although there have been various offloading decision algorithms proposed in the literature, our work has some original and advanced characters. We put the focus on the offloading decision making phase, with the optimization objective of improving the task execution efficiency and reducing the energy consumption of mobile devices by determining the "best fit" execution mode at each decision epoch.

3. Problem Formulation

3.1. Computing Architecture. In this paper, we consider a computing architecture, where the MT can execute its computation task in one of the following three modes:

(1) *locally*: executing the task at the MT's processor;

(2) *remotely*: offloading the task to a remote computation server via the wireless network;

(3) *combinedly*: executing the task with the former two options simultaneously.

Modes 2 and 3 involve the practice of offloading, which is modeled into a three-step process as follows.

(i) *Data Uploading*. MT sends the task specification and input data to the remote computation server.

(ii) *Server Computing*. The remote computation server performs the task.

(iii) *Results Downloading*. The MT downloads the computation results from the remote computation server.

3.2. Execution Completion Time. Let C denote the local computation volume of the task. r_l is the speed of local processor, which is a constant value determined by the MT's CPU capacity. The task execution completion time under the local execution mode can be expressed as

$$t_l = \frac{C}{r_l}. \tag{1}$$

In the remote execution mode, let D_u and D_d denote the data volume for uplink and downlink transmission and the corresponding average transmission capacities of the wireless system are r_u and r_d, respectively. r_u and r_d are determined by multiple parameters, for example, the wireless bandwidth, transmission power, and average channel fading coefficient. For simplicity, the remote server is assumed to have a perfect process capacity so that the computation delay can be ignored. Thus the execution completion time under the remote execution mode can be expressed as the sum of the uplink transmission delay t_u and the downlink transmission delay t_d; that is,

$$t_r = t_u + t_d = \frac{D_u}{r_u} + \frac{D_d}{r_d}. \tag{2}$$

When the task is executed combinedly, the local processor and the remote server work simultaneously. The execution process ends when *either* of them finishes the task. So the execution completion time can be expressed as

$$t_c = \min\{t_l, t_r\} = \min\left\{\frac{C}{r_l}, \frac{D_u}{r_u} + \frac{D_d}{r_d}\right\}. \tag{3}$$

3.3. Energy Consumption of MT. Energy is primary constraint for mobile devices; therefore, we only consider the energy consumption of the MT. In the local execution mode, p_l is the power of the local processor; the energy consumption of the MT is

$$e_l = p_l \cdot t_l. \tag{4}$$

In the remote execution mode, the power of transmitting antenna and receiving antenna is p_u and p_d, respectively. The energy consumption of the MT is

$$e_r = e_u + e_d = p_u \cdot t_u + p_d \cdot t_d, \tag{5}$$

where e_u and e_d are power consumption of the data uploading and results downloading phases, respectively. When executed combinedly, the energy consumption of the MT can be expressed as

$$e_c = \begin{cases} (p_l + p_u)\, t_u + (p_l + p_d)\,(t_c - t_u), & t_c > t_u, \\ (p_l + p_u)\, t_c, & t_c \le t_u. \end{cases} \tag{6}$$

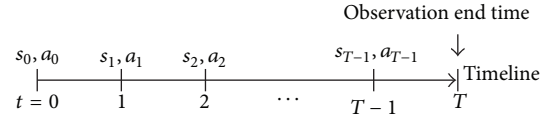


Figure 1: Timing in MDP.

3.4. Adaptive Execution Mode Selection. Under the previous assumptions, the offloading decision problem can be translated into an execution mode selection problem. Based on the optimization object, that is, minimizing task execution delay as well as reducing MTs' energy consumption, an adaptive execution mode selection scheme (AEMSS) is proposed. Functions of the offloading decision maker include

(i) determining the most appropriate execution mode for a specific computing task, that is, local execution, remote execution, or combined execution;

(ii) determining if the execution mode needs to be adjusted during the execution process when the unstable factors, for example, the wireless environment condition, have dramatically changed.

The AEMSS makes decisions based on a series of parameters, including the task properties (C, D_u, D_d), MT's CPU capacity, and the wireless transmission capacity (r_l, r_u, r_d). The problem is formulated into a finite-horizon Markov decision process (MDP) model and solved by the backward induction algorithm (BIA). Details are elaborated in the next section.

4. The Finite-Horizon MDP-Based AEMSS

In general, an MDP model consists of five elements, that is, (1) decision epochs; (2) states; (3) actions; (4) transition probabilities; and (5) rewards. In this section we will describe how the offloading decision problem can be formulated into a finite-horizon MDP model from these five aspects.

4.1. Decision Epochs and State Space. For simplicity, the time is slotted into discrete decision epochs and indexed by $t \in \{0, 1, 2, \ldots, T\}$. As Figure 1 shows, time point $t = T$ denotes the *observation end time* in the MDP model, not the moment the task is just being completed. Time T has another sense, that is, the longest task completion time that the user can sustain. The task is considered failed if it is still uncompleted upon time T.

At decision epoch t, the system state s_t reflects the current task completion progress and the real-time wireless condition, which can be expressed as a tuple

$$s_t = \left(\phi_t, C'_t, D'_{u,t}, D'_{d,t}, r_t\right), \tag{7}$$

the elements in which are explained below:

(i) ϕ_t: the execution mode adopted in the last time slot $[t-1, t)$; that is,

$$\phi_t = \begin{cases} 1, & \text{local execution}, \\ 2, & \text{remote execution}, \\ 3, & \text{combined execution}, \end{cases} \tag{8}$$

Table 1: Subspaces of the state space.

$\mathcal{S}_1$[1]	$\{s = (\phi, C', D'_u, D'_d, r) \mid \phi = 1, C' \in \{1, 2, \ldots, C-1\}, D'_u = D'_d = 0, r \in \mathcal{R}\}$
$\mathcal{S}_2$[2]	$\{s = (\phi, C', D'_u, D'_d, r) \mid \phi = 2, C' = 0, D'_u \in \{1, 2, \ldots, D_u - 1\}, D'_d = D_d, r \in \mathcal{R}\}$ $\cup \{s = (\phi, C', D'_u, D'_d, r) \mid \phi = 2, C' = 0, D'_u = 0, D'_d \in \{1, 2, \ldots, D_d - 1\}, r \in \mathcal{R}\}$
$\mathcal{S}_3$[3]	$\{s = (\phi, C', D'_\mu, D'_d, r) \mid \phi = 3, C' \in \{1, 2, \ldots, C-1\}, D'_u \in \{1, 2, \ldots, D_u - 1\}, D'_d = D_d, r \in \mathcal{R}\}$ $\cup \{s = (\phi, C', D'_u, D'_d, r) \mid \phi = 3, C' \in \{1, 2, \ldots, C-1\}, D'_u = 0, D'_d \in \{1, 2, \ldots, D_d\}, r \in \mathcal{R}\}$
$\mathcal{S}_{\text{initial}}$[4]	$\{s_{\text{initial}} = (0, C, D_u, D_d, r) \mid r \in \mathcal{R}\}$
$\mathcal{S}_{\text{terminal}}$[5]	$\{s_{\text{terminal}} = (4, 0, 0, 0, \not{r})\}$

[1] $\mathcal{S}_1$ is the set of system states indicating that the task is in the local execution process; that is, $\phi = 1$. $\mathcal{R} = \{r_{\min}, r_{\min} + 1, \ldots, r_{\max}\}$, where $r_{\min}$ and $r_{\max}$ denote the minimum and maximum transmission capacities the wireless network can provide, respectively.
[2] $\mathcal{S}_2$ is the set of system states indicating that the task is in the remote execution process; that is, $\phi = 2$. $\mathcal{S}_2$ can be seen as the union of two state sets; that is, the task is in the uplink transmission process ($D'_d = D_d$) and the task is in the downlink receiving process ($D'_u = D_u$), respectively.
[3] $\mathcal{S}_3$ is the set of system states indicating that the task is in the combined executing process; that is, $\phi = 3$. $\mathcal{S}_3$ can be seen as the union of two state sets; that is, the task is in the uplink transmission process ($D'_d = D_d$) and the task is in the downlink receiving process ($D'_u = D_u$), respectively. Meanwhile, the task is also under local processing.
[4] $\mathcal{S}_4$ is the set of initial states, which are distinguished by different wireless conditions, that is, different values of r.
[5] $\mathcal{S}_5$ contains a single state, that is, s_{terminal}, indicating that the task execution process is already finished. $\not{r}$ means that when the task has been completed, the wireless transmission capacity can be disregarded.

(ii) C'_t: the remaining computation volume for local processing by time t;

(iii) $D'_{u,t}$: the remaining data volume for uplink transmission by decision epoch t; if $\phi_t = 1$ or the uplink transmission has already finished, $D'_{u,t} = 0$;

(iv) $D'_{d,t}$: the remaining data volume for downlink transmission by decision epoch t; if $\phi_t = 1$, $D'_{d,t} = 0$;

(v) r_t: the transmission capacity the wireless network can provide at decision epoch t, which is assumed to be static within a time slot and i.i.d. between slots.

In addition, there are two kinds of special states in the state space, that is, the initial states $s_{\text{initial}} \in S_{\text{initial}}$ and a terminal state s_{terminal}. The initial states are specific at decision epoch $t = 0$ and indicate that the task is untreated, while the terminal state indicates that the task execution process has already been completed. Therefore, the state space $\mathcal{S}$ can be expressed as

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_{\text{initial}} \cup \mathcal{S}_{\text{terminal}}, \tag{9}$$

where $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_{\text{initial}}$, and $\mathcal{S}_{\text{terminal}}$ are subspaces of $\mathcal{S}$, the definitions of which are listed in Table 1.

4.2. Action Space and Policy. In this model, there are four actions in the action space $\mathcal{A}$; that is,

$$\mathcal{A} = \{0, 1, 2, 3\}. \tag{10}$$

At decision epoch t, AEMSS chooses an action based on the current state s_t. Different actions represent the different execution modes the task will adopt in the following time slot; that is,

$$a_t = \begin{cases} 0, & \text{null}, \\ 1, & \text{local execution}, \\ 2, & \text{remote execution}, \\ 3, & \text{combined execution}, \end{cases} \tag{11}$$

where $a_t = 0$ indicates that the task has already been completed and nothing needs to be done in the following time slot.

At decision epoch t, the decision maker chooses an action from the feasible action set $\mathcal{A}_t$ according to the decision rule $d_t(s_t) = a_t$. In MDP, a policy $\pi = (d_0, d_1, \ldots, d_T)$ specifies the decision rule to be used at all decision epochs. It provides the decision maker with a prescription for action selection under any possible future state [17]. In this model, the decision rules at all T decision epochs are different; for example, when the task has already been completed, $a_t = 0$ is the only available action. Therefore, the policy obtained is a "nonstationary policy." In the following parts we will show how the actions can transform the system states.

At time t, the system state is $s_t = (\phi_t, C'_t, D'_{u,t}, D'_{d,t}, r_t)$. After action a_t is taken the system will transform to the state $s_{t+1} = (\phi_{t+1}, C'_{t+1}, D'_{u,t+1}, D'_{d,t+1}, r_{t+1})$ by time $t+1$. The feasible cases of states transformation are listed in Table 2. The cases can be split into two categories, that is, $a_t = \phi_t$ (cases 1–3, the current execution mode will continue to be adopted) and $a_t \neq \phi_t$ (cases 4–9, the current execution mode will be adjusted).

In Table 2, cases 4–9 indicate that the execution mode is changed within two successive time slots as follows.

(i) In cases 4 and 5, the task is being executed locally/remotely at time t when the decision maker decides to change the execution mode to the remote/local one. In these cases the execution progress before time t will be cleared and the task will be forced to be executed from scratch with a different execution mode.

(ii) In cases 6 and 7, when the task is being executed locally or remotely, the decision maker wants it to be executed with both modes simultaneously in the next time slot. In these cases the current execution progress will be preserved and a new execution process with another execution mode will begin in the next slot.

Table 2: Actions and states transformation.

Number	Cases	Illustration
1	$\phi_t = 1, a_t = 1$	$s_t = (1, C'_t, 0, 0, r_t), a_t = 1 \Rightarrow s_{t+1} = (1, C'_t - r_l, 0, 0, r_{t+1})$
2	$\phi_t = 2, a_t = 2$	$s_t = (2, 0, D'_{u,t}, D_d, r_t), a_t = 2 \Rightarrow s_{t+1} = (2, 0, D'_{u,t} - r_t, D_d, r_{t+1})$[1]
3	$\phi_t = 3, a_t = 3$	$s_t = (3, C'_t, D'_{u,t}, D_d, r_t), a_t = 3 \Rightarrow s_{t+1} = (3, D'_{u,t} - r_t, D_d, C'_t - r_l, r_{t+1})$[2]
4	$\phi_t = 1, a_t = 2$	$s_t = (1, C'_t, 0, 0, r_t), a_t = 2 \Rightarrow s_{t+1} = (2, 0, D_u - r_t, D_d, r_{t+1})$
5	$\phi_t = 2, a_t = 1$	$s_t = (2, 0, D'_{u,t}, D_d, r_t), a_t = 1 \Rightarrow s_{t+1} = (1, C - r_l, 0, 0, r_{t+1})$
6	$\phi_t = 1, a_t = 3$	$s_t = (1, C'_t, 0, 0, r_t), a_k = 3 \Rightarrow s_{t+1} = (3, C'_t - r_l, D_u - r_t, D_d, r_{t+1})$
7	$\phi_t = 2, a_t = 3$	$s_t = (2, 0, D'_{u,t}, D_d, r_t), a_t = 3 \Rightarrow s_{t+1} = (3, C - r_l, D'_{u,t} - r_t, D_d, r_{t+1})$
8	$\phi_t = 3, a_t = 1$	$s_t = (3, C'_t, D'_{u,t}, D_d, r_t), a_t = 1 \Rightarrow s_{t+1} = (1, C'_t - r_l, 0, 0, r_{t+1})$
9	$\phi_t = 3, a_t = 2$	$s_t = (3, C'_t, D'_{u,t}, D_d, r_t), a_t = 2 \Rightarrow s_{t+1} = (2, 0, D'_{u,t} - r_t, D_d, r_{t+1})$

[1,2] C, D_u, D_d denote the task properties, that is, the total computation volume for local processing and the total data volume for wireless transmission while $C'_t, D'_{u,t}, D'_{d,t}$ are real-time values at decision epoch t, that is, the remaining computation volume after a period of local processing and remaining data volume after a period of transmission. Therefore, $s_t = (2, 0, D'_{u,t}, D_d, r_k)$ and $s_k = (2, 0, 0, D'_{d,t}, r_k)$ denote that the task is in uplink transmission process and in downlink transmission process, respectively. In this table, cases which involve the remote executing are all in the uplink transmission process.

(iii) In cases 8 and 9, the task is being executed with local mode and remote mode simultaneously, but the decision maker judges that one of them is unnecessary. In the next time slot the execution progress of this mode will be cleared and another one will continue.

4.3. State Transition Probabilities. From Table 2 we can conclude that when a specific action a_t is selected, the system state of the next decision epoch can be determined except the element of wireless transmission capacity r_{t+1}. Therefore the state transition probability between two successive decision epochs can be written as

$$p\left(s_{t+1} \mid s_t, a_t\right) = p\left(r_{t+1} \mid r_t\right). \tag{12}$$

The long-term probability distribution of the wireless transmission capacities is denoted as $p^{\infty}(r_{\min}), p^{\infty}(r_{\min}+1), \ldots, p^{\infty}(r_{\max})$; thus, the steady state probability distributions of each task state at each decision epoch are

$$\begin{aligned} p_0^{\infty}\left(s_{\text{initial}} = (0, C, D_u, D_d, r_0)\right) &= p^{\infty}\left(r_0\right), \\ r_0 &\in \{r_{\min}, r_{\min}+1, \ldots, r_{\max}\}, \\ p_{t+1}^{\infty}\left(s'\right) &= \sum_{k=0}^{t} \sum_{\substack{s,s' \in S/S_{\text{initial}} \\ a \in A_t}} p_k^{\infty}(s)\, p\left(s' \mid s, a\right). \end{aligned} \tag{13}$$

$p_t^{\infty}(s)$ denotes the steady state probability of state $s \in \mathcal{S}_t$ at decision epoch $t \in \{0, 1, \ldots, T-1\}$. $\mathcal{S}_t$ denotes the set containing all feasible states at time epoch t, that is, states with a steady state probability of $p_t^{\infty}(s) \neq 0$.

4.4. Rewards and Value Function. In this model, the system reward function within time interval $t \sim t+1, t < T$, is defined as

$$\begin{aligned} r_t(s, a) &= \sum_{\substack{s \in S_t, s' \in S_{t+1} \\ a \in A_t}} p\left(s' \mid s, a\right) r_t\left(s' \mid s, a\right), \\ r_t\left(s' \mid s, a\right) &= \omega_d f_{d,t}\left(s' \mid s, a\right) - \omega_e f_{e,t}\left(s' \mid s, a\right) - \sigma_t. \end{aligned} \tag{14}$$

$f_{d,k}(s' \mid s, a)$ is the delay reward function and $f_{e,k}(s' \mid s, a)$ is the energy consumption function. ω_d, ω_e are weight factors satisfying $\omega_d + \omega_e = 1$. σ_k is the penalty factor for exceeding the execution delay limit, which is defined as

$$\sigma_t = \begin{cases} 0, & t \neq T, \\ 0, & t = T,\ s_t = s_{\text{terminal}}, \\ \sigma, & t = T,\ s_t \neq s_{\text{terminal}}. \end{cases} \tag{15}$$

It can be seen that the task will be regarded as a failure if it is still uncompleted at the final decision epoch $t = T$. The delay reward function is given by

$$f_{d,t}\left(s' \mid s, a\right) = \begin{cases} \rho_{t+1} - \rho_t, & \phi_t \neq 4, \\ \rho^*, & \phi_t = 4, \end{cases} \tag{16}$$

where $s \in \mathcal{S}_t, s' \in \mathcal{S}_{t+1}, a \in \mathcal{A}_t$, and ρ_t is the task completion factor at decision epoch t, the definition of which is

$$\rho_t = \begin{cases} 1 - \dfrac{C'_t}{C}, & \phi_t = 1, \\ 1 - \dfrac{D'_{u,t} + D'_{d,t}}{D_u + D_d}, & \phi_t = 2, \\ 1 - \min\left\{\dfrac{D'_{u,t} + D'_{d,t}}{D_u + D_d}, \dfrac{C'_t}{C}\right\}, & \phi_t = 3. \end{cases} \tag{17}$$

It can be seen that ρ_t is the percentage completion of the task. ρ^* is the extra reward the system gains each time slot after the task is completed. The penalty factor σ_t and extra reward ρ^* have the same function, that is, promoting the task to be completed as early as possible.

At decision epoch t, the energy consumption function is given by

$$f_{e,t}\left(s_{t+1} \mid s_t, a_t\right) = \begin{cases} p_l, & a_t = 1, \\ p_u, & a_t = 2,\ D'_{u,t} \neq 0, \\ p_d, & a_t = 2,\ D'_{u,t} = 0, \\ p_l + p_u, & a_t = 3,\ D'_{u,t} \neq 0, \\ p_l + p_d, & a_t = 3,\ D'_{u,t} = 0, \\ 0 & \phi_t = 4. \end{cases} \tag{18}$$

It can be seen that, under the combined execution mode, the task can be accomplished fastest with the price of the highest energy consumption. During the whole time domain from $t = 0 \sim T$, the expected total reward, that is, the value function, can be expressed as

$$v^{\pi}(s) = E_s^{\pi}\left\{\sum_{t=0}^{T-1} r_t\left(s_t, a_t\right)\right\}, \quad s \in \mathcal{S}_{\text{initial}},\ s_t \in \mathcal{S}_t,\ a_t \in \mathcal{A}_t, \tag{19}$$

where $E_s^{\pi}\{*\}$ denotes the expectation value of $*$ under policy $\pi = (d_0, d_1, \ldots, d_{T-1})$ with the initial state s. The optimization objective is to find an optimal policy $\pi^* = (d_0^*, d_1^*, \ldots, d_{T-1}^*)$ which satisfies

$$v^{\pi^*}(s) \geq v^{\pi}(s) \tag{20}$$

for all initial states $s \in \mathcal{S}_{\text{initial}}$ and all $\pi \in \Pi$. Π is the set containing all feasible policies.

4.5. Solution for Finite-Horizon MDP. For an infinite-horizon MDP, there are various mature algorithms available for reference, for example, value iteration, policy iteration, and action elimination algorithms [17]. In this part, the solution for the proposed finite-horizon MDP model is discussed.

On time domain $0 \sim t$, sequence $h_t = (s_0, a_0, s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t)$ is called a "history" of the system and $h_{t+1} = (h_t, a_t, s_{t+1})$. Let $u_t^{\pi}(h_t)$ for $t < T$ denote the total expected reward obtained by using policy π at decision epochs $t, t+1, \ldots, T-1$ with a history of h_t; that is,

$$u_t^{\pi}\left(h_t\right) = E_{h_t}^{\pi}\left\{\sum_{k=t}^{T-1} r_k\left(s_k, a_k\right)\right\}, \quad s_k \in \mathcal{S}_k,\ a_k \in \mathcal{A}_k. \tag{21}$$

When $h_1 = s$, $u_0^{\pi}(s) = v^{\pi}(s)$. From [17], the optimal equations are given by

$$u_t^*\left(h_t\right) = \max_{a \in A_t}\left\{r_t(s,a) + \sum_{s' \in S_{t+1}} p\left(s' \mid s, a\right) u_{t+1}^*\left(h_t, a, s'\right)\right\} \tag{22}$$

for $t = 0, 1, \ldots, T-1$ and $h_t = (h_{t-1}, a_{t-1}, s)$, $s \in \mathcal{S}_t$.

From above we can see that $u_0^*(h_T)$ corresponds to the maximum total expected reward $v^{\pi^*}(s)$. The solutions satisfying the optimal equations are the actions $a_{s,t}^*$, $s \in \mathcal{S}_t$, and $t \in \{0, 1, \ldots, T-1\}$ which can achieve the maximum total expected reward; that is,

$$a_{s,t}^* = \arg\max_{a \in A_t}\left\{r_t(s,a) + \sum_{s' \in S_{t+1}} p\left(s' \mid s, a\right) u_{t+1}^*\left(h_t, a_t, s'\right)\right\}. \tag{23}$$

In this paper, the BIA [17] is employed to solve the optimization problem given by (23) as follows.

The Backward Induction Algorithm (BIA)

(1) Set $t = T - 1$ and

$$u_{T-1}^*(s) = \max_{a \in A_{T-1}}\left\{r_{T-1}(s,a)\right\}$$
$$a_{s,T-1}^* = \arg\max_{a \in A_{T-1}}\left\{r_{T-1}(s,a)\right\} \tag{24}$$

for all $s \in \mathcal{S}_{T-1}$.

(2) Substitute $t-1$ for t and compute $u_t^*(s)$ for each $s \in S_t$ by

$$u_t^*(s) = \max_{a \in A_{S_t}}\left\{r_t\left(s_t, a_t\right) + \sum_{s_{t+1} \in S_{t+1}} p\left(s_{t+1} \mid s_t, a_t\right) u_{t+1}^*\left(s_{t+1}\right)\right\}. \tag{25}$$

Set

$$a_{s,t}^* = \arg\max_{a \in A_{S_t}}\left\{r_t\left(s_t, a_t\right) + \sum_{s_{t+1} \in S_{t+1}} p\left(s_{t+1} \mid s_t, a_t\right) u_{t+1}^*\left(s_{t+1}\right)\right\}. \tag{26}$$

(3) If $t = 0$, stop. Otherwise return to step 2.

5. Model Implementation Issues

In this section, discussions are made on some issues occurring when the proposed MDP-based AEMSS is implemented to a real system.

5.1. The Offloading Decision Process. Figure 2 illustrates the workflow of the AEMSS, which can be partitioned into two parts, that is, the offline policy making and the online action mapping.

(i) *Offline Policy Making*. The execution mode selection policy is calculated offline via BIA, the input data of which includes task properties, wireless network characteristics, and the MT's capacities. Before execution, the computation task needs to be profiled first to determine the description parameters, that is, C, D_u, D_d, and the time threshold T. Then the parameters will be submitted to the server for policy determination. The server will also evaluate the wireless channel condition to determine the transition probability matrix of the wireless capacities. In this model, the optimal policy achieved, that is, the output of BIA, is a nonstationary policy. Thus for each decision epoch t, there will be a matrix reflecting the correspondence between states and actions and all the T matrixes can form a T-dimension state-to-action mapping table.

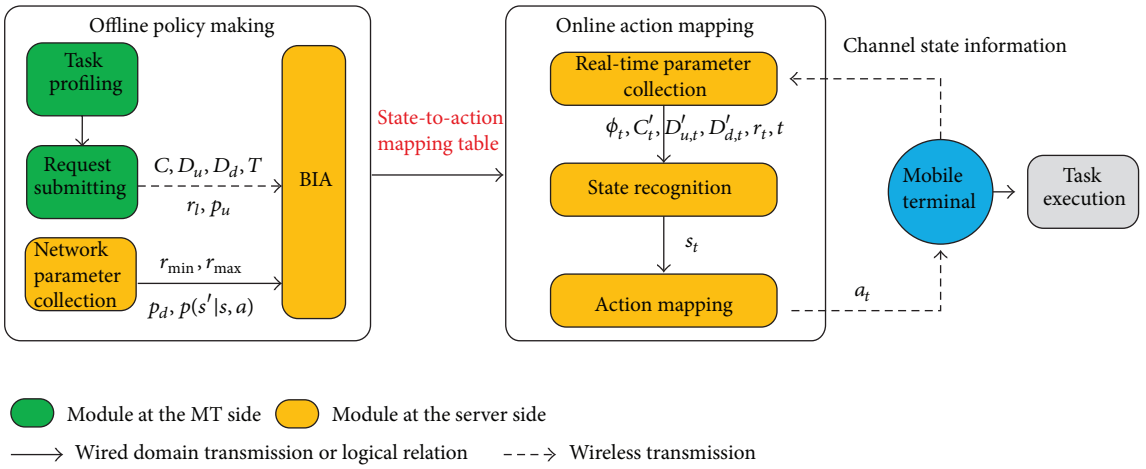


Figure 2: Workflow of the AEMSS.

Table 3: Parameters in the simulation for state transition probabilities.

Parameters	Values
Intercell distance	500 m
Bandwidth	5 MHz
Scheduling algorithm	Round Robin
Transmitter power of base station	43 dBm
Maximum transmitter power of MT	21 dBm
Poisson intensity for service requests	5

Table 4: Offloading decision policies adopted in the performance evaluation.

Policies	Classification	Description
π^*	Dynamic	The MDP-based AEMSS
π^{AL}	Static	Always local
π^{AO}	Static	Always offloading
π^{AC}	Static	Always combined
π^{DY}	Dynamic	Offloading if network capacity reaches the average level; otherwise it executes locally

(ii) *Online Action Mapping*. By employing BIA, the optimal execution mode selection policy can be obtained and stored in advance. At decision epoch t after the execution begins, the MT evaluates the channel state information by measuring the fading level of the reference signal and reports it to the parameter collection module, which monitors the task execution progress and collects necessary parameter values such as $\phi_t, C'_t, D'_{u,t}, D'_{d,t}$. All the real-time parameter values are mapped into a certain system state s_t. Then by looking up the state-to-action mapping table, the optimal action a_t can be chosen and the task execution mode in the following time slot is determined.

5.2. Measurement of the State Transition Probabilities. The changing process of a wireless network is complicated and difficult to predict. There are many factors that can influence the wireless transmission capacity, for example, bandwidth, user number, and resource allocation scheme. In this paper, a system simulation for a typical 3GPP network is conducted to estimate the state transition probabilities, where users arrive and depart from the network according to a Poisson process. Main parameters in simulation are listed in Table 3. The proposed scheme is adaptive and applicable to a wide range of conditions. Different wireless networks have different state transform situations, which can be obtained by changing the parameter set in the simulation.

6. Numerical Results and Analysis

In this section, the performance of the proposed MDP-based AEMSS is evaluated with four other offloading schemes, including three static schemes and a dynamic one. The policy achieved by solving the finite-horizon MDP is denoted as π^*; other policies for comparison are referred to as $\pi^{\mathrm{AL}}, \pi^{\mathrm{AO}}, \pi^{\mathrm{AC}}$, and π^{DY}, the descriptions of which are listed in Table 4.

Firstly, a qualitative analysis on the optimal policy π^* is provided to reflect the relationship between the chosen execution modes and the task characteristics. A series of computation tasks with different characteristics (C, D_u, and D_d) are analyzed. The probability of choosing a certain action is defined as

$$p(a) = \frac{1}{T}\sum_{t=0}^{T-1}\sum_{s\in\mathcal{S}_t} p^{\infty}(s)\cdot I\left[d_t^*(s)=a\right], \quad a=1,2,3, \tag{27}$$

where $d_t^*(s)$ is the action decision rule at decision epoch t. The definition of operator $I[*]$ is

$$I[*] = \begin{cases} 1, & * \text{ is true} \\ 0, & * \text{ is false.} \end{cases} \tag{28}$$

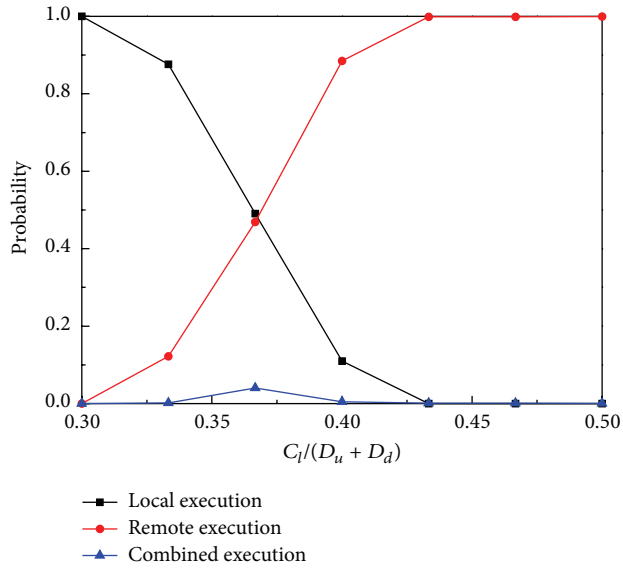


Figure 3: Probability of adopting three execution modes.

Table 5: Parameters in performance evaluation.

Notation	Parameter definition	Value
ω_d, ω_e	Weight factors in reward function	0~1
T	Task execution time limit	20
r_l	Speed of MT's processor	1
p_l	Power of MT's processor	2
p_u, p_d	Power of MT's transmitting and receiving antenna	3, 1
$r_{\min}, r_{\max}$	Minimum and maximum transmission capacities of wireless network	1, 5
σ	Penalty for timeout	100

Figure 3 shows the probability of adopting the three execution modes versus the ratio $C_l/(D_u+D_d)$. It can be seen that when C is relatively small to $D_u + D_d$, the probability of adopting the local execution mode is high. With the rising of $C/(D_u+D_d)$, the probability of adopting the remote execution mode goes to 1. The conclusion is obvious: offloading is beneficial when large amounts of local computation volume are needed with relatively small amounts of data transmission volume, and vice versa.

Next we consider a computation task with a comparative local computation volume and data transmission volume so that making offloading decisions is relying more on the real-time environmental information. The task description parameters adopted in the simulation are

$$(C, D_u, D_d) = (15, 20, 20); \tag{29}$$

other parameters are listed in Table 5.

The performance metric adopted is the expected total reward defined in Section 4 with different weight factors and different initial states. Figure 4 shows the performance of the MDP-based AEMSS and other four schemes, that is, always local, always offloading, and always combined schemes and a dynamic scheme that makes offloading decisions based on the wireless transmission capacity at the beginning of the execution (called DY scheme afterwards).

From Figures 4(a)–4(d) it can be concluded that (a) with higher wireless transmission capacity, the MDP-based policy gains a better performance, while the performance of always local scheme stays at a consistent level for the wireless transmission condition has no effect on the local execution process. (b) The always offloading scheme gains a pretty good performance, almost equivalent with the proposed AEMSS when the wireless transmission capacity is high whereas when the wireless transmission capacity decreases, the performance gap between them gets wider. (c) When the weight factor of energy consumption function is high, the performance of always combined policy is poor, because executing a task in local and remote modes simultaneously is an energy-intensive practice. However, when the weight factor of delay reward function increases, its performance improves and is equal to the AEMSS when the weight factor of delay reward function is 1. Under these circumstances, the combined execution mode is the optimal mode for its task completion time is shortest. (d) The performance of the DY mechanism is superior to the other three static policies, for it can react to the real-time environment condition. The performance gap between it and the AEMSS is caused mainly by the execution mode adjustment mechanism of AEMSS.

We integrate the expected total reward with different initial states by

$$v^{\pi} = \sum_{s \in S_0} p_0^{\infty}(s)\, v^{\pi}(s), \tag{30}$$

and the integrated performance of different policies is shown in Figure 5. Figures 4 and 5 reflect a phenomenon that the expected total reward increases linearly with the weight factor ω_d. This is driven by the design of the reward function, not indicating that a higher weight factor of the delay reward function is better. As defined in (16), the system will gain an extra reward ρ^* at each time slot after the task execution is completed. A higher ω_d will push the task execution to be finished earlier; therefore, the system can gain more extra reward until time T. When employing the AEMSS, the weight factors are determined by the designer's preference, that is, delay oriented or energy oriented.

$s_t = s_{\text{terminal}}$ indicates that the task execution process has been completed at time epoch t. Therefore, the completion probability of the task can be estimated by the steady state probability of the terminal state at decision epoch t; that is,

$$p_{c,t} = p_t^{\infty}(s_{\text{terminal}}). \tag{31}$$

Figure 6(a) depicts the task completion probabilities at each decision epoch with different policies when $\omega_d = \omega_e = 0.5$. We can see that the always combined scheme can complete the task fastest. The local computation volume is set as $C = 15$ in the simulation; therefore by time $t = 15$ the always local and always combined schemes can achieve a task completion probability of 1. The always offloading policy can complete

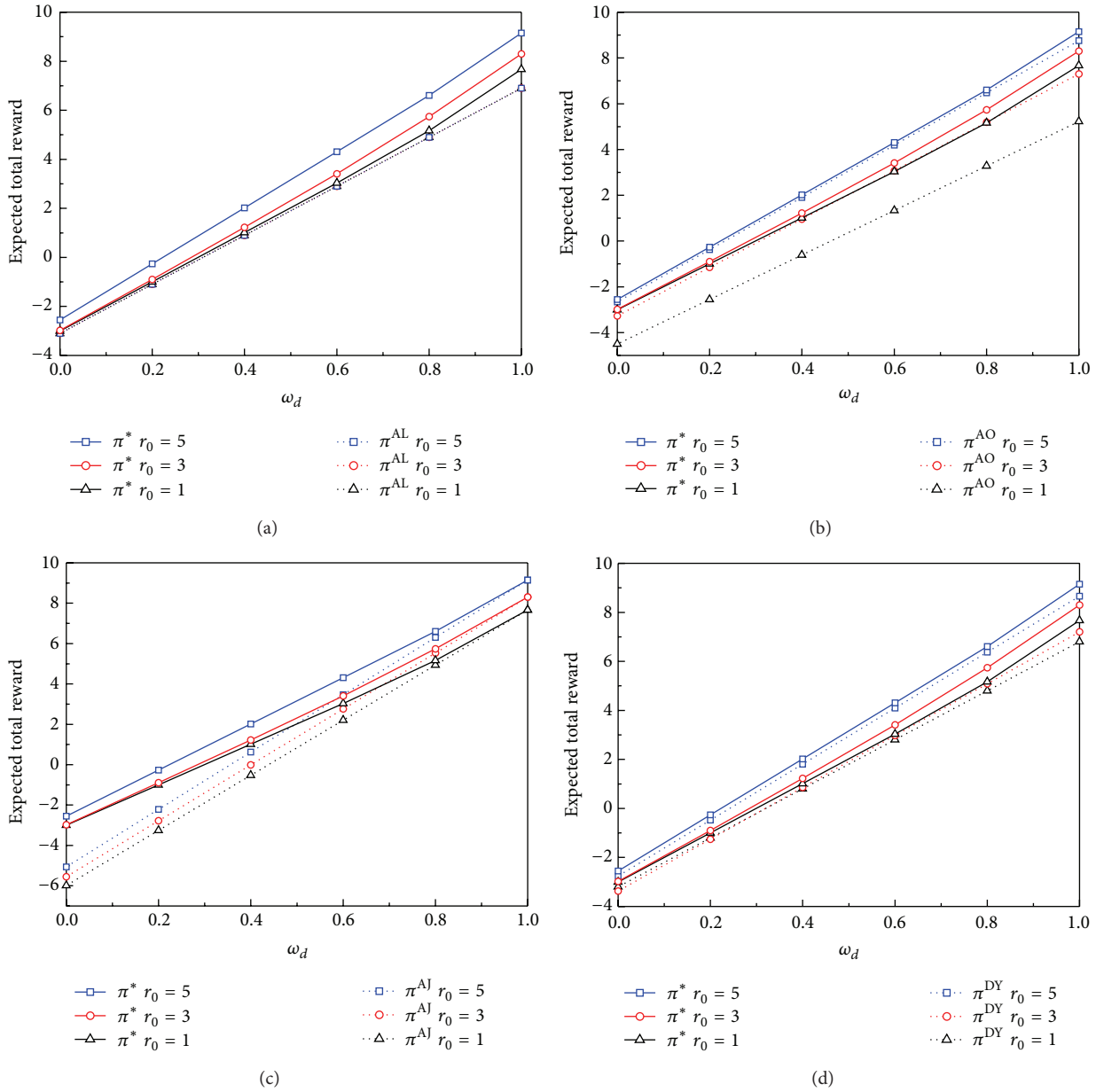


Figure 4: Expected total reward with different initial states and different policies.

the task with the highest probability when $t < 15$, but this may also leave the task uncompleted when $t > 15$ with the highest probability. The delay performance of proposed MDP-based AEMSS is at an intermediate level because it also takes the energy consumption into consideration. Figure 6(b) illustrates the task completion probabilities with different weight factors. We can see that, with a higher weight factor of the delay reward function, the task execution will be finished faster. When $\omega_d = 1$, the combined execution mode will be adopted with probability of 1; therefore, the task will be finished with probability 1 at time $t = 15$ ($C/p_l = 15$).

Figure 7 illustrates the tradeoff between the time saving and energy consumption of the AEMSS when the weight factors are varying. At $t = 15$, the delay performance and the cumulative energy consumption under the optimal policy π^* are plotted. It can be concluded that, with a higher ω_d, the task will be completed faster and the energy consumption will increase accordingly. This is because the combined execution mode is more likely to be adopted when the delay requirement is strict and executing the task both locally and remotely is energy intensive.

As described in Section 4, the AEMSS can adjust the execution mode during the task execution process when the wireless condition has dramatically changed. That is the main reason behind the performance improvement in our proposed scheme compared to the general dynamic offloading schemes. An observation is taken on the execution mode adjustment frequency at all the T decision epochs.

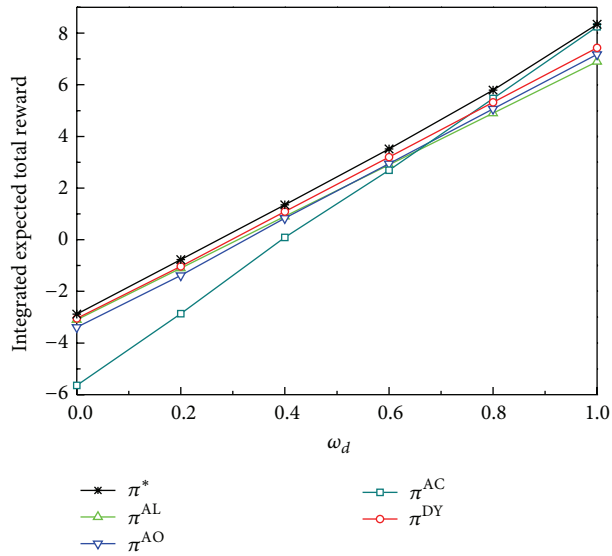
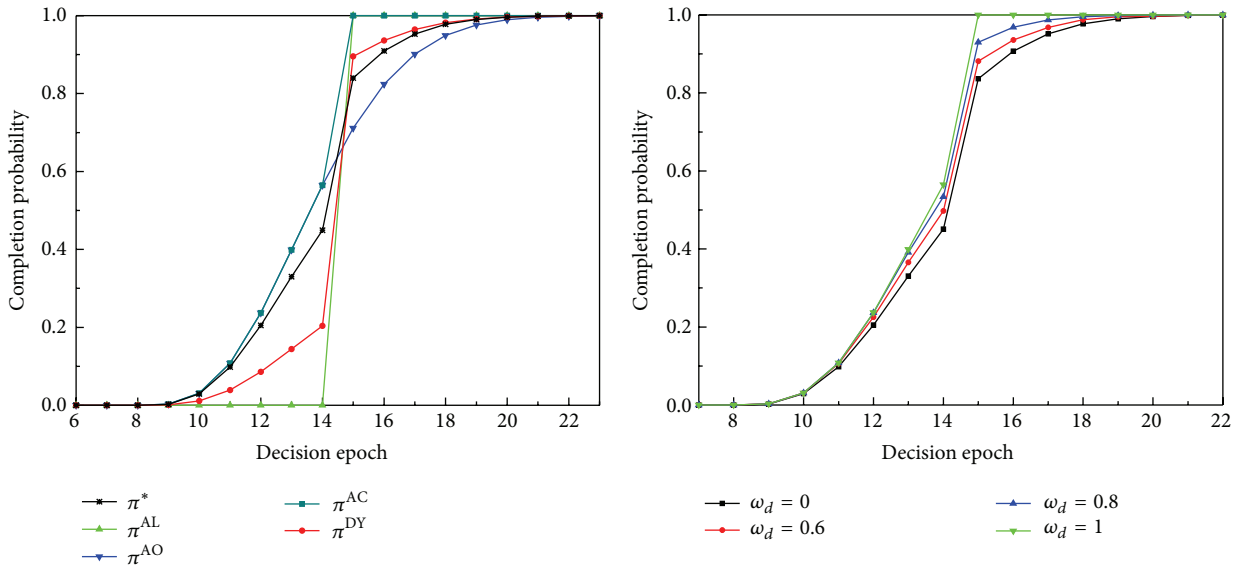


Figure 5: Integrated expected total reward.



(a) Task completion probabilities under different policies

(b) Task completion probability with different weight factors

Figure 6: Task completion probability.

At decision epoch t, an "execution mode adjustment" event, which is denoted as δ, occurs when

$$d_t^*(s) \neq a_t \neq \phi_t, \quad s = \left(\phi_t, C_t', D_{u,t}', D_{d,t}', r_t\right) \in \mathcal{S}_t, \tag{32}$$

and the occurrence probability of event δ at decision epoch t is defined as

$$p_t(\delta) = \sum_{s \in S_t} p_t^{\infty}(s) \cdot I\left[d_t^*(s) \neq \phi_t\right], \quad t \in \{0, 1, \ldots, T\}. \tag{33}$$

Figure 8 shows the execution mode adjustment probability at all decision epochs. Along with the timeline, the execution mode adjustment probabilities reduce to zero gradually. The reason is that, with the growth of the execution progress, adjusting the execution mode will cost a heavier price.

7. Conclusion

In this paper, MTs can execute their computation tasks either (1) locally; (2) remotely; or (3) combinedly. To determine the most appropriate execution mode, a dynamic offloading decision scheme, that is, the AEMSS, is proposed. The problem is formulated into a finite-horizon MDP with the objectives of minimizing the execution delay and reducing the energy consumption of MTs. Offloading decisions are

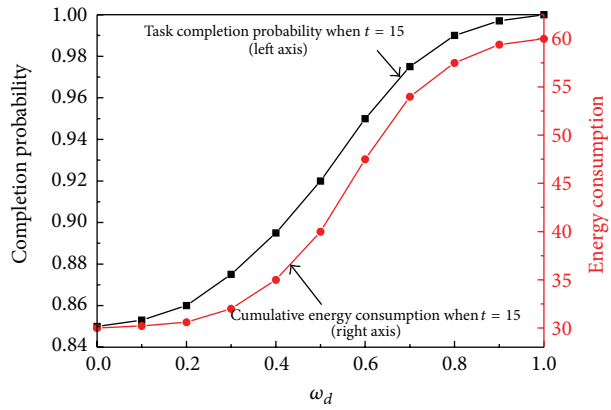


FIGURE 7: Task completion probability and energy consumption versus different weight factors when $t = 15$.

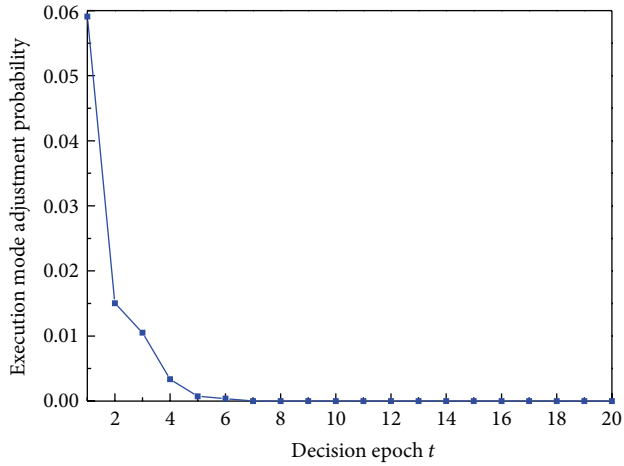


FIGURE 8: Execution mode adjustment probabilities at all decision epochs.

made by taking the task characteristic and the current wireless transmission condition into an overall consideration. In the design of reward function, an execution threshold time is introduced to make sure that the task execution can be completed with an acceptable delay. In addition, a novel execution mode adjustment mechanism is introduced to make the task execution process more flexible for the real-time environment variation. By solving the optimization problem with the BIA, a nonsteady policy describing the correspondence of states and actions is obtained. The policy is equivalent to a state-to-action mapping table which can be stored for looking up during the decision making phase. The performance of the proposed scheme is evaluated with other several offloading schemes and the numerical results indicate that the proposed scheme can outperform other algorithms in an energy-efficient way.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities (no. 2014ZD03-02), National Key Scientific Instrument and Equipment Development Project (2013YQ20060706), and National Key Technology R&D Program of China (2013ZX03003005).

References

[1] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–7, ACM Press, May 1996.

[2] K. W. Tracy, "Mobile application development experiences on Apples iOS and Android OS," *IEEE Potentials*, vol. 31, no. 4, pp. 30–34, 2012.

[3] D. Datla, X. Chen, T. Tsou et al., "Wireless distributed computing: a survey of research challenges," *IEEE Communications Magazine*, vol. 50, no. 1, pp. 144–152, 2012.

[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[5] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: can offloading computation save energy?" *Computer*, vol. 43, no. 4, Article ID 5445167, pp. 51–56, 2010.

[6] S. Gitzenis and N. Bambos, "Joint task migration and power management in wireless computing," *IEEE Transactions on Mobile Computing*, vol. 8, no. 9, pp. 1189–1204, 2009.

[7] N. I. Md Enzai and M. Tang, "A taxonomy of computation offloading in mobile cloud computing," in *Proceedings of the 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pp. 19–28, Oxford, UK, April 2014.

[8] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '01)*, pp. 238–246, November 2001.

[9] Z. Li, C. Wang, and R. Xu, "Task allocation for distributed multimedia processing on wirelessly networked handheld devices," in *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS '02)*, pp. 79–84, 2002.

[10] C. Xian, Y. H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, pp. 1–8, December 2007.

[11] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (PDPS '08)*, pp. 1–8, April 2008.

[12] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[13] H. Eom, P. S. Juste, R. Figueiredo, O. Tickoo, R. Illikkal, and R. Iyer, "Machine learning-based runtime scheduler for mobile offloading framework," in *Proceedings of the IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC '13)*, pp. 17–25, December 2013.

[14] A. Y. Ding, B. Han, Y. Xiao et al., "Enabling energy-aware collaborative mobile data offloading for smartphones," in *Proceedings*

of the 10th Annual IEEE Communications Society Conference on Sensing and Communication in Wireless Networks (SECON '13), pp. 487–495, June 2013.

[15] B. Jose and S. Nancy, "A novel application model and an off loading mechanism for efficient mobile computing," in *Proceedings of the IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '14)*, pp. 419–426, Larnaca, Cyprus, October 2014.

[16] P. Rong and M. Pedram, "Extending the lifetime of a network of battery-powered mobile devices by remote processing: a Markovian decision-based approach," in *Proceedings of the 40th Design Automation Conference*, pp. 906–911, June 2003.

[17] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, Hoboken, NJ, USA, 1994.