Jin *et al. EURASIP Journal on Wireless Communications and Networking* (2020) 2020:217
https://doi.org/10.1186/s13638-020-01827-w

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Task admission control for application service operators in mobile cloud computing

Xiaomin Jin[1,2*], Wenqiang Hua[1,2] and Zhongmin Wang[1,2]

Abstract

The resource constraint has become an important factor hindering the further development of mobile devices (MDs). Mobile cloud computing (MCC) is a new approach proposed to extend MDs' capacity and improve their performance by task offloading. In MCC, MDs send task requests to the application service operator (ASO), which provides application services to MDs and needs to determine whether to accept the task request according to the system condition. This paper studies the task admission control problem for ASOs with the consideration of three features (two-dimensional resources, uncertainty, and incomplete information). A task admission control model, which considers radio resource variations, computing, and radio resources, is established based on the semi-Markov decision process with the goal of maximizing the ASO's profits while guaranteeing the quality of service (QoS). To develop the admission policy, a reinforcement learning-based policy algorithm, which develops the admission policy through system simulations without knowing the complete system information, is proposed. Experimental results show that the established model adaptively adjusts the admission policy to accept or reject different levels and classes of task requests based on the ASO load, available radio resources, and event type. The proposed policy algorithm outperforms the existing policy algorithms and maximizes the ASO's profits while guaranteeing the QoS.

Keywords: Mobile cloud computing, Application service operator, Task admission control, Radio resource variation

1 Introduction

In recent years, with the rapid development of wireless network and computer technologies, the use of mobile devices (MDs) (e.g., smart phones, wearable devices, and smart vehicles) has become very popular in many industries. Cisco predicted that the number of MDs worldwide will grow from 8.6 billion in 2017 to 12.3 billion in 2022 [1]. At the same time, with the increasing popularity of mobile Internet, a large number of mobile applications providing different types of services are developed. People are spending more and more time on their MDs and want to do everything with the help of mobile applications. For instance, according to Internet Trend Report 2019 [2], the number of mobile users in China has exceeded 817 million with a year-on-year growth rate of 9%, and their mobile data traffic consumption increased by 189%. To ease the traffic burden of cellular networks, the research (e.g., [3]) on data offloading, which offloads data traffic to other complementary networks, has attracted the attention of scholars. MDs are equipped with more powerful CPUs and larger memories due to the improvements of chip manufacturing techniques. However, higher CPU frequency results in more energy consumption due to the fact that CPU power increases super-linearly with its frequency [4]. MDs are powered by batteries, whose capacity is limited because their size is limited to support MDs' portability. For example, as one case, compared with the previous generation of feature phones, today's smart phones have shorter working hours after one charge. As the other case, compared with combustion engine vehicles, the traveling distance of electric vehicles are limited by their battery volume [5]. Unlike the semiconductor

*Correspondence: xmjin@xupt.edu.cn
[1]School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Chang'an West Street, Xi'an, 710121 China
[2]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications, Chang'an West Street, Xi'an, 710121 China

Springer Open

© The Author(s). 2020 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

technology, the battery technology has not made breakthroughs in the short term, and the annual growth rate of the battery capacity is only 5% [6]. The development of the battery technology lags far behind the semiconductor technology developed by Moore's Law. On the other hand, due to a series of factors such as architecture and heat dissipation, although MD processing capacity has been improved, it is still weak compared with the ordinary computer, making MDs take much time and energy to execute some applications, and even cannot execute heavy applications. As a result, these constraints bring a poor user experience and prevent the further development of MDs.

Task offloading, which offloads computing tasks to the external platform to extend available MD resources, is an effective way to solve the problem of limited MD resources. Cloud computing, as the foundation of the future information industry, is a business computing model, which provides powerful external computing resources to MDs. On the basis of cloud computing, mobile cloud computing (MCC), which offloads application tasks to the cloud via task offloading, is proposed to address the problem that MD resources are limited. MCC provides a rich pool of resources that can be accessed through wireless networks. MCC has attracted wide attention from industry and academia because of its tremendous potential. There have been many mobile cloud applications for mobile healthcare [7], e-commerce [8], and mobile education [9]. According to the assessment of Allied Analytics LLP [10], the mobile cloud market is valued at $12.07 billion in 2016 and is expected to reach $72.55 billion by 2023, with a compound annual growth rate of 30.1% from 2017 to 2023. It is believed that the mobile cloud market will become more prosperous with the development of MCC.

The task offloading architecture, the offloading policy, and the offloading granularity are three main branches of the current research on MCC [11]. How to develop offloading policies has been studied in many previous works [12–16]. The offloading policy aims to improve the MD performance and determines whether a task should be offloaded to the cloud. If the offloading policy indicates that a task should be offloaded to the cloud, a task request is sent to the application service operator (ASO), which provides application services for mobile users. The target of ASOs is to maximize their profits, and ASOs try to accept as many tasks as possible to increase their income. However, if an ASO accepts all incoming task, it leads to resource overloading and then affects the quality of service (QoS). ASOs need task admission control to determine whether to accept a new task according to their current load conditions. The features of task admission control problem in MCC can be summarized as three points: (a) Two-dimensional resources. Mobile users in MCC are connected to the cloud via wireless networks, which have a serious impact on MCC [17]. Therefore, the task admission control in MCC has to consider both computing and radio resources. (b) Uncertainty. Wireless networks are not stable and vary for many reasons, such as the wireless channel fading and channel interference [18]. Wireless network variations lead to uncertainty to the task admission control in MCC. In addition, dynamic resource extension, uncertain task arrivals, and departures also lead to uncertainty. (c) Incomplete information. In real-life MCC, some information of the task admission control problem is often unclear or hard to obtain.

This paper strives to tackle the task admission control problem in MCC and aims to maximize the ASO's profits while ensuring the QoS. For features (a) and (b), a task admission control model, which considers radio resource variations, computing, and radio resources, is established based on the semi-Markov decision process (SMDP) with the long-term average criterion. SMDP is a powerful tool for solving the sequential decision-making problems and provides a mathematical framework that selects an action according to the state observed at each decision epoch. The SMDP policy, composed by a set of state-action pairs, can be developed offline and applied online. For feature (c), a policy algorithm based on the reinforcement learning (RL) is proposed to develop the admission policy. SMDP problems can be solved by using classical dynamic programming methods. However, these dynamic programming methods require the exact transition probabilities, which are often hard to obtain and need the extra storage to store [19]. At the same time, the complete system information of the task admission control problem is often unclear or hard to obtain in real-life MCC. Therefore, a RL-based policy algorithm is proposed. RL is a machine learning framework for solving sequential decision-making problems and can solve SMDP problems approximately without the complete system information. The main contributions of this paper are summarized as follows:

(1) The task admission control problem is formulated as a SMDP, and a SMDP-based model, which aims to maximize the ASO's profits while ensuring the QoS, is established. To describe the task admission control problem in MCC accurately, the established model considers two-dimensional resources (computing and radio resources), system uncertainty (radio resource variations, task uncertainty, and dynamic resource extension), and multi-level and multi-class application services.
(2) A RL-based policy algorithm is proposed to develop the admission policy. The proposed policy algorithm develops the admission policy through system simulations without requiring the complete system information. The policy can be developed offline and applied

online, and the admission control depends only on the current system state. These advantages make the proposed policy algorithm efficient for the task admission control problem in real-life MCC, whose complete information is often unclear or hard to obtain.

(3) Extensive simulation experiments are conducted to verify the established system model and proposed policy algorithm. The impact of system parameters on the ASO's profits and QoS is evaluated and analyzed. To verify the efficiency of the proposed algorithm, it is compared with existing algorithms such as the threshold-based policy algorithm, the greedy policy algorithm, and the random policy algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the related work. In Section 3, we first describe the system model and then illustrate the RL-based policy algorithm. In Section 4, the established model and proposed policy algorithm are evaluated. Section 5 concludes this paper.

2 Related work

The offloading policy, which is developed by the offloading decision-making algorithm, determines whether a task request is sent to the ASO. We first review the work that focuses on offloading decision-making briefly. The admission control problem is involved in wireless networks. Therefore, we also review the work on admission control in wireless networks after reviewing the work on task admission control in MCC.

2.1 Offloading decision-making

As mentioned above, how to make offloading decisions is a main branch of current research on MCC, and this problem is usually described as an application partitioning problem. Many works proposed algorithms to develop offloading policies to improve the MD performance. Zheng et al. investigated the problem of multi-user task offloading for MCC under dynamic environment, wherein mobile users become active or inactive dynamically, and the wireless channels for mobile users to offload task vary randomly [12]. They formulated the mobile users' offloading decision process under dynamic environment as a stochastic game and proved that the formulated stochastic game is equivalent to a weighted potential game which has at least one Nash Equilibrium. Mahmoodi et al. proposed an energy-efficient joint scheduling and task offloading scheme for MDs using applications with arbitrary component dependency graphs [13]. They defined a net utility that trades off the energy saved by the mobile, subject to constraints on the communication delay, overall application execution time, and component precedence ordering. Kumari et al. considered a trade-off between time and cost for offloading in MCC and proposed a two-step algorithm known as the cost and time constraint offloading algorithm, the task scheduling algorithm based on teaching, learning-based optimization, and the energy saving using the dynamic voltage and frequency scaling technique [14]. Hong and Kim proposed optimal transmission scheduling and optimal service class selection of task offloading while capturing the trade-off between energy, latency, and pricing [15]. They formulated the transmission scheduling problem as dynamic programming, and its optimal scheduling and two suboptimal scheduling algorithms have been derived. Hekmati et al. considered the multi-decision problem when task execution completion times are subject to hard deadline constraints, and when the wireless channel can be modeled as a Markov process [16]. They proposed an online mobile task offloading algorithm named MultiOpt to develop the offloading policy. In [3], Zhou et al. studied the data offloading problem and formulated it as an optimization problem, which also involves offloading decision-making that the mobile network operator decides whether a WiFi access point is selected to offload traffic. They designed an effective reverse auction-based incentive mechanism to stimulate WiFi access points to participate in the data offloading process.

2.2 Task admission control

After offloading decisions are made, requests of offloadable tasks are sent to the ASO. The ASO needs to use admission control to determine whether to accept a task and to allocate resources for it. Several works studied the task admission control from different aspects. Guo et al. established a ASO resource model using queuing theory and optimized admission control for multi-type task requests [20]. They modeled the admission control problem as an NP-hard optimization problem and used the moment-based convex linear matrix inequality relaxation to develop the admission policy. Lyu et al. studied the task admission problem with the aim of minimizing the total energy consumption while guaranteeing the latency requirements of MDs [21]. They transformed the admission control problem to an integer programming problem with the optimal substructure by pre-admitting resource-restrained MDs and proposed a quantized dynamic programming algorithm to develop the admission policy. Liu and Lee studied the admission control and resource allocation problem for partitioned mobile applications in MCC [22]. A discounted SMDP-based model was proposed to solve the admission control and resource allocation problem. They used the policy iteration approach to develop the optimal policy. Liu et al. focused on the resource allocation problem for the cloudlet-based MCC system with resource-intensive and latency-sensitive mobile applications [23]. They proposed a joint multi-resource allocation framework based on the

SMDP and used the linear programming to obtain the optimal resource allocation policy among multiple mobile users. Wang et al. studied the admission control problem in the multi-server and multi-user situation with the aim of minimizing the total energy consumption of MDs while guaranteeing their latency requirements [24]. They formulated the admission control problem to a multi-choice integer program and utilized Ben's genetic algorithm to solve it. Chen et al. proposed a comprehensive framework consisting of a resource-efficient computation offloading mechanism for users and a joint communication and computation resource allocation mechanism for network operator [25]. They formulated the admission control problem as a NP-hard optimization problem and designed an approximation algorithm based on the user ranking criteria to develop the admission policy. Qi et al. proposed a multi-level computing architecture coupled with admission control to meet the heterogeneous requirements of vehicular services and modeled the admission control problem as a MDP that optimizes the network throughput [26]. Khojasteh et al. proposed two task admission algorithms to keep the cloud system in the stable operating region by using two controlling parameters, that is, the full rate task acceptance threshold and the filtering coefficient [27]. Their first admission algorithm, which is based on the long-term estimation of the average utilization and offered load, is lightweight and appropriate for the cloud systems with a stable task arrival rate. Their second admission algorithm, which is based on the instantaneous utilization, is computation-intensive and appropriate for the systems with a varying task arrival rate. Lyazidi et al. focused on the admission control and resource allocation problem in the cloud radio access network [28]. They formulated the problem as an optimization problem constrained by mobile users' QoS requirements, the maximum transmission power, and the fronthaul links capacity. They reformulated the original nonlinear optimization problem as a mixed integer linear program and proposed a two-stage algorithm to solve it.

2.3 Admission control in wireless networks

Admission control is also studied in the research field of wireless networks. Mirahsan et al. studied the admission control problem for wireless virtual networks in heterogeneous wireless networks with the goal of improving the QoS of network operators and proposed an admission method including the feedback information of virtual network users [29]. They formulated the admission control problem as a convex optimization problem, which allows the general multi-association between users and base stations, and proposed a solution algorithm for heterogeneous traffic distribution networks. Dromard et al. proposed an admission control model combining the dynamic link scheduling for the bandwidth limitation problem of wireless Mesh networks and transformed it as a 0-1 linear programming problem, aiming to optimize the network bandwidth usage [30]. Zhang et al. studied the admission control problem in sensor nodes and developed stochastic models in wireless sensor networks to explore admission control with the sleep/active scheme [31]. Shang et al. established an admission control model based on the matching game and multi-attribute decision-making according to the network's attribute and system resource allocation [32]. They proposed an algorithm, which balances the interests of the network and user, reflects the superiority of the balanced decision of both parties, and guarantees the common interests of the network and user.

This paper focuses on the task admission control problem in MCC and fully considers features (a)–(c) in model establishment and algorithm design, which goes beyond existing works. Existing models on task admission control in MCC ignore the feature (b) uncertainty and assume that the wireless networks are stable. At the same time, existing works on admission control in wireless networks only need to consider the radio resource load conditions and cannot highlight the feature (a) two-dimensional resources. Different from these works, two-dimensional resources (computing and radio resources), system uncertainty (radio resource variations, task uncertainty, and dynamic resource extension), and multi-level and multi-class application services are considered in the established model, making it more accurate in describing the task admission control problem in MCC. Furthermore, existing works do not pay enough attention to the feature (c) incomplete information and use dynamic programming methods, which need the complete system information to develop the admission policy. The proposed RL-based policy algorithm develops the admission policy through system simulations without requiring the complete system information.

3 Methods

3.1 System model

3.1.1 Mobile cloud computing system architecture

The MCC system architecture, illustrated in Fig. 1, is divided into 3 parts, namely the mobile users, the ASO, and the cloud operator. The cloud operator manages physical resources and provides the virtual resource renting service. The ASO obtains virtual resources from the resource pool and does not need to have its own hardware equipments, which frees the ASO from the equipment purchase and maintenance and helps the ASO pay more attention on the application service development. The ASO provides application services (e.g., augmented reality (AR), virtual reality (VR), and speech and image recognition) for mobile users. Task requests from mobile users are sent to the ASO. After receiving a task request, the

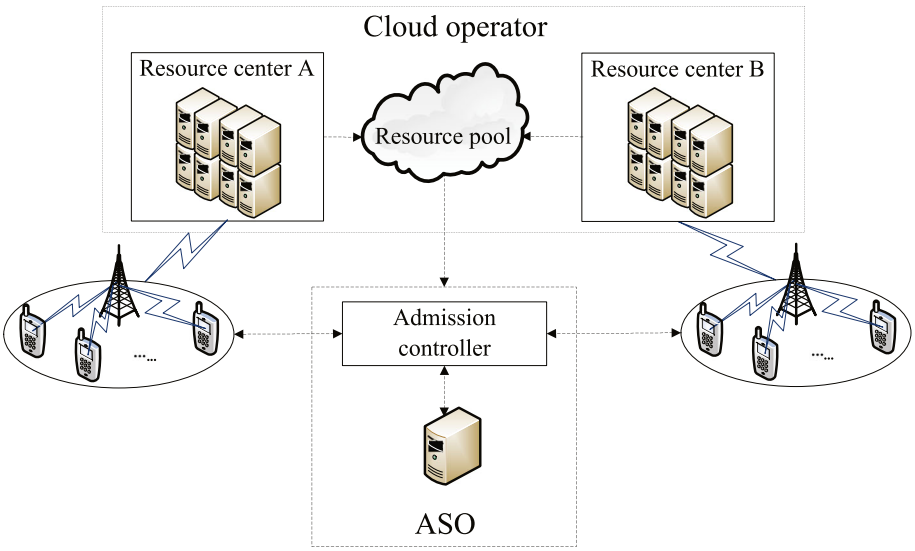

Fig. 1 Mobile cloud computing system architecture. The MCC system architecture is divided into 3 parts, including the mobile users, the ASO, and the cloud operator

admission controller determines whether to accept the task.

3.1.2 SMDP-based task admission control model

In this section, the SMDP-based task admission control model is illustrated. The application service, the state space, the action space, and the reward function of the SMDP-based model are defined. The task admission control model is abstracted as the model shown in Fig. 2. The admission controller decides whether to accept the task request according to the system state when a task request arrives. If the task is accepted, the ASO allocates resources and executes the task. Many ASOs offer same services for mobile users in the cloud service market [33]. If a task request is rejected, it leaves the current ASO and switches to another ASO that provides the same application service. The system state is updated after a system event occurs.

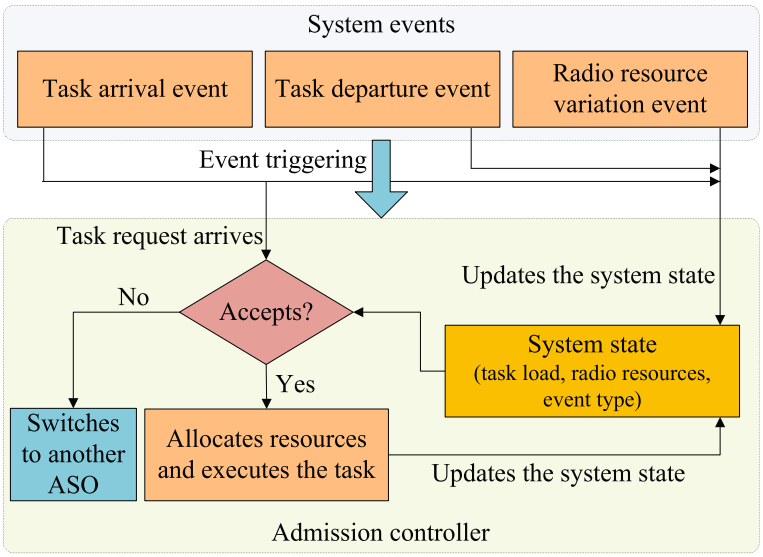

Fig. 2 Task admission control model. The task admission control model is abstracted as the model shown in this figure. The admission controller decides whether to accept the task request according to the system state when a task request arrives. If the task is accepted, the ASO allocates resources and executes the task

(1) *Application service*

The ASO provides multi-level and multi-class application services to mobile users. This paper assumes that the ASO provides L levels and M classes application services, and thus, the ASO will receive $L \times M$ types of task requests from mobile users. If a task request is accepted, the ASO allocates resources to execute the task according to the level it requires. Let $\{(c_l, b_l) \,|1 \leq l \leq L\}$ denote the resources that the ASO provides. $c_l = x_l c^u$ and $b_l = y_l b^u$ represent the computing and radio resources provided by the ASO for l-level application services, respectively. c^u (b^u) represents one computing (radio) resource unit, which is the minimum computing (radio) resources provided to the mobile users. For example, $c^u = 1$ GHz and $b^u = 100$ kbps. In the following description, the unit is omitted. That is to say, $c_l = x_l$ ($b_l = y_l$) has the same meaning to $c_l = x_l c^u$ ($b_l = y_l b^u$).

The task request rejecting probability (referred to "rejecting probability" for simplicity in the following description) is taken as the indicator of QoS. Let P_l^r denote the rejecting probability that the ASO guarantees for the l-level application services. In general, the higher the application service level, the higher the QoS ASO guarantees. Also, if the mobile user purchases a higher-level application service, the ASO will allocate more resources. Mathematically, if $1 \leq l_1 < l_2 \leq L$, then $c_{l_1} < c_{l_2}$, $b_{l_1} < b_{l_2}$ and $P_{l_1}^r > P_{l_2}^r$. The notations used in this paper are summarized in Table 1.

(2) *State space*

At a decision epoch, the state is the system descriptor, and the admission controller makes decisions according to the current state. The state space, represented by set S, is composed of all system states. The state s ($s \in S$) is expressed as $s = (Z(s), B(s), E(s))$. $Z(s) = \left(Z_l^m(s)\right)_{l=1,m=1}^{L,M}$ represents the numbers of tasks that are being executed in the ASO, and its element $Z_l^m(s)$ represents the number of l-level and m-class tasks that are being executed in the ASO. $B(s)$ represents the available ASO radio resources. As mentioned above, radio resources vary for many reasons, such as the channel fading and channel interference [18]. Moreover, the ASO radio resources rented from the cloud operator may vary within the service-level agreement (SLA) between the ASO and the cloud operator. This paper assumes that $B(s)$ varies from B_L to B_U, in which B_L and B_U represent the lower and upper radio resource bounds, respectively. $E(s)$ represents the event type, and $E(s) \in \{O\} \cup \bigcup_{l=1,m=1}^{L,M} \left\{A_l^m, D_l^m\right\}$. $E(s) = A_l^m$ represents the arrival event of the l-level and m-class task request. After the completion of a task, the task departs from the ASO, and $E(s) = D_l^m$ represents the departure event of the l-level and m-class task. $E(s) = O$ represents the radio resource variation event. One noticeable

Table 1 Notations

Name	Description
L	The number of application service levels
M	The number of application service classes
c_l	The computing resources allocated for the l-level application service
b_l	The radio resources allocated for the l-level application service
P_l^r	The rejecting probability guaranteed for the l-level application service
B_L	The lower radio resource bound
B_U	The upper radio resource bound
A_l^m	The arrival event of the l-level and m-class task request
D_l^m	The departure event of the l-level and m-class task
O	The radio resource variation event
C	The base computing resources
C_U	The upper bound of the extendable computing resources
a_a	The action to accept the task request
a_r	The action to reject the task request
a_d	The action taken when a departure event occurs
a_o	The action taken when a radio resource variation event occurs
R_l^m	The income from accepting and executing a l-level and m-class task
F_0	The penalty coefficient
$f_c(\cdot)$	The cost coefficient of occupying the computing resources
f_b	The cost coefficient of occupying the radio resources
α, β	Two learning rates
N, V	Two maximum iterations in the RL-based policy algorithm
a_{greedy}	The greedy action that results in the highest action-value function
a_{random}	The random action selected from the action space
ρ	The average reward
T	The current temperature
T_0	The initial temperature
T_γ	The temperature dropping coefficient
r_n	The accumulated reward until the nth iteration
τ_n	The accumulated time until the nth iteration
ω_l	The Lagrange multiplier
$P_{\omega_l}^r$	The rejecting probability with ω_l
δ_l	The coefficient to update ω_l
ε_l	The accuracy requirement
λ_l^m	The arrival rate of the l-level and m-class task request
μ_l^m	The departure rate of the l-level and m-class task
λ_o	The occurrence rate of the radio resource variation event

advantage of the cloud computing is that it allows dynamic resource extension, and the ASO computing resources can be further extended. Let C denote the base computing resources and C_U denote the upper bound of the extendable computing resources. Computing and radio resources occupied by the tasks that are being executed should not exceed the computing and radio resource upper bounds, which are ensured by constraints

$$\forall s \in S, 0 \leq \sum_{l=1}^{L} \sum_{m=1}^{M} Z_l^m(s) c_l \leq C_U \tag{1}$$

and

$$\forall s \in S, 0 \leq \sum_{l=1}^{L} \sum_{m=1}^{M} Z_l^m(s) b_l \leq B_U. \tag{2}$$

Constraint

$$\forall E(s) = D_l^m, Z_l^m(s) \geq 1 \tag{3}$$

ensures that at least one task is being executed when a departure event occurs.

(3) *Action space*

The action space for state s is a set of all possible actions that can be taken at state s. When an event $E(s)$ occurs, the admission controller selects an action from the action space $A(s)$. The action space $A(s)$ is a subset of $A = \{a_d, a_o, a_a, a_r\}$, which represents all actions in the system model. If a task arrival event occurs $\left(E(s) = A_l^m\right)$, the task request can be accepted or rejected, which are denoted as taking actions a_a or a_r. However, the task request must be rejected when the required ASO computing or radio resources exceed their upper bounds. Therefore, if $E(s) = A_l^m$, action space $A(s)$ is expressed as

$$A(s) = \begin{cases} \{a_r\}, & \sum_{l=1}^{L} \sum_{m=1}^{M} Z_l^m(s) c_l + c_l > C_U \vee \\ & \sum_{l=1}^{L} \sum_{m=1}^{M} Z_l^m(s) b_l + b_l > B_U \\ \{a_a, a_r\}, \text{ otherwise} \end{cases}. \tag{4}$$

If $E(s) = D_l^m$, $A(s) = \{a_d\}$, which means only action a_d can be taken when a departure event occurs. If $E(s) = O$, $A(s) = \{a_o\}$, which means only action a_o can be taken when a radio resource variation event occurs. Unlike actions a_a and a_r, a_d and a_o do not affect the profits and QoS. However, a_d and a_o are the important elements to ensure the integrity and correctness of the system model. When a departure event or a radio resource variation event occurs, at least one optional action is required.

(4) *Reward function*

The reward function represents the profits of the decision-making at current state. The reward function $r(k, s, a)$, expressed as

$$r(k, s, a) = f_r(k, s, a) - \tau(k, s, a) \left[o_1(k, s, a) + o_2(k, s, a)\right], \tag{5}$$

is defined to represent the profits resulted from taking action a at state s with next state k. $f_r(k, s, a)$, expressed as

$$f_r(k, s, a) = \begin{cases} R_l^m, E(s) = A_l^m \wedge a = a_a \\ 0, \text{otherwise} \end{cases}, \tag{6}$$

represents the income from taking action a at state s. R_l^m represents the income from accepting and executing a l-level and m-class task. The system cost is caused by two parts, which are the penalty for short of radio resources and the cost of occupying resources to execute tasks, respectively. The second portion of Eq. (5) represents the system cost, in which $\tau(k, s, a)$ represents the sojourn time from current state s to next state k after taking action a, $o_1(k, s, a)$ represents the penalty per unit time for short of radio resources, and $o_2(k, s, a)$ represents the system cost per unit time of occupying the computing and radio resources rented from the cloud operator to execute tasks. After action a is taken at state s, the number of tasks that are being executed is equal to $Z_l^m(s)$, and the available radio resources are $B(k)$.

$o_1(k, s, a)$ is calculated by

$$o_1(k, s, a) = \max\left(\sum_{l=1}^{L} \sum_{m=1}^{M} Z_l^m(k) b_l - B(k), 0\right) F_0, \tag{7}$$

in which F_0 represents the penalty coefficient, and the other portion denotes the radio resource shortage, which may be caused by a radio resource variation event. If there are enough radio resources, $o_1(k, s, a)$ is equal to zero; otherwise, $o_1(k, s, a)$ is positive. When the radio resources are short, the ASO cannot allocate sufficient radio resources to the task, and the ASO's punitive cost is paid to mobile users as the compensation for the radio resource shortage. One simple way to solve the problem of radio resource reallocating and compensation dividing among mobile users is reducing the radio resource allocation and dividing

the compensation both equally.

$o_2(k,s,a)$ is calculated by

$$o_2(k,s,a) = f_c\left(\sum_{l=1}^{L}\sum_{m=1}^{M} Z_l^m(k)c_l\right)\sum_{l=1}^{L}\sum_{m=1}^{M} Z_l^m(k)c_l + f_b\min\left(\sum_{l=1}^{L}\sum_{m=1}^{M} Z_l^m(k)b_l, B(k)\right), \quad (8)$$

in which $f_c(\cdot)$ and f_b represents the cost coefficient of occupying the computing and radio resources, respectively. Generally, dynamically extended computing resources are more expensive, and

$$f_c(x) = \begin{cases} f_0^c, 0 < x \le C \\ \frac{f_0^c C + f_1^c(x-C)}{x}, C < x \le C_U \end{cases} \quad (9)$$

is used to calculate the cost coefficient of occupying computing resources, in which $f_0^c \le f_1^c$. $f_c(x)x$ represents the cost of occupying the computing resources x. If $0 < x \le C$, the cost $f_c(x)x = f_0^c x$. If $C < x \le C_U$, the cost $f_c(x)x = f_0^c C + f_1^c(x - C)$, in which $f_0^c C$ represents the cost of occupying the base computing resources C, and $f_1^c(x-C)$ represents the cost of occupying the dynamically extended computing resources $(x - C)$. f_0^c, f_1^c and f_b are constants.

On the basis of the system model, the decision process of the admission controller can be represented as the process shown in Fig. 3 [34]. At time t_i, the admission controller observes current state s_i and selects action a_i from action space $A(s_i)$. This action-taking has two effects on the system: one is to receive the corresponding reward, and the other is that the system state is affected by this action and enters next state s_{i+1}. At time t_{i+1}, the admission controller faces the same problem as the previous decision-making time, that is, selecting an action according to the current system state. The decision-making process will go on in this form and generate a policy made up of state-action pairs and a reward sequence. The problem of policy solving is that a policy is required to maximize the value of a function (criterion) of the reward sequence under this policy. A long-term average criterion, expressed as

$$\rho^{\pi(s)} = \lim_{I\to\infty} \frac{\sum_{i=0}^{I} r\left(s_{i+1}, s_i, a_i\right)}{\sum_{i=0}^{I} \tau\left(s_{i+1}, s_i, a_i\right)}, \quad (10)$$

in which $\pi(s)$ represents the policy, is used in the system model and aims to maximize the profit per unit time.

3.2 Task admission control policy algorithm

In this paper, a RL-based task admission control policy algorithm, whose pseudo-code is shown in Fig. 4, is proposed. It can be seen that the RL-based policy algorithm has two loops, whose execution times are N and V. Also, the RL-based policy algorithm needs space to store $Q(\widehat{s}, a)$, $Q^+(\widehat{s}, a)$, and some intermediate variables. Therefore, the RL-based policy algorithm has time complexity $o(NV)$ and space complexity $o\left(\left|\widehat{S}\right||A|\right)$, in which $\left|\widehat{S}\right|$ represents the aggregated state space size and $|A|$ represents the action space size. The RL model is illustrated in Fig. 5, and the admission controller learns through interacting with the system environment. The RL-based policy algorithm is a simulation-based algorithm, which develops the approximate optimal policy using the observed data from the real-life system or through system simulations without the complete system information. Therefore, it has a wider range of use and can be used in more problem scenarios. The long-term average Q-learning [19], which belongs to the value iteration-based RL methods, is used to develop the policy. By using Q-learning, the admission controller can make decisions after learning an action-value function, which gives a value of each state-action pair. For a state, the admission controller selects the action that has the highest value obtained from the action-value

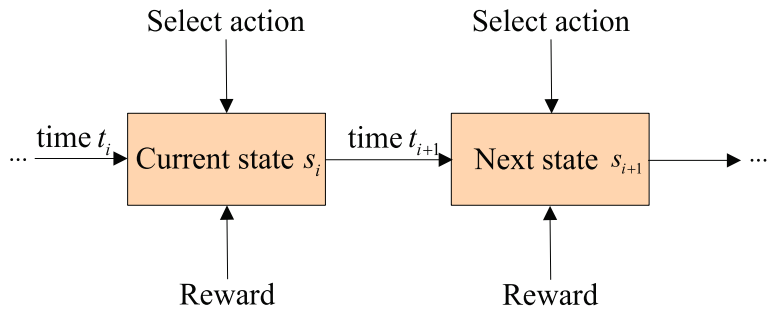

Fig. 3 The decision process of the admission controller. The decision process of the admission controller is represented as the process shown in this figure. The decision-making process will go on in this form and generate a policy made up of state-action pairs and a reward sequence

Algorithm: RL-based Policy Algorithm

01: initialize $n=1, v=1, Q(\hat{s},a)=0, Q^{+}(\hat{s},a)=0, \rho=0, \rho^{+}=0, r_n=0, \tau_n=0$;
02: **repeat**
03: **repeat**
04: $\hat{s} \leftarrow$ currents state ;
05: **if**($\varphi \le p(a_{greedy} \rightarrow a_{random})$)**then**
06: $a \leftarrow a_{greedy}$;
07: **else then**
08: $a \leftarrow a_{random}$;
09: **endif**
10: simulates the next state $\hat{k}$;
11: $r_n \leftarrow r_n + r(\hat{k},\hat{s},a,\omega)$;
12: $\tau_n \leftarrow \tau_n + \tau(\hat{k},\hat{s},a)$;
13: updates ρ by (15);
14: updates α by (16);
15: updates β by (17);
16: updates $Q(\hat{s},a)$ by (12);
17: $n \leftarrow n+1$;
18: **until** $n = N+1$
19: **if**($P^r_{\omega_l} - P^r_l \le \varepsilon_l$ && $\rho > \rho^{+}$)**then**
20: $\rho^{+} \leftarrow \rho$;
21: $Q^{+}(\hat{s},a) \leftarrow Q(\hat{s},a)$;
22: **endif**
23: updates ω by (20);
24: $v \leftarrow v+1$;
25: **until** $v = V+1$
26: **return** admission policy $\pi(\hat{s})$ derived from $Q^{+}(\hat{s},a)$.

Fig. 4 Pseudo-code of the RL-based policy algorithm. This figure illustrates the pseudo-code of the proposed RL-based task admission control policy algorithm

function as the optimal action. In the learning process, the QoS constraint should be considered, and thus, the system state is modified to handle the QoS constraint. As the core components of the RL-based policy algorithm, the action-value function and the QoS constraint are illustrated in the following description.

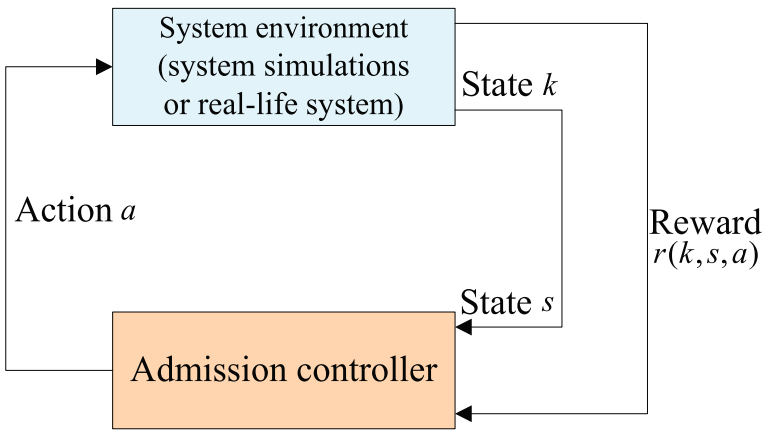

Fig. 5 Reinforcement learning model. This figure illustrates the RL model used in the proposed task admission control policy algorithm, which learns through interacting with the system environment

3.2.1 Action-value function

Let $Q(s,a)$ represent the action-value function, whose value denotes the average adjusted value of taking action a at state s. According to [19], the Bellman equation for the long-term average reward SMDP-based problem can be expressed as

$$Q^*(s,a) = \sum_{k\in S} p(k|s,a)\left[r(k,s,a) - \rho^*\tau(k,s,a) + \max_{g\in A(k)} Q^*(k,g)\right], \quad (11)$$

in which $Q^*(s,a)$ represents the average adjusted value by taking actions optimally, $p(k|s,a)$ represents the transition probability that the system state transfers from state s to state k after taking action a, and ρ^* represents the optimal average reward. The optimal policy $\pi^*(s) = \arg\max_{a\in A(s)} Q^*(s,a)$.

Based on the Robbins-Monro algorithm, a temporal difference method, expressed as

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r(k,s,a) - \rho\tau(k,s,a) + \max_{g\in A(k)} Q(k,g)\right], \quad (12)$$

is used to update $Q(s,a)$ in the iterations, in which α represents the learning rate, ρ represents the average reward, and k represents the next state. Equation (12) shows that the RL-based policy algorithm does not require transition probabilities and can run with the data from simulations or real-life system.

At the initial step, all $Q(s,a)$ are set to a same value (e.g., 0). When visiting a state, RL-based policy algorithm needs to simulate an action-taking. To avoid falling into the local optimal policy and improve the probability to achieve the global optimal policy, the simulated annealing algorithm [35], which simulates the annealing process of heating solids and brings random factors in the selecting process, is used to select an action for the state-action pair whose action-value function is to be updated. The random action that may be worse than the greedy action is selected with a certain probability. When selecting an action for a state-action pair, a random number $\varphi \in [0,1)$ is generated and compared with

$$p\left(a_{\text{greedy}} \rightarrow a_{\text{random}}\right) = e^{\left[Q(s,a_{\text{random}}) - Q(s,a_{\text{greedy}})\right]/T}, \quad (13)$$

which represents the probability of selecting action a_{greedy} instead of selecting action a_{random}. a_{greedy} represents the greedy action that results in the highest action-value function, a_{random} represents a random action selected from the action space, and T represents the current temperature. T is calculated by

$$T = T_0 T_\gamma^{n-1}, \quad (14)$$

in which T_0 represents the initial temperature, T_γ represents the temperature dropping coefficient, and n represents the number of iterations. If $\varphi \le p\left(a_{\text{greedy}} \rightarrow a_{\text{random}}\right)$, a_{random} is selected; otherwise, a_{greedy} is selected.

After the simulative action a is taken, the system goes to the next simulative state k. The average reward is updated by

$$\rho \leftarrow (1-\beta)\rho + \beta\frac{r_n}{\tau_n}, \quad (15)$$

in which β represents another learning rate, and r_n and τ_n represent the accumulated reward and time until the nth iteration, respectively. The two learning rates (α and β) are calculated by

$$\alpha = \log(n) / n \quad (16)$$

and

$$\beta = H_\beta^1 / \left(H_\beta^2 + n\right), \quad (17)$$

respectively. After a certain number of iterations, the action-value function is learnt, and the approximate optimal policy is developed by the action-value function.

3.2.2 QoS constraint

The QoS constraint is formulated as

$$\lim_{I\to\infty} \frac{\sum_{i=0}^{I} P_l^r(s_{i+1})\,\tau(s_{i+1},s_i,a_i)}{\sum_{i=0}^{I} \tau(s_{i+1},s_i,a_i)} \le P_l^r \quad (18)$$

with the long-term average criterion. Equation (18) indicates that the long-term time-average rejecting probability of the l-level application services is no more than P_l^r. In Eq. (18), s_{i+1} represents the next state of state s_i, a_i represents the action taken at state s_i, and $P_l^r(s_{i+1})$ represents the rejecting probability at state s_{i+1}. The Lagrange multiplier framework [36] is used to deal with the QoS constraint. According to Eq. (18), the QoS constraint depends on the rejecting probability and sojourn time. Therefore, the expression of state s is extended as $\bar{s} = \left[N^t(\bar{s}), N^r(\bar{s}), \tau, s\right]$, in which $N^t(\bar{s}) = \left(N_l^t(\bar{s})\right)_{l=1}^L$ represents the total number of task requests, $N^r(\bar{s}) = \left(N_l^r(\bar{s})\right)_{l=1}^L$ represents the total number of rejected task requests, and τ represents the sojourn time between decision epochs. However, $N_l^t(\bar{s})$ and $N_l^r(\bar{s})$ $\left(N_l^r(\bar{s}) \le N_l^t(\bar{s})\right)$ can be any nonnegative integers, and τ is a decimal, making the extended state space infinite. To add the QoS constraint into the RL-based policy algorithm, the extended state space must be finite. The quantization method [37] of the rejecting probability and sojourn time is used to aggregate the extended states to make the extended state space finite. The aggregated state is denoted as $\hat{s} =$

$[h_1(\widehat{s}), h_2(\widehat{s}), s]$. $h_1(\widehat{s}) = (h_l^1(\widehat{s}))_{l=1}^L$, in which $h_l^1(\widehat{s})$ represents the quantized rejecting probability of l-level task requests, and $h_2(\widehat{s})$ represents the quantized sojourn time. The rejecting probability $\left(N_l^r(\overline{s}) / N_l^t(\overline{s})\right)$ is quantized into 100 levels, and τ is quantized into 2 levels. If $\tau \leq \overline{\tau}$ ($\overline{\tau}$ represents the average sojourn time), τ is quantized to level 1; otherwise, τ is quantized to level 2.

After the extended states are aggregated, the action-value function with the QoS constraint is denoted as $Q(\widehat{s}, a)$. In the Lagrange multiplier framework, the reward function, expressed as

$$r\left(\widehat{k}, \widehat{s}, a, \omega\right) = r\left(\widehat{k}, \widehat{s}, a\right) - \sum_{l=1}^{L} \omega_l q_l\left(\widehat{k}, \widehat{s}, a\right), \tag{19}$$

is adjusted with the Lagrange multiplier $\omega = (\omega_l)_{l=1}^L$. In Eq. (19), $r\left(\widehat{k}, \widehat{s}, a\right)$ is equal to the original reward function, that is, $r\left(\widehat{k}, \widehat{s}, a\right) = r(k, s, a)$. $q_l\left(\widehat{k}, \widehat{s}, a\right)$ represents the cost function associated with the QoS constraint, and $q_l\left(\widehat{k}, \widehat{s}, a\right) = f\left(h_l^1\left(\widehat{k}\right)\right)\tau\left(\widehat{k}, \widehat{s}, a\right)$, in which $f\left(h_l^1\left(\widehat{k}\right)\right)$ denotes the rejecting probability level that $h_l^1\left(\widehat{k}\right)$ represents. To find an optimal ω_l, ω_l is updated by

$$\omega_l \leftarrow \omega_l + \delta_l\left(P_{\omega_l}^r - P_l^r\right), \tag{20}$$

in which δ_l is a updating coefficient, and $P_{\omega_l}^r$ represents the rejecting probability with ω_l.

4 Results and discussion

In this section, extensive simulation experiments are conducted to evaluate the established system model and the proposed policy algorithm. The arrival of the l-level and m-class task request is assumed to follow a Poisson process with mean rate λ_l^m. If a task request is accepted, the ASO allocates resources and executes the task. The resource occupation time of the task is assumed to follow the exponential distribution. The mean occupation time of the l-level and m-class task is represented by $1 / \mu_l^m$, and thus, the mean rate of the task departure is μ_l^m. The occurrence of the radio resource variation event is assumed to follow a Poisson process with mean rate λ_o, and the radio resources vary uniformly between its upper and lower bounds. Based on this experimental settings, the cumulative event rate at state s with action a, denoted by $\gamma(s, a)$, is the sum of all event rates. $\gamma(s, a)$ is calculated by

$$\gamma(s,a) = \begin{cases} \gamma_0 + \mu_l^m, E(s) = A_l^m, a = a_a \\ \gamma_0, E(s) = A_l^m, a = a_r \\ \gamma_0 - \mu_l^m, E(s) = D_l^m, a = a_d \\ \gamma_0, E(s) = O, a = a_o \end{cases}, \tag{21}$$

in which $\gamma_0 = \sum_{l=1}^{L}\sum_{m=1}^{M}\left(\lambda_l^m + Z_l^m(s)\mu_l^m\right) + \lambda_o$. According to the property of exponential distribution that the minimum of exponential random variables is also exponentially distributed with the cumulative rate parameters [38], sojourn time of the earliest event follows the exponential distribution with rate parameter $\gamma(s, a)$. The sojourn time $\tau(k, s, a)$ is a random variable and is generated randomly according to its distribution, in which k is the next state after the earliest event occurs. R_l^m represents the income from accepting a l-level and m-class task and is set as $R_l \frac{1}{\mu_l^m}$, in which R_l represents the income per unit time, and η_l ($\eta_l = R_l / R_1$) represents the ratio of R_l and R_1. Two indicators, the system reward/profits (SR) and rejecting probability (RP), are concerned. The unit of the SR is UM (Unit Money). This section first evaluates the established system model and then compares the performance of the proposed policy algorithm and other algorithms. The default simulation parameters are listed in Table 2.

4.1 Evaluation of the system model

4.1.1 Impacts of the arrival rate

Figure 6 shows the SR and RPs under different task request arrival rates. In this experiment, the arrival rates are set to be equal, and λ_{sum} represents the sum of the arrival rates, that is, $\lambda_1^1 = \lambda_1^2 = \lambda_2^1 = \lambda_2^2 = \lambda_{\text{sum}} / 4$. It can be observed that the SR increases with the increasing λ_{sum}, and its increments are 7.31UM, 6.83UM, 6.38UM, 5.72UM, 5.27UM, 4.68UM, 3.95UM, 3.5UM, 2.71UM, 2.38UM, 1.62UM, and 0.59UM, respectively, which shows that the SR increases slowly when λ_{sum} becomes large. The RP (RP($l = 1$)) of 1-level task requests increases from 1.77% to 8.07%, and the RP (RP($l = 2$)) of 2-level task requests increases from 0.33% to 4.77%. When λ_{sum} is 24.75, RP($l = 1$) (8.07%) is slightly larger than the maximum allowable rejecting probability $\left(P_1^r = 8\%\right)$ with a difference of 0.07%. The RL-based policy algorithm searches for the approximate optimal admission policy while guaranteeing the QoS requirement iteratively. It is considered to meet the requirement within the accuracy range, which is 0–0.1% in this paper. When λ_{sum} is small, the ASO resources are enough, and the QoS requirement is easy to be met. Therefore, more task requests are accepted with the increasing λ_{sum}, and the SR increases

Table 2 Default simulation parameters

L	M	c_1	c_2	b_1	b_2	λ_1^1	λ_1^2	λ_2^1	λ_2^2	λ_o	μ_1^1	μ_1^2
2	2	1	2	1	2	5	5	5	5	5	2	4
μ_2^1	μ_2^2	C	C_U	B_L	B_U	P_1^r	P_2^r	R_1	η_2	F_0	f_b	f_0^c
4	8	8	16	8	16	8%	5%	12	6.5	10^2	0.5	0.5
f_1^c	H_β^1	H_β^2	T_0	T_γ	δ_1	δ_2	N	V	ε_1	ε_2	ω_1	ω_2
1	90	10^2	10^3	0.95	10	10	10^6	10^3	0.1%	0.1%	500	500

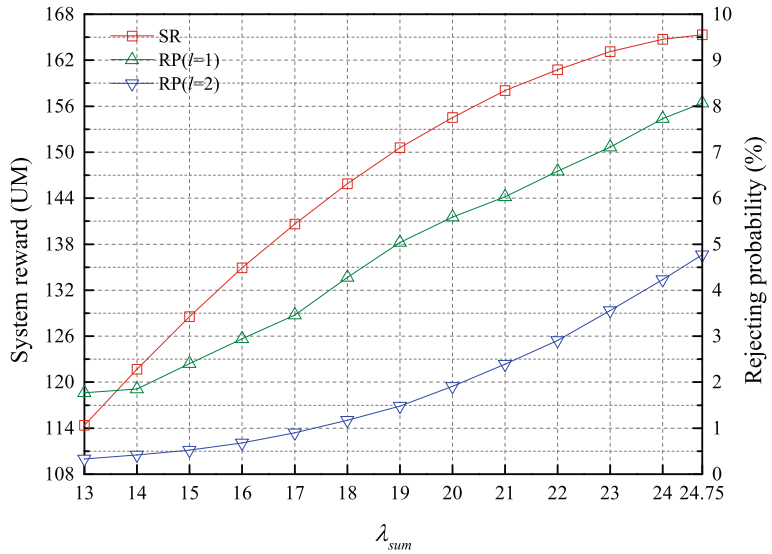

Fig. 6 SR and RPs under different task request arrival rates. This figure shows the evaluation results under different task request arrival rates

rapidly. When λ_{sum} becomes large, many task requests are rejected because of the heavy ASO load. With the help of the RL-based policy algorithm, the system model adaptively adjusts the admission policy depending on the arrival rates while satisfying the QoS, making the SR increase and the QoS requirement satisfied. The average increment rates of RP($l = 1$) and RP($l = 2$) are 0.54% and 0.38%, respectively, showing that RP($l = 1$) increases faster than RP($l = 2$). This is because the income from accepting a 2-level task is larger than the income from accepting a 1-level task, and thus, more 1-level task requests are rejected.

Figures 7 and 8 show the SR and RPs under different arrival rates of the 1-level and 2-level task request, respectively. In the experiment of Fig. 7, arrival rates of the 1-level task request are set to be equal, and arrival rates of the 2-level task request are set as the default simulation parameters, that is, $\lambda_1^1 = \lambda_1^2 = \lambda_{l=1}$, $\lambda_2^1 = \lambda_2^2 = 5$. From Fig. 7, it can be observed that the SR first increases and then decreases with the increasing $\lambda_{l=1}$. At the same time,

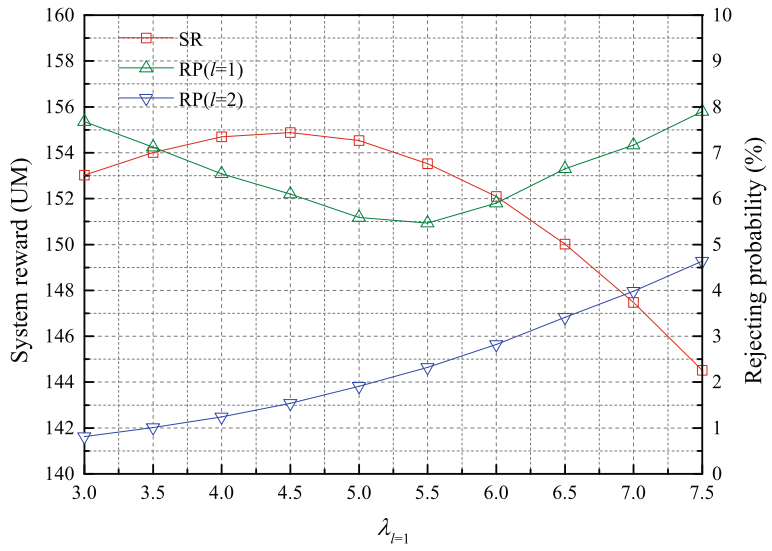

Fig. 7 SR and RPs under different arrival rates of the 1-level task request. This figure shows the evaluation results under different arrival rates of the 1-level task request

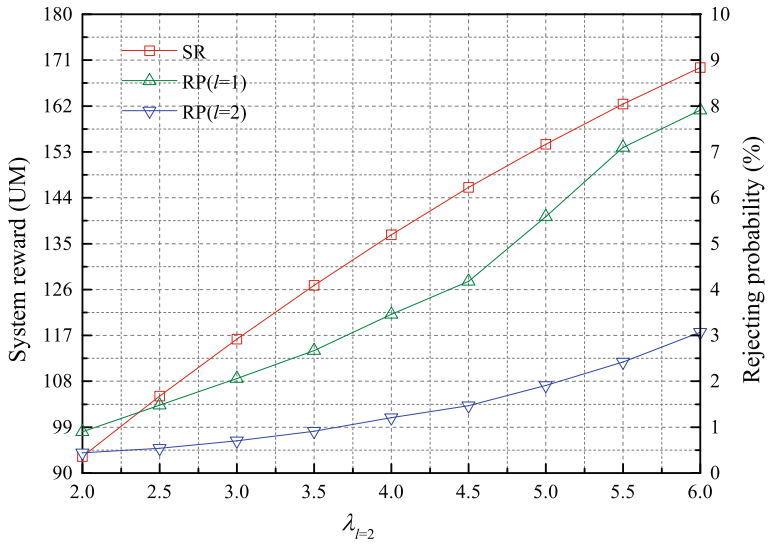

Fig. 8 SR and RPs under different arrival rates of the 2-level task request. This figure shows the evaluation results under different arrival rates of the 2-level task request. Figures 7 and 8 show the evaluation results under different arrival rates of the *l*-level task request

RP($l = 1$) first decreases and then increases. When $\lambda_{l=1}$ is small, with the increasing $\lambda_{l=1}$, RP($l = 1$) decreases, and more 1-level task requests are accepted to increase the SR. At the same time, a small $\lambda_{l=1}$ allows fewer 2-level task requests are rejected, making RP($l = 2$) increase slowly. When $\lambda_{l=1}$ is large, with the increasing $\lambda_{l=1}$, RP($l = 1$) increases, and more 1-level task requests are rejected to balance the ASO load. Also, a large $\lambda_{l=1}$ leads to the rapid increment of RP($l = 2$).

In the experiment of Fig. 8, arrival rates of the 2-level task request are set to be equal, and the arrival rates of the 1-level task request are set as the default simulation parameters, that is, $\lambda_1^1 = \lambda_1^2 = 5$, $\lambda_2^1 = \lambda_2^2 = \lambda_{l=2}$. From Fig. 8, it can be observed that the SR, RP($l = 1$), and RP($l = 2$) increase with the increasing $\lambda_{l=2}$. The average increment rates of RP($l = 2$) are 0.85% and 0.66% in Figs. 7 and 8, respectively, which means that RP($l = 2$) increases faster in Fig. 7. The reason is that the increasing 1-level tasks occupy too much resources, and more 2-level task requests are rejected to balance the ASO load.

Figures 9 and 10 show the SR and RPs under different arrival rates of the 1-class and 2-class task request,

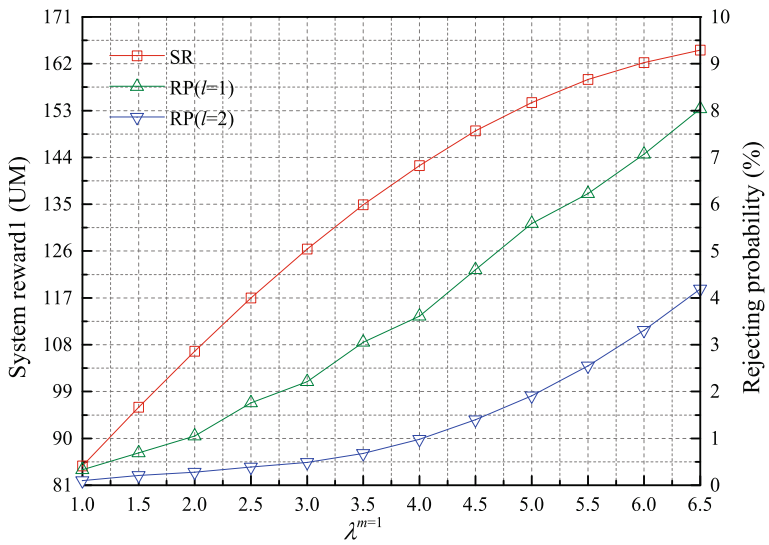

Fig. 9 SR and RPs under different arrival rates of the 1-class task request. This figure shows the evaluation results under different arrival rates of the 1-class task request

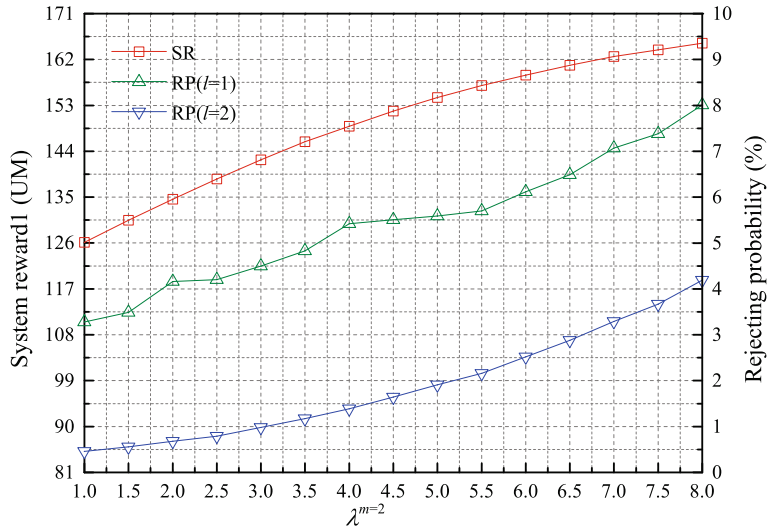

Fig. 10 SR and RPs under different arrival rates of the 2-class task request. This figure shows the evaluation results under different arrival rates of the 2-class task request. Figures 9 and 10 show the evaluation results under different arrival rates of the *m*-class task request

respectively. In the experiment of Fig. 9, arrival rates of the 1-class task request are set to be equal, and arrival rates of the 2-class task request are set as the default simulation parameters, that is, $\lambda_1^1 = \lambda_2^1 = \lambda^{m=1}$, $\lambda_1^2 = \lambda_2^2 = 5$. In the experiment of Fig. 10, arrival rates of the 2-class task request are set to be equal, and the arrival rates of the 1-class task request are set as the default simulation parameters, that is, $\lambda_1^1 = \lambda_2^1 = 5$, $\lambda_1^2 = \lambda_2^2 = \lambda^{m=2}$. From Figs. 9 and 10, it can be observed that the SR, RP($l = 1$), and RP($l = 2$) increase with the increasing $\lambda^{m=1}$ and $\lambda^{m=2}$. In Fig. 9, the average increment rates of the SR, RP($l = 1$), and RP($l = 2$) are 14.53UM, 1.40%, and 0.74%, respectively. In Fig. 10, the average increment rates of the SR, RP($l = 1$), and RP($l = 2$) are 5.58UM, 0.68%, and 0.53%, respectively. The income from accepting a l-level and m-class task request is $R_l \frac{1}{\mu_l^m}$, and $\mu_l^1 < \mu_l^2$, which indicates that accepting a 1-class task request results in more income. As reflected in the SR increment, the average SR increment rate in Fig. 9 is larger than that in Fig. 10, which shows that the SR increases faster with the increasing $\lambda^{m=1}$. The mean occupation time of a l-level and m-class task is $1 / \mu_l^m$, and $\mu_l^1 < \mu_l^2$, which indicates that the 1-class task takes more ASO resources. Therefore, more task requests are rejected with the increasing $\lambda^{m=1}$. As reflected in the increments of RP($l = 1$) and RP($l = 2$), the average increment rates of RP($l = 1$) and RP($l = 2$) in Fig. 9 are larger than those in Fig. 10, which shows that RP($l = 1$) and RP($l = 2$) increase faster with the increasing $\lambda^{m=1}$.

4.1.2 Impacts of the resources

Figure 11 shows the impacts of the resources on the SR and RPs. A larger B_L and C make resources ample and allow the ASO to provide more resources, which leads to fewer penalties for radio resource shortages, fewer costs for extended computing resources, and more task request acceptances. As reflected in Fig. 11, the SRs (SR($\eta_2 = 6.5$) and SR($\eta_2 = 4.5$)) increase, and the RPs (RP($l = 1, \eta_2 = 6.5$), RP($l = 2, \eta_2 = 6.5$), RP($l = 1, \eta_2 = 4.5$), and RP($l = 2, \eta_2 = 4.5$)) decrease with the increasing B_L and C. η_2 is the ratio of R_2 and R_1, and the larger η_2 means more income from accepting a 2-level task request. SR($\eta_2 = 6.5$) is larger than SR($\eta_2 = 4.5$) for this reason. For the same reason, more 1-level task requests are rejected to accept enough 2-level task requests to optimize the SR, making RP($l = 1, \eta_2 = 6.5$) larger than RP($l = 1, \eta_2 = 4.5$), and RP($l = 2, \eta_2 = 6.5$) smaller than RP($l = 2, \eta_2 = 4.5$). In addition, it can be observed that RP($l = 2, \eta_2 = 6.5$) and RP($l = 2, \eta_2 = 4.5$) first decrease and then keep stable. At the same time, the average decrease rates of RP($l = 1, \eta_2 = 6.5$) are 0.66% and 0.18% when $6 \le B_L = C \le 10$ and $10 \le B_L = C \le 14$, respectively. The average decrease rates of RP($l = 1, \eta_2 = 4.5$) are 0.99% and 0.38% when $6 \le B_L = C \le 10$ and $10 \le B_L = C \le 14$, respectively. This shows that the RPs decrease slowly when the resources are ample. The reason is when the resources are ample, the resources will no longer be the main factor limiting the task request acceptances, but the resource occupation cost.

Figure 12 shows the SR and RPs under different radio resource variation rates. It can be observed that the SRs (SR($F_0 = 100$) and SR($F_0 = 175$)) decrease with the increasing λ_o. The radio resources become more unstable with the increasing λ_o, and more penalties for radio resource shortages are generated. The large λ_o results in the large possibility of radio resource variations during

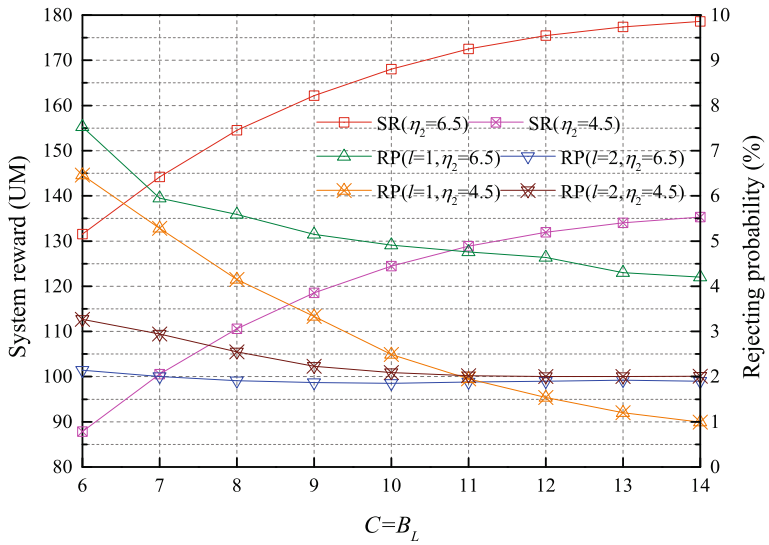

Fig. 11 SR and RPs under different lower bounds of resources. This figure shows the impact of the lower bounds of resources on the evaluation results

the task execution, which leads to more punitive cost. For example, if the radio resources are enough and stable during a period, there will be no penalty during this period. In the same period, if λ_o is large, the radio resources are unstable and easily become less than required, which leads to the penalty. To eliminate the cost caused by the penalties, more task requests are accepted, which is reflected in the decreasing trend of RPs (RP($l = 1, F_0 = 100$), RP($l = 2, F_0 = 100$), RP($l = 1, F_0 = 175$), and RP($l = 2, F_0 = 175$)). F_0 is the penalty coefficient, and a larger F_0 leads to more punitive cost. Therefore, SR($F_0 = 100$) is larger than SR($F_0 = 175$). From Fig. 12, it can be observed that RP($l = 1, F_0 = 175$) and RP($l = 2, F_0 = 175$) are both larger than RP($l = 1, F_0 = 100$) and RP($l = 2, F_0 = 100$) when $\lambda_o \geq 4$. The reason is that, when F_0 is large, more penalties are generated because of radio resource variations, and more task requests are rejected. From Figs. 11 and 12, it can be concluded that stable and ample resources are crucial to improve the SR and reduce the RPs.

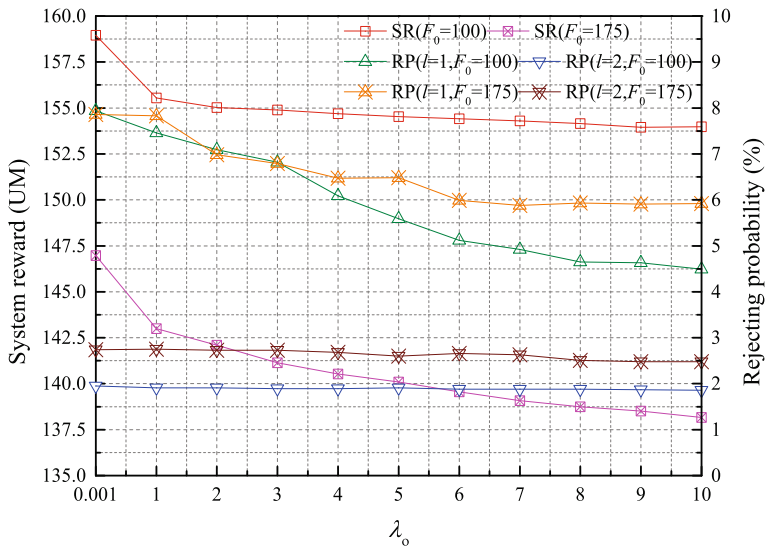

Fig. 12 SR and RPs under different radio resource variation rates. This figure shows the impacts of the radio resource variation rate on the evaluation results

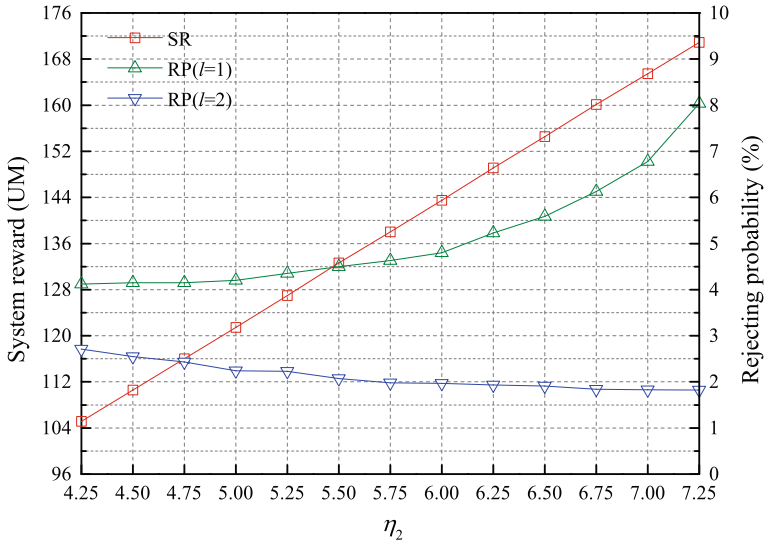

Fig. 13 SR and RPs under different η_2. This figure shows the impacts of the income on the evaluation results, in which η_2 represents the ratio of R_2 and R_1

4.1.3 Impacts of the income and cost

Figure 13 shows the impacts of the income on the SR and RPs, in which η_2 represents the ratio of R_2 and R_1. It can be observed that the SR increases with the increasing η_2, and the average increment rates of the SR is 21.93UM. The income from accepting a 2-level and m-class task request is $R_2 \frac{1}{\mu_2^m} = \eta_2 R_1 \frac{1}{\mu_2^m}$, and a larger η_2 results in more income from accepting a 2-level task request. Therefore, the SR increases rapidly with the increasing η_2. Correspondingly, it can be observed that RP($l = 1$) increases, and RP($l = 2$) decreases with the increasing η_2. With the increasing η_2, more 1-level task requests are rejected to accept more 2-level task requests, which have more income.

Figure 14 shows the SR and RPs under different penalty coefficients. It can be seen that the SR decreases with the increasing F_0. F_0 represents the penalty coefficient for the radio resource shortage, and the system cost increases when F_0 becomes large. On the other hand, as F_0 increases, the ASO rejects more task requests to eliminate the penalty caused by the radio resource shortage, and thus, the RPs increase with the increasing F_0. Both of these two factors reduce the system reward. The incre-

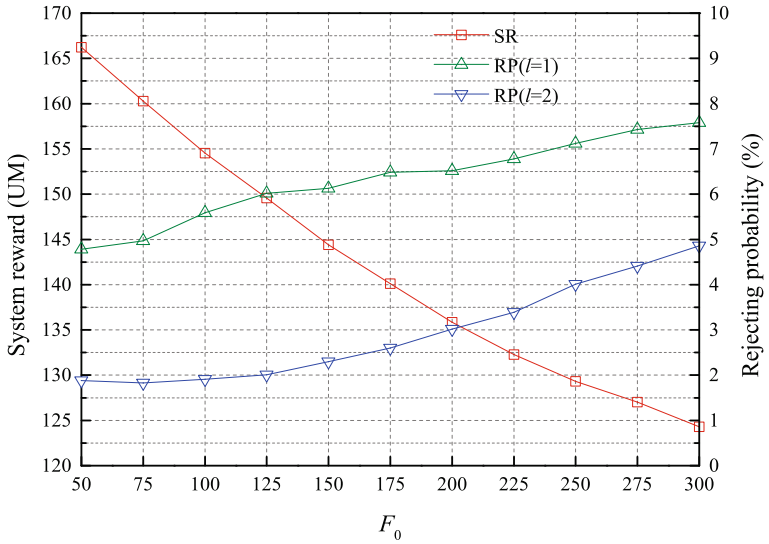

Fig. 14 SR and RPs under different F_0. This figure shows the impacts of the penalty coefficient(F_0) on the evaluation results

ment of RP($l = 2$) (2.98%) is larger than that of RP($l = 1$) (2.8%) when F_0 increases from 50 to 300, which shows that RP($l = 2$) increases faster. This is because that the 2-level task occupies more radio resources, and a large F_0 has more impact on it.

Figures 15 and 16 show the SR and RPs under different resource occupation cost coefficients. In the experiment of Fig. 15, three occupation cost coefficients, f_0^c, f_1^c, and f_b are set as $f_0^c = f_1^c = f_b = f$. With the increasing f, the computing and radio resource occupation cost increases. As reflected in Fig. 15, the SRs (SR($\eta_2 = 6.5$) and SR($\eta_2 = 4.5$)) decrease with the increasing f. The RPs (RP($l = 1, \eta_2 = 6.5$), RP($l = 2, \eta_2 = 6.5$), RP($l = 1, \eta_2 = 4.5$) and RP($l = 2, \eta_2 = 4.5$)) first keep relatively stable and then increase with the increasing f. To combat the increasing resource occupation cost, the ASO reduces the acceptance of task requests so that less resources are allocated. It can be observed that RP($l = 2, \eta_2 = 6.5$) increases slightly. This is because $\eta_2 = 6.5$ is large so that the occupation cost can be eliminated by the income from accepting 2-level task requests, and fewer 2-level task requests are rejected, which is also confirmed by the obvious increment of RP($l = 2, \eta_2 = 4.5$). It can be seen that when η_2 decreases from 6.5 to 4.5, RP($l = 1, \eta_2 = 6.5$) is larger than RP($l = 1, \eta_2 = 4.5$), and RP($l = 2, \eta_2 = 6.5$) is smaller than RP($l = 2, \eta_2 = 4.5$). When η_2 decreases, the income from accepting a 2-level task requests decreases while the income from accepting a 1-level task request increases relatively. Therefore, fewer 1-level task requests are rejected, and more 2-level task requests are rejected.

Figure 16 shows the SR and RPs under different f_1^c, which represents the cost coefficient of occupying the dynamically extended computing resources. From Fig. 16, it can be seen that the SRs (SR($\eta_2 = 6.5$) and SR($\eta_2 = 4.5$)) decrease, and the RPs (RP($l = 1, \eta_2 = 6.5$), RP($l = 2, \eta_2 = 6.5$), RP($l = 1, \eta_2 = 4.5$), and RP($l = 2, \eta_2 = 4.5$)) increase with the increasing f_1^c. Dynamically extended computing resources are more expensive, and the large f_1^c leads to more occupation cost. Therefore, the ASO rejects more task requests to reduce the possibility of extending the computing resources. Similar to Fig. 15, when $\eta_2 = 6.5$, fewer 2-level task requests are rejected for the reason that the occupation cost can be eliminated by the income from accepting 2-level task requests. When η_2 decreases from 6.5 to 4.5, the income from accepting 2-level task requests decreases while the income from accepting 1-level task requests increases relatively. Therefore, the ASO reduces the rejection of 1-level task requests and improves the rejection of 2-level task requests, which is reflected in Fig. 16 that RP($l = 1, \eta_2 = 6.5$) is larger than RP($l = 1, \eta_2 = 4.5$), and RP($l = 2, \eta_2 = 6.5$) is smaller than RP($l = 2, \eta_2 = 4.5$).

4.2 Performance comparisons of the policy algorithms

In this section, performance of the admission control policy algorithms is compared, and four policy algorithms are evaluated.

(1) RACPA, the random admission control policy algorithm, which accepts and rejects task requests randomly.
(2) TACPA, the threshold admission control policy algorithm, which rejects task requests when the ASO resource occupation ratio exceeds 95%; otherwise, it accepts all task requests.

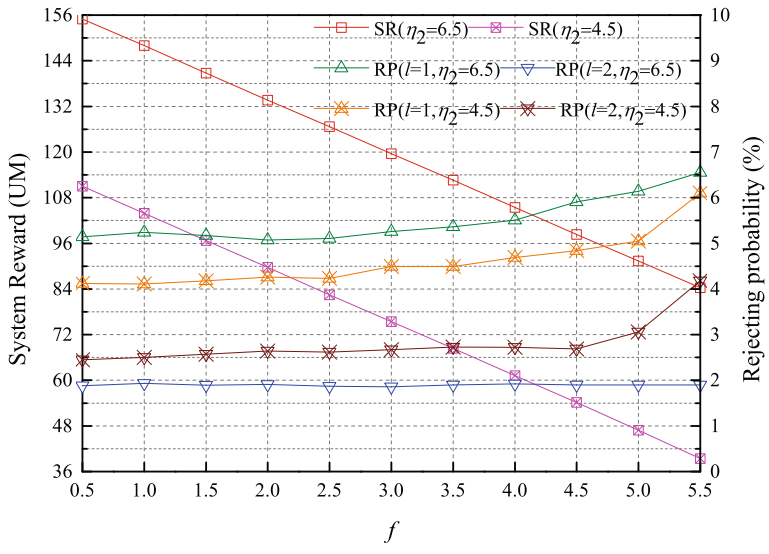

Fig. 15 SR and RPs under different f. This figure shows the impacts of the resource occupation cost coefficient(f) on the evaluation results

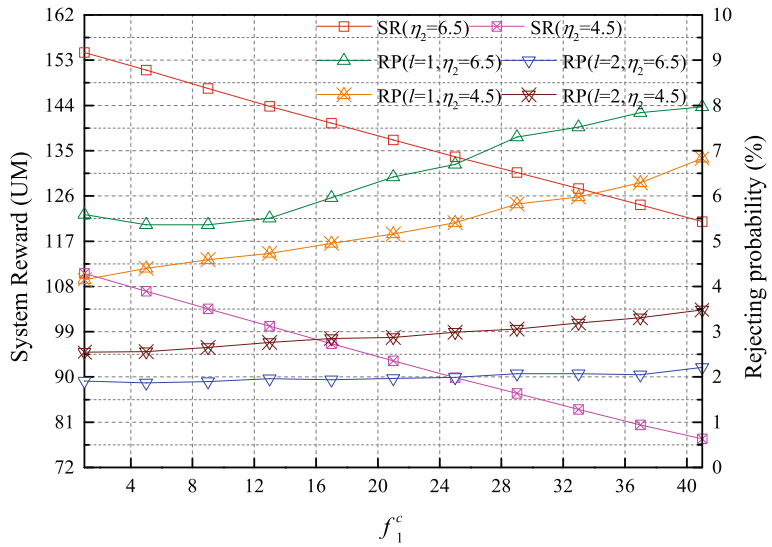

Fig. 16 SR and RPs under different f_1^c. This figure shows the impacts of the cost coefficient (f_1^c) of occupying the dynamically extended computing resources on the evaluation results

(3) GACPA, the greedy admission control policy algorithm, which takes the action that leads to more reward when receiving a task request.
(4) RLACPA, the proposed RL-based admission control policy algorithm.

The performance of the proposed policy algorithms and other commonly used policy algorithms is compared. The scenario parameters are shown in Table 3, and other parameters are the default simulation parameters listed in Table 2. The SRs in different scenarios are shown in Table 4, and their box-plot is shown in Fig. 17, which helps to visualize the data from Table 4. Table 5 shows the RPs in different scenarios.

As shown in Tables 4 and 5, SRs of RACPA are the smallest, and RPs of RACPA are the largest. The reason is that RACPA does not consider the system condition when making admission control decisions and accepts or rejects task requests randomly without optimizing the SR. In this experiment, RACPA generates the random policy evenly, and thus, its RPs are about 50%. TACPA is a commonly used policy algorithm and makes admission control decisions based on the system resource occupation ratio. TACPA does not optimize the SR and RPs from a long-term perspective. Therefore, TACPA cannot obtain the optimal SR while satisfying the QoS requirement. SRs of TACPA are smaller than those of RLACPA. In S2, S3, and S5, RPs of TACPA satisfy the QoS requirement. In S1, S4, S6–S10, although RP($l = 1$)s of TACPA satisfy the QoS requirement, RP($l = 2$)s exceed 37.80%, 41.67%, 59.60%, 60.0%, 54.0%, 67.20%, and 66.0% of P_2^rs, respectively. GACPA is another commonly used policy

Table 3 Scenario parameters

Scenario	Parameter										
	λ_1^1	λ_1^2	λ_2^1	λ_2^2	λ_o	η_2	C	B_L	F_0	P_1^r (%)	P_2^r (%)
S1	6	6	6	6	0.001	6.5	8	8	100	10	5
S2	4	4	6	6	2	7.5	6	6	100	8	5
S3	6	4	6	4	1	5.5	7	7	150	8	5
S4	6	4	6	4	5	6.5	10	10	100	5	3
S5	7	7	3	3	2	7	8	8	125	5	3
S6	7	7	5.5	5.5	4	8	8	8	75	10	5
S7	7	7	5.5	5.5	10	9	6	6	50	8	5
S8	7	4.5	7	4.5	3	6.5	8	8	100	10	5
S9	8.5	8.5	4.25	4.25	2.5	10	12	12	150	10	5
S10	6.25	7.25	4.25	5.25	4	6.5	7	7	50	5	3

Table 4 SR (UM) in different scenarios

Scenario	Policy algorithm			
	RACPA	TACPA	GACPA	RLACPA
S1	106.64	160.40	154.35	171.30
S2	86.26	125.00	124.21	177.01
S3	80.56	96.63	95.92	111.47
S4	73.46	128.77	127.54	129.08
S5	73.75	116.78	115.35	123.01
S6	122.76	195.15	194.67	200.51
S7	132.47	209.57	211.07	214.88
S8	109.28	161.34	158.39	166.36
S9	128.70	227.40	226.58	230.43
S10	90.46	150.93	151.10	153.18

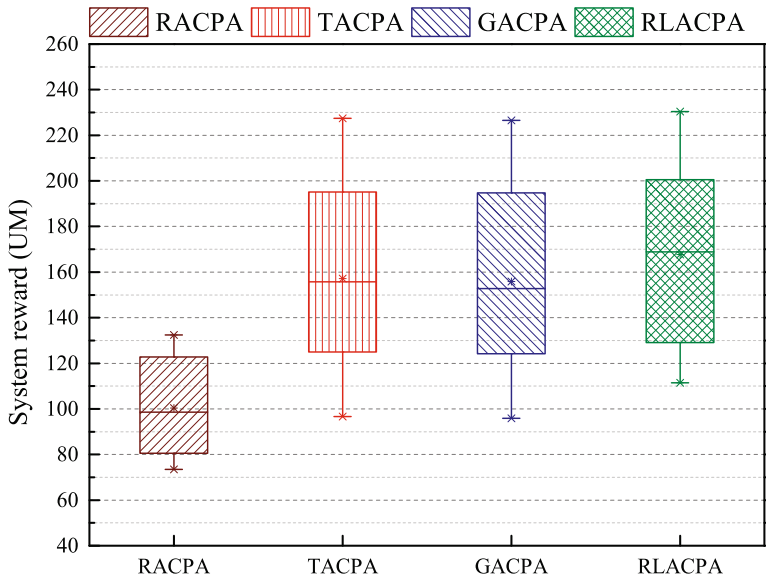

Fig. 17 SR in different scenarios. This figure is the box-plot of Table 4 and helps to visualize the data from Table 4

algorithm and makes admission control decisions greedily. GACPA selects the optimal action for each step but does not optimize the SR and RPs from a global perspective. Therefore, GACPA cannot obtain the optimal SR while satisfying the QoS requirement either. In S1–S5, RPs of GACPA satisfy the QoS requirement. In S6–S10, although RP($l = 1$)s of TACPA satisfy the QoS requirement, RP($l = 2$)s exceed 10.20%, 10.20%, 6.60%, 13.60%, and 7.0% of P_2^rs, respectively. The average range exceeding P_2^r of GACPA (9.52%) is noticeably smaller than that of TACPA (55.18%). As shown in Table 4, SRs of RLACPA are larger than those of other algorithms. The average relative difference between RACPA, TACPA, GACPA, and RLACPA are 67.29%, 8.06%, and 9.05%, respectively. As shown in Table 5, in S2, S4–S6, and S8–S10, RPs of RLACPA can meet the QoS requirement. In S1, S3, and S7, RP($l = 1$)s of RLACPA are slightly larger than P_1^rs with the difference of 0.05%, 0.06%, and 0.03%, respectively, which meets the accuracy requirement of 0.1%. As explained in Fig. 6, the reason is that RLACPA is iteratively searching for the optimal SR while satisfying the QoS requirement. It is considered to meet the requirement within the accuracy range, which is 0–0.1% in this paper.

5 Conclusion

In MCC, mobile users send task requests to the ASO according to the offloading policy provided by the offloading decision-making algorithms, and the ASO needs the task admission controller to decide whether to accept the

Table 5 RPs (%) in different scenarios

Scenario	Policy algorithm							
	RACPA		TACPA		GACPA		RLACPA	
	RP($l = 1$)	RP($l = 2$)	RP($l = 1$)	RP($l = 2$)	RP($l = 1$)	RP($l = 2$)	RP($l = 1$)	RP($l = 2$)
S1	49.97	50.05	2.90	6.89	1.86	4.69	10.05	4.28
S2	49.98	49.97	1.11	2.94	0.65	1.75	7.55	1.82
S3	49.99	49.99	1.66	4.25	3.19	2.65	8.06	4.17
S4	50.02	50.00	1.68	4.25	1.03	2.71	3.11	2.81
S5	49.98	50.04	0.99	2.66	0.55	1.53	4.69	1.57
S6	50.05	49.99	3.33	7.98	2.24	5.51	8.28	4.66
S7	50.02	49.97	3.31	8.00	2.22	5.51	8.03	4.81
S8	50.04	50.00	3.21	7.70	2.18	5.33	8.87	4.71
S9	50.01	49.97	3.45	8.36	2.29	5.68	8.17	4.74
S10	50.02	49.94	1.97	4.98	1.23	3.21	4.80	2.97

task request. The features of the task admission control problem in MCC are summarized as three points: (a) two-dimensional resources, (b) uncertainty, and (c) incomplete information. Considering these three features, a SMDP-based task admission control model, which considers radio resource variations, computing, and radio resources, is established. Also, a RL-based policy algorithm, which develops the admission policy through system simulations without the complete system information, is proposed to develop the admission policy. The established system model and proposed policy algorithm can be extended to more general admission control problems with one or more of the above features. Experimental results show that the SMDP-based task admission control model adaptively adjust the admission policy to accept or reject different levels and classes of tasks according to the ASO load, available radio resources and event type. The proposed RL-based policy algorithm outperforms the existing policy algorithms. The experimental results also show that stable and ample radio resources improve the ASO performance.

As mentioned above, wireless networks have a serious impact on MCC. In the current version of the problem, we only consider one type of radio resources. The concurrent multipath transfer (CMT) technology can use multiple physical wireless interfaces to transfer data in MCC to combat the challenge that wireless links have limited bandwidth and low robustness. Therefore, in the future, we will study the admission control problem with the consideration of CMT.

Abbreviations

MD: Mobile device; MCC: Mobile cloud computing; ASO: Application service operator; QoS: Quality of service; SMDP: Semi-Markov decision process; RL: Reinforcement learning; AR: Augmented reality; VR: Virtual reality; SLA: Service-level agreement; SR: System reward; RP: Rejecting probability; UM: Unit money; RACPA: The random admission control policy algorithm, which accepts and rejects task requests randomly; TACPA: The threshold admission control policy algorithm, which rejects task requests when the ASO resource occupation ratio exceeds 95%; otherwise, it accepts all task requests; GACPA: The greedy admission control policy algorithm, which takes the action that leads to more reward when receiving a task request; RLACPA: The proposed reinforcement learning-based admission control policy algorithm

Acknowledgements

The research presented in this paper was supported by Education Department of Shaanxi Province, Science and Technology Department of Shaanxi Province, and Xi'an University of Posts and Telecommunications.

Authors' contributions

XJ established the system model and designed the RL-based policy algorithm. WH contributed to the design of the experiment and helped to review the related work. ZW contributed to the manuscript revision and the expansion of literature review. All authors commented on the work and approved the final manuscript.

Authors' information

XJ received the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2018. Now he is a lecturer at School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His research interests include mobile devices and mobile cloud computing.

WH received the Ph.D. degree from Xidian University, Xi'an, China, in 2018. Now he is a lecturer at School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His research interests include machine learning, deep learning, and PolSAR image processing.

ZW received the Ph.D. degree from Beijing Institute of Technology, Beijing, China, in 2000. Now he is a professor at School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His current research interests include affective computing, big data processing and application, and embedded intelligent perception.

Funding

This work was supported by the Special Scientific Research Program of Education Department of Shaanxi Province (No. 19JK0806), the Key Research and Development Program of Shaanxi Province (No. 2019ZDLGY07-08), the Young Teachers Research Foundation of Xi'an University of Posts and Telecommunications, and the Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

Availability of data and materials

The data and material used to support the findings of this study are available from the corresponding author upon request.

Competing interests

The authors declare that they have no competing interests.

Received: 29 March 2020 Accepted: 1 October 2020
Published online: 27 October 2020

References

1. Cisco visual networking index: forecast and trends. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. Accessed 27 Nov 2018
2. Internet trend report 2019. https://techcrunch.com/2019/06/11/internet-trends-report-2019/. Accessed 12 June 2019
3. H. Zhou, X. Chen, S. He, J. Chen, J. Wu, DRAIM: a novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading. IEEE J. Sel. Areas Commun. **38**(4), 711–722 (2020)
4. J. Kwak, Y. Kim, J. Lee, S. Chong, DREAM: dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J. Sel. Areas Commun. **33**(12), 2510–2523 (2015)
5. Y. Cao, T. Jiang, O. Kaiwartya, H. Sun, H. Zhou, R. Wang, Toward pre-empted EV charging recommendation through V2V-based reservation system. IEEE Trans. Syst. Man Cybern. Syst., 10–110920192917149 (2020). https://doi.org/10.1109/TSMC.2019.2917149
6. Energy technology perspectives 2017. https://www.iea.org/etp/tracking2017/energystorage/. Accessed 16 July 2017
7. Y. Karaca, M. Moonis, Y. Zhang, C. Gezgez, Mobile cloud computing based stroke healthcare system. Int. J. Inf. Manage. **45**(4), 250–261 (2019)
8. M. Almasri, H. Alshareef, in *Proc. of 2019 ACM Int. Conf. Proc. Ser.* Mobile cloud-based e-payment systems in Saudi Arabia: a case study (ACM Press, New York, 2019), pp. 5–10
9. I. Arpaci, A hybrid modeling approach for predicting the educational use of mobile cloud computing services in higher education. Comput. Hum. Behav. **90**(1), 181–187 (2019)
10. Mobile cloud market by application-global opportunity analysis and industry forecast. https://www.researchandmarkets.com/reports/4333216/mobile-cloud-market-by-application-global. Accessed June 2017
11. G. Lewis, P. Lago, Architectural tactics for cyber-foraging: results of a systematic literature review. J. Syst. Softw. **107**(2015), 158–186 (2015)
12. J. Zheng, Y. Cai, Y. Wu, X. Shen, Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach. IEEE Trans. Mob. Comput. **18**(4), 771–786 (2019)
13. S. E. Mahmoodi, R. N. Uma, K. P. Subbalakshmi, Optimal joint scheduling and cloud offloading for mobile applications. IEEE Trans. Cloud Comput. **7**(2), 301–313 (2019)
14. R. Kumari, S. Kaushal, N. Chilamkurti, Energy conscious multi-site computation offloading for mobile cloud computing. Soft Comput. **22**(20), 6751–6764 (2018)
15. S. T. Hong, H. Kim, QoE-aware computation offloading to capture energy-latency-pricing tradeoff in mobile clouds. IEEE Trans. Mob. Comput. **18**(9), 2174–2189 (2019)

16. A. Hekmati, P. Teymoori, T. D. Todd, D. Zhao, G. Karakostas, in *Proc. of 2019 Int. Symp. Comput. Commun*. Optimal multi-decision mobile computation offloading with hard task deadlines (IEEE, Piscataway, 2019), pp. 1–8
17. K. Kumar, Y. Lu, Cloud computing for mobile users: can offloading computation save energy? Computer. **43**(4), 51–56 (2010)
18. R. Kaewpuang, D. Niyato, P. Wang, E. Hossain, A framework for cooperative resource management in mobile cloud computing. IEEE J. Sel. Areas Commun. **31**(12), 2685–2700 (2013)
19. A. Gosavi, Reinforcement learning for long-run average cost. Eur. J. Oper. Res. **155**(3), 654–674 (2004)
20. S. Guo, D. Wu, H. Zhang, D. Yuan, Resource modeling and scheduling for mobile edge computing: a service provider' perspective. IEEE Access. **6**(6), 35611–35623 (2018)
21. X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, R. Li, Energy-efficient admission of delay-sensitive tasks for mobile edge computing. IEEE Trans. Commun. **66**(6), 2603–2616 (2018)
22. Y. Liu, M. J. Lee, in *Proc. of 2015 IEEE Int. Symp. Serv.-Oriented Syst. Eng*. An adaptive resource allocation algorithm for partitioned services in mobile cloud computing (IEEE, Piscataway, 2015), pp. 209–215
23. Y. Liu, M. J. Lee, Y. Zheng, Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. IEEE. Trans. Mob. Comput. **15**(10), 2398–2410 (2016)
24. J. Wang, Y. Yue, R. Wang, M. Yu, J. Yu, H. Liu, X. Ying, R. Yu, in *Proc. of 2019 Int. Conf. Parallel Distrib. Syst*. Energy-efficient admission of delay-sensitive tasks for multi-mobile edge computing servers (IEEE, Piscataway, 2019), pp. 747–753
25. X. Chen, W. Li, S. Lu, Z. Zhou, X. Fu, Efficient resource allocation for on-demand mobile-edge cloud computing. IEEE Trans. Veh. Technol. **67**(9), 8769–8780 (2018)
26. Y. Qi, L. Tian, Y. Zhou, J. Yuan, Mobile edge computing-assisted admission control in vehicular networks: the convergence of communication and computation. IEEE Veh. Technol. Mag. **14**(1), 37–44 (2019)
27. H. Khojasteh, J. Misic, V. B. Misic, in *Proc. of 2015 Int. Wirel. Commun. Mob. Comput. Conf*. Task filtering as a task admission control policy in cloud server pools (IEEE, Piscataway, 2015), pp. 727–732
28. M. Y. Lyazidi, N. Aitsaadi, R. Langar, in *Proc. of 2016 GLOBECOM*. Resource allocation and admission control in OFDMA-based cloud-RAN (IEEE, Piscataway, 2016), pp. 1–6
29. M. Mirahsan, G. Senarath, H. Farmanbar, N. D. Dao, H. Yanikomeroglu, Admission control of wireless virtual networks in HetHetNets. IEEE Trans. Veh. Technol. **67**(6), 4565–4576 (2018)
30. J. Dromard, L. Khoukhi, R. Khatoun, Towards combining admission control and link scheduling in wireless mesh networks. Telecommun. Syst. **66**(1), 9–54 (2017)
31. X. Zhang, D. Li, W. W. Li, W. Zhao, An optimal dynamic admission control policy and upper bound analysis in wireless sensor networks. IEEE Access. **7**(4), 53314–53329 (2019)
32. F. Shang, D. Zhou, D. He, An admission control algorithm based on matching game and differentiated service in wireless mesh networks. Neural Comput. Appl. **32**(4), 2945–2962 (2020)
33. G. Baranwal, D. P. Vidyarthi, in *Proc. of 2014 IEEE Int. Adv. Comput. Conf*. A framework for selection of best cloud service provider using ranked voting method (IEEE, Piscataway, 2014), pp. 831–837
34. K. Liu, *Applied Markov Decision Processes*. (Tsinghua University Press, Beijing, 2004)
35. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simmulated annealing. Science. **220**(4598), 671–680 (1983)
36. F. J. Beutler, K. W. Ross, Time-average optimal constrained semi-Markov decision processes. Adv. Appl. Probab. **18**(2), 341–359 (1986)
37. H. Tong, T. X. Brown, Adaptive call admission control under quality of service constraints: a reinforcement learning solution. IEEE J. Sel. Areas Commun. **18**(2), 209–221 (2000)
38. A. W. Marshall, I. Olkin, A multivariate exponential distribution. J. Am. Stat. Assoc. **62**(317), 30–44 (1967)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com