Personality- and Value-aware Scheduling of User Requests in Cloud for Profit Maximization

Peijin Cong, Guo Xu, Junlong Zhou, Mingsong Chen, Tongquan Wei, and Meikang Qiu

Abstract—The main goal of a cloud provider is to make profits by providing services to users. Existing profit optimization strategies employ homogeneous user models in which user personality is ignored, resulting in fewer profits and particularly notably lower user satisfaction that in turn, leads to fewer users and reduced profits. In this paper, we propose efficient personality-aware request scheduling schemes to maximize the profit of the cloud provider under the constraint of user satisfaction. Specifically, we first model the service requests at the granularity of individual personality and propose a personalized user satisfaction prediction model based on questionnaires. Subsequently, we design a personality-guided integer linear programming (ILP)-based request scheduling algorithm to maximize the profit under the constraint of user satisfaction, which is followed by an approximate but lightweight value assessment and cross entropy (VACE)-based profit improvement scheme. The VACE-based scheme is especially tailored for applications with high scheduling resolution. Extensive simulation results show that our satisfaction prediction model can achieve the accuracy of up to 83%, and our profit optimization schemes can improve the profit by at least 3.96% as compared to the benchmarking methods while still obtaining a speedup of at least 1.68x.

Index Terms—Cloud computing, profit maximization, user request scheduling, user personality, multiserver system, customer satisfaction, present value, opportunity cost.

1 Introduction

Driven by virtual technology, cloud computing has become a popular IT business model that delivers various hardware and software resources to the cloud users on demand in a pay-as-you-go manner over the Internet [1]. Cloud computing has facilitated the development of cloud providers across the globe and has created a variety of computation models for pervasive and ubiquitous applications [2]. In the context of cloud computing, the cloud provider aims at maximizing its profit while users demand a high quality of cloud services. As more and more cloud providers are available to cloud users, maximizing profit while satisfying heterogeneous users in a competitive cloud market has become a huge challenge for cloud providers.

Profit improvement can be achieved by either increasing revenue or reducing cost. Effective pricing mechanisms are used to increase revenue, while diverse cloud resource management schemes are used to reduce costs. With regard to pricing, various models [3], [4], [5] have been proposed to better adapt to changing cloud service demand, which brings great benefits to the increase in cloud providers' revenues. However, existing pricing mechanisms are user-independent, thus fail to incorporate user characteristics or personality. User personality in psychology is used to describe the individual differences in the pattern of thinking, emotion, and behavior [6]. In the cloud service market, different users have different requirements for service price and quality of service (QoS). Service price and QoS are two key factors that users are most concerned about. Thus, existing pricing strategies that ignore user personality may result in degraded QoS for users and reduced revenue for cloud providers. With respect to cloud resource management, diverse energy-efficient cloud platform configuration mechanisms and request scheduling schemes [7], [8], [9], [10] have been designed to avoid energy waste incurred by over-provisioning of available resources or inefficient scheduling of requests. However, these works often adopt uniform service level agreements (SLAs) standard that assumes a unified personality or QoS for all customers, leading to increased cloud service platform maintenance costs. To further the profit maximization, we can model and exploit the heterogeneity of user personality via modeling user preference as a function of price and QoS.

In this paper, we investigate a personalized cloud pricing strategy and two service request scheduling mechanisms. Essentially, the cloud provider will greatly benefit by prioritizing resources towards critical users with higher QoS requirements as opposed to the users who could accept a lower QoS. For example, for users who prefer QoS to service price, the cloud provider could provide users with higher QoS and charge a higher service price to increase revenue. For users who prefer service price to QoS, the cloud provider could reduce infrastructure operation costs by using lower configurations of servers and charge users a lower service price. Thus, for users with different personalities, it is crucial to design personalized pricing strategy and judiciously schedule the service requests of personalized users such that the profit of the cloud provider is optimized while personalized user requirements are satisfied.

T. Wei is the corresponding author. Email: tqwei@cs.ecnu.edu.cn.

- *P. Cong, G. Xu, and T. Wei are with the School of Computer Science and Technology, East China Normal University, Shanghai 200062, China.*
- *J. Zhou is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.*
- *M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China.*
- *M. Qiu is with the Computer Science Department of Pace University.*

This work is partially supported by Shanghai Municipal Natural Science Foundation under Grant 16ZR140900.

The main contributions of this paper are listed as follows.

- We propose a personalized service request model to capture users' differences in terms of service execution requirements, service price, response time, and so on. We also develop a user satisfaction model to predict user satisfaction under different service price and QoS levels.
- We present an integer linear programming (ILP)-based task scheduling scheme to optimize the cloud provider's profit. Since the ILP-based algorithm is time-consuming, a lightweight value assessment and cross entropy (VACE)-based scheduling scheme is further proposed to obtain a near-optimal solution for profit maximization.
- Extensive simulation experiments have been conducted to validate the effectiveness of the proposed algorithms. Results show that our proposed schemes are superior to the benchmarking schemes in terms of profits by at least 3.96% while achieving at least 1.68 times of speedup.

The remainder of the paper is organized as follows. Section 2 reviews the related work on cloud profit optimization. Section 3 describes the cloud platform and models. Section 4 formulates the profit optimization problem and overviews the proposed schemes. Section 5 describes the personalized user satisfaction prediction, and Section 6 presents the personality-guided profit maximization schemes. The effectiveness of the proposed schemes is validated in Section 7. Section 8 discusses the possible directions of applying our schemes to real cloud providers, and Section 9 concludes the work and gives future research directions.

2 Related Work

In this section, we conduct a literature review on cloud profit optimization from perspectives of cloud pricing strategies, cloud resource management, and cloud customer satisfaction, respectively.

Most cloud providers adopt static pricing schemes [11]. Under this pricing strategy, service price is fixed, thus opportunities for increasing revenue by raising price under high market demand may be missing. Amazon EC2 [12] has launched a pricing mechanism called "spot pricing" that dynamically adjusts spot price for a virtual machine (VM) to reflect supply and demand in cloud market. However, Xu et al. [3] observe that the spot pricing may not work well to match supply and demand. Thus, they propose a market-driven dynamic pricing scheme to maximize revenue. Similarly, Mac et al. [13] propose a genetic algorithm for dynamic resource pricing by capturing the changes of cloud market. However, the complexity of cloud market greatly increases the computational cost of pricing algorithms and the market details are not readily available. Resource auction is a pricing method that does not need market details and allows users to compete and price freely. Zhang et al. [14] propose an auction mechanism based on smooth analysis and random reduction for dynamic VM pricing in a geographically distributed cloud data center. Also, Jiang et al. [15] design a novel bidding language and an online cloud auction framework to efficiently price cloud resources. However, users may influence the auction results and gain unfair advantages by malicious bidding or hiding their preferences for cloud resources. These behaviors destroy the auction experience of other normal users and significantly reduce the efficiency of auction.

Energy-efficient cloud resource provisioning schemes and user request scheduling mechanisms [16], [17], [18], [19] are important for cloud providers to manage cloud resources. Cao et al. [7] investigate the problem of optimal multiserver configurations (e.g., server size and server speed) to avoid over-provisioning of available resources for profit maximization in cloud computing. However, their multiserver platform is homogeneous. Liu et al. [8] explore cost-minimized heterogeneous multiserver configurations for cloud providers, in which the servers are different in terms of CPU cores, memory sizes, etc. These works adopt simple task scheduling discipline, i.e., First Come First Serve (FCFS), to serve the service requests submitted by users. Nevertheless, different service request may have different requirements for execution delay, that is, some may be delay-sensitive while some may be delay-insensitive. In this way, FCFS may be an inefficient scheduling scheme to serve requests with different delay requirements. Sundar et al. [20] propose a heuristic algorithm to schedule dependent tasks of an application under the constraints of communication delay and deadline to minimize the cost for cloud providers. Different from the above works, Yuan et al. [9] investigate cost-effective task scheduling problem in the hybrid cloud environment, and design a temporal task scheduling algorithm to schedule tasks under the constraint of execution delay to reduce energy consumption of cloud data center. These works save the cost of cloud providers via reasonable resource provisioning and efficient task scheduling. However, they ignore the role of customer satisfaction in cloud profit optimization. Customer satisfaction influences customer retention, thus further affects its revenue.

Low customer satisfaction will result in a decreased number of customers and reduced revenues of providers. For cloud providers, a naive solution to improving customer satisfaction is over-provisioning of available resources to meet the peak demand of users. However, this solution may result in low resource utilization during periods of low user demand. Thus, a more effective solution is to guarantee customer satisfaction of a limited number of users during peak workloads [21], [22]. For example, in [23], the spare cloud resources are provided to the second-class users with discounted prices at the cost of low QoS. Nonetheless, this method leads to unpredictable request delays, rejections, terminations, and price fluctuations for these users. In order to guarantee customer satisfaction of all users, Mei et al. [24] propose a double resource renting scheme that combines long-term renting and short-term renting to serve users. In this way, all service requests could be completed before their execution deadline. Instead of adopting double renting scheme, Cong et al. [4] develop a reward model for users and a penalty model for cloud providers to guarantee service quality and satisfy users. The above works do not take user personality into account. In essence, user personality has a great impact on cloud profit optimization since users with different personalities usually have different requirements for service price and QoS with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

respect to cloud services. In this way, these personalized users would achieve high customer satisfaction at different service prices and QoS levels. Thus, fully exploring the characteristics of user personality is critical for cloud providers to improve customer satisfaction.

In this paper, we propose a personality-aware user request scheduling scheme to maximize the profit of the cloud provider. In psychology, the personality of a user is described by Big Five personality traits [25]. Ten-Item Personality Inventory (TIPI) [26] is a prevalent personality measure method that includes 10-item to score each personality trait in Big Five personality traits (see Section 5 for details). Below, we first study the effect of user personality on its preference for service price and QoS by investigating user satisfaction under different service price and QoS levels. According to the research results, we establish personalized user model and cloud provider model, based on which a profit optimization problem is formulated. We then present a personality-based prediction mechanism to predict user satisfaction under different service price and QoS levels. Finally, we design an ILP-based optimal scheduling algorithm followed by an approximate but lightweight value assessment and cross entropy (VACE)-based profit improvement scheme to solve the profit optimization problem.

3 Cloud Computing Platform and Models

In this study, we consider a three-tire cloud computing architecture that contains users, a cloud provider, and a infrastructure provider. As shown in Figure 1, heterogeneous users submit their service requests to the cloud provider, receive the desired results from the cloud provider, and pay for the service based on service amount and service quality. The cloud provider handles users' service requests by renting resources (e.g., servers) from the infrastructure provider. These heterogeneous servers are modeled as a multiserver system, in which each server is characterized by a given supply voltage-frequency pair. A service request queue with infinite capacity is maintained by the multiserver system for waiting requests when all the servers are busy.

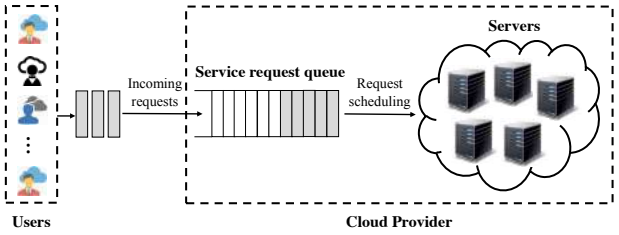


Figure 1: A cloud service platform containing multiple heterogeneous users and a cloud provider.

3.1 Cloud User Model

Users with different personalities have different preferences for service price and QoS, and the preferences further have an impact on user satisfaction with regard to services. Thus, in this section, we model users from perspective of service request model and user satisfaction model, respectively.

3.1.1 User service request model

We assume the arrival of service requests at the multiserver system follows a Poisson process. Once the users submit requests to the cloud provider, these requests are first placed in the request queue before they can be processed by any servers. We use a tuple $\tau_m : \{W_m, D_m^{des}, c_m^{des}, d_m, D_m\}$ to characterize a service request submitted by user m. In the tuple, W_m denotes the number of instructions in τ_m, and D_m^{des} represents the finish time of τ_m beyond which user satisfaction declines and charge rate decays. We denote D_m^{des} as the desired finish time of request τ_m. The c_m^{des} refers to the service price per instruction when request τ_m is finished before D_m^{des}, while d_m is the charge decay rate when request τ_m is finished later than D_m^{des}. D_m indicates the deadline of request τ_m. Let ft_m be the finish time of request τ_m, and $R(\tau_m, ft_m)$ be the fee charged by the cloud provider to request τ_m. Then, $R(\tau_m, ft_m)$ is given by

$$R(\tau_m, ft_m) = \begin{cases} W_m c_m^{des}, 0 < ft_m \le D_m^{des} \\ W_m c_m^{des} - d_m(ft_m - D_m^{des}), D_m^{des} < ft_m \le D_m \\ 0, ft_m > D_m \end{cases} \quad (1)$$

The service request model is illustrated in Figure 2. If the finish time ft_m of request τ_m is earlier than its desired finish time D_m^{des}, the cloud provider is deemed to process τ_m successfully with the required QoS and the user will pay the cloud provider the maximum fee c_m^{des}. If the finish time ft_m of request τ_m is later than its desired finish time D_m^{des} but earlier than its deadline D_m, it is believed that the cloud provider delivers an acceptable result for request τ_m and the user will pay the cloud provider based on charge decay rate d_m. However, if the finish time ft_m of request τ_m is later than its deadline D_m, the cloud provider delivers an unacceptable result for request τ_m and the user will not pay the cloud provider due to low QoS.

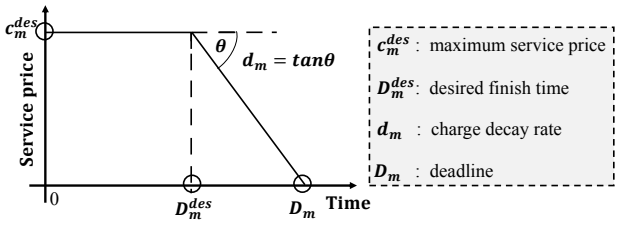


Figure 2: An illustration of service request charge model.

Existing pricing schemes only adopt a single function for different types of service requests, while in our personalized pricing model, each service request has its own charge function. That is to say, for request τ_m, it has its own service execution requirement W_m, desired finish time D_m^{des}, maximal fee charged by the cloud provider $W_m c_m^{des}$, charge decay rate d_m, and request deadline D_m. The model in Figure 2 can effectively capture the differences of individual users in terms of service price and QoS.

3.1.2 User satisfaction model

With the explosive growth of cloud providers, providing a good service experience to users with different personalities makes the cloud providers more competitive in the cloud market. For users, service price and QoS are two important factors that affect their experience. Users naturally hope to

get the best QoS at a low price. However, from the perspective of cloud providers, users' requirements for service price and QoS are difficult to meet at the same time since lower service price usually result in worse QoS and improving QoS will inevitably increase service price. Many studies aim to achieve a good trade-off between service price and QoS to improve user satisfaction [9], whereas these works ignore user personality, which also plays an important role in user satisfaction. For example, some users may want a low price, while others may need high QoS. In this section, we model user satisfaction based on user personality.

We use a latent variable l to measure a user's preference for service price and QoS. The latent variable l can be expressed as a weighted sum of service price and QoS, thus can be expressed as:

$$l = \xi * price + (1 - \xi) * qos, \tag{2}$$

where $price$ denotes service price and qos indicates QoS. In this study, QoS refers to the response time of a service request. The weight ξ is used to measure a user's preference between service price and QoS, and satisfies $0 < \xi < 1$. A higher weight ξ indicates more preference to $price$ while a lower weight ξ means more preference to qos.

We assume that user satisfaction score is a quantitative representation of latent variable l. Thus, user satisfaction score can be expressed as a linear function of l, i.e.,

$$\boldsymbol{s} = \alpha_1 + \alpha_2 * l, \tag{3}$$

where $\boldsymbol{s}$ represents user satisfaction, and α_1 and α_2 are two parameters that will be determined in Section 5.

Based on latent variable l and user satisfaction $\boldsymbol{s}$, next, we calculate the attribute values (i.e., d_m and D_m) of user request $\tau_m : \{W_m, D_m^{des}, c_m^{des}, d_m, D_m\}$ to capture user personality. Note that the value of W_m is set by the user, and the values of D_m^{des} and c_m^{des} can be obtained via questionnaires. In order to obtain the charge decay rate d_m of request τ_m, i.e., the user satisfaction attenuation rate caused by the degradation of QoS, we insert the service price c_m^{des} into Eq. (2), and substitute Eq. (2) into Eq. (3), then we have

$$\boldsymbol{s} = \alpha_1 + \alpha_2 * (\xi * c_m^{des} + (1 - \xi) * qos), \tag{4}$$

which can be further re-expressed as:

$$\boldsymbol{s} = \alpha_2 * \xi * qos + (\alpha_1 + \alpha_2 * (1 - \xi) * c_m^{des}), \tag{5}$$

where c_m^{des} is a known variable, α_1, α_2, and ξ can be obtained based on user satisfaction model. Eq. (5) is a first degree polynomial, in which the first part is composed of slope $\alpha_2 * \xi$ and variable qos, while the second part (i.e., $\alpha_1 + \alpha_2 * (1 - \xi) * c_m^{des}$) is a constant and referred to as the intercept. Thus, the charge decay rate d_m of request τ_m, i.e., the user satisfaction attenuation rate, could be set to $\alpha_2 * \xi$. The deadline D_m of request τ_m refers to the response time (i.e., qos) when the user satisfaction score becomes 0. Thus, substituting $\boldsymbol{s} = 0$ into Eq. (5), we have

$$D_m = qos = \frac{(\alpha_1 + \alpha_2 * (1 - \xi) * c_m^{des})}{\alpha_2 * \xi}. \tag{6}$$

3.2 Cloud Provider Model

In this section, we model the cloud provider's revenue and cost, in which revenue is the sum of charges of all users' service requests while the cost is the sum of the rental fee and the electricity bill.

3.2.1 Revenue

The cloud provider provides services to users and charge them for the completion of service requests. Since users have different personalities and the attributes of service requests are different, it is natural to charge users at the granularity of service request. We define the cloud provider's revenue as the sum of charges of all service requests. Thus, the total revenue of a cloud provider is expressed as:

$$Revenue = \sum_{m=1}^{M} R(\tau_m, ft_m), \tag{7}$$

where M is the number of requests during a time interval.

3.2.2 Cost

The cloud provider's cost includes two parts, i.e., the rental cost of resources and the utility cost of energy consumption.

Rental cost: In this paper, the cloud provider maintains a multiserver system that consists of N heterogeneous servers S. We use Φ to denote the server set, i.e., $\Phi = \{S_1, S_2, ..., S_N\}$. To maintain normal operations of the multiserver system, the cloud provider needs to rent hardware facilities from the infrastructure provider in advance. Let δ_n be the rental cost of S_n during a time interval, and $Rent_{cos}$ denote the total rental expenses of all servers during a time interval. Then, $Rent_{cos}$ can be calculated as:

$$Rent_{cos} = \sum_{n=1}^{N} \delta_n, \tag{8}$$

where N denotes the number of servers.

Energy cost: The power consumption P_{tol} of a server can be modeled as the sum of static power consumption P_{sta} and dynamic power consumption P_{dyn}, that is,

$$P_{tol} = P_{sta} + P_{dyn}. \tag{9}$$

P_{sta} is independent of switching activity and can be regarded as a server-dependent constant and P_{dyn} is related to charging and discharging of gates in the circuits. The dynamic power consumption of server S_n when executing requests at the supply voltage-frequency (v_n, f_n) pair can be expressed as [27]:

$$P_{dyn} = \mu_n \eta_n v_n^2 f_n, \tag{10}$$

where μ_n and η_n are the activity factor and the effective switching capacitance of server S_n, respectively. Let M_n be the number of service requests assigned to server S_n and W_m be the number of instructions of request τ_m. Then, the total energy consumption of server S_n during a time interval can be calculated as:

$$E_n = \sum_{m=1}^{M_n} (\mu_n \eta_n v_n^2 W_m + P_{sta}(n) \frac{W_m}{f_n}), \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

where $P_{sta}(n)$ represents the static power consumption of server S_n. Let $Elec_{cos}$ be the electricity cost of N servers during a time interval, then it can be given by

$$Elec_{cos} = e_c \cdot \sum_{n=1}^{N} E_n, \tag{12}$$

where e_c refers to the electricity price charged per unit of consumed energy.

4 Problem Definition and Overview of the Proposed Approach

In this section, we first define the problem of profit maximization under the constraint of user satisfaction, then briefly outline our proposed approach.

4.1 Problem Definition

Based on the revenue model and cost model, we define the cloud provider's profit as revenue minus costs, that is,

$$Profit = Revenue - Elec_{cos} - Rent_{cos}, \tag{13}$$

where $Revenue$, $Elec_{cos}$, and $Rent_{cos}$ are given in Eqs. (7), (8), and (12), respectively. Given users with different personalities, this paper aims to optimize the profit of the cloud provider under the constraint of user satisfaction. The optimization problem is thus defined as:

$$\text{maximize: } Profit, \tag{14}$$

$$\text{subject to: } \boldsymbol{s} \geq \boldsymbol{s_{th}}, \tag{15}$$

where $\boldsymbol{s}$ is given in Eq. (3). $\boldsymbol{s_{th}}$ is a constant that represents the user satisfaction to be guaranteed. Eq. (15) ensures that user satisfaction cannot be lower than $\boldsymbol{s_{th}}$.

4.2 Overview of the Proposed Approach

Figure 3 presents an overview of our proposed solution to the profit optimization problem given in Eq. (14). For cloud users, we first study the personality of a user using Big Five personality traits, in which each trait corresponds to a character. The score of each trait could be measured by TIPI, which is a questionnaire consisting of 10 items that inquiry questions about user personality. Based on user personality score, we then explore the effect of user personality on the preference for service price and QoS by investigating user satisfaction under different service price and QoS levels. Finally, we construct a personalized service request model and propose a user satisfaction model to predict QoS and service price users are satisfied with. For the cloud provider, based on personalized service request model and user satisfaction model, we build revenue model and cost model, respectively. Afterwards, we formulate the profit maximization problem for the cloud provider under the constraint of user satisfaction.

To solve the optimization problem, two efficient request scheduling schemes are designed, respectively. The first one is an optimal but time-consuming integer linear programming (ILP)-based algorithm while the second one is an approximate but lightweight value assessment and cross entropy (VACE)-based algorithm. In particular, the VACE-based algorithm adopts the concepts of present value and opportunity cost in economics to evaluate the values of service requests during the scheduling process, thus speeding up the optimization process. If an application demands high scheduling resolution (i.e., small scheduling interval), the VACE-based algorithm is an appropriate option to obtain a near-optimal solution with low time complexity. Otherwise, the ILP-based algorithm could be adopted to obtain an optimal solution at the cost of high time complexity.

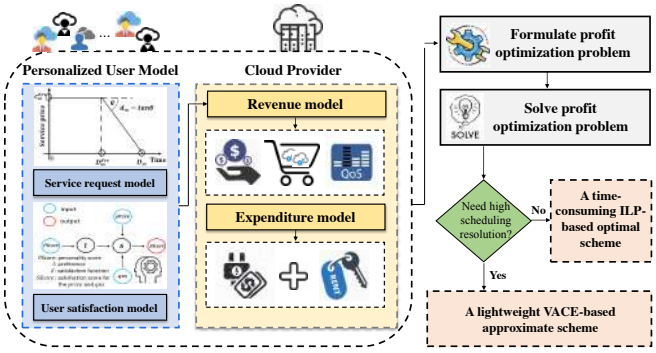


Figure 3: An overview of the proposed approach.

5 User Personality-Guided Satisfaction Prediction

In this section, we first explore the impact of a user's personality on his/her satisfaction with respect to the service. Subsequently, we develop a personality-guided user satisfaction prediction model to achieve prediction of user satisfaction under different service price and QoS levels.

5.1 The Impact of Personality on User Satisfaction

We have known that personality refers to individual differences in the pattern of thinking, emotion, and behavior [6]. These differences make people with different personalities react differently when faced with the same situation and have a great influence on people's choices. User personality is usually described by Big Five personality traits summarized in Table 1 [28]. Big Five personality traits is one of the most influential personality models in psychology and has been used in many research. The five personality traits included in the Big Five personality traits are generally applicable to all types of people and races, and will not change due to differences in geography, culture, and language. The five factors included in the Big Five personality traits are extraversion (E), agreeableness (A), conscientiousness (C), neuroticism (N), and openness (O), respectively.

In order to assess personality, a questionnaire with ten questions (i.e., TIPI) is commonly used [29], [30]. Using TIPI, a user's personality can be described by a personality vector of five feature scores, each ranging from 1 to 7 [29]. Based on these personality scores, we can explore the impact of personality on user satisfaction through a "Price-QoS-User Satisfaction" questionnaire. As reported in [30], the user satisfaction score is an integer and is generally set to be varied in the range from 1 (lowest satisfaction level) to 10 (highest satisfaction level), and according to the principle proposed in [30], we set the baseline and other levels of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

Table 1: Big Five personality traits and their description [28].

Trait	Description
Extraversion (E)	Enthusiasm, talkativeness, decisiveness, activity, risk and optimism
Agreeableness (A)	Trust, altruism, straightforwardness, compliance, modesty and empathy
Conscientiousness (C)	Seriousness, justice, organization, due diligence, diligence, self-discipline, caution and restraint
Neuroticism (N)	Balance anxiety, hostility, depression, self-awareness, impulsively and vulnerability
Openness (O)	Imagination, aesthetics, emotional enrichment, diversity, creation, and curiosity

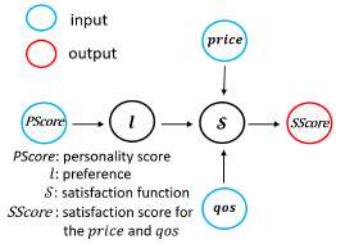


Figure 4: User satisfaction prediction model.

qos and $price$. In [30], the case with the best or worst performance is selected as the baseline and other cases can be represented using a percentage of the baseline. Following this, we select the shortest (best) response that a server can support as the baseline of qos, and set the value of baseline to 100%. We also investigate other qos levels that are 110%, 120%, 130%, 140%, and 150% of the baseline. Similarly, we choose the cloud provider's expected sale price (the best price for the cloud provider) as the baseline of $price$, set the value of baseline to 100%, and also consider other sale prices that are 95%, 90%, 85%, 80%, and 75% of the baseline. Combining 6 levels of qos with 6 levels of $price$, a total of 36 $price$-qos levels are obtained. Under each level of $price$-qos, we ask users to give the corresponding satisfaction score.

Based on the analysis of questionnaires, we found that the satisfaction scores of users with high "C" trait usually show a regular trend with the changes in service price and QoS, whereas users with high "A" trait can tolerate longer service waiting times when service price is low enough. This shows that user personality has a great influence on user satisfaction, i.e., users with different personalities usually achieve maximum satisfaction at different service prices and QoS levels. Thus, the cloud provider can utilize this characteristic to enable personalized service request scheduling.

5.2 Personality-Oriented User Satisfaction Prediction

In this section, we establish a personality-oriented user satisfaction prediction model to achieve prediction of user satisfaction with respect to a service request. As shown in Figure 4, the user satisfaction prediction model takes service price $price$, QoS qos, and a user's Big Five personality score $PScore$ as the inputs, and the satisfaction score $SScore$ for the given service price $price$ and service quality qos as the output. In order to construct the personality-oriented user satisfaction prediction model, we first establish the relationship between the user satisfaction function $\boldsymbol{s}$ and user preference l, then analyze the relationship between user preference l and personality score $PScore$.

Establish the relationship between satisfaction function $\boldsymbol{s}$ and preference l: The satisfaction function $\boldsymbol{s}$ has been defined in Eq. (3). Here, we calculate the values of α_1 and α_2 in Eq. (3). We use N_e and N_l to represent the number of participants in the questionnaire and the number of $price$-qos levels, respectively. For participant i ($i \in [1, N_e]$), his/her preference weight with respect to service price and QoS is denoted as ξ_i. For each $price$-qos level j ($j \in [1, N_l]$), the service price, QoS, preference, and satisfaction score of participant i are denoted by $price_{(i,j)}$, $qos_{(i,j)}$, $l_{(i,j)}$ and $SScore_{(i,j)}$, respectively. Then, the values of α_1 and α_2 can be obtained by solving the following mean square error equation

$$\min_{\alpha_1,\alpha_2,\xi_1,\ldots,\xi_{N_e}} = \sum_{i=1}^{N_e}\sum_{j=1}^{N_l}(SScore_{(i,j)} - \alpha_1 - \alpha_2 * l_{(i,j)})^2, \tag{16}$$

where

$$l_{(i,j)} = \xi_i * price_{(i,j)} + (1-\xi_i) * qos_{(i,j)}. \tag{17}$$

Since Eq. (16) has too many unknown variables, it cannot be solved directly. Therefore, we develop an iterative method to solve it. We first initialize preference weight ξ_i for each participant and calculate preference $l_{(i,j)}$ according to Eq. (2). Then, we substitute $l_{(i,j)}$ into Eq. (16) and derive α_1 and α_2 by solving the following linear regression

$$\min_{\alpha_1,\alpha_2} = \sum_{i=1}^{N_e}\sum_{j=1}^{N_l}(SScore_{(i,j)} - \alpha_1 - \alpha_2 * l_{(i,j)})^2. \tag{18}$$

After obtaining the values of α_1 and α_2, the scenario of only one participant is considered. Eq. (16) is thus converted to

$$\min_{\xi_i^{'}} = \sum_{j=1}^{N_l}(\frac{SScore_{(i,j)} - \alpha_1}{\alpha_2} - (1-\xi_i^{'}) * qos_{(i,j)} - \xi_i^{'} * price_{(i,j)})^2. \tag{19}$$

Eq. (19) can be solved by a linear regression solver. According to Eq. (19), we can obtain the updated preference weight $\xi_i^{'}$ of participant i. In particular, if the updated preference weight $\xi_i^{'}$ and the initial preference weight ξ_i for all participants i ($i \in [1, N_e]$) are close enough, the iteration process stops and we can obtain the preference weights of all participants. Note that α_1 and α_2 are used to establish the relationship between preference l and satisfaction function $\boldsymbol{s}$, while the weight ξ is used to establish the relationship between personality score $PScore$ and user preference l. Next, we establish the relationship between personality score $PScore$ and preference l.

Establish the relationship between personality score $PScore$ and preference l: The user preference l has been defined in Eq. (2). In Eq. (2), given $price$ and qos, the weight ξ determines the user preference l. To model the relationship between $PScore$ and l, we need to associate $PScore$ with ξ. For participant i ($i \in [1, N_e]$), we use $PScore_i = [PScore_{(i,E)}, PScore_{(i,A)}, PScore_{(i,C)}, PScore_{(i,ES)}, PScore_{(i,O)}]$ to indicate his/her personality score, each of which corresponds to a personality trait (i.e., E, A, C, ES, and O). For N_e participants, we use matrix

PS of $N_e \times 5$ to denote their personality scores, and vector PW of $N_e \times 1$ to denote their preference weights. Given PS and PW, a linear regression model is used to establish the relationship between PS and PW, i.e.,

$$\min_{Y} = \|PW - PS \times Y\|^2. \tag{20}$$

The vector $Y_{5\times 1}$ reflects the effects of scores of five personality traits on user preferences, and can be obtained by solving Eq. (20) using a linear regression solver. Given PS and Y, user preference weight PW can be inferred by

$$PW = PS \times Y. \tag{21}$$

Combining Eqs. (21) and (3), we can calculate user satisfaction score $SScore$ based on user personality score $PScore$, service price $price$ and service quality qos.

6 Personality-Guided Profit Maximization under User Satisfaction Constraint

In this section, we present an ILP-based optimal request scheduling solution followed by an value assessment- and cross entropy (VACE)-based approximate request scheduling scheme to maximize the cloud provider's profit.

6.1 ILP-based Profit Maximization Algorithm

Before introducing the ILP-based request scheduling algorithm, we first define the scheduling vector T as:

$$T \triangleq [T_1, T_2, ..., T_K], \tag{22}$$

where K is the scheduling horizon and T_k is the k^{th} time slot of the scheduling horizon. T_k can be months, days, hours, minutes, or even seconds, depending on the available pricing information and computing capability of the schedulers. In this study, the scheduling resolution refers to the scheduling interval, and a high scheduling resolution refers to a small scheduling interval.

Let $A_{n,m}$ denote the assignment of request τ_m on server S_n. If request τ_m is assigned to server S_n, $A_{n,m} = 1$, otherwise, $A_{n,m} = 0$. $B_{i,j}$ indicates the execution order of request τ_i and request τ_j. If request τ_i starts after request τ_j, $B_{i,j} = 1$, otherwise $B_{i,j} = 0$. $\boldsymbol{s}_m$ denotes the user satisfaction with respect to request τ_m. b_i and b_j represent the start time of request τ_i and request τ_j, respectively, and $b_{max(i,j)}$ denotes the maximum of b_i and b_j. $\mathcal{H}$ is a large constant. Two auxiliary variables $h_{i,j}$ and $g_{i,j}$ are introduced as the pseudo-linear constraints to facilitate the formulation.

Our optimization problem is described below: Given a request set Γ of independent requests τ_m defined by $\{W_m, D_m^{des}, c_m^{des}, d_m, D_m\}$ for $m = 1, 2, \cdots, M$, and a server set Φ of heterogeneous servers S_n with supply voltage-frequency pair (v_n, f_n) for $n = 1, 2, \cdots, N$, the goal is to find the optimal request-to-server assignment $A_{n,m}$ and the start execution time b_m of τ_m for maximizing the profit of the cloud provider under the user satisfaction constraint. Note that the cloud provider only adopts the long-term renting scheme since the rental price of long-term renting is much cheaper than that of short-term renting [24]. In such a case, we can safely assume that the number N of servers is constant in different scheduling time slots of a given scheduling horizon. This is because that the length of a scheduling horizon is far shorter than the duration of a renting contract. The request set Γ is varied in different scheduling time slots since there are arrivals of new requests in each time slot [4], [24]. The ILP formulation of the optimization problem for a scheduling time slot is given as:

$$\text{maximize: } Profit, \tag{23}$$

$$\text{subject to: } \sum_{n=1}^{N} A_{n,m} = 1, \tag{24}$$

$$b_m + \sum_{n=1}^{N} A_{n,m} \frac{W_m}{f_n} \leq D_m, \tag{25}$$

$$\frac{\sum_{m=1}^{M} \boldsymbol{s}_m}{M} \geq \boldsymbol{s_{th}}, \tag{26}$$

$$bmax_{(i,j)} = B_{j,i} \times b_i + B_{i,j} \times b_j, \tag{27}$$

$$h_{i,j} = \mathcal{H} \times (b_i - bmax_{(i,j)}) + b_j, \tag{28}$$

$$b_i - h_{i,j} \geq \sum_{n=1}^{N} A_{n,j} \frac{W_j}{f_n}, \tag{29}$$

$$g_{i,j} = \mathcal{H} \times (b_j - bmax_{(i,j)}) + b_i, \tag{30}$$

$$b_j - g_{i,j} \geq \sum_{n=1}^{N} A_{n,i} \frac{W_i}{f_n}, \tag{31}$$

where $Profit$ is given in Eq. (13). Eq. (24) indicates that request τ_m can only be assigned to one server. Eq. (25) guarantees that all accepted requests will produce profits, i.e., the provider will reject the requests that can not produce profit. Eq. (26) ensures that average user satisfaction score is not less than $\boldsymbol{s_{th}}$. Eq. (27) ensures that $bmax_{(i,j)} = \max(b_i, b_j)$ holds. Eqs. (28)-(31) indicate that the execution of request τ_i and request τ_j has no overlapping on server S_n.

Algorithm 1 describes the ILP-based profit maximization scheme. The inputs to the algorithm include user request set Γ, server set Φ, and scheduling vector $T \triangleq [T_1, T_2, ..., T_K]$. The size of the server set Φ is constant. The scheduling vector T is specified by the cloud provider. The output is the total profit T_{Profit} generated in all time slots. T_{Profit} is initialized to 0 on line 1. For each time slot in the scheduling vector, the algorithm first updates the user request set Γ in the service request queue Q considering the new arrival of requests on line 3. Line 4 sets the properties of the service requests based on user personality model defined in Section 3.1.2. The available time of a server is defined as the earliest time instance that a server is available for executing user requests. It is clear that at the beginning of the first time slot, the available time of each server is same and given. But in any following time slot, the available time of each server can be different and depends on the server's assigned requests in the previous time slot and the server's frequency (i.e., processing speed). The available time of a server is important to the scheduling of user requests assigned to this server since the execution of requests follow the first-come-first-serve rule. Thus, line 5 updates the states of servers in Φ by calculating the servers' available time. Line 6 then decides the optimal schedule that maximizes the profit of the cloud provider by solving the ILP formulated in Eqs. (23)-(31). Line 7 calculates $Profit_i$ derived in time slot T_i using Eq. (13). Line 8 updates the total profit T_{Profit}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

Algorithm 1: ILP-based profit maximization algorithm

Input: User request set Γ, server set Φ, and scheduling vector $T \triangleq [T_1, T_2, ..., T_K]$
Output: Total profit T_{Profit} generated in all time slots

1 $T_{Profit} = 0$;
2 **for** $i = 1$ *to* K **do**
3 Update request set Γ in service request queue Q;
4 Set request properties based on user personality;
5 Update state of servers in Φ by calculating servers' available time;
6 Generate request allocation $\Gamma \rightarrow S$ by addressing the problem defined in Eqs. (23)-(31) using ILP solver;
7 Calculate $Profit_i$ using Eq. (13);
8 $T_{Profit} = T_{Profit} + Profit_i$;
9 $i = i + 1$;
10 **end**
11 **return** T_{Profit};

generated in the whole schedule. The process repeats until all scheduling slots are examined (lines 2-10). Finally, the algorithm returns the total profit T_{Profit} on line 11.

It is obvious that ILP consumes excessive computing resources when there are a large number of requests and servers, which increases the cloud provider's cost. In addition, when the time slot is small, the ILP optimization problem may not be solved before the next time slot, thus is not suitable for applications with high scheduling resolution (i.e., small scheduling interval) [31]. To this end, we propose a value assessment and cross entropy (VACE)-based profit maximization scheme to reduce the computation complexity and speed up the scheduling process.

6.2 VACE-based Profit Maximization Scheme

In this section, we first introduce the cross-entropy technique and describe how it can be applied to solve the optimization problem. Then, we assess the values of service requests based on the concepts of present value and opportunity cost in economics. Finally, we propose VACE, a lightweight cross entropy-based profit optimization scheme that is accelerated by using value assessment of service requests, to reduce the computation complexity and speed up the scheduling process. In particular, our value assessment- and cross entropy (VACE)-based profit maximization algorithm can be used to schedule the service requests of applications with high scheduling resolution.

6.2.1 Cross-entropy

Cross-entropy (CE) is used to transform an original deterministic optimization problem into a stochastic problem. It uses an adaptive sampling algorithm to solve the stochastic problem. In each iteration of the sampling algorithm, a sequence of random solutions are constructed that converge probabilistically to the optimal or a near-optimal solution.

Consider the following optimization problem: assuming Ω is a finite set of states and $S(x)$ is a real-valued function defined on Ω, the goal is to find the maximum value γ^* of $S(x)$ on Ω and the corresponding state x^*, that is,

$$S(x^*) = \gamma^* = \max_{x \in \Omega} S(x). \tag{32}$$

In order to solve the deterministic optimization problem given in Eq. (32), we first formulate a family of probability density functions (PDFs) (denoted by $\{f(x,u), u \in \boldsymbol{U}\}$) distributed in Ω. Then, the deterministic optimization problem is transformed to the estimation problem as:

$$\lambda(\gamma) = P_v\{S(X) \geq \gamma\} = E_v(I_{\{S(X) \geq \gamma\}}), \tag{33}$$

where γ is a threshold or a level parameter, and $X = \{X_1, X_2, ..., X_Z\}$ is a sample vector consisting of Z random samples generated by the corresponding PDF $f(x,v)$. $f(x,v)$ is a parametric class of probabilities $\{f(x,u), u \in \boldsymbol{U}\}$ with parameter u being set to v. P_v denotes the probability of $S(X) \geq \gamma$ and E_v denotes the expectation value of $I_{\{S(X) \geq \gamma\}}$. $I_{\{S(X) \geq \gamma\}}$ is the indicator function, i.e., $I_{\{S(X) \geq \gamma\}} = 1$ if and only if $S(X) \geq \gamma$.

After converting the deterministic optimization problem into the estimation problem given in Eq. (33), the main idea of CE is to find the smallest γ^* such that $\lambda(\gamma^*)$ is close to 0, i.e., the probability $P_v\{S(X) \geq \gamma^*\}$ is close to 0. That is to say, when $S(X)$ is less than γ^* with high probability, γ^* is regarded as the optimal or a near-optimal solution of the original deterministic optimization problem given in Eq. (32). In order to obtain γ^*, the CE method iteratively generates a sequence of tuples $\{\gamma_t, f(x,p_t)\}$, which converge to a small neighborhood of the optimal tuple $\{\gamma^*, f(x,p^*)\}$. γ_t is the threshold generated by the PDF $f(x,p_t)$ at the tth iteration. γ^* is the optimal solution of Eq. (32) and $f(x,p^*)$ is the corresponding PDF. The CE optimization method can be summarized as follows:

1) Initialize p_0 and set $t = 1$.
2) Generate Z samples $X_1, ...X_Z$ based on $f(x,p_{t-1})$ and calculate the associated objective function values $S(X_1), ..., S(X_Z)$.
3) Select N_{elite} samples with the maximum values, and denote δ as the set of these best samples.
4) Derive threshold γ_t by averaging the N_{elite} best samples, that is,

$$\gamma_t = \frac{1}{N_{elite}} \sum_{z \in \delta} S(X_z). \tag{34}$$

5) Update p_t of PDF $f(x,p_t)$ of the next iteration using

$$p_t = \underset{p}{\operatorname{argmax}} \frac{1}{Z} \sum_{z=1}^{Z} I_{\{S(X) \geq \gamma\}} \ln f(X_i, p). \tag{35}$$

6) If the predefined stop criteria are met, the iteration exits. Otherwise, set $t = t + 1$ and return to step 2).

6.2.2 Value assessment

In this section, we evaluate the values of service requests from aspects of revenue and cost. First, we introduce the concepts of present value and opportunity cost in economics. On this basis, we then describe the method of assessing the value of service requests.

Present value (PV). PV states that an amount of money today is worth more than that same amount in the future. In

this work, we use PV to convert the values of requests with different completion time to the values of current time. In this way, we can choose the request with the highest value for scheduling. In economics, present value is defined as:

$$PV = FV/(1+r)^n, \tag{36}$$

where PV and FV refer to the present value and the future value, respectively. r denotes the rate of return and n denotes the number of periods. Here, FV represents the revenue of service requests and n refers to the completion time of service requests. Based on Eq. (36), the PV of request m can be expressed as:

$$PV_m = \frac{R(\tau_m, ft_m)}{(1+r)^{ft_m}}, \tag{37}$$

where $R(\tau_m, ft_m)$ represents the future revenue and ft_m denotes the completion time of service request τ_m.

Opportunity cost (OC). OC refers to the benefits an individual, investor or business misses out when choosing one alternative over another. The cloud provider can leverage it to make educated decisions when having multiple options. For example, when we execute request τ_m, we will lose the benefit of executing other requests during the execution of τ_m. Thus, the OC of request τ_m can be defined as:

$$OC_m = \sum_{i=1;i\neq m}^{M} d_i * \max(0, RT_m - D_i^{des}), \tag{38}$$

where d_i is the charge decay rate of request τ_i, RT_m denotes the response time of request τ_m, $RT_m - D_i^{des}$ represents the decay time of request τ_i due to the execution of request τ_m. When $RT_m - D_i^{des}$ is negative, request τ_i is completed with satisfactory QoS and the benefit of request τ_i is not lost. $d_i * \max(0, RT_m - D_i^{des})$ denotes the lost benefit of request τ_i due to the execution of request τ_m.

Based on Eqs. (37) and (38), the value of request τ_m can be assessed by

$$Val_m = PV_m - OC_m. \tag{39}$$

Note that Val_m refers to the profit metric of request τ_m rather than the actual profit.

6.2.3 VACE-based profit maximization algorithm

The value assessment and cross entropy (VACE)-based algorithm uses the cross entropy technique to iteratively generate the allocations of requests to servers. In each iteration, the value assessment method is used to determine the order in which requests are executed at each server. Algorithm 2 shows the VACE-based profit maximization scheme.

Inputs to the algorithm are request set Γ and server set Φ. Line 1 initializes the PDF to $f(x, p_0)$ and the iteration counter t to 1. Lines 2-18 iteratively assign the requests to servers and schedule the allocated requests on servers for profit maximization. Line 3 generates Z allocations based on the PDF $f(x, p_{t-1})$ using the Latin hypercube importance sampling technique. For each allocation, the requests are scheduled on servers by prioritizing them based on the value assessment method on lines 4-8. Line 9 selects K out of Z schedules that satisfy the user satisfaction constraint in Eq. (26) and line 10 calculates the profit of K schedules using Eq. (13). Among these schedules, N_{elite} schedules with higher profit are chosen to update the profit threshold γ_t and the PDF $f(x, p_t)$ of the next iteration on lines 11-13. The profit threshold γ_t is deemed to be optimal if the results of the successive k iterations remain unchanged (lines 14-16). If γ_t is optimal, the iteration terminates, and line 19 returns γ_t and the corresponding schedule. Otherwise, the iteration counter t is updated on line 17, and the iteration terminates until t is greater than the maximum value t_{max}.

Algorithm 2: VACE-based profit maximization

```
   Input: User request set Γ and server set Φ
   Output: Schedule X and resultant profit γ
1  Initialize PDF to f(x, p_0) and iteration count t to 1;
2  while t ≤ t_max do
3      Generate Z allocations based on PDF f(x, p_{t-1}) using
       Latin Hypercube Importance Sampling;
4      for Z allocations do
5          Assess values of requests on servers by Eq. (39);
6          Prioritize requests based on value assessment;
7          Generate request schedule;
8      end
9      Select K schedules satisfying the constraint in Eq. (26);
10     Calculate the profits of K schedules using Eq. (13);
11     Select N_elite schedules with maximum profit;
12     Update the profit threshold γ_t using Eq. (34);
13     Update PDF f(x, p_t) of the next iteration by Eq. (35);
14     if γ_{t-k} = γ_{t-k+1} = ... = γ_t then
15         break;
16     end
17     t = t + 1;
18 end
19 return γ_t and corresponding schedule;
```

7 Evaluation

Extensive simulation experiments have been conducted to verify the effectiveness of the personality-guided user satisfaction prediction model and the satisfaction-constrained profit optimization schemes. Below, we first describe the experimental settings, then verify the effectiveness of the proposed user satisfaction prediction model, followed by the validation of the proposed profit optimization schemes and a comparison study with benchmarking algorithms in terms of profit improvement and time complexity.

7.1 Experimental Settings

In order to establish user satisfaction model, we collect 100 participants' personality scores and their satisfaction scores under 36 *price-qos* levels using questionnaires. A 5-fold cross-validation is used to train the user satisfaction model. To validate the effectiveness of the model, we recruit another 100 participants. The total 200 participants are between 23 to 48 years old with an average of 26.8 years and most of them are under 34 (the highest percentage of individuals using cloud computing services was recorded among people under 34 years of age [32]). The occupations of the total 200 participants include students, teachers, engineers, researchers, and accountants. These participants come from different countries and half of them are male. Thus, the 200 participants should be able to represent the general public. The server configuration parameters given in [33], which are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

Table 2: Configurations of the simulated server model built on 65nm technology [33].

Level	$v(V)$	$f(GHz)$	$\eta(nF)$
1	0.85	0.8010	13.0
2	0.90	0.8291	12.0
3	0.95	0.8553	14.0
4	1.00	0.8797	15.0
5	1.05	0.9027	17.0
6	1.10	1.0000	16.0
7	1.15	1.0527	18.0
8	1.20	1.1236	16.0
9	1.25	1.1867	19.0
10	1.30	1.2500	18.0

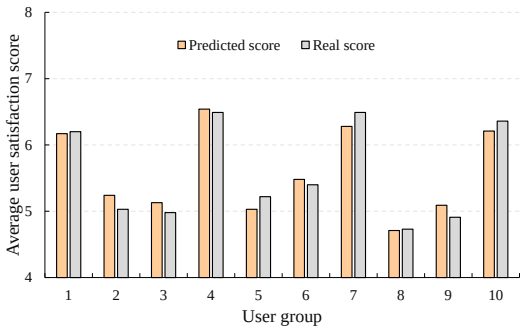


Figure 5: Satisfaction score prediction of new participants.

also shown in Table 2, are used in this work. The configuration of each server is randomly selected from the table. In our experiment, we empirically divide the participants into 10 different groups (i.e., 10 different service request sets) to test the performance of the proposed model and algorithms in different cases. The number of instructions W_m of request τ_m is randomly selected between $[4\times10^7, 6\times10^8]$ [33]. Other attributes such as service price c_m^{des} and desired finish time D_m^{des} are obtained based on questionnaires. The simulations are performed on a machine with a 3.5GHz Intel i7 quad-core processor and 16GB of memory.

In order to verify the effectiveness of the proposed satisfaction-constrained profit optimization algorithms, we compare and analyze the proposed algorithms with four benchmarking algorithms, i.e., RAN, GA [34], CORA [35], and NFA [36], which are briefly described as follows.

- RAN : RAN is an algorithm that randomly schedules requests to meet user satisfaction requirements.
- GA (Genetic Algorithm): GA obtains the optimal solution by simulating the natural evolution process. It abstracts the candidate solution into chromosome by coding, evaluating advantages and disadvantages of the chromosome using fitness function, and using reproduction and mutation to generate new candidate solutions until the termination condition is reached.
- CORA (Completion-time Optimal Resource Allocation): CORA optimizes the average completion time of the service requests, and obtains the optimal solution by converting the optimization problem from integer programming to linear programming.
- NFA (Novel Firefly Algorithm): NFA minimizes the makespan of tasks in a hybrid cloud environment. It follows the evolutionary mechanism of standard FA, but achieves better effectiveness and efficiency by using a distance-based mapping operator for mapping fireflies to solutions, a composite heuristic to generate an initial solution, and a new movement scheme to explore a wide range in the search space.

7.2 Experimental Results and Analysis

7.2.1 User satisfaction model prediction

We divide 100 new participants into 10 groups for verifying the user satisfaction prediction model. The verification results are shown in Figure 5. It can be seen from the figure that even for new participants who are not involved in model training, the predicted satisfaction score and the real satisfaction score are almost equal. Most users are not satisfied with high service price and low service quality, thus the average user satisfaction score is between 4.5 and 6.5, which is far below the highest satisfaction score (i.e., 10). Moreover, the differences in user personality lead to varying average satisfaction scores for each group of samples.

Figure 6 shows a confusion matrix of the predicted and real satisfaction scores of new participants, in which x-axis denotes the predicted satisfaction score while y-axis denotes the real satisfaction score. The element (i, j) of the i-th row and the j-th column in the matrix represents the probability that the real satisfaction score is i and the predicted satisfaction score is j. For example, $(2, 1)$ = 15% indicates that when the real satisfaction score is 2 and the predicted satisfaction score is 1, the probability is 15%. We can see from the figure that the satisfaction model can accurately predict satisfaction scores in more than 67% of the cases.

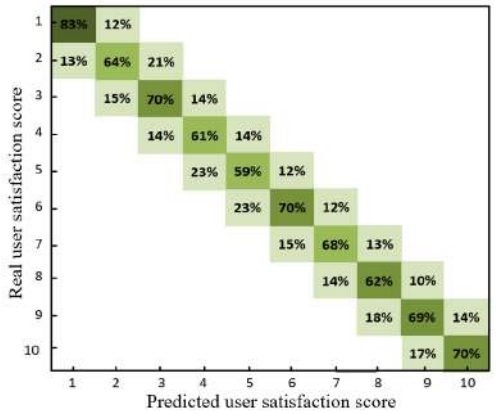


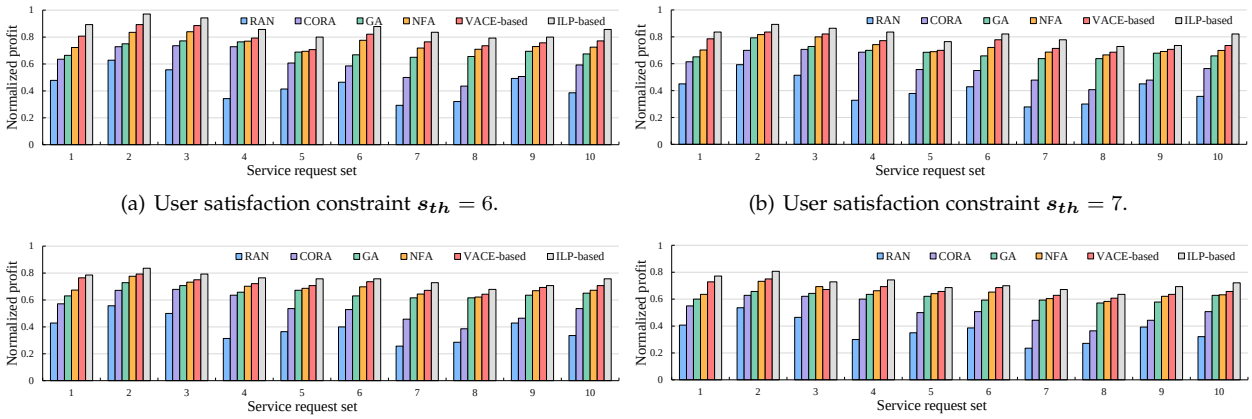
Figure 6: The confusion matrix of the predicted and real satisfaction scores of new participants.

7.2.2 Satisfaction constrained profit optimization

In this section, we compare our algorithms with four benchmarking algorithms in terms of profit and time complexity under different user satisfaction constraints, respectively.

Comparison of profit. We have known that user satisfaction scores are integers and set to be varied in the range from 1 (lowest satisfaction level) to 10 (highest satisfaction level) [30]. When the user satisfaction score is higher than 5 (the medium level), the cloud provider is considered to provide a good user experience. Thus, in the experiment, we set the user satisfaction constraint s_{th} to 6, 7, 8, and 9, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing



(a) User satisfaction constraint $s_{th} = 6$.

(b) User satisfaction constraint $s_{th} = 7$.

(c) User satisfaction constraint $s_{th} = 8$.

(d) User satisfaction constraint $s_{th} = 9$.

Figure 7: The normalized profit of 10 service request sets using our algorithms and four benchmarking algorithms.

The reason why we do not take the value of 10 is that, achieving the highest user satisfaction level typically has an unacceptable cost which is against the objective of cloud providers for profit maximization. Note that for each request set under each satisfaction constraint, we run six algorithms 1000 times separately and take the normalized average for comparison. Figure 7 shows the normalized average profit of six algorithms under different satisfaction constraints.

From this figure, we can see that our ILP-based algorithm outperforms four benchmarking methods in terms of profit. For example, when user satisfaction constraint $s_{th} = 6$ (see Figure 7(a)), compared to RAN, CORA, GA, and NFA, our ILP-based algorithm increases the profit of the cloud provider by 97.06%, 42.45%, 23.58%, and 14.71%, respectively. The reason behind lies in that our ILP-based algorithm adopts integer linear programming to optimize the profit, while RAN, GA, and NFA are approximate methods and CORA is a linear programming based approach.

Our VACE-based request scheduling algorithm also outperforms four benchmarking methods in terms of profit. For example, in the case of $s_{th} = 7$ in Figure 7(b), compared to RAN, CORA, GA, and NFA, our VACE-based algorithm increases the profit of the cloud provider by 84.76%, 31.22%, 10.35%, and 4.43%, respectively. This is because our VACE-based algorithm generates the schedule schemes based on the CE method with strong global search capability and takes advantages of user personality. RAN randomly dispatches service requests, ignoring users with high budges and urgency, thus, has least profit. CORA optimizes the profit by improving the service quality of all users, but does not take the impact of user personality on profit optimization into account. GA uses profit as the fitness function so that users with high budges and urgency are not missed, thus, obtains more profit than CORA and RAN. However, GA is easy to prematurely converge and fall into a local optimal solution, thus, achieves less profit than our VACE-based algorithm. For example, in the case of $s_{th} = 9$, as shown in Figure 7(d), GA achieves 9.68% less profit compared to our VACE-based algorithm. NFA uses a composite heuristic to generate an initial solution, and adopts a new movement scheme to explore a wide range in the search space to avoid falling into a locally optimal solution, whereas it does not take user personality into account. Thus, it achieves more profit than GA but less profit than our VACE-based algorithm.

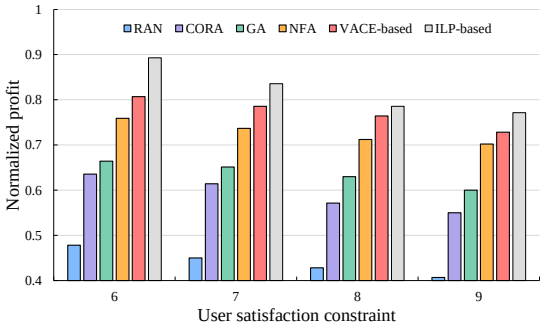


Figure 8: Compare the normalized profit of six algorithms under different user satisfaction constraints.

From Figure 8, we can see that when user satisfaction constraint s_{th} increases, the cloud provider's profit for all six algorithms decreases. The reason is that a higher s_{th} forces the cloud provider to prioritize the requests with high urgency to avoid the violation of SLA. In this way, the requests with low urgency but high profit will be delayed or discarded. Moreover, we can also see that the time-consuming ILP-based algorithm outperforms the VACE-based algorithm in terms of profit. For example, in the case of $s_{th} = 9$, the normalized average profit of the ILP-based algorithm under 10 request sets is 0.7157 while that of the VACE-based algorithm under 10 request set is 0.6714. Nevertheless, our lightweight VACE-based algorithm uses the methods of CE and value assessment for acceleration, thus is suited for applications with high scheduling resolution.

Comparison of time complexity. We run six algorithms 1000 times under different user satisfaction constraints. Table 3 compares the average running time of six algorithms under different user satisfaction constraints.

It can be seen from Table 3 that RAN has the least average running time since RAN only needs to satisfy users without optimizing profit. Among remaining algorithms that consider profit optimization, our VACE-based algorithm is superior to NFA, GA, CORA, and ILP-based

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3000792, IEEE Transactions on Cloud Computing

Table 3: Compare the average running time of six algorithms under different user satisfaction constraints (Unit of Measurement (UoM): seconds).

s_{th}	RAN	CORA	GA	NFA	VACE-based	ILP-based	Min.Im	Max.Im
6	1	28.75	5.87	3.85	2.29	31.09	1.68x	13.58x
7	1.25	29.09	7.31	4.79	2.84	33.92	1.69x	11.94x
8	1.41	31.98	8.49	5.63	3.2	36.19	1.76x	11.31x
9	1.73	32.11	10.14	6.72	3.84	38.55	1.75x	10.04x

algorithms in terms of average running time, and achieves a speedup range from 1.68x to 13.58x. The reason behind lies in that our VACE-based algorithm uses the CE technique for fast convergence, which is further accelerated using the value assessment of the requests. In addition, when user satisfaction constraint s_{th} increases, the average running time of six algorithms increases as well. The reason is that a higher s_{th} will increase the difficulty of searching for an efficient solution, thus incurring longer running time.

Figure 9 shows the normalized running time of six algorithms under different number of requests when $s_{th} = 6$. As can be seen from the figure, the ILP-based and CORA algorithms not only run much slower than the RAN, GA, NFA, and VACE-based algorithms, but also increase more rapidly in running time as the number of requests increases. The reason behind lies in that both the ILP-based and CORA algorithms use the linear programming technique, and thus have higher complexity than the RAN, GA, NFA, and VACE-based approximate algorithms. Moreover, the CORA algorithm converts the optimization problem from integer programming to linear programming, which simplifies the optimization at the cost of profit reduction, thus has less running time than our ILP-based algorithm.

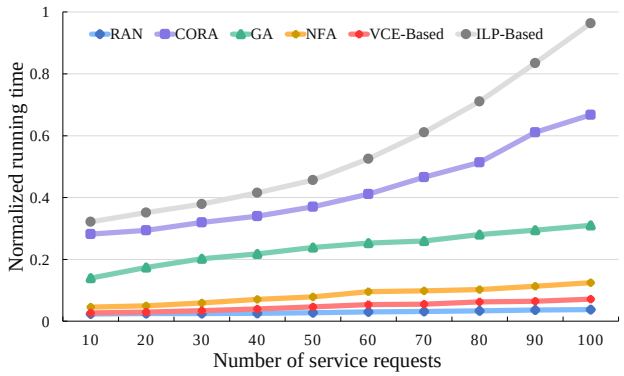


Figure 9: Compare the normalized running time of six algorithms under different number of service requests.

In summary, though our ILP-based algorithm has longer running time, it maximizes the provider's profit under the user satisfaction constraint and obtains a higher profit than other benchmarking algorithms. Our VACE-based algorithm is suboptimal in terms of profit optimization. However, it incurs shorter running time by adopting a value assessment-based CE technique, thus is well suited for applications with high scheduling resolution.

8 Discussion

In this section, we discuss the possible directions of applying our proposed user satisfaction model and request scheduling algorithms to the real cloud service providers in the future. Specifically, we provide two real examples below for the sake of better understanding.

- CRM [37]: As a combination of practices, strategies and technologies that companies use to manage and analyze their interactions with current and potential customers, customer relationship management (CRM) has been widely adopted in many service industries [38], [39], [40]. Using the CRM system, companies can greatly improve customer satisfaction for driving sales growth and ultimately gaining more profits. For attracting more customers, the CRM system should offer personalized and customized services so that each customer will experience differently according to their needs and interests [39].

As profit-oriented business enterprises, cloud service providers also can use such CRM systems to manage customer relationships for improving customer satisfaction and profitability [40], [41], [42]. In this paper, we propose a personality-guided user satisfaction model that not only predicts the user satisfaction based on factors such as user personality, service price, and service quality, but also captures the personalized execution requirements of service requests submitted by different users. Owing to the large amounts of historical transaction data and customer profile data generated by the interaction between cloud service providers and their customers and stored in the CRM cloud systems, our proposed personality-guided model can be integrated into the CRM systems of cloud service providers for providing personalization and customization of cloud services, and hence for getting more profits.

- DRM [43]: Real cloud service providers (e.g., Amazon EC2 and IBM blue cloud) usually manage their resources by using the cloud managers such as Eucalyptus, OpenNebula, and Nimbus [24]. These cloud managers can help virtualized data centers to build private, public and hybrid implementations of infrastructure as a service [44]. In cloud computing, resources are provided to cloud users in the form as virtual machines (VMs). When users submit their requests, the cloud service providers would use distributed resource management (DRM) systems such as Sun Grid Engine [45], Portable Batch System [46], and Condor [47] to schedule and assign these requests to different VMs in a centralized way. Taking Condor as an example, it provides a job queueing mechanism, scheduling scheme, priority policy, resource monitoring, and resource management. Users submit their requests to Condor, then Condor places them into a queue and chooses when and where to run them based upon a scheduling scheme [48].

In this paper, we model the cloud service platform as a multiserver system with a service request queue and propose a personality-guided user satisfaction model. Based on these models, we develop two effective scheduling schemes, i.e., ILP-based optimal request scheduling scheme and VACE-based approximate request scheduling scheme, to maximize the cloud service provider's profit. In real world, the cloud service providers can deploy different DRM systems for achieving different goals. Thus, at least for the cloud service

providers with the goal of maximizing profit, our proposed request scheduling schemes could be alternative approaches for their DRM systems.

In the future, we will focus on designing a personality-guided CRM system and a profit-aware DRM system for real cloud service providers.

9 Conclusions and Future Work

In this paper, we propose efficient personality-guided user request scheduling schemes for cloud profit maximization. More specifically, we first establish personalized user model and cloud provider model, based on which a profit optimization problem is formulated. Then, we propose a user personality-guided satisfaction prediction technique based on questionnaires. Subsequently, we formulate a satisfaction constrained profit maximization problem and design a time-consuming ILP-based optimal request scheduling scheme, which is followed by a lightweight VACE-based approximate scheme tailored for applications with higher scheduling resolution. Extensive experimental results show that our personality-guided user satisfaction prediction technique can achieve an accuracy of up to 83%. Our satisfaction constrained profit optimization algorithms can increase the profit by at least 3.96% compared to four benchmark algorithms while still achieving a speedup of at least 1.68x.

In the future, first, we will study the user satisfaction prediction model in depth by taking more factors into account, and improve the accuracy of prediction results by adopting neural network techniques to train the user satisfaction model. Second, we will consider a more flexible multiserver system model that allows long-term and short-term resource leasing mechanisms for further improving user satisfaction. Finally, based on these models, we plan to solve the profit maximization problem by determining the optimal request scheduling as well as the optimal multiserver configuration (i.e., the optimal size and speed for servers). In addition, we aim to design a personality-guided CRM system and a profit-aware DRM system for real cloud service providers.

References

[1] P. Cong, G. Xu, T. Wei, and K. Li, "A survey of profit optimization techniques for cloud providers", *ACM Computing Surveys*, vol. 53, no. 2, pp. 1-35, 2020.

[2] R. Buyya, S. Srirama, G. Casale, et al., "A manifesto for future generation cloud computing: Research directions for the next decade", *ACM Computing Surveys*, vol. 51, no. 5, pp. 1-38, 2018.

[3] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization", *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 158-171, 2013.

[4] P. Cong, L. Li, J. Zhou, K. Cao, T. Wei, M. Chen, and S. Hu, "Developing user perceived value based pricing models for cloud markets", *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 12, pp. 2742-2756, 2018.

[5] V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou, and A. Ailamaki, "Optimal service pricing for a cloud cache", *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1345-1358, 2011.

[6] R. Corsini and B. Ozaki, "Encyclopedia of psychology", *Wiley New York*, vol. 1, 1994.

[7] J. Cao, K. Hwang, K. Li, and A. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing", *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087-1096, 2013.

[8] C. Liu, K. Li, and K. Li, "Minimal cost server configuration for meeting time-varying resource demands in cloud centers", *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2503-2513, 2018.

[9] H. Yuan, J. Bi, and M. Zhou, "Temporal task scheduling of multiple delay-constrained applications in green hybrid cloud", *IEEE Transactions on Services Computing*, 2018.

[10] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud", *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 14-28, 2014.

[11] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach", *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017-1025, 2013.

[12] AmazonEC2SpotInstances. [Online]. Available: http://aws.amazon.com/ec2/spotinstances/.

[13] M. Macías and J. Guitart, "A genetic model for pricing in cloud computing markets", *in Proceedings of ACM Symposium on Applied Computing*, pp. 113-118, 2011.

[14] X. Zhang, C. Wu, Z. Li, and F. Lau, "A truthful (1-ϵ)-optimal mechanism for on-demand cloud resource provisioning", *IEEE Transactions on Cloud Computing*, 2018.

[15] H. Zhang, H. Jiang, B. Li, F. Liu, A. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands", *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805-818, 2016.

[16] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided min-min scheduling algorithm for load balancing in cloud computing", *in Proceedings of National Conference on Parallel Computing Technologies*, pp. 1-8, 2013.

[17] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows", *in Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-12, 2011.

[18] R. Cunha, M. Assuncao, C. Cardonha, and M. Netto, "Exploiting user patience for scaling resource capacity in cloud services", *in Proceedings of International Conference on Cloud Computing*, pp. 448-455, 2014.

[19] T. Wang, J. Zhou, G. Zhang, T. Wei, and S. Hu, "Customer perceived value- and risk-aware multiserver configuration for profit maximization", *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 5, pp. 1074-1088, 2020.

[20] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint", *in Proceedings of IEEE Conference on Computer Communications*, pp. 37-45, 2018.

[21] B. Rochwerger, D. Breitgand, A. Epstein, et al., "Reservoir-when one cloud is not enough", *Computer*, vol. 44, no. 3, pp. 44-51, 2011.

[22] B. Rochwerger, D. Breitgand, E. Levy, et al., "The reservoir model and architecture for open federated cloud computing", *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 1-11, 2009.

[23] J. Chen, C. Wang, B. Zhou, L. Sun, Y. Lee, and A. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud", *in Proceedings of International symposium on High performance distributed computing*, pp. 229-238, 2011.

[24] J. Mei, K. Li, A. Ouyang, and K. Li, "A profit maximization scheme with guaranteed quality of service in cloud computing", *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3064-3078, 2015.

[25] Big Five personality traits. [Online]. Available: https://en.wikipedia.org/wiki/Big_Five_personality_traits.

[26] E. Romero, P. Villar, J. Gómez-Fraguela, and L. López-Romero, "Measuring personality traits with ultra-short scales: A study of the Ten Item Personality Inventory (TIPI) in a Spanish sample", *Personality and Individual Differences*, vol. 53, no. 3, pp. 289-293, 2012.

[27] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint", *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 2s, pp. 363-368, 2014.

[28] R. McCrae and O. John, "An introduction to the five-factor model and its applications", *Journal of personality*, vol. 60, no. 2, pp. 175-215, 1992.

[29] S. Gosling, P. Rentfrow, and W. Swann Jr, "A very brief measure of the big-five personality domains", *Journal of Research in personality*, vol. 37, no. 6, pp. 504-528, 2003.

[30] K. Yan, X. Zhang, J. Tan, and X. Fu, "Redefining QoS and customizing the power management policy to satisfy individual mobile users", *in Proceedings of 49th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1-12, 2016.

[31] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions", *in Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 912-920, 2010.
[32] statista. [Online]. Available: https://www.statista.com/statistics/1022962/poland-use-of-cloud-computing-services-by-age/.
[33] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 8, pp. 1269-1282, 2016.
[34] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent tasks scheduling based on genetic algorithm in cloud computing", *in Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, 2009.
[35] Z. Huang, B. Balasubramanian, M. Wang, T. Lan, M. Chiang, and D. Tsang, "Need for speed: Cora scheduler for optimizing completion-times in the cloud", *in Proceedings of IEEE Conference on Computer Communications*, pp. 891-899, 2015.
[36] Y. Zhang, J. Zhou, L. Sun, J. Mao, and J. Sun, "A novel firefly algorithm for scheduling bag-of-tasks applications under budget constraints on hybrid clouds", *IEEE Access*, vol. 7, pp. 151888-151901, 2019.
[37] Customer Relationship Management. [Online]. Available: http://en.wikipedia.org/wiki/Customer_relationship_management.
[38] R. Schmidt, M. Möhring, and B. Keller, "Customer relationship management in a public cloud environment-Key influencing factors for European enterprises", *in Proceedings of the 50th Hawaii International Conference on System Sciences*, pp. 4241-4250, 2017.
[39] M. Anshari, M. Almunawar, S. Lim, and A. Al-Mudimigh, "Customer relationship management and big data enabled: Personalization & customization of services", *Applied Computing and Informatics*, vol. 15, no. 2, pp. 94-101, 2019.
[40] I. Chen and K. Popovich, "Understanding customer relationship management (CRM): People, process and technology", *Business Process Management Journal*, vol. 9, no. 5, pp. 672-688, 2003.
[41] R. Bose, "Customer relationship management: key components for IT success", *Industrial management & Data systems*, vol. 102, no. 2, pp. 89-97, 2002.
[42] R. Rahimi and M. Kozak, "Impact of customer relationship management on customer satisfaction: The case of a budget hotel chain", *Journal of Travel & Tourism Marketing*, vol. 34, no. 1, pp. 40-51, 2017.
[43] Y. Yan and B. Chapman, "Comparative study of distributed resource management systems-SGE, LSF, PBS Pro, and LoadLeveler", *Technical Report-Citeseerx*, 2008.
[44] OpenNebula. [Online]. Available: https://en.wikipedia.org/wiki/OpenNebula.
[45] Y. Yang and Y. Chen, "Sun Grid Engine (SGE) and its application", *in Proceedings of International Symposium on Computers & Informatics*, pp. 975-982, 2015.
[46] T. Baer and D. Johnson, "pbsacct: A workload analysis system for pbs-based hpc systems", *in Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, pp. 1-6, 2014.
[47] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience", *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323-356, 2005.
[48] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor: A distributed job scheduler", *in Beowulf Cluster Computing with Linux. Cambridge, MA, USA: MIT Press*, pp. 307-350, 2002.

Peijin Cong received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, shanghai, China, in 2016. She is currently pursuing her Ph.D. degree with the School of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interests are in the area of mobile device power management, cloud computing, mobile cloud computing, and mobile edge computing.

Guo Xu is currently pursuing the master degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interest is in the area of cloud computing.

Junlong Zhou (S'15-M'17) received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014-2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include embedded systems, edge computing and Internet-of-Things, and cyber physical systems.

Mingsong Chen (S'08–M'11) received the B.S. and M.E. degrees from Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006 respectively, and the Ph.D. degree in Computer Engineering from the University of Florida, Gainesville, in 2010. He is currently a full Professor with the Department of Embedded Software and Systems of East China Normal University. His research interests include design automation of cyber-physical systems and mobile cloud computing.

Tongquan Wei (M'11-SM'19) received his Ph.D. degree in Computer Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University. His research interests are in the areas of internet of things (IoT), edge computing, cloud computing, and design automation of intelligent systems and cyber physical systems. He is a senior member of the IEEE.

Meikang Qiu (SM07) received the BE and ME degrees from Shanghai Jiao Tong University, China. He received the MS and PhD degree in computer science from the University of Texas at Dallas in 2003 and 2007, respectively. Currently, he is an associate professor of computer engineering at Pace University. He has worked at Chinese Helicopter R&D Institute, IBM, etc. His research interests include cyber security, embedded systems, cloud computing, smart grid, microprocessor, data analytics, etc.