Wireless Netw (2018) 24:1491–1508
https://doi.org/10.1007/s11276-016-1416-7

A hierarchical approach for resource allocation in hybrid cloud environments

Zhe Liu[1] · Changle Li[1] · Weijie Wu[2] · Riheng Jia[3]

Published online: 29 November 2016

Abstract Cloud computing is a key technology for online service providers. However, current online service systems experience performance degradation due to the heterogeneous and time-variant incoming of user requests. To address this kind of diversity, we propose a hierarchical approach for resource management in hybrid clouds, where local private clouds handle routine requests and a powerful third-party public cloud is responsible for the burst of sudden incoming requests. Our goal is to answer (1) from the online service provider's perspective, how to decide the local private cloud resource allocation, and how to distribute the incoming requests to private and/or public clouds; and (2) from the public cloud provider's perspective, how to decide the optimal prices for these public cloud resources so as to maximize its profit. We use a Stackelberg game model to capture the complex interactions between users, online service providers and public cloud providers, based on which we analyze the resource allocation in private clouds and pricing strategy in public cloud. Furthermore, we design efficient online algorithms to determine the public cloud provider's and the online service provider's optimal decisions. Simulation results validate the effectiveness and efficiency of our proposed approach.

Keywords Stackelberg game · Resource allocation · Hybrid cloud

✉ Changle Li
clli@mail.xidian.edu.cn

Zhe Liu
zliuxidian@gmail.com

Weijie Wu
wuwjpku@gmail.com

Riheng Jia
jiariheng@sjtu.edu.cn

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Shaanxi 710071, China

[2] Future Network Theory Lab, Huawei Technologies Co. Ltd, Hong Kong, Hong Kong

[3] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

1 Introduction

Online services are applied into many aspects of our daily life. For example, people perform communication, shopping, entertainment and professional activities over the Internet. The online service providers (OSPs) need to develop new technologies, aiming at providing QoS guaranteed service while at the same time, minimizing its operating cost. In general, an OSP operates a number of distinct services simultaneously, so it needs to answer requests from various applications that differ a lot in pattern. To this end, many OSPs develop private cloud systems so as to provision virtualized environment where each service can be properly addressed.

However, private cloud is still not enough. Users' requests can be of high variance over time. In general, we can roughly classify the customers' request patterns as following two categories:

- Routine case: Normal and periodical user requests, such as webpage browsing or video viewing, can be roughly estimated over the time span according to their regularity.
- Burst case: Sudden increase in request amount can incur a much higher burden, e.g., hundreds of times higher, than the routine case.

These requests can undoubtedly bring extensive revenue for the OSP, but in the meanwhile, it brings huge burden for the server to answer these requests. A naive way for an OSP to cope with these requests is to prepare a very powerful private cloud system, such that even when the burst occurs, it can still answer all requests. However, this approach is neither economically efficient nor robust. The hardware and management cost is surely very high, and for most of the time, the request amount is much lower than the peak time, so many servers remain idle. Even if one prepares such a system, it can hardly be guaranteed to work normally at the peak time. In fact, there have been many cases that online shopping sites crashed in shopping festivals (e.g., the Black Friday in western countries or Double Eleven in China, which draws much more requests than the routine workloads [1]).

Alternatively, hybrid cloud is a promising solution to this problem. Hybrid cloud means a combination of a private cloud operated by the OSP, and a public cloud system operated by a third-party cloud service provider (CSP). To address the routine case, most requests can be answered by the local private cloud; and for the burst case, the OSP can resort those requests beyond its capacity to the public cloud.

The fundamental questions of the hybrid cloud approach are resource allocation and pricing. In particular, given the request pattern of users, from the OSP's perspective, it has to decide how to build up a private cloud in terms of the amount of each resource (e.g., CPU, memory, storage, etc.) to be prepared; and facing the incoming requests, how to properly distribute them into private/public clouds. From the CSP's perspective, it needs to decide how to set prices to sell cloud services so as to maximize its profit.

These questions are non-trivial to answer due to the following challenges. First, the stochastic nature of the request arrival pattern makes it difficult to determine the local resource arrangement. Second, since the arrival rate of requests is very high in a large scale system, the resource allocation algorithm must be dynamic and time efficient, but finding the optimal allocation is a very complex problem: even if we know the request distribution a priori, it is still NP-complete. Last but not least, the CSP's pricing decision and the OSP's allocation and purchasing decisions are highly coupled, so we need have a deep understanding on their economic correlations and dynamics. All these challenges make the resource allocation and pricing problem in hybrid cloud an open question so far.

To tackle these challenges, we propose a hierarchical approach to analyze the hybrid cloud framework. In particular, we use a Stackelberg game to capture the interactions between the OSP and the CSP, based on which we design algorithms to decide the OSP's strategy in purchasing the public cloud resources, as well as its allocation mechanism for the incoming requests. We also design an algorithm for the CSP to decide its pricing scheme that approximately maximizes its profit. Due to the NP-completeness of each sub-problem, we design efficient and effective approximation algorithms, and use theoretic analysis and simulations to validate their high accuracy. We believe this gives important guidelines to design practical hybrid cloud systems.

To sum up, we design a comprehensive framework to design resource allocation and pricing in hybrid cloud. Our main contributions are:

- We reveal the interactions of OSP and CSP, and present a hierarchical analytical framework for the hybrid cloud.
- We propose approximation algorithms for resource allocation and pricing, and show their efficiency and accuracy both theoretically and empirically.
- Base on real-trace simulation, we show our design can achieve satisfactory performance in practice.

This is the outline of this paper. In Sect. 2, we formulate the hybrid cloud as a Stackelberg game model. Section 3 proposes two approximation algorithms to address the request distribution problem for the OSP. Based on this, we derive the dynamic resource allocation algorithm for the OSP in Sect. 4. In Sect. 5 we use an approximation algorithm to decide the optimal pricing scheme for the CSP. Real-trace evaluation is given in Sect. 6. Section 7 discusses some possible extensions to our framework. Section 8 states related work and Sect. 9 concludes.

2 System and game model

In this section, we present our system model and problem formulation, provide background material on a game-theoretic model called the Stackelberg model, and finally develop this game-theoretic model as a framework for capturing the complex interactions between the OSP and CSP in hybrid cloud environments.

2.1 System model and problem formulation

In this paper, we consider the hybrid cloud as a hierarchical model (as shown in Fig. 1), which includes the users, OSP and CSP in every individual layer. The primary process is that the OSP collects requests and serves network users. However, when requests cannot be fully satisfied in the private cloud of OSP, they are resorted to the public cloud in CSP. In what follows, we will give an overview of each part of the system, and show mathematical analysis respectively.

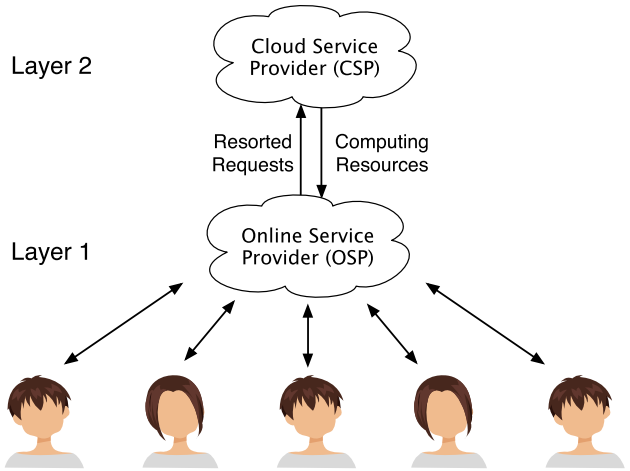


Fig. 1 Hierarchical system model

2.1.1 Layer 1: users and OSP

Layer 1 depicts the interactions between network users and OSP. In this layer, users request for network services and try to access OSP's computing resources. These requests are heterogenous and meanwhile, they call for different combination of computing resources to fulfill. After collecting users' requests, the OSP assembles its computing resources and allocates them to serve users' heterogenous requests. This process happens in local area, i.e., only between users and their corresponding OSP. Now we formulate the interactions between users and OSP in following paragraphs.

We assume that users' requests can be classified into n types. In reality, they can represent various applications. Each request type requires a specific collection of different computing resources, for example, CPU, memory and storage. We can define $\mathcal{R}$ as the set of all request types, i.e.,

$$\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_n\}, i = 1, 2, \ldots, n, \tag{1}$$

where $\mathcal{R}_i = (r_{i1}, \ldots, r_{ij}, \ldots, r_{im})$ is a type i request. Each element r_{ij} denotes the amount of resource j required to satisfy $\mathcal{R}_i$.

Assume that the number of incoming request in type i, i.e., $\mathcal{R}_i$, is a random variable K_i in any given time t. Note that the number of each incoming request is calculated in a time interval rather than a time instant. Without loss of generality, we formulate K_i to be an independent Gaussian distribution[1] as

$$K_i^t : \mathcal{G}_i^t(k_i) = \frac{1}{\sqrt{2\pi}\sigma_i^t} \exp\left[-\frac{(k_i - \mu_i^t)^2}{2(\sigma_i^t)^2}\right] \sim N(\mu_i^t, \sigma_i^t), \tag{2}$$

[1] The reason why we choose the Gaussian distribution is discussed in Sect. 7.2.

where the parameter μ_i^t and σ_i^t are given distributions over t, and they represent the request incoming patterns. We use a truncated Gaussian distribution to characterize the number of incoming request, where $K_i^t \in [k_{imin}, k_{imax}]$, where $0 \leqslant k_{imin} \ll \mu_i^t$ and $k_{imax} \gg \mu_i^t$. Since the probability $K_i^t < 0$ or $K_i^t > k_{imax}$ can be neglected, we can still use a Gaussian distribution to approximate K_i^t. Figure 2 shows μ_i^t in a routine or burst case, respectively.

We use $\boldsymbol{\varphi}_t$ to represent the instantaneous amount of incoming requests in all types, which can be formulated as

$$\boldsymbol{\varphi}_t = (K_1^t, \ldots, K_i^t, \ldots, K_n^t). \tag{3}$$

Now let us consider how to capture the property of the private cloud system operated by the OSP. Let $\mathcal{N}$ be the amount of resources possessed by this private cloud, then it can be represented by

$$\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_j, \ldots, \mathcal{N}_m), \quad j = 1, 2, \ldots, m, \tag{4}$$

where each element represents the amount of each resource available in the private cloud, e.g., CPU, memory, storage, etc. For each kind of resource, it is associated with a unit operating cost, denoted by $\mathcal{P}_l = (p_{l1}, \ldots, p_{lj}, \ldots, p_{lm})$.

2.1.2 Layer 2: OSP and CSP

When requests cannot be fully satisfied in the private cloud, they are resorted to the public cloud. The public cloud of CSP allocates its own computing resources to serve sorted requests and in the meanwhile, charges for its leased resources to OSP. Finally, the OSP rents CSP's computing resource and serves resorted requests.

A classic methodology, vector bin packing [2, 3] provisions an optimal combination of virtual machines (VMs) to cover any given set of request demands. On measuring the fee paid by the OSP to the CSP, we can view it as the sum of fee paid for each kind of resource, where the unit price vector is $\mathcal{P}_c = (p_{c1}, \ldots, p_{cj}, \ldots, p_{cm})$.

Now let us focus on the profit of the OSP and the CSP at a particular time point t. We denote the allocation strategy used to distribute incoming requests as $\mathcal{A}$. Let $\mathbf{I}_t$ be the set of requests from users at time t. The OSP allocates a subset of these requests $\mathbf{L}_t(\mathcal{A})$ to its private cloud, and the rest $\mathbf{C}_t(\mathcal{A})$ to the public cloud. Resource demand of $\mathbf{I}_t$, $\mathbf{L}_t(\mathcal{A})$ and $\mathbf{C}_t(\mathcal{A})$ are denoted as $\mathcal{S}_I^t$, $\mathcal{S}_L^t(\mathcal{A})$ and $\mathcal{S}_C^t(\mathcal{A})$, respectively, each demand being a vector of resources needed to satisfy the corresponding requests. Obviously, we have $\mathbf{I}_t = \mathbf{L}_t(\mathcal{A}) \bigcup \mathbf{C}_t(\mathcal{A})$ and $\mathcal{S}_I^t = \mathcal{S}_L^t(\mathcal{A}) + \mathcal{S}_C^t(\mathcal{A})$. In essence, $\mathbf{L}_t(\mathcal{A})$, $\mathbf{C}_t(\mathcal{A})$, $\mathcal{S}_L^t(\mathcal{A})$ and $\mathcal{S}_C^t(\mathcal{A})$ are functions of the allocation strategy $\mathcal{A}$. Given any deterministic $\mathcal{A}$, the issue of request allocation will be settled immediately.

Upon satisfying a request, the OSP owns a profit for this service provided. We denote $\mathcal{U}_t$ as the sum of this utility at

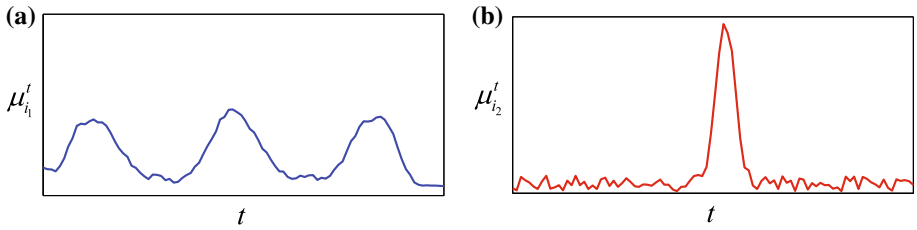


Fig. 2 Examples of μ_i^t. **a** a routine case **b** a burst case

Table 1 Main notations

Notation	Explanation
$\mathcal{N}$	Resources in the private cloud
$\mathcal{R}$	Set of request types
$\boldsymbol{\varphi}_t$	Instantaneous amount of incoming requests
$\mathcal{P}_l$	OSP's unit operating cost
$\mathcal{P}_c$	Price for public cloud resources
$\mathbf{I}_t$	Set of users' requests
$\mathbf{L}_t$	Subset of requests allocated to OSP
$\mathbf{C}_t$	Subset of requests allocated to CSP
$\mathcal{S}_I^t$	Resource demand of $\mathbf{I}_t$
$\mathcal{S}_L^t$	Resource demand of $\mathbf{L}_t$
$\mathcal{S}_C^t$	Resource demand of $\mathbf{C}_t$
Π_L^t	Utility of OSP
Π_C^t	Utility of CSP

time t. We can express the utility (or payoff[2]) of the OSP and the CSP as

$$\Pi_L^t = \mathcal{U}_t - \mathcal{N}\mathcal{P}_l - \mathcal{S}_C^t(\mathcal{A})\mathcal{P}_c = \Pi_L^t(\mathcal{P}_c, \mathcal{N}, \mathcal{A}), \quad (5)$$

$$\Pi_C^t = \mathcal{S}_C^t(\mathcal{A})\mathcal{P}_c = \Pi_C^t(\mathcal{P}_c, \mathcal{N}, \mathcal{A}), \quad (6)$$

where the OSP's utility equals the utility of satisfying all requests minus its cost, and the CSP's utility equals its revenue. The terms $\mathcal{N}\mathcal{P}_l$ and $\mathcal{S}_C^t(\mathcal{A})\mathcal{P}_c$ are indeed scalar products, and we have $\mathcal{N}\mathcal{P}_l = \sum_{j=1}^{m} \mathcal{N}_j p_{lj}$ and $\mathcal{S}_C^t(\mathcal{A})\mathcal{P}_c = \sum_{j=1}^{m} S_{cj}^t(\mathcal{A}) p_{cj}$, where $S_{cj}^t(\mathcal{A})$ is the element of $\mathcal{S}_C^t(\mathcal{A})$.

We should note that the operating cost of OSP (i.e., $\mathcal{P}_l$) is used to maintain the running of private resources, for example, CPU, memory and storage. In other words, the cost indicates the power consumption. Table 1 lists important notations introduced in this section.

2.2 Stackelberg's model of duopoly

In game theory and economics terms, Stackelberg's duopoly game [4] was proposed to model the interaction between the leader firm and follower firm. Considering a market in which there are two firms and they produce the same kind of good. Assume that Firm 1 is the leader firm who occupies the majority of market share and Firm 2 is the follower to Firm 1. Each firm's strategic variable is its good output respectively, and different with other game models, the firms in Stackelberg game make their own decisions sequentially rather than simultaneously. That is to say, Firm 1 (i.e., the leader) chooses its output and then, Firm 2 (i.e., the follower) handles its output by knowing Firm 1's output. It indicates that Firm 2's strategy is a function which associates its own output with each Firm 1's output.

In order to find the subgame perfect equilibria, backward induction is used in Stackelberg game. The basic idea of backward induction is listed as follows:

- *Step 1*: find the outputs of Firm 2 that maximize its profit, given any output of Firm 1;
- *Step 2*: find Firm 1's output that maximizes its profit, given the output strategy of Firm 2.

By applying backward induction, we can obtain the subgame perfect equilibria of a Stackelberg game, and guarantee some useful properties, for example, the first-mover's equilibrium profit and equilibrium outputs. Interested readers can refer to reference [4] for more details.

2.3 Game theoretic approach to hybrid cloud system

In this paper, we use a Stackelberg game [4] to capture the interactions between the OSP and the CSP, where the CSP is the leader and the OSP is the follower. Meanwhile, the OSP/CSP's computing resource can be regarded as good in market. We use this game theoretical model because in reality, the CSP decides its public cloud resource sale scheme first. Once the decision is made, the CSP usually does not change it frequently. Based on this scheme, the OSP decides its local resource preparation accordingly in order to maximize its own utility. Thus, we can claim that the two players in game move sequentially. Given the allocation strategy $\mathcal{A}$, $\Pi_L^t(\mathcal{P}_c, \mathcal{N}, \mathcal{A})$ and $\Pi_C^t(\mathcal{P}_c, \mathcal{N}, \mathcal{A})$ are deterministic and therefore, we can omit the notation $\mathcal{A}$ for simplicity in the rest of this paper. We formally describe the game as follows.

- *Players.* The online service provider (OSP) and the cloud service provider (CSP).

[2] In this paper, we use "utility" and "payoff" interchangeably.

- *Payoff*. The payoff of the OSP is

$$\Pi_L(\mathcal{P}_c, \mathcal{N}) = \int_t \Pi_L^t(\mathcal{P}_c, \mathcal{N})\mathrm{dt}, \tag{7}$$

and the payoff of the CSP is

$$\Pi_C(\mathcal{P}_c, \mathcal{N}) = \int_t \Pi_C^t(\mathcal{P}_c, \mathcal{N})\mathrm{dt}. \tag{8}$$

We will formally describe these payoff functions in later sections.

- *Game strategy*. The CSP decides the pricing strategy $\mathcal{P}_c$ for the public cloud. The OSP decides the resource $\mathcal{N}$ powered by the private cloud.

The solution to this game is called Stackelberg equilibrium (or SE), which can be solved by applying the backward induction [5] as follows.

1. Assume the CSP fixes $\mathcal{P}_c$. The OSP solves the problem $\mathcal{N}^*(\mathcal{P}_c) = \arg\max_{\mathcal{N}} \Pi_L(\mathcal{P}_c, \mathcal{N})$, where the item $\mathcal{N}^*(\mathcal{P}_c)$ is the optimal resource allocation for OSP given the pricing strategy $\mathcal{P}_c$ of CSP.
2. By knowing $\mathcal{N}^*(\mathcal{P}_c)$, the CSP solves $\mathcal{P}_c^* = \arg\max_{\mathcal{P}_c} \Pi_C(\mathcal{P}_c, \mathcal{N}^*(\mathcal{P}_c))$.

Our following formulation will show that both payoff functions are continuous with respect to the decision variables, thus the two steps are both optimization problems over a compact set, so the existence of SE is guaranteed. Note that one may also meet the discrete case when modeling a similar problem. The option to tackle this point is approximating the discrete function with a continuous one or modifying the payoff function to a discrete manner.

Correspondingly, the next sections are organized in a backward manner. In Sect. 3, we propose efficient solutions to address $\mathcal{A}$ and decide $\mathbf{L}_t$ for any given $\mathcal{N}$ and $\mathcal{P}_c$. This serves as the basis of our game analysis. Section 4 analyzes the OSP's problem for any given CSP's strategy. Section 5 decides the CSP's optimal pricing scheme so as to obtain the Stackelberg equilibrium.

3 Request distribution: private or public

Prior to solving the Stackelberg game, let us first consider given the strategies of the OSP and the CSP, how can the OSP allocate the incoming requests into the private and public clouds so as to maximize its utility. In this section, we formulate this problem into an optimization problem, prove its NP-completeness, and then present two approximation algorithms for efficient solutions. In other words, we focus on how to design a rational and efficient $\mathcal{A}$ in following parts of this section.

3.1 The static optimization problem

Given private cloud resource $\mathcal{N}$, an OSP needs to decide which requests to be allocated in the private cloud for processing, and which are resorted to the public cloud. Note that due to the various nature of requests, they have different "appropriateness" to be put in local. For example, requests with sensitive information, high security requirement or high data transmission cost should be processed locally with a high priority. In here, we use a notation λ_{ip} to represent this appropriateness of putting the *p th* request of the *i th* type in the private cloud. Later we use Λ_t to represent the matrix consisting of elements λ_{ip}. We use a binary variable x_{ip} to denote whether this request is allocated in the private cloud, i.e., $x_{ip} = 1$ if yes or 0 otherwise. In general, the OSP aims at solving the following problem, denoted by *OPT*:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \sum_{p=1}^{K_i^t} \lambda_{ip} x_{ip} \\ \text{subject to} \quad & \sum_{i=1}^{n} \sum_{p=1}^{K_i^t} r_{ij} x_{ip} \leqslant \mathcal{N}_j, \forall 1 \leq j \leq m, \\ & x_{ip} \in \{0, 1\}. \end{aligned} \tag{9}$$

In this subsection, we treat *OPT* as a static optimization at t, and based on this we will further determine the OSP's best choice in the next section. Now let us prove that *OPT* is NP-complete.

Theorem 1 *OPT is NP-complete.*

Proof Given any solution to *OPT*, we can verify the solution in a polynomial time. Thus *OPT* is NP.

Then we build *OPT* from the Binary Knapsack Problem (BKP), one of Karp's 21 NP-complete problems [6, 7]. The decision of BKP is NP-complete and its optimization is NP-hard. A dimensional extension of BKP leads to the Multiple Knapsack Problem (MKP), which is at least NP-hard [8, 9]. Suppose we have a knapsack with a limit of $\{\mathcal{N}_1, \ldots, \mathcal{N}_j, \ldots, \mathcal{N}_m\}$. Given a set of items, each with a weight of $(r_{i1}, \ldots, r_{ij}, \ldots, r_{im})$ and a value of λ_i. Next we determine an item collection to maximize its value under the constraint of the sum of weights. Therefore, *OPT* reduces to MKP and is thus NP-complete. □

Given that *OPT* is NP-complete, we design approximation algorithms to achieve the sub-optimality in an efficient manner. In particular, we propose the greedy mechanism and the load balancing mechanism. The first one has an advantage in the low computational complexity, while the latter is nearer to the optimality.

3.2 A greedy sorting mechanism for *OPT*

Now we design the following Approximated Greedy Sorting (*APPGS*) algorithm depicted in Algorithm 1. Let $\{\mathbf{L}_t, \mathbf{C}_t\} = APPGS(\boldsymbol{\varphi}_t, \Lambda_t, \mathcal{N})$ be the output of this algorithm. The algorithm consists of three steps:

- *Step 1*: Initialization. In this step, all user requests are randomly arranged and denoted by the term m_v.
- *Step 2*: Sorting. We sort the requests in a sequence such that $\frac{\lambda_v^*}{\sqrt{\sum_{j=1}^{m} v_{vj}^2}}$ is non-decreasing in v. We use $\mathbf{I}$ to denote this sequence and m_v^* to denote the index of each request.
- *Step 3*: Allocation. We use the greedy algorithm to generate $\mathbf{L}_t$, the set of requests to process in the private cloud, and also $\mathbf{C}_t$, the set of requests to the public cloud, as $\mathbf{I} - \mathbf{L}_t$.

We next analyze the time complexity of Algorithm 1.

Algorithm 1: *APPGS* Mechanism

Input: φ_t, Λ_t, $\mathcal{N}$
Output: private set $\mathbf{L}_t$, public set $\mathbf{C}_t$

Initialization:
$V = \sum_{i=1}^{n} K_i^t$, $m_v = \{r_{v1}, r_{v2}, \ldots, r_{vm}\}$ $(v = 1, 2, \ldots, V)$, $\mathbf{L}_t = \emptyset$, $\mathbf{C}_t = \emptyset$, $\mathbf{T} = \emptyset$
Sorting request list $\{m_1^*, m_2^*, \ldots, m_V^*\}$ such that $\frac{\lambda_1^*}{\sqrt{\sum_{j=1}^{m} r_{1j}^2}} \geqslant \frac{\lambda_2^*}{\sqrt{\sum_{j=1}^{m} r_{2j}^2}} \geqslant \ldots \geqslant \frac{\lambda_V^*}{\sqrt{\sum_{j=1}^{m} r_{Vj}^2}}$
$\mathbf{I} = \{m_1^*, m_2^*, \ldots, m_V^*\}$
for $v = 1 : V$ **do**
  if $\mathbf{T} + m_v^* \leqslant \mathcal{N}$ **then**
    $\mathbf{L}_t := \mathbf{L}_t \bigcup \{m_v^*\}$
    $\mathbf{T} := \mathbf{T} + m_v^*$
  end
end
Return $\mathbf{L}_t, \mathbf{C}_t = \mathbf{I} - \mathbf{L}_t$

Theorem 2 *The time complexity of Algorithm 1 is polynomial.*

Proof Let the number of user requests be *N*. In Step 2, the sorting process is of $O(N \log N)$ complexity. In Step 3, we need to traverse *N* user requests, and its computational complexity is $O(N)$. In conclusion, the computational complexity of Algorithm 1 is $O(N \log N)$. □

The *APPGS* is based on one of our previous works, and readers interested in the approximation ratio and other details can refer to [10].

3.3 A load balancing mechanism for *OPT*

In this subsection, we present another approximation algorithm with a better utility performance but a bit higher complexity. We call it Approximated Load Balancing (*APPLB*) algorithm, which derives from [11].

Let us use a toy example to illustrate the intuition of our algorithm. We consider requests associated with two dimensional resource requirements, i.e., CPU and storage. In Fig. 3a, the private cloud accepts requests m_1 and m_2, but the rest available resources are not enough for any other incoming requests. But if the OSP chooses m_1, m_3 and m_4 into the private cloud, its resource utilization ratio is higher (see Fig. 3b). Thus, our objective is to design an algorithm that maximizes the resource utilization in the private cloud.

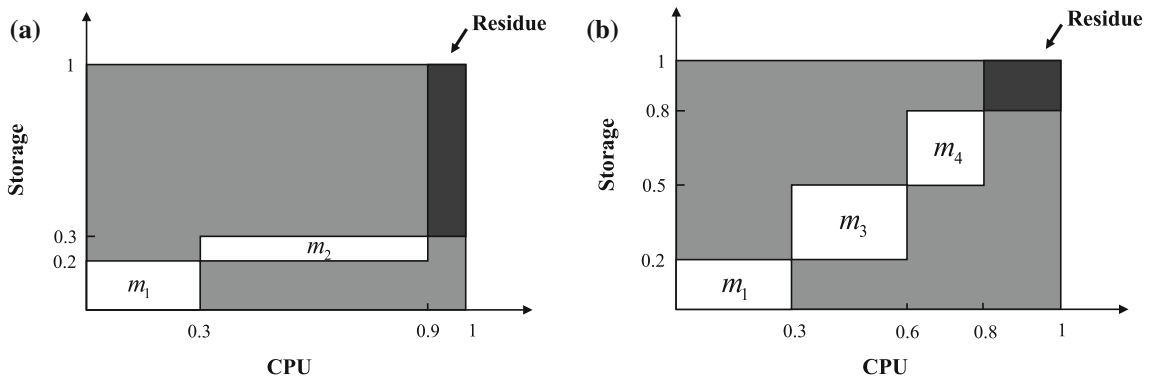


Fig. 3 Comparison of load decision. **a** unbalancing load **b** balancing load

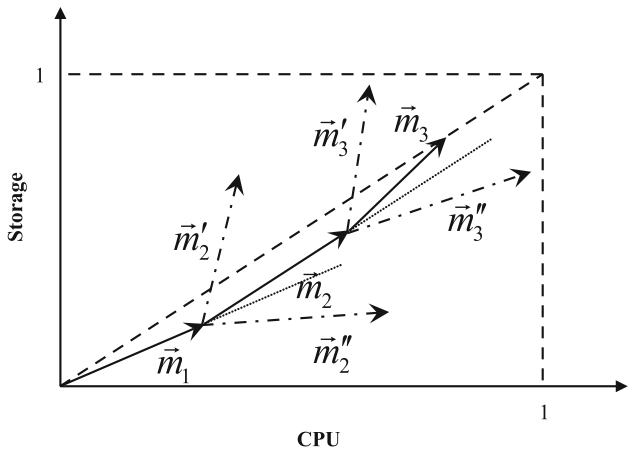


Fig. 4 A simple diagram of vector selection

Now we derive the request selection for m-dimensional resource in a vector view. Each request can be denoted as

$$\begin{aligned}\mathbf{m}_v &= \alpha_v^1\mathbf{e}_1 + \ldots + \alpha_v^j\mathbf{e}_j + \ldots + \alpha_v^m\mathbf{e}_m \\ &= \left(\alpha_v^1, \ldots, \alpha_v^j, \ldots, \alpha_v^m\right), \quad v = 1, 2, \ldots, V\end{aligned} \tag{10}$$

- *Step 2*: We solve $\arg\min_{\mathbf{m}_\theta \in \mathbf{T}} \langle \mathbf{m}_v, \mathbf{m}_\theta \rangle$ and place the result into $\mathbf{H}$. If more than one solution exist, choose the one with the largest appropriateness. Denote our choice as θ^*.
- *Step 3*: Store θ^* into $\mathbf{L}_t$, and update $\mathcal{N}$, $\mathbf{m}_{v+1}$, $\mathbf{T}$ and $\mathbf{I}_t$ for the next loop.

Comparing with Algorithm 1, or *APPGS*, the *APPLB* algorithm is more efficient in terms of the OSP's payoff. Although a bit more complex than *APPGS*, Theorem 3 shows that *APPLB* is still efficient in time consumption.

Theorem 3 *The time complexity of Algorithm 2 is polynomial.*

Proof Assume the OSP has received N user requests. At the v^*-th loop round, we need to find the candidates who fit the current resource constraint, i.e., traversing $N - v^*$ residual requests. Until the termination time, the algorithm runs for $N + (N-1) + \ldots + 1 = \frac{N^2+N}{2}$ times, where each time is only a dot product computing. In conclusion, *APPLB* has a polynomial computational complexity of $O(N^2)$. □

Algorithm 2: *APPLB* Mechanism

Input: φ_t, Λ_t, $\mathcal{N}$
Output: private set $\mathbf{L}_t$, public set $\mathbf{C}_t$

Initialization:
$V = \sum_{i=1}^{n} K_i^t$, $\mathbf{m}_v = r_{v1}\mathbf{e}_1 + \ldots + r_{vm}\mathbf{e}_m$ $(v = 1, 2, \ldots, V)$, $\mathbf{L}_t = \emptyset$, $\mathbf{C}_t = \emptyset$, $\mathbf{T} = \emptyset$, $\mathbf{H} = \emptyset$, $\mathbf{I}_t = \{\mathbf{m}_1, \ldots, \mathbf{m}_V\}$, $\mathbf{m}_0 = \mathbf{0}$
for $v = 0 : V - 1$ **do**
  for $\mathbf{m}_j \in \mathbf{I}$ **do**
    if $\mathcal{N} - \mathbf{m}_j \geqslant \mathbf{0}$ **then**
      $\mathbf{T} = \mathbf{T} \bigcup \{\mathbf{m}_j\}$
    end
  end
  $\mathbf{H} = \arg\min_{\mathbf{m}_\theta \in \mathbf{T}} \langle \mathbf{m}_v, \mathbf{m}_\theta \rangle$
  $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \mathbf{H}} \lambda(\boldsymbol{\theta})$
  $\mathbf{L}_t := \mathbf{L}_t \bigcup \{\boldsymbol{\theta}^*\}$, $\mathcal{N} := \mathcal{N} - \boldsymbol{\theta}^*$
  $\mathbf{m}_{v+1} := \mathbf{m}_v + \boldsymbol{\theta}^*$, $\mathbf{T} := \emptyset$
  $\mathbf{I} := \mathbf{I}_t - \{\boldsymbol{\theta}^*\}$
end
Return $\mathbf{L}_t, \mathbf{C}_t = \mathbf{I}_t - \mathbf{L}_t$

where $\mathbf{e}_j$ is the unit vector of the *j th* resource type. When selecting the next request denoted by $\mathbf{m}_{v+1}$, we minimize the dot product $\langle \mathbf{m}_v, \mathbf{m}_{v+1} \rangle$. The intuition of our approach is shown in Fig. 4, where we select the vector $\mathbf{m}_2$ by minimizing the dot product $\langle \mathbf{m}_1, \mathbf{m}_2 \rangle$. Selection of $\mathbf{m}_3$ also follows the same rule. Now let us depict the load balancing algorithm $\{\mathbf{L}_t, \mathbf{C}_t\} = APPLB(\boldsymbol{\varphi}_t, \Lambda_t, \mathcal{N})$.

In each outer *for*-loop, Algorithm 2 runs as follows:

To summarize the two algorithms, *APPGS* has a lower computational complexity while *APPLB* is nearer to the optimality. In the remainder of this paper, we use them as a combination, called *APP*: when the computational size is not very large, we choose *APPLB* to obtain a higher utility, but if the amount of requests is very large, we opt to apply *APPGS* instead for better time efficiency. In later sections, we will further use simulations to validate the accuracy as well as efficiency of the two algorithms.

4 OSP's decision on the private cloud

Recall that in the previous section, we have established a Stackelberg game model which requires a backward induction to seek its solution. In this section, based on the results in the previous section, we solve the second stage of this game, i.e., the OSP's decision on constructing the private cloud. In short, we will solve $\mathcal{N}^*(\mathcal{P}_c) = \arg\max_{\mathcal{N}} \Pi_L(\mathcal{N}, \mathcal{P}_c)$.

4.1 Decision of private cloud

Considering the *APP* algorithm in Sect. 3, we can rapidly compute the private set $\mathbf{L}_t$ and the public set $\mathbf{C}_t$ as

$$\{\mathbf{L}_t, \mathbf{C}_t\} = APP(\varphi_t, \Lambda_t, \mathcal{N}). \tag{11}$$

Since $\mathbf{I}_t = \mathbf{L}_t \bigcup \mathbf{C}_t$, and $\mathcal{S}_I^t = \mathcal{S}_L^t + \mathcal{S}_C^t$, we can simplify Eq. (11) to be

$$\mathcal{S}_L^t = APP(\varphi_t, \Lambda_t, \mathcal{N}) = APP(\mathcal{N}, \varphi_t). \tag{12}$$

Combining Eqs. (5) and (12), we obtain the OSP's utility as

$$\begin{aligned} \Pi_L^t(\mathcal{P}_c, \mathcal{N}) =& \mathcal{U}_t - \mathcal{N}\mathcal{P}_l - \mathcal{S}_C^t \mathcal{P}_c \\ =& \mathcal{U}_t - \left\{\mathcal{N}\mathcal{P}_l + \left[\mathcal{S}_I^t - APP(\mathcal{N}, \varphi_t)\right]\mathcal{P}_c\right\}. \end{aligned} \tag{13}$$

In Eq. (13), the terms $\mathcal{P}_l$ and $\mathcal{P}_c$ are constant and do not change over time t. Although the terms $\mathcal{U}_t$ and φ_t are time-related, they only depend on the incoming requests of users and are irrelevant to OSP's resource $\mathcal{N}$. This implies that $\mathcal{U}_t$ and φ_t can be regarded as constants when we derive the optimal $\mathcal{N}$. Therefore, the OSP's utility is a function of $\mathcal{N}$ for any given time t. For the OSP, the resources it need to prepare in the private cloud is determined by

$$\mathcal{N}^* = \arg\max_{\mathcal{N}} \int_t \Pi_L^t(\mathcal{P}_c, \mathcal{N}) \mathrm{dt}. \tag{14}$$

Since $\mathcal{U}_t$ and $\mathcal{S}_I^t \mathcal{P}_c$ are constants, we have

$$\begin{aligned} \mathcal{N}^* =& \arg\max_{\mathcal{N}} \int_t APP(\mathcal{N}, \boldsymbol{\varphi}_t)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{dt} \\ =& \arg\max_{\mathcal{N}} \int_{t\in[0,T]} \int_{\boldsymbol{\varphi}_t \in \Theta} APP(\mathcal{N}, \boldsymbol{\varphi}_t)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{d}\boldsymbol{\varphi}_t \mathrm{dt} \\ =& \arg\max_{\mathcal{N}} \mathcal{F}(\mathcal{N}, \mathcal{P}_c), \end{aligned} \tag{15}$$

where $\mathcal{F}(\mathcal{N}, \mathcal{P}_c) = \int_{t\in[0,T]} \int_{\boldsymbol{\varphi}_t \in \Theta} APP(\mathcal{N}, \boldsymbol{\varphi}_t)\mathcal{P}_c - \mathcal{N}\mathcal{P}_l \mathrm{d}\boldsymbol{\varphi}_t \mathrm{dt}$, $\Theta = \left(\mathcal{G}_1^t(k_1), \mathcal{G}_2^t(k_2), \ldots, \mathcal{G}_n^t(k_n)\right)$. In order to simplify the calculation, we proceed to show

$$\begin{aligned} \mathcal{F} =& \sum_{j=1}^{m} \int_{t\in[0,T]} \int_{\mathcal{S}_j^t} APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) p_{cj} - \mathcal{N}_j p_{lj} \mathrm{d}\mathcal{S}_j^t \mathrm{dt} \\ =& \sum_{j=1}^{m} \mathcal{F}_j(\mathcal{N}_j, p_{cj}), \end{aligned} \tag{16}$$

where $\mathcal{S}_j^t$ denotes the *j-th* resource demand at time t, and obviously, we have

$$\mathcal{S}_j^t : \mathcal{G}_j\left(s_j^t\right) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_j^t} \exp\left[-\frac{\left(s_j^t - \hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right] \sim N\left(\hat{\mu}_j^t, \hat{\sigma}_j^t\right), \tag{17}$$

where the parameter $\hat{\mu}_j = \sum_{i=1}^{n} r_{ij}\mu_i^t$, $\hat{\sigma}_j = \sqrt{\sum_{i=1}^{n} r_{ij}^2 (\sigma_i^t)^2}$.

If $\mathcal{S}_j^t < \mathcal{N}_j$, then the OSP can fully address the resource demand without purchasing public cloud resources. When $\mathcal{S}_j^t \geqslant \mathcal{N}_j$, the private cloud is fully utilized but still cannot satisfy all demands, so public cloud is needed. We can simply define $APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right)$ as

$$APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) = \begin{cases} \mathcal{S}_j^t & \sum_i^n k_{imin} \leqslant \mathcal{S}_j^t < \mathcal{N}_j, \\ \mathcal{N}_j & \mathcal{N}_j \leqslant \mathcal{S}_j^t \leqslant \sum_i^n k_{imax}, \end{cases} \tag{18}$$

where k_{imim} (k_{imax}) denotes the minimal (maximal) number of requests of type i.

We assume that $\hat{\mu}_j$ and $\hat{\sigma}_j$ have certain distributions over time t, but we do not specify any particular form for them. We can still estimate the lower bound of $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$. Combining Eqs. (16), (17) and (18), we have

$$\begin{aligned} \inf \mathcal{F}_j(\mathcal{N}_j, p_{cj}) =& \frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}} \int_0^T \rho(\mathcal{N}_j, t)\mathrm{dt} - \frac{\mathrm{p_{cj}} + 2\mathrm{p_{lj}}}{2}\mathcal{N}_\mathrm{j}\mathrm{T} \\ & - \frac{p_{cj}}{2} \int_0^T \hat{\mu}_j^t \mathrm{dt}, \end{aligned} \tag{19}$$

where

$$\begin{aligned} \rho(\mathcal{N}_j, t) =& \exp\left[-\left(\mathcal{N}_j - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right] \\ & - \exp\left[-\left(\sum_i k_{imin} - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right]. \end{aligned} \tag{20}$$

Interested readers may prefer to our appendix for more detailed derivations.

For any j, we take the derivative of $\inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)$ and have

$$\begin{aligned} \frac{\partial \inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)}{\partial \mathcal{N}_j} =& -\frac{p_{cj} + 2p_{lj}}{2} T \\ & + \frac{p_{cj}}{\sqrt{2\pi}} \int_0^T \frac{\left(\hat{\mu}_j^t - \mathcal{N}_j\right)}{\hat{\sigma}_j^t} \exp\left[-\frac{\left(\mathcal{N}_j - \hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right] \mathrm{dt}. \end{aligned} \tag{21}$$

Let the derivative equal zero and we can obtain the result $\mathcal{N}_j^*$. In conclusion, the best arrangement for OSP is $\mathcal{N}^* = \left\{\mathcal{N}_1^*, \ldots, \mathcal{N}_j^*, \ldots, \mathcal{N}_m^*\right\}$, where

$$\mathcal{N}_j^* = \arg\max_{\mathcal{N}_j} \quad \inf \mathcal{F}_j(\mathcal{N}_j, p_{cj}). \tag{22}$$

Up till now, we can see that for any given p_{cj}, $\mathcal{N}_j^*$ is determined by the above function. Since we remain general forms of the distribution on $\hat{\mu}_j^t$ and $\hat{\sigma}_j^t$, it is hard for us to obtain any closed-form solution. Specifically, our goal of this paper is to build a theoretical game model to capture the interactions between the OSP and CSP. In other words, we mainly focus on the game-based framework in hybrid cloud systems. We admit that one can still obtain the optimal $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$ with some other methods, for example, by using the numerical integration or re-formulating the system model. Analysis of different methods to approach optimal $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$ is not included in this paper, and considering the lower bound of $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$ is easy to estimated relatively, we use this method to determine $\mathcal{N}_j^*$. In what follows, we design an algorithm to approach the optimality.

4.2 An approximation algorithm to OSP

The previous subsection presents a theoretical expression of $\mathcal{N}^*$. Runge–Kutta scheme [12] is an efficient method to search the optimum, and is widely applied in practice. Therefore, we propose an efficient and accurate algorithm based on the Runge–Kutta scheme.

Let $F_j(\mathcal{N}_j) = \frac{\partial \inf \mathcal{F}_j(\mathcal{N}_j)}{\partial \mathcal{N}_j}$ and the previous problem turns to finding zeros of $F_j(\mathcal{N}_j)$. A fourth order Runge–Kutta (*RK*4) scheme of $F_j(\mathcal{N}_j)$ is depicted as Algorithm 3.

Algorithm 3: $\hat{\mathcal{N}}_j^* = RK4\left(F_j, p_{cj}\right)$ Scheme

Input: initial-value $F_j(0), F_j' = f\left(\mathcal{N}_j, F_j\right), h, \varepsilon$
Output: $\hat{\mathcal{N}}_j^*$

Initialization: $L_1 = L_2 = F_j(0), \mathcal{N}_0 = 0$
while $|L_2| > \varepsilon$ **do**
  $\mathcal{N}_0 := \mathcal{N}_0 + h$
  Calculate $F_j(\mathcal{N}_0)$
  $a_1 = f\left(\mathcal{N}_0, F_j(\mathcal{N}_0)\right)$
  $a_2 = f\left(\mathcal{N}_0 + \frac{h}{2}, F_j(\mathcal{N}_0) + \frac{h}{2}a_1\right)$
  $a_3 = f\left(\mathcal{N}_0 + \frac{h}{2}, F_j(\mathcal{N}_0) + \frac{h}{2}a_2\right)$
  $a_4 = f\left(\mathcal{N}_0 + h, F_j(\mathcal{N}_0) + ha_3\right)$
  $L_2 = L_1 + \frac{h}{6}\left(a_1 + 2a_2 + 2a_3 + a_4\right)$
end
$\hat{\mathcal{N}}_j^* = \mathcal{N}_0$
Return $\hat{\mathcal{N}}_j^*$

In Algorithm 3, h is the step size and ε is pre-fixed small real value to control the termination of this algorithm. In every loop of this algorithm, we calculate the intermediate variables a_1, a_2, a_3 and a_4, and estimate a slope to calculate the next iteration value. According to [12], the *RK*4 based algorithm has an error bound of h^5, and the total error bound is as small as h^4. When we apply a small value of h, we can approach the result very close to the optimality.

5 CSP's decision on pricing strategy

In the previous section, the OSP decides how it implements its private cloud for any given public cloud resource pricing strategy $\mathcal{P}_c$. In this section, by knowing the OSP's best response, we answer what is the optimal pricing strategy of the CSP, such that its profit can be maximized.

5.1 Problem formulation

Given the $\hat{\mathcal{N}}_j^*$ in the previous section, the CSP needs to solve $\mathcal{P}_c^* = \arg\max_{\mathcal{P}_c} \Pi_C(\mathcal{P}_c, \mathcal{N}^*(\mathcal{P}_c))$. Therefore, we have

$$\begin{aligned}
\mathcal{P}_c^* &= \arg\max_{\mathcal{P}_c} \int_0^T \mathcal{S}_C^t \mathcal{P}_c \mathrm{dt} \\
&= \arg\max_{\mathcal{P}_c} \int_0^T \left(\mathcal{S}_I^t - \mathcal{S}_L^t\right)\mathcal{P}_c \mathrm{dt} \\
&= \arg\max_{\mathcal{P}_c} \sum_{j=1}^{m} \int_0^T \left[\mathcal{S}_j^t - APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right)\right] p_{cj} \mathrm{dt} \\
&= \arg\max_{\mathcal{P}_c} \sum_{j=1}^{m} \mathcal{H}_j(p_{cj}).
\end{aligned} \tag{23}$$

The previous section gives $\mathcal{N}_j = \hat{\mathcal{N}}_j^*(p_{cj})$. Combining Eqs. (17) and (18), we have

$$\begin{aligned}
\mathcal{H}_j(p_{cj}) = &\int_0^T \int_{\sum_i k_{imin}}^{\hat{\mathcal{N}}_j^*(p_{cj})} \left(s_j^t - s_j^t\right)\mathcal{G}_j\left(s_j^t\right) p_{cj} \mathrm{ds_j^t dt} \\
&+ \int_0^T \int_{\hat{\mathcal{N}}_j^*(p_{cj})}^{\sum_i k_{imax}} \left[s_j^t - \hat{\mathcal{N}}_j^*(p_{cj})\right]\mathcal{G}_j\left(s_j^t\right) p_{cj} \mathrm{ds_j^t dt}.
\end{aligned} \tag{24}$$

Note that the first term equals 0, because the case $APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) = \mathcal{S}_j^t$ implies the OSP does not need to purchase public resources. Then we can obtain the lower bound of $\mathcal{H}_j(p_{cj})$ as

$$\begin{aligned}
\inf \mathcal{H}_j(p_{cj}) = &\frac{p_{cj}}{\sqrt{2\pi}} \int_0^T \hat{\sigma}_j^t \omega(p_{cj}, t) \mathrm{dt} \\
&+ \frac{p_{cj}}{2} \int_0^T \hat{\mu}_j^t \mathrm{dt} - \frac{1}{2}\hat{\mathcal{N}}_j^*(\mathrm{p_{cj}})\mathrm{p_{cj}T},
\end{aligned} \tag{25}$$

where

$$\omega(p_{cj},t)=\exp\left[-\left(\sum_i k_{imax}-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right] - \exp\left[-\left(\hat{\mathcal{N}}_j^*(p_{cj})-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right]. \tag{26}$$

Interested readers can find the detailed derivatives in the appendix. The result we obtain is: the solution to the CSP's utility maximization problem is $\mathcal{P}_c^*=\left(p_{c1}^*,\ldots,p_{cj}^*,\ldots,p_{cm}^*\right)$, where

$$p_{cj}^*=\arg\max_{p_{cj}} \quad \inf\mathcal{H}_j(p_{cj}). \tag{27}$$

5.2 Public pricing strategy

To further observe the feature of the solution to CSP, let us first take the derivative of $\inf\mathcal{H}_j(p_{cj})$, so we have

$$\frac{\mathrm{d}\inf\mathcal{H}_j(p_{cj})}{\mathrm{d}p_{cj}}=\frac{1}{\sqrt{2\pi}}\int_0^T\hat{\sigma}_j^t\omega(p_{cj},t)\mathrm{d}t + \frac{p_{cj}}{\sqrt{2\pi}}\int_0^T\hat{\sigma}_j^t\omega(p_{cj},t)'\mathrm{d}t+\frac{1}{2}\int_0^T\hat{\mu}_j^t\mathrm{d}t - \frac{1}{2}\hat{\mathcal{N}}_j^*(p_{cj})T-\frac{1}{2}\left[\hat{\mathcal{N}}_j^*(p_{cj})\right]'p_{cj}T. \tag{28}$$

A direct idea for public pricing strategy is to compute the price value such that the above equation equals zero. However, it is very hard to obtain a closed-form solution due to the complicated forms of the formulas. Therefore, we apply an efficient and accurate algorithm to approach the optimum. This algorithm also follows the *RK*4 methodology and is similar to Algorithm 3.

Let us present our approach in Algorithm 4, each iteration of which follows the *RK*4 method. Different from Algorithm 3, this algorithm searches the optimum point of $\inf\mathcal{H}_j(p_{cj})$. Similarly this algorithm has an error bound of h^5 in each step, so the overall error bound is h^4. By carefully choosing the value of h, we can guarantee a very small error bound.

5.2.1 Summary

Up till now, we have proposed an integrated Stackelberg game model and presented guidelines for the OSP and the CSP to make their optimal decisions. In particular, the CSP decides the pricing scheme as $\left(p_{c1}^*,\ldots,p_{cj}^*,\ldots,p_{cm}^*\right)$ for each of its public cloud resources, and the OSP decides how to implement its private cloud system by deciding the amount of resources it need to prepare, i.e., $\left(\mathcal{N}_1^*,\ldots,\mathcal{N}_j^*,\ldots,\mathcal{N}_m^*\right)$. Due to the generality of problem formulation, as well as NP-completeness of the resource allocation problem, we do not provide closed-form solutions. Alternatively, we design efficient and accurate algorithms, and derive the theoretic formulas and bounds for these decisions.

Our goal in this paper is to build a Stackelberg game model for the hybrid cloud system, aiming at capturing the interactions between a single OSP and a single CSP. Future extensions to this model could include more complex interactions that may occur between multiple OSPs and multiple CSPs.

6 Performance evaluation

In this section, we use real-trace data to evaluate the performance of our design. We first show the performance of our designed algorithms *APPGS* and *APPLB*, and then

Algorithm 4: $\hat{p}_{cj}^*=RK4S\left(\inf\mathcal{H}_j\left(p_{cj}\right)\right)$ Scheme

Input: initial-value $\inf\mathcal{H}_j(0),\left[\inf\mathcal{H}_j\left(p_{cj}\right)\right]'=g\left(p_{cj},\inf\mathcal{H}_j\left(p_{cj}\right)\right),h,\varepsilon$
Output: $\hat{p}_{cj}^*$

```
Initialization: L_1 = 0, L_2 = inf H_j(0), p_0 = 0
while |L_2 - L_1| > ε do
    p_0 := p_0 + h
    Calculate inf H_j(p_0)
    b_1 = g(p_0, inf H_j(p_0))
    b_2 = g(p_0 + h/2, inf H_j(p_0) + (h/2) b_1)
    b_3 = g(p_0 + h/2, inf H_j(p_0) + (h/2) b_2)
    b_4 = g(p_0 + h, inf H_j(p_cj) + h b_3)
    L_1 = L_2
    L_2 = L_1 + (h/6)(b_1 + 2b_2 + 2b_3 + b_4)
end
p̂*_cj = p_0
Return p̂*_cj
```

compare the utility of the OSP and the CSP when using our algorithms to decide their strategies.

6.1 Comparison of static allocation algorithms

We compare the OSP's utility when using the *APPGS* and *APPLB* algorithms, compared with its maximal possible utility (denoted by *OPT*, and obtained by exhaustive search). We assume that users' requests can be classified into three types, and each request type requires three different computing resources. We vary the request number from 1 to 150, and the proportion of each request type is fixed. Results are shown in Fig. 5, where we can see that *APPGS* achieves 85.71% utility compared to the maximal value, while *APPLB* achieves 92.65%. We run our algorithms in a normal PC with 3.20*GHz* CPU, and the total running time of *APPGS* and *APPLB* are 1.05 and 107.04 ms, respectively. Therefore, we can use *APPLB* for a higher utility when computational time is allowed, or *APPGS* for a faster calculation. In the following, we use the combination of them, denoted by *APP*.

6.2 Private cloud utility

In this and the next subsections, we evaluate our Algorithms 3 and 4. We consider an OSP with three different type of services. *Type 1* is a constant request over time. *Type 2* is a time-variant service with strong periodicity. *Type 3* represents a burst request (e.g., the Black Friday flooding in online sale websites). The data traces are depicted in Fig. 6, where the x-axis is the time domain and y-axis represents the number of requests from users. *Type 1* and *Type 3* curves are artificially generated by us, while *Type 2* curve is a real-trace data we obtained from a particular online video service company, collected every 30 min from November 4 to 18, 2012. Due to the requests from that company, we anonymize the company's title, and have normalized the values.

We consider three typical resources in cloud service, i.e., the CPU, memory, and storage; and they comprise the resource vector. Considering above type of services, we give a rational assumption that $\mathcal{R}_1 = (0.6, 0.6, 0.8)$, $\mathcal{R}_2 = (1.0, 1.0, 1.0)$ and $\mathcal{R}_3 = (1.5, 1.0, 1.0)$, respectively. Checking the simple economic relation of computing cost, we set the unit operating cost of OSP as $\mathcal{P}_l = (1.0, 0.8, 0.6)$ and the CSP's pricing strategy is $\mathcal{P}_c = \gamma \mathcal{P}_l$, where $\gamma \geqslant 1$. The coefficient γ implies the economic difference between computing resources of OSP and CSP.

There are numbers of related works dealing with tasks scheduling in hybrid cloud (see Sect. 8), but they have very different settings, and their algorithms are not comparable to ours due to different objectives. In here, we compare our design with two baseline approaches. The algorithms we consider are:

- *Stackelberg Game Model* (SGM): our approach.
- *Always-Fit-Most scheme* (AFM): the private cloud is powerful enough to deal with all requests, even at the peak time point.
- *AVeraGe scheme* (AVG): the private cloud deals with half of total requests; the other half are resorted to the public cloud.

Figure 7 shows the OSP's utility in different schemes, and the accumulative results are depicted in Fig. 8. Although possessing a peak utility value, the AFM scheme has the worst performance. The reason is that the OSP has to preserve large number of idle resources, and it is a huge waste for most of the time. The AVG scheme does not consider the cost difference using private and public cloud, and thus leads to a lower utility due to high fee charged by the CSP. Our SGM scheme achieves the greatest accumulative utility, and is obviously the best of the three approaches.

6.3 Public cloud utility

In this subsection, we compare the pricing strategy of the CSP using our approach with the optimal pricing scheme. Since we need to use exhaustive search to find the optimality, we apply a simplified setting: we consider only *Type 2* requests but omit the rest. We use Algorithm 4 to

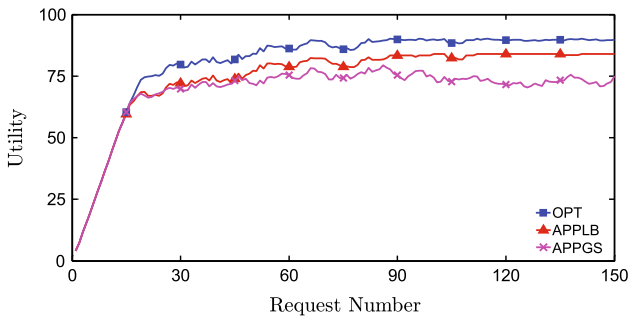


Fig. 5 Comparison of static algorithms

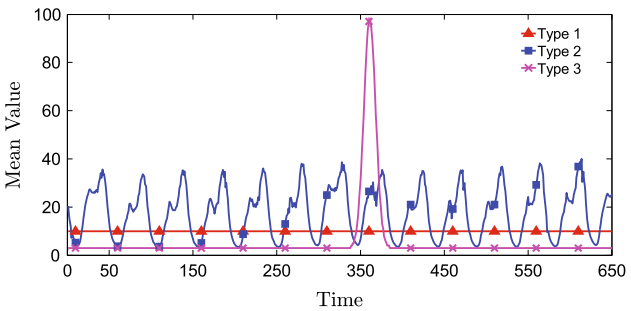


Fig. 6 Service types

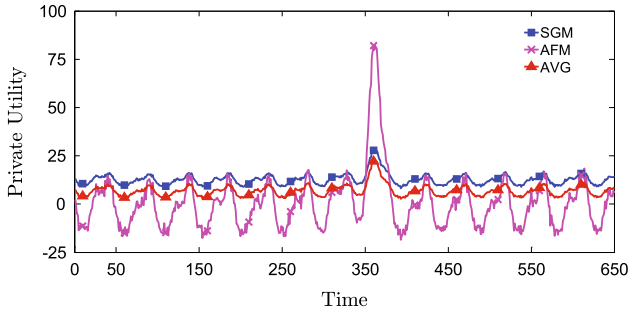


Fig. 7 Comparison of private utility

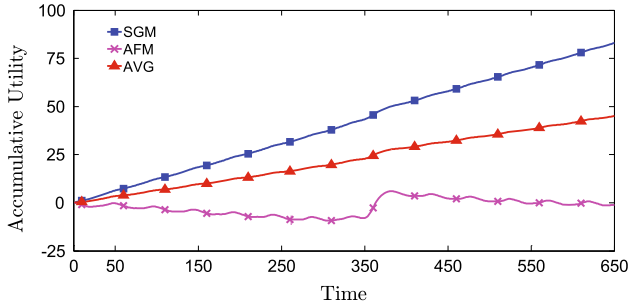


Fig. 8 Accumulative private utility

reach our solution, and exhaustive search to find the optimal price. We present our results in Fig. 9. We find that although our solution on the unit price is lower than the optimal choice, however, the CSP's utility under our solution reaches 94.7% to the maximal possible value. To summarize, via extensive simulations we validate that our algorithms can achieve near-optimal solutions with low time complexity.

6.4 Performance impact of γ

In this subsection, we show how parameters affect the private utility. Specifically, we compare the accumulative utility with different coefficient, where γ equals 1, 3, 5 respectively. Note that $\gamma = 1$ indicates that it is an ideal case where the OSP pays same for leasing resources from CSP with its local operating. The results are shown in Figs. 10, 11, 12.

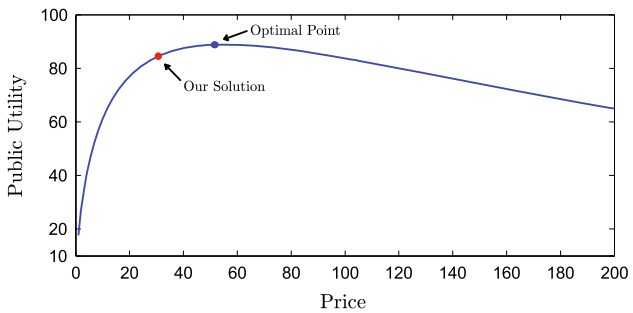


Fig. 9 Comparison of public utility

The SGM scheme obtains the best performance in above three figures. The performance of AVG decreases with the increase of coefficient γ since it pays more to the CSP. In particular, AVG's utility coincides with that of SGM in Fig. 10, and the reason is that the operating cost of OSP is same as the cost to lease resources from CSP, i.e., $\gamma = 1$. The AFM scheme is the worst when $\gamma = 1$ and $\gamma = 3$, while it overtakes the AVG in Fig. 12 since it prepares sufficient resources for the burst case. However, the AFM is inferior to the SGM scheme especially in the routine case, which is common in network service.

7 Discussion

In this section, we discuss some possible extensions to our framework and deal with some issues of previous sections.

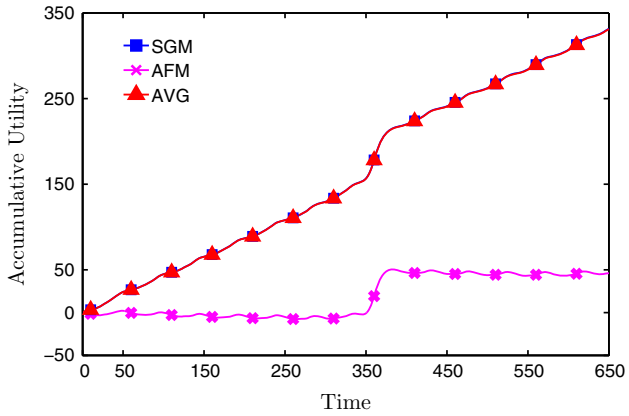


Fig. 10 $\gamma = 1.0$

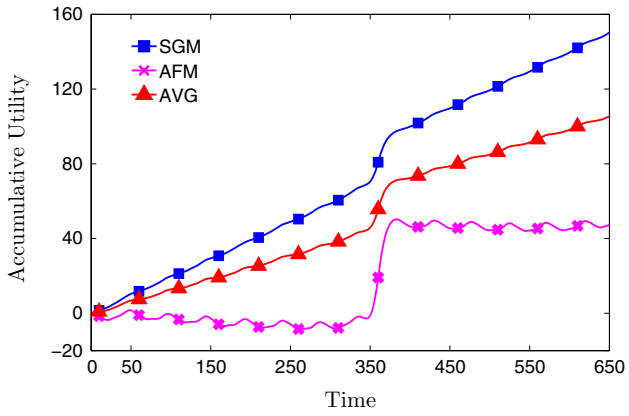


Fig. 11 $\gamma = 3.0$

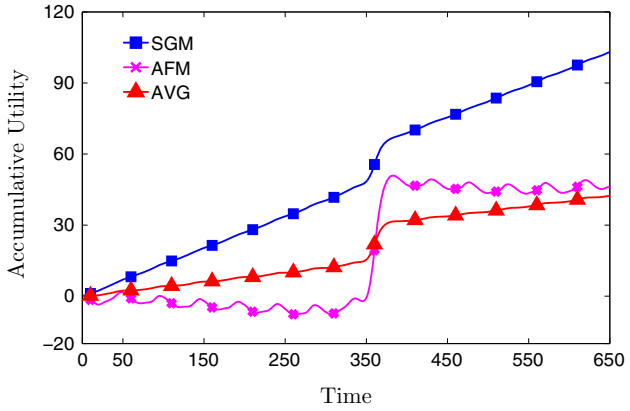


Fig. 12 $\gamma = 5.0$

7.1 Incomplete information game

In the previous sections, we assume that the CSP may decide its pricing strategy to maximize its own profit by knowing the OSP's best response (see Sect. 5). In here, an implicit condition is that the CSP can obtain the internal information from the private cloud in the OSP. One may argue how can this be achieved. One way is to do market investigation. Through multiple probe, the CSP can estimate the best response of the OSP under different scenarios.

Another possibility is to use an imperfect information game to model the interactions between the CSP and the OSP, or the Bayesian Stackelberg game. Let us now analyze the hybrid cloud with a classical algorithm, i.e., DOBSS in [13]. For the leader CSP, its pure strategy j indicates the unit price p_{cj}, while the follower OSP's pure strategy i is $\mathcal{N}_i$, where $i, j = 1, 2, \ldots, m$. We denote the CSP's policy as y, which consists of a vector of its pure strategies. The proportion of used j in policy y is denoted as y_j. For any OSP's policy type $l \in L$, x^l illustrates its vector of pure strategies, where the index set L contains all possible policies of the OSP's.

Given a priori probability set $\mathbf{H} = \{h_l : l \in X\}$, where each element h_l corresponds to each OSP's policy, the CSP solves the following problem to decide its pricing strategy:

$$\begin{aligned} \max_{y,x,a} \quad & \sum_j \sum_l \sum_i h_l \Pi_C(j,i) y_j x_i^l \\ \text{subjectto} \quad & \sum_j y_j = 1, \\ & \sum_i x_j^l = 1, \\ & 0 \leqslant a^l - \sum_j \Pi_L(j,i) y_j \leqslant (1 - x_i^l) M, \\ & y_j \in [0,1], x_i^l \in \{0,1\}, l \in \Re \end{aligned} \tag{29}$$

where a^l is the upper bound on the OSP's utility given the OSP's policy l and the CSP's policy y. We simply omit details about this optimization problem since they are similar to [13].

7.2 General request distribution

7.2.1 Poisson distribution approximation

In Sects. 4 and 5, we assume the number of incoming request to be an independent Gaussian distribution. This assumption is for analytical tractability, and it also represents some realistic properties of incoming request. Now we will show why we use the independent Gaussian distribution to depict the request's properties.

Given the request type $\mathcal{R}_i$, the number of $\mathcal{R}_i$ over a time interval is a discrete random variable that is often modeled by a Poisson distribution, where $i = 1, 2, \cdots, n$. In most analytical cases, we want the corresponding distribution to be a continuous one, whereas the Poisson distribution is obviously discrete. A simple and direct idea is that we can use some other distributions to approximate a Poisson distribution, for example, the Gaussian distribution which is widely applied in engineering statistics. In reality, we have the following theorem.

Theorem 4 *Suppose that X_i is the number of type $\mathcal{R}_i$ and $X_i \sim P(\lambda_i)$, we have that the variable*

$$K_i = \frac{X_i - \lambda_i}{\sqrt{\lambda_i}}$$

is approximately a standard Gaussian random variable. And the approximation is good for $\lambda_i > 5$.

We simply omit the proof and interested readers may refer to references [14, 15] for more details. Without of generality, we extend the standard case to a general one, i.e., $K_i \sim N(\mu_i, \sigma_i)$. Based on this approximation, we can use the Gaussian distribution to depict the incoming request since $\lambda_i > 5$ holds in our scheme and therefore, we obtain the continuity correction.

We should also note that in some other scenarios, the arrival rate of request is naturally modeled as a Gaussian distribution and there is no need to apply previous approximation. For example, when staffing a calling center [16] or modeling the arrival of public information [17]. In next part, we will show how to deal with the situation where the request distribution does not follow Gaussian.

7.2.2 Extend to general case

Previous parts make the Gaussian distribution to depict the properties of incoming request. However, we cannot rule out

possibilities that the request distribution does not follow Gaussian. Now let us discuss how to address the general case of request distribution. Suppose the number of request type $\mathcal{R}_i$ is a random variable $K_i \sim \mathcal{J}_i(k_i)$. The definition and the instantaneous amount of incoming requests are the same as Eqs. (1) and (3). Now we present a framework for the general situation.

- *Step 1: Static Optimization.* In this step, the OSP allocates the incoming requests into the private and public clouds, by applying the approximation algorithms in Sect. 3. This can be expressed as

$$\{\mathbf{L}_t, \mathbf{C}_t\} = APP(\varphi_t, \Lambda_t, \mathcal{N}), \tag{30}$$

  where $\mathbf{L}_t$ and $\mathbf{C}_t$ are the private and cloud set, respectively.
- *Step 2: OSP's Construction.* Based on the private set $\mathbf{L}_t$, the OSP can obtain the its own required resource denoted as $\mathcal{S}_L^t$. The OSP's utility is

$$\Pi_L^t = \mathcal{U}_t - \mathcal{N}\mathcal{P}_l - \left(\mathcal{S}_I^t - \mathcal{S}_L^t\right)\mathcal{P}_c. \tag{31}$$

  Therefore, the OSP can arrange its private cloud as

$$\mathcal{N}^* = \arg\max_{\mathcal{N}} \int_T \Pi_L^t \mathrm{d}t. \tag{32}$$

- *Step 3: CSP's Decision.* The CSP can compute its required resource $\mathcal{S}_C^t$ when knowing the cloud set $\mathbf{C}_t$. Therefore, the CSP's utility can be shown as

$$\Pi_C^t = \mathcal{S}_C^t \mathcal{P}_c|_{\mathcal{N}^*}, \tag{33}$$

  and it can update its pricing strategy as

$$\mathcal{P}_c^* = \arg\max_{\mathcal{P}_c} \int_T \Pi_C^t|_{\mathcal{N}^*} \mathrm{d}t. \tag{34}$$

The above steps present a framework for general distributions of incoming request. It is difficult to obtain a closed-form solution, but can be addressed by heuristic algorithms. Deriving the heuristics of general cases is beyond the scope of this paper.

8 Related work

The resource/request allocation problems in the cloud have attracted a lot of attentions. Zhen et al. [18] presented a virtualization system to dynamically allocate data center resources and support green computing. Alicherry and Lakshman [19] developed efficient resource allocation algorithms in distributed clouds. The objective is minimizing the latency between selected data centers. With a ranking mechanism, Ergu et al. [20] modeled the task-oriented problem of resource allocation in a cloud computing environment. Dán et al. [21] considered the dynamic content allocation problem for a content delivery system, which combines cloud-based storage with low cost dedicated servers. In [22], Hao et al. proposed a non-prior method for VMs allocation in a distributed cloud to decrease the network cost. Shi et al. in [23] represented the first online combinatorial and truthful auction mechanism to model dynamic provisioning of VMs in a cloud paradigm.

Meanwhile, a number of research has been focusing on the hybrid cloud system as well. In [24], Armbrust et al. showed the hybrid cloud can achieve better performance than single public or private cloud. Bittencourt et al. showed the main characteristics of scheduling in [25], and proposed a cost optimization algorithm for the hybrid cloud in [26]. Concerning resource allocation and performance, Lee et al. proposed a cluster-based architecture to allocate resources in [27]. Authors in [28] discussed hybrid cloud management with deadline constraints. Zhou et al. [29] addressed the privacy issue in a hybrid cloud, where sensitive data are kept in trusted private cloud while insensitive data are moved to the public cloud. Beloglazov et al. [30] proposed several resource allocation policies and algorithms to realize Green IT.

Up till now, we have not found any work that combine the design of resource allocation and hybrid cloud. Different from all these previous works, we consider an online service application with specific request arrival patterns, and design the combinatorial resource allocation and pricing strategies in a hybrid cloud system.

9 Conclusion and future work

In this paper, we propose a hybrid cloud design to address the heterogeneous and time varying requests in online service applications. In particular, we use a Stackelberg game model to capture the interactions between the public cloud and the private cloud, based on which we answer (1) for the online service provider, how to implement the private cloud system in terms of local resource preparation, and (2) for the public cloud provider, how to decide the optimal prices for the cloud resources so as to maximize its profit. We prove the NP-completeness of the resource allocation problem, and propose efficient and effective algorithms to approach the optimality. We show, both theoretically and empirically, that our algorithms is time-efficient and achieves near-optimal solutions. We believe this gives important guidelines to design practical hybrid cloud systems.

This paper is merely an initial work on the particular topic of hybrid systems, and we only consider one OSP and one CSP in the hybrid cloud system. Future extensions to this paper could include more complex interactions that may occur between multiple OSPs and multiple CSPs.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant Nos. 61271176, 61401334, 61571350 and 61402287, the Fundamental Research Funds for the Central Universities (BDY021403), the 111 Project (B08038) and Shanghai Yangfan Project (No. 14YF1401900).

Appendix 1: Derive the infimum of $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$

In Sect. 4, we combine Eqs. (16), (17) and (18), and thus we obtain each dimensional $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$ in Eq. (35). Note that we divide the $\mathcal{F}_j(\mathcal{N}_j, p_{cj})$ into three individual parts, i.e., **A**, **B** and **C**.

$$
\begin{aligned}
\mathcal{F}_j &= \int_{t\in[0,T]}\int_{s_j^t}\left[APP_j\left(\mathcal{N}_j,\mathcal{S}_j^t\right)p_{cj}-\mathcal{N}_j p_{lj}\right]\mathcal{G}_j\left(s_j^t\right)\mathrm{ds_j^t dt}\\
&=\int_0^T\int_{\sum_i k_{imin}}^{\mathcal{N}_j}\left(s_j^t p_{cj}-\mathcal{N}_j p_{lj}\right)\mathcal{G}_j\left(s_j^t\right)\mathrm{ds_j^t dt}\\
&\quad+\int_0^T\int_{\mathcal{N}_j}^{\sum_i k_{imax}}\left(\mathcal{N}_j p_{cj}-\mathcal{N}_j p_{lj}\right)\mathcal{G}_j\left(s_j^t\right)\mathrm{ds_j^t dt}\\
&=\int_0^T\int_{\sum_i k_{imin}}^{\mathcal{N}_j}\left[s_j^t p_{cj}-\mathcal{N}_j p_{lj}\right]\cdot\frac{1}{\sqrt{2\pi}\hat{\sigma}_j^t}\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t dt}\\
&\quad+\int_0^T\int_{\mathcal{N}_j}^{\sum_i k_{imax}}\left[\mathcal{N}_j p_{cj}-\mathcal{N}_j p_{lj}\right]\cdot\frac{1}{\sqrt{2\pi}\hat{\sigma}_j^t}\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t dt}\\
&=\int_0^T\frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\mathcal{N}_j}s_j^t\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t dt}\\
&\quad+\int_0^T\frac{\mathcal{N}_j p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\mathcal{N}_j}^{\sum_i k_{imax}}\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t dt}\\
&\quad-\int_0^T\frac{\mathcal{N}_j p_{lj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\sum_i k_{imax}}\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t dt}\\
&=\int_0^T\mathbf{A}\mathrm{dt}+\int_0^T\mathbf{B}\mathrm{dt}-\int_0^T\mathbf{C}\mathrm{dt}.
\end{aligned}
\tag{35}
$$

We consider the error function in the probability analysis: $erf(z)=\frac{2}{\sqrt{\pi}}\int_0^z\exp(-x^2)\mathrm{dx}$. Then **A**, **B** and **C** can be computed as Eq. (36–38), respectively.

$$
\begin{aligned}
\mathbf{A} &= \frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\mathcal{N}_j}s_j^t\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\
&=\frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\mathcal{N}_j}\left(s_j^t-\hat{\mu}_j^t+\hat{\mu}_j^t\right)\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\
&=\frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\mathcal{N}_j}\left(s_j^t-\hat{\mu}_j^t\right)\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\
&\quad+\frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\mathcal{N}_j}\hat{\mu}_j^t\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}
\end{aligned}
\tag{36}
$$

$$
\begin{aligned}
&=\frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\exp\left[-\left(\mathcal{N}_j-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right]\\
&\quad-\frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\exp\left[-\left(\sum_i k_{imin}-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right]\\
&\quad+\frac{p_{cj}\hat{\mu}_j^t}{2}erf\left[\left(\mathcal{N}_j-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right]\\
&\quad-\frac{p_{cj}\hat{\mu}_j^t}{2}erf\left[\left(\sum_i k_{imin}-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right],
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{B} &= \frac{\mathcal{N}_j p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\mathcal{N}_j}^{\sum_i k_{imax}}\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\
&=\frac{\mathcal{N}_j p_{cj}}{2}\cdot\frac{2}{\sqrt{\pi}}\int_0^{\frac{\sum_i k_{imax}-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}}\exp\left(-\tau^2\right)\mathrm{d}\tau\\
&\quad-\frac{\mathcal{N}_j p_{cj}}{2}\cdot\frac{2}{\sqrt{\pi}}\int_0^{\frac{\mathcal{N}_j-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}}\exp(-\tau^2)\mathrm{d}\tau\\
&=\frac{\mathcal{N}_j p_{cj}}{2}erf\left[\left(\sum_i k_{imax}-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right]\\
&\quad-\frac{\mathcal{N}_j p_{cj}}{2}erf\left[\left(\mathcal{N}_j-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right],
\end{aligned}
\tag{37}
$$

and with the last part

$$\begin{aligned}\mathbf{C} &= \frac{\mathcal{N}_j p_{lj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\sum_i k_{imin}}^{\sum_i k_{imax}} \exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\ &= \frac{\mathcal{N}_j p_{lj}}{2}\cdot\frac{2}{\sqrt{\pi}}\int_0^{\frac{\sum_i k_{imax}-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}} \exp\left(-\tau^2\right)\mathrm{d}\tau\\ &\quad -\frac{\mathcal{N}_j p_{lj}}{2}\cdot\frac{2}{\sqrt{\pi}}\int_0^{\frac{\sum_i k_{imin}-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}} \exp\left(-\tau^2\right)\mathrm{d}\tau\\ &= \frac{\mathcal{N}_j p_{lj}}{2} erf\left[\left(\sum_i k_{imax}-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right]\\ &\quad -\frac{\mathcal{N}_j p_{lj}}{2} erf\left[\left(\sum_i k_{imin}-\hat{\mu}_j^t\right)/\sqrt{2}\hat{\sigma}_j^t\right].\end{aligned} \tag{38}$$

Conclude Eq. (36) to (38), we can derive Eq. (35) as

$$\mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right) = \int_0^T \mathbf{A}+\mathbf{B}-\mathbf{C}\mathrm{dt} \tag{39}$$

Due to the generality of problem formulation, it is hard to obtain an accurate formulas to Eq. (35). Instead, combined with Eq. (35) and properties of $erf(z)$, we can obtain the lower bound of $\mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)$ as

$$\begin{aligned}\inf \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right) =& \frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\int_0^T \rho\left(\mathcal{N}_j, t\right)\mathrm{dt}\\ &-\frac{p_{cj}}{2}\int_0^T \hat{\mu}_j^t\mathrm{dt} - \frac{\mathrm{p_{cj}}+2\mathrm{p_{lj}}}{2}\mathcal{N}_\mathrm{j}\mathrm{T}\\ \leqslant& \mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right),\end{aligned} \tag{40}$$

where

$$\begin{aligned}\rho\left(\mathcal{N}_j, t\right) =& \exp\left[-\left(\mathcal{N}_j-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right]\\ &-\exp\left[-\left(\sum_i k_{imin}-\hat{\mu}_j^t\right)^2/2\left(\hat{\sigma}_j^t\right)^2\right].\end{aligned} \tag{41}$$

Note that a natural assumption is $\sum_i k_{imin} \leqslant \hat{\mu}_j^t \leqslant \sum_i k_{imax}$. Up till now, we obtain the infimum of $\mathcal{F}_j\left(\mathcal{N}_j, p_{cj}\right)$ in Sect. 4.

Appendix 2: Derive the infimum of $\mathcal{H}_j\left(p_{cj}\right)$

In Sect. 5, we formulate each dimensional $\mathcal{H}_j\left(p_{cj}\right)$ as Eq. (24). Note that the first term equals 0, because the case $APP_j\left(\mathcal{N}_j, \mathcal{S}_j^t\right) = \mathcal{S}_j^t$ implies the OSP does not need to purchase public resources. Thus, we can rewrite Eq. (24) as follows:

$$\begin{aligned}\mathcal{H}_j\left(p_{cj}\right) =& \int_0^T \int_{\sum_i k_{imin}}^{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)} \left(s_j^t-s_j^t\right)\mathcal{G}_j\left(s_j^t\right)p_{cj}\mathrm{ds_j^t dt}\\ &+\int_0^T \int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[s_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]\mathcal{G}_j\left(s_j^t\right)p_{cj}\mathrm{ds_j^t dt}\\ =& 0+\int_0^T \mathbf{D}\mathrm{dt}.\end{aligned} \tag{42}$$

Now we compute the term **D** as Eq. (43).

$$\begin{aligned}\mathbf{D} =& \int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[s_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]\mathcal{G}_j\left(s_j^t\right)p_{cj}\mathrm{ds_j^t}\\ =& \frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[s_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\ =& \frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[s_j^t-\hat{\mu}_j^t\right]\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\ &+\frac{p_{cj}}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \left[\hat{\mu}_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]\exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\ =& \frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\int_{-\frac{\left(\hat{\mathcal{N}}_j^*\left(p_{cj}\right)-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}}^{\frac{\left(\sum_i k_{imax}-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}} \exp(\tau)\mathrm{d}\tau\\ &+\frac{p_{cj}\left[\hat{\mu}_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]}{\sqrt{2\pi}\hat{\sigma}_j^t}\int_{\hat{\mathcal{N}}_j^*\left(p_{cj}\right)}^{\sum_i k_{imax}} \exp\left[-\frac{\left(s_j^t-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\mathrm{ds_j^t}\\ =& \frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\exp\left[-\frac{\left(\sum_i k_{imax}-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\\ &-\frac{p_{cj}\hat{\sigma}_j^t}{\sqrt{2\pi}}\exp\left[-\frac{\left(\hat{\mathcal{N}}_j^*\left(p_{cj}\right)-\hat{\mu}_j^t\right)^2}{2\left(\hat{\sigma}_j^t\right)^2}\right]\\ &+\frac{p_{cj}\left[\hat{\mu}_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]}{2} erf\left(\frac{\sum_i k_{imax}-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}\right)\\ &-\frac{p_{cj}\left[\hat{\mu}_j^t-\hat{\mathcal{N}}_j^*\left(p_{cj}\right)\right]}{2} erf\left(\frac{\hat{\mathcal{N}}_j^*-\hat{\mu}_j^t}{\sqrt{2}\hat{\sigma}_j^t}\right).\end{aligned} \tag{43}$$

Similar to the previous derivations in Appendix 1, we can have

$$\begin{aligned}\inf \mathcal{H}_j(p_{cj}) =& \frac{p_{cj}}{\sqrt{2\pi}} \int_0^T \hat{\sigma}_j^t \omega(p_{cj}, t) \mathrm{dt} \\ &+ \frac{p_{cj}}{2} \int_0^T \hat{\mu}_j^t \mathrm{dt} - \frac{1}{2} \hat{\mathcal{N}}_{\mathrm{j}}^* (\mathrm{p_{cj}}) \mathrm{p_{cj}} \mathrm{T} \\ &\leqslant \mathcal{H}_j(p_{cj}),\end{aligned} \tag{44}$$

where

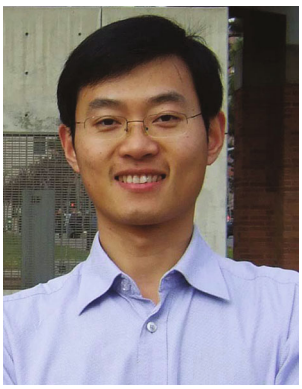
$$\begin{aligned}\omega(p_{cj}, t) = & \exp\left[-\left(\sum_i k_{imax} - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right] \\ & - \exp\left[-\left(\hat{\mathcal{N}}_j^*(p_{cj}) - \hat{\mu}_j^t\right)^2 / 2\left(\hat{\sigma}_j^t\right)^2\right].\end{aligned} \tag{45}$$

Up till now, we obtain the infimum of $\mathcal{H}_j(p_{cj})$ in Sect. 5.

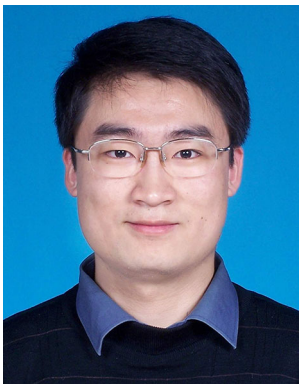
References

1. How Alibaba catered to USD 3 billion sales in a day. http://www.infoq.com/news/2012/12/interview-taobao-tmall.
2. de Castro Silva, J. L., Soma, N. Y., & Maculan, N. (2003). A greedy search for the three-dimensional bin packing problem: The packing static stability case. *International Transactions in Operational Research*, *10*(2), 141–153.
3. Panigrahy, R., Talwar, K., Uyeda, L., & Wieder, U. (2011). *Heuristics for vector bin packing*. Microsoft: Technical Report.
4. Osborne, M. J. (2004). *An introduction to game theory*. Oxford: Oxford University Press.
5. Wu, W., Lui, J. C., & Ma, R. T. (2013). On incentivizing upload capacity in P2P-VoD systems: Design, analysis and evaluation. *Computer Networks*, *57*(7), 1674–1688.
6. Jünger, M., Liebling, T. M., Naddef, D., et al. (2009). *50 Years of integer programming 1958–2008*. NewYork: Springer.
7. Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations series. The IBM research symposia series* (pp. 85–103). Springer.
8. Fréville, A. (2004). The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, *155*(1), 1–21.
9. Puchinger, J., Raidl, G. R., & Pferschy, U. (2006). The core concept for the multidimensional knapsack problem. In *Evolutionary computation in combinatorial optimization, series*. Lecture Notes in Computer Science (vol. 3906, pp. 195–208).
10. Li, C., Liu, Z., Geng, X., Dong, M., Yang, F., Gan, X., et al. (2014). Two dimension spectrum allocation for cognitive radio networks. *IEEE Transactions on Wireless Communications*, *13*(3), 1410–1423.
11. Singh, A., Korupolu, M., & Mohapatra, D. (November 2008). Server-storage virtualization: integration and load balancing in data centers. In *Proceedings of the ACM/IEEE conference on supercomputing* (pp. 1–12).
12. Jameson, A., Schmidt, W., & Turkel, E. (June 1981). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *Fluid and plasma dynamics conference* (pp. 1–15).
13. Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., & Kraus, S. (2008). Efficient algorithms to solve Bayesian Stackelberg games for security applications. In *ACM AAMAS* (pp. 895–902).
14. Montgomery, D. C., Runger, G. C., & Hubele, N. F. (2009). *Engineering statistics* (5th ed.). Hoboken: Wiley.
15. Montgomery, D. C., & Runger, G. C. (2010). *Applied statistics and probability for engineers* (5th ed.). Hoboken: Wiley.
16. Whitt, W. (2006). Staffing a calling center with uncertain arrival rate and absenteeism. *Production and Operations Management*, *15*(1), 88–102.
17. Melvin, M., & Yin, X. (2000). Public information arrival, exchange rate volatility, and quote frequency. *The Economic Journal*, *110*(465), 644–661.
18. Xiao, Z., Song, W., & Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, *24*(6), 1107–1117.
19. Alicherry, M., & Lakshman, T. (2012). Network aware resource allocation in distributed clouds. In *IEEE INFOCOM* (pp. 963–971).
20. Ergu, D., Kou, G., Peng, Y., Shi, Y., & Shi, Y. (2013). The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing*, *64*(3), 835–848.
21. Dán, G., & Carlsson, N. (2014). Dynamic content allocation for cloud-assisted service of periodic workloads. In *IEEE INFOCOM* (pp. 853–861).
22. Hao, F., Kodialam, M., Lakshman, T. V., & Mukherjee, S. (April 2014). Online allocation of virtual machines in a distributed cloud. In *IEEE INFOCOM* (pp. 10–18).
23. Shi, W., Zhang, L., Wu, C., Li, Z., & Lau, F. C. (June 2014). An online auction framework for dynamic resource provisioning in cloud computing. In *ACM SIGMETRICS* (pp. 71–83).
24. Armbrust, M., Fox, A., Griffith, R., et al. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58.
25. Bittencourt, L. F., Madeira, E. R. M., & da Fonseca, N. L. S. (2012). Scheduling in hybrid clouds. *IEEE Communications Magazine*, *50*(9), 42–47.
26. Bittencourt, L. F., & Madeira, E. R. M. (2011). HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, *2*(3), 207–227.
27. Lee, G., Chun, B., & Katz, R. H. (2011). Heterogeneity-aware resource allocation and scheduling in the cloud. In *Proceedings of HotCloud* (pp. 1–5).
28. den Bossche, R. V., Vanmechelen, K., & Broeckhove, J. (2010). Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In *International conference on cloud computing (CLOUD)* (pp. 228–235).
29. Zhou, Z., Zhang, H., Du, X., Li, P., & Yu, X. (2013). Prometheus: Privacy-aware data retrieval on hybrid cloud. In *IEEE INFOCOM* (pp. 2643–2651).
30. Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, *28*(5), 755–768.

Zhe Liu received the B.Eng. degree in Telecommunication Engineering from Hebei University, Baoding, China, in 2012, and is currently pursuing the Ph.D. degree in Telecommunication Engineering, at Xidian University, Xi'an, China. His current research interests include the dynamic radio resource management in cognitive radio networks, big data, and information centric networks.



Changle Li received the Ph.D. degree in communication and information systems from Xidian University, China in 2005. Since then, he conducted his postdoctoral research in Canada and the National Institute of Information and Communications Technology (NICT), Japan, respectively. He has been a visiting scholar at the University of Technology Sydney (UTS) and is currently a professor with the State Key Laboratory of Integrated Services Networks, Xidian University. He is an IEEE Senior Member and his research interests include intelligent transportation systems, vehicular networks, mobile ad hoc networks, and wireless sensor networks.



Weijie Wu received the B.Sc. degree in electronic and information science and technology from Peking University, Beijing, China, in July 2008, and the Ph.D. degree in computer science from The Chinese University of Hong Kong, Hong Kong, in August 2012. He is now a Researcher with Future Network Theory Laboratory of 2012 Labs, Huawei Technologies Co. Ltd., Hong Kong. Before that, he was an Assistant Professor with Shanghai Jiao Tong University, Shanghai, China, a Research Fellow with National University of Singapore, Singapore, and a Postdoctoral Fellow with The Chinese University of Hong Kong. His research interests include computer networks from mathematical modeling and economic perspectives, network science, network economics, and network optimization.

Riheng Jia received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology, China, in 2012. He is currently pursuing the Ph.D. degree in computer science from Shanghai Jiao Tong University. His research of interests is in the area of wireless networks and energy harvesting communication.