Peer-to-Peer Netw. Appl. (2016) 9:28–41
DOI 10.1007/s12083-014-0309-8

A scalable and automatic mechanism for resource allocation in self-organizing cloud

Xiaotong Wu · Meng Liu · WanChun Dou · Longxiang Gao · Shui Yu

Received: 15 June 2014 / Accepted: 29 August 2014 / Published online: 2 October 2014

Abstract Taking advantage of the huge potential of consumers' untapped computing power, self-organizing cloud is a novel computing paradigm where the consumers are able to contribute/sell their computing resources. Meanwhile, host machines held by the consumers are connected by a peer-to-peer (P2P) overlay network on the Internet. In this new architecture, due to large and varying multitudes of resources and prices, it is inefficient and tedious for consumers to select the proper resource manually. Thus, there is a high demand for a scalable and automatic mechanism to accomplish resource allocation. In view of this challenge, this paper proposes two novel economic strategies based on mechanism design. Concretely, we apply the Modified Vickrey Auction (MVA) mechanism to the case where the resource is sufficient; and the Continuous Double Auction (CDA) mechanism is employed when the resource is insufficient. We also prove that aforementioned mechanisms have dominant strategy incentive compatibility. Finally, extensive experiment results are conducted to verify the performance of the proposed strategies in terms of procurement cost and execution efficiency.

Keywords Self-organizing cloud · Mechanism design · Dynamic pricing · Resource allocation · Peer-to-peer networks

X. Wu · M. Liu · W. Dou (✉)
The State Key Laboratory for Novel Software Technology, the Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
e-mail: douwc@nju.edu.cn

X. Wu
e-mail: wxt199003@gmail.com

M. Liu
e-mail: mengliunju@gmail.com

L. Gao · S. Yu
School of Information Technology, Deakin University, Burwood, VIC 3125, Australia

L. Gao
e-mail: longxiang.gao@deakin.edu.au

S. Yu
e-mail: shui.yu@deakin.edu.au

1 Introduction

Cloud computing is an emerging paradigm, which enables on-demand provisioning of computational and storage resources [1, 2]. Existing business cloud platforms, such as Amazon EC2 [3], Microsoft Azure [4], GoogleFS [5], offer an effective way to reduce both capital and operational expenditure of consumers. However, in these cloud platforms, the huge potential of consumers' untapped computing power has been ignored, especially with the fast growth of their computing capacities. To overcome this disadvantage, self-organizing cloud (or called customer-provided cloud) has been proposed [6, 7]. Meanwhile, this newborn paradigm has attracted an increasing number of practitioners from industry [8]. While this paradigm makes cloud computing better, it also brings new and challenging problems. One important problem in self-organizing cloud is the resource allocation problem. The focus of the resource allocation problem is on how to select the proper instances for consumers. Actually, resource allocation is an important but not yet well unexplored area in self-organizing cloud.

In recent years, various techniques have been proposed to address the aforementioned problem. In self-organizing cloud, every joined host, either a public server or a desktop computer, serves as an individual node on a structured P2P overlay network [9]. It brings the resource query and

allocation problems. Faced with these problems, Di et al. [9] designed a resource discovery protocol, Proactive Index Diffusion CAN (PID-CAN), which can proactively diffuse resource indexes over the nodes and randomly route query messages among them. Furthermore, Di et al. [7] proposed a fully distributed, VM-multiplexing resource allocation scheme to manage decentralized resources. On the other hand, Wang et al. [6] investigated SpotCloud, the first business self-organizing cloud, and found that it is very friendly to individual customers when they seek to run short-term tasks at minimum costs. They also proposed a mechanism for cloud service providers to recommend short-listed instances to customers. In addition, some similar works have been proposed in [10, 11]. These advanced techniques enable computing resources to be allocated to meet the elastic needs of cloud consumers.

However, much less attention in previous works is paid to the following characteristics and limitations of participants in self-organizing cloud. First of all, there is a huge difference among resources provided by participants in self-organizing cloud in terms of quality and volume, where providers have different expected prices about their resources. As a result, it is not appropriate to use a unified unit price for all resources. In addition, when a consumer submits a task, there are lots of providers who offer various versions of resources at different prices and with varying quality-of-service parameters. It is quite complex and challenging for consumers to obtain the optimal resource with a desired quality within the budget. More specifically, the number of providers and their offerings in self-organizing cloud changes continually. Given the large and varying multitudes of providers, it is inefficient and tedious to select the most appropriate one manually. Therefore, there is a high demand for a scalable and automatic mechanism to accomplish resource allocation in self-organizing cloud.

We consider that *dynamic pricing* turns out to be a good choice for these problems. Dynamic pricing is a pricing strategy where buyers and sellers actively engage in the price discovery process. Besides, dynamic pricing increases the welfare of the user, where it facilitates healthy competition among vendors and increases the efficiency of cloud resource usage [12]. One way to implement dynamic pricing is the auction [13]. In this paper, the proposed mechanisms based on the *Vickrey auction* (also called the second-price sealed-bid auction). Under the Vickrey auction, bidders submit sealed bids after they have been told that the highest bidder wins the item and pays a price equal to the second-highest bid [14]. Moreover, the mechanisms proposed in this paper utilize *reverse auction* and *double auction*. In a reverse auction, when sellers compete to obtain business from the buyers, they undercut with each other and prices will typically decrease. Double auction is a multilateral process in which buyers and sellers submit bid and ask prices to an auctioneer simultaneously, which determines a clearing price for the sale.

Besides, we introduce mechanism design for accomplishing the resource allocation. Mechanism design is the sub-field of microeconomics and game theory. It considers how to implement good system-wide solutions to problems that involve multiple self-interested agents, one of which has private information about preferences [15]. The main focus of mechanism design is on the design of institutions that satisfy certain objectives, assuming that individuals interacting through the institution will act strategically and may hold private information that is relevant to the decision at hand [16]. Actually, mechanism design has been widely applied to cloud computing [17–19].

To address the resource allocation problem in self-organizing cloud, our work utilizes the aforementioned techniques. To the best of our knowledge, the work presented in this paper is the first to propose a scalable and automatic mechanism to perform resource allocation in self-organizing cloud. To this end, the contributions of the proposed work can be summarized as follows:

- We propose two novel mechanisms to accomplish resource allocation in self-organizing cloud. They are MVA and CDA mechanisms, respectively. They offer incentive for the providers to truthfully report the price according to their individual conditions. A consumer gets an appropriate resource with the minimized cost.
- Our mechanisms enable an automatic resource allocation mechanism from cloud planner. It helps consumers to select an appropriate resource within the budget and of the desired quality.
- We prove that the above mechanisms have dominant strategy incentive compatibility. Experiment results demonstrate that CDA mechanism is suitable when the resource is insufficient, while MVA mechanism is fit when the resource is sufficient.

The remainder of the paper is organized as follows. The system description is presented in Section 2. We formulate the problem in Section 3. We propose our MVA and CDA mechanisms in Section 4. The cloud planner module is specified in Section 5. Experiments' results are shown in Section 6. We discuss related work in Section 7. Finally, we conclude this paper with future work in Section 8.

2 System description

For simplicity, we consider a self-organizing cloud as a model involving three different entities as illustrated in Fig. 1. The *resource consumer* is a player using resources; the *resource provider* offers resources for usage on payment; the *cloud planner* is a middleware that interacts with

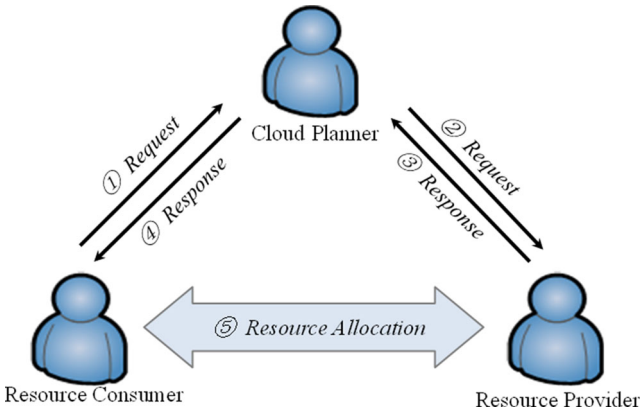

Fig. 1 The model of self-organizing cloud

the resource consumer and the resource provider, respectively, and responsible for choosing a proper resource from providers for consumers. We note that in this paper the resource refers to an instance comprising the combination of CPU, memory, storage, and networking capacity and so on.

When a consumer wants to execute a task, a resource request is submitted by the consumer to the cloud planner. Then the cloud planner sends the request to all resource providers. The winner and the transaction price of this resource request are determined by the cloud planner based on bids from resource providers. The result is informed to the resource consumer, and resource allocation between this consumer and the selected provider is executed.

We suppose that the participants in self-organizing cloud, providers and consumers, have the following characteristics.

- Interactions among the participants is in a strategic way. The participants are rational in the sense of having the explicit objective of maximizing their own individual payoffs.
- Each provider is independent and autonomy. Certain information held by each provider, such as the price interval, cost and resource quality, is private, and only the provider would know about this; wher-eas other participants do not know this information. Moreover, the provider has the ability to determine whether to sell the resources or not.
- Each participant has a set of choice strategies that are available to her/him. We assume that the participants are intelligent where they can determine their best response strategy.

In a word, resource providers have respective resources and corresponding unit prices, and are able to determine whether to sell the resources or not. Resource consumers give the request and gain the most appropriate instance within the budget. The cloud planner is responsible for choosing the proper resource from providers for consumers.

3 Problem formulation

Before formulating our problem, we briefly introduce basic concepts of mechanism design. Afterwards, we formulate the problem based on mechanism design.

3.1 Basic Concepts of Mechanism Design

In this paper, a general mechanism design setting is shown as follows [15, 16]:

- A finite group of *players* interact with each other. This set is denoted by $P = \{1, 2, ..., p\}$. The individuals are rational and intelligent.
- X is denoted as a set of *alternatives* or *outcomes* from which the players are required to make a collective choice.
- Prior to making the collective choice, each player privately observes his/her preferences over the alternatives in X. This is modeled by supposing that player i, $i = 1, 2, ..., p$, privately observes a parameter or signal b_i that determines his/her preferences.
- Θ_i is denoted the set of private *types* of player i. Then, the set of all type profiles is given by $\Theta = \Theta_1 \times \Theta_2 \times ... \times \Theta_p$. A type profile is represented as $b = (b_1, ..., b_p), b_1 \in \Theta_1, ..., b_p \in \Theta_p$.
- It is assumed that there is a common prior distribution $\Phi \in \Delta(\Theta)$. To maintain consistency of beliefs, individual belief functions describe the beliefs that the player has regarding the type profiles of the rest of the players. Meanwhile, all of these functions can all be derived.
- Individual players have preferences over outcomes that are represented by a *utility function* $u_i : X \times \Theta_i \rightarrow R$.

In addition, we assume that P, X, Θ_i, $\Phi \in \Delta(\Theta)$, u_i, are all common knowledge. In other words, each player knows them and each player knows that every player knows them, etc. The value of b_i is known to player i but is not known to the other players.

Since the player's preferences depend on the realization of their types, it is natural to make the collective decision depending on b [15]. So, we induce the notion of the *social choice function* from the literature [15].

Definition ***Social Choice Function*** Given a set of players $P = \{1, 2, ..., p\}$, their type sets $\Theta_1, ..., \Theta_p$, and a set of outcomes X, a social choice function is a mapping

$f : \Theta_1 \times ... \times \Theta_p \rightarrow X$

that assigns to each possible type profile $(b_1, ..., b_p)$ a collective choice from the set of alternatives. □

The design of the social choice function (SCF) depends on the goal of the whole system. Therefore, we focus mainly on the design of the SCF in this paper.

We summarize the key notations in Table 1. In the following paper, we might omit the symbols for simplicity if thus would not lead to the ambiguity.

3.2 Mechanism design formulation

Suppose that there are n resource providers, each is denoted as sp_i, where $1 \leq i \leq n$. Each provider owns an instance which comprises a combination of R different resources managed by a Virtual Machine Monitor (VMM). $sum(sp_i) = (sum_1(sp_i), ..., sum_R(sp_i))^T$ is denoted as sp_i's capacity vector, while $cur(sp_i) = (cur_1(sp_i), ..., cur_R(sp_i))^T$ denotes the sp_i's current capacity vector. For example, if a provider offers an instance, which owns a 2.0 Gflops single-core CPU, 2 GB memory, 500 GB storage, and a 8 Mbps network bandwidth, his capacity vector is $(2.0, 2, 500, 8)^T$.

Similarly, there are m resource consumers, each is denoted by sc_j, where $1 \leq j \leq m$. Consumer sc_j submits a task t_j requesting an instance as $t(t_j) = (t_1(t_j), .. ., t_R(t_j))^T$.

In this paper, we utilize reverse auction, where the consumers are auctioneers and the providers are bidders. That is, given a task t_j, provider sp_i competes the task via the auction. At the same time, sp_i must satisfy

$$cur(sp_i) \succeq t(t_j), 1 \leq i \leq n, 1 \leq j \leq m, \quad (1)$$

which implies that $\forall k\ cur_k(sp_i) \geq t_k(t_j), k = 1, 2, ..., R$. By using the virtualization technique, a provider could reallocate and gain an instance equal to $t(sc_j)$. In this way, all providers which satisfy $cur(sp_i) \succeq t(t_j)$ are able to compete for the task t_j according to their bids.

For the task t_j, both providers and the consumer should have an approximate estimation of the instance. However, it is difficult for them because they have little information of the price. In essence, the resource providers in self-organizing cloud are not the professional service providers (e.g. Amazon EC2 [3], Microsoft Azure [4]), which can set their own prices. They are even not aware of proper prices about their resources. So, it requires a mechanism to solve this problem, and we thus recommend a price-setting mechanism to gain the current price for the resource based on market conditions. SpotCloud [8] employs this mechanism. An example in SpotCloud is shown in Table 2.

In Table 2, SpotCloud lists VMs' sell prices, including fixed cost of one physical server per month, the sell price about RAM per GB, etc. Moreover, the providers have the basic understanding of prices about respective instances. Similarly, the consumers have the rough estimation about the budget of instances that they need. According to this table, the resource provider could change the bold item to set their own sell prices. Even so, some key factors are still ignored. The quality and cost of respective resources should be taken into consideration by providers. Meanwhile, it doesn't consider the competition among providers via their sell prices. Thus, it requires a method to guide them to set a reasonable price.

For a desired instance of task t_j, provider sp_i has a known valuation v_{ij} via the above mechanism as well as his own condition. This valuation does not depend on the choice of the provider from whom the instance is purchased. Assuming Θ_i is the numerical interval $[\underline{v_{ij}}, \overline{v_{ij}}]$ and denoted as the set of private types of provider sp_i. Please note that the types of providers are randomly distributed from the interval $[\underline{v_{ij}}, \overline{v_{ij}}]$, which is common knowledge among all participants. This type is also viewed

Table 1 Table of Key Notations

Notation	Description
n	The number of resource providers
m	The number of resource consumers
sp_i	A resource provider, where $i = 1, 2, ..., n$
sc_j	A resource consumer, where $j = 1, 2, ..., m$
$sum(sp_i)$	Total capacity vector of provider sp_i
$cur(sp_i)$	Current capacity vector of provider sp_i
t_j	A task submitted by consumer sc_j
$t(t_j)$	Resource vector required by task t_j
X	A set of alternatives or outcomes
Θ_i	A set of types of provider sp_i
v_{ij}	The real valuation of sp_i for t_j
b_{ij}	The price reported by sp_i for t_j
$f(b)$	Social Choice Function
$[\underline{v_{ij}}, \overline{v_{ij}}]$	The price interval estimated by sp_i for t_j
$[\underline{c_j}, \overline{c_j}]$	The budget interval accepted by sc_j for t_j

Table 2 An Example of the price-setting mechanism in SpotCloud

Item	Price
Fixed Cost / Physical Server / Month	**$219**
Average Hours / Month	730
RAM (GB) / Server	**8**
Hourly Cost / Server	$0.30
Hourly Cost / GB of RAM	$0.04
Markup	**80 %**
Marked-up Hourly Sell Price / Server	0.54
Sell Price / GB /hour	0.07
Utilization Rate (on SpotCloud)	**50 %**
Gross Revenue Per Server /Month	$137.97
Cluster Size (Physical Servers)	**200**
Gross Revenue / Month	$27594.00

as the willingness to sell (minimum price below which the seller is not interested in selling the instance). $b_{ij} \in \Theta_i$ is denoted as the provider sp_i's reported bid for the task t_j. Similarly, the consumers also have a budget interval of individual tasks as $[c_j, \overline{c_j}]$.

We formulate the model as follows. For the task t_j submitted by consumer sc_j and provider $sp_1, ..., sp_n$, an outcome is defined as

$$x = (y_{0j}, y_{1j}, ..., y_{nj}, t_{0j}, t_{1j}, ..., t_{nj}) \tag{2}$$

where y_{0j} is denoted as whether consumer sc_j buys the resource or not, t_{0j} is denoted as monetary transfer received by consumer sc_j, $y_{ij}, i = 1, 2, ..., n$ is denoted as whether the resource is supplied by provider sp_i and $t_{ij}, i = 1, 2, ..., n$ is denoted as monetary transfer received by provider sp_i.

Associated with this model, we have

$$\begin{aligned} y_{0j} &= \begin{cases} 0 \text{ if } sc_j \text{ buys an instance} \\ 1 \text{ otherwise} \end{cases} \\ t_{0j} &= \text{monetary transfer received by } sc_j \end{aligned} \tag{3}$$

For $sp_i, i = 1, 2, ..., n$, we have

$$\begin{aligned} y_{ij} &= \begin{cases} 1 \text{ if } sp_i \text{ supplies an instance to } sc_j \\ 0 \text{ otherwise} \end{cases} \\ t_{ij} &= \text{monetary transfer received by } sp_i \end{aligned} \tag{4}$$

The set X of all feasible outcomes is given by

$$\begin{aligned} X = \Big\{ &(y_{0j}, y_{1j}, ..., y_{nj}, t_{0j}, t_{1j}, ..., t_{nj}) \mid \\ &\textstyle\sum_{i=0}^{n} y_{ij} = 1, \sum_{i=0}^{n} t_{ij} \le 0 \Big\} \end{aligned} \tag{5}$$

$s.t.$

$$y_{ij} \in \{0, 1\} \; \forall i = 0, 1, ..., n \tag{6}$$

$$t_{0j} \le 0, 1 \le j \le m \tag{7}$$

$$t_{ij} \ge 0 \; \forall i = 1, ..., n \tag{8}$$

The constraint $\sum_{i=0}^{n} t_{ij} \le 0$ in equation (5) refers that the total money received by all participants is less than or equal to zero. That is, total money paid by all providers is greater than or equal to zero (that is, the consumer pays at least as much as the providers receive. The excess between the payment and receipts is the surplus).

4 Mechanisms for resource allocation

In this section, we specify two mechanisms based on mechanism design, Modified Vickrey Auction mechanism and Continuous Double Auction mechanism, respectively. The former one is applied to the case where the resource is sufficient, while the latter one is employed in the case of resource insufficiency.

4.1 Modified Vickrey auction mechanism

In order to elicit the truthful information from the providers, incentive compatibility is used. In this case, truth revelation is the best response for each provider irrespective of what is reported by the others. This is called dominant strategy incentive compatibility. However, the Gibbard-Satterthwaite impossibility theorem points out that the social choice function is a dominant strategy incentive compatible if and only if it is dictatorial [20, 21]. One of the solutions is that SCFs are nondictatorial in a quasilinear environment. In order to achieve this goal, we assume that resource providers aim is to maximize their profit. Thus, the providers are risk neutral, and this can refer to quasilinearity. A bidder is said to be risk neutral if his utility is a linear function of his wealth. The similar assumption is also made by others in case of Grid [15] and cloud computing [17].

For simplicity, we assume that bid b_{ij} of provider sp_i for the task t_j is less than $\overline{c_j}$, i.e., $b_{ij} \le \overline{c_j}$. This guarantees that the consumer will eventually buy the instance due to the price within the budget. Meanwhile, we also suppose that all the providers satisfy $cur(sp_i) \ge t(t_j), i = 1, 2, ..., n$. So, the social choice function $f(b) = (y_{0j}(b), y_{1j}(b), ..., y_{nj}(b), t_{0j}(b), t_{1j}(b), ..., t_{nj}(b))$ is written as follows:

$$y_{0j}(b) = 0, \; \forall b. \tag{9}$$

For $sp_i, i = 1, 2, ..., n$, we have

$$y_{ij} = \begin{cases} 1 \; b_{ij} = min\{b_{1j}, ..., b_{nj}\} \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

where $b = (b_{1j}, ..., b_{nj})$. More specifically, bidder i will receive payment only when the resource is procured from him.

Next, we should confirm the utility function based on the MVA. The MVA has a condition where the bidders must be symmetric. That is, the bidders are symmetric if $\Theta_1 = \cdots = \Theta_n = \Theta$ and $\Theta_1(\cdot) = \cdots = \Theta_n(\cdot) = \Theta(\cdot)$. This assumption implies that all the bidders have the same set of possible valuations and further they draw their valuations using the same probability density [15]. Obviously, as all bidders' resources meet the basic requirement of the buyer, it is reasonable to assume that it satisfies the condition.

The utility function is given as follows:

$$\begin{aligned} u_{ij}(x, b_{ij}) &= u_{ij}\Big(\big(y_{0j}, y_{1j}, ..., y_{nj}, t_{0j}, t_{1j}, ..., t_{nj}\big), b_{ij}\Big) \\ &= -y_{ij} b_{ij} + t_{ij} \qquad i = 1, 2, ..., n \end{aligned} \tag{11}$$

where $b_{ij} \in R$ can be viewed as provider sp_i's valuation of the resource. Given $x \in X$ and $b_{ij} \in \Theta_{ij}$, the value $u_{ij}(x, b_{ij})$ denotes the payoff of player i. Concretely, u_{ij} depends not only on the outcome and the type of player i, but also on the types of the other players. Assume the bidder k is the winner. Then, the payments of all bidders are defined as:

$$t_{ij}(b) = \begin{cases} min(b_{1j}, ..., b_{(k-1)j}, b_{(k+1)j}, ..., b_{nj}) y_{ij}(b) & i = 1, ..., n \\ -\sum_{z=1}^{n} t_{zj}(b) & i = 0 \end{cases} \tag{12}$$

Therefore, the eventual transaction price is

$$price_{trans} = min(b_{1j}, ..., b_{(k-1)j}, b_{(k+1)j}, ..., b_{nj}) \tag{13}$$

Obviously, $price_{trans} \leq \overline{c_j}$, due to $b_{ij} \leq \overline{c_j}, i = 1, 2, ..., n$. The whole MVA mechanism is presented in Algorithm 1.

Algorithm 1 MVA mechanism

Input: A set of bidders $\{sp_1, ..., sp_n\}$
Output: Winner and payment for participants $(u_{1j}, ..., u_{nj})$

```
min ← ∞;
winner ← 0;
dealprice ← ∞;
// determining the winner of the bidders
for i ← 1 to n do
    if b_ij < min then
        min ← b_ij;
        winner ← i;
    end if
end for
dealprice ← get_second_min(b);
// computing the utility of each bidder
for i ← 1 to n do
    if winner = i then
        u_ij ← dealprice;
    else
        u_ij ← 0
    end if
end for
return winner and (u_1j, ..., u_nj)
```

In Algorithm 1, the function *get_second_min*(*arguments*) is to find the second minimum value among the arguments. The time complexity of this algorithm is $O(n)$. Besides, for MVA mechanism, we obtain Theorem 1.

Theorem 1 *MVA mechanism has dominant strategy incentive compatibility.*

Proof See Appendix A. □

Based on Theorem 1, truth telling is the best response for each provider irrespective of what is reported by the others. That is, providers report their real evaluations about the instance. However, someone may think that this method looks counter intuitive. It may seem from the consumer point of view that receiving the lowest bid (The *first-price reverse auction* usually adopts this.) is better than receiving the second lowest bid. On the contrary, it is noted that bidders act differently in different auction situations. The truth is that the bidders will bid more aggressively in the MVA mechanism than in the first-price reverse auction. The literature [22] has researched this and proved the MVA is the optimal auction.

For consumers, the strategy adopted in this paper is based on the "First-Come, First-Served (FCFS)"rule. FCFS is a service policy whereby the requests of consumers are attended to in the order that they arrived, without other biases or preferences. By using this way, the order of tasks scheduled by the cloud planner is corresponding to their arrival sequence.

In addition, there may be a situation where two or more providers win the auction, due to their same lowest bid. Then, who is the final winner between/among these candidates? Our recommendation is that the candidate list is submitted to the consumer, and the consumer determines the final provider.

Here, we discuss the advantages and disadvantages of MVA mechanism. It is obvious that the mechanism has the ability to offer incentive for providers to report their real evaluations. The consumers thus gain the appropriate instance with lower reported price. Unfortunately, this strategy ignores the case where the resource is insufficient. When the resource is sufficient, every consumer finally gets resources irrespective of the service order. However, when resource is insufficiency, the strategy seems to be unreasonable. It is possible that many urgent tasks can't be executed in time, especially when some tasks are not in hurry to be performed. Considering these situations, we propose the following mechanism.

4.2 Continuous double auction mechanism

When the system is faced with resource insufficiency, not only the providers but also the consumers need to compete with each other. Here, we introduce the Double Auction (DA) concept. The DA is a multilateral process in which buyers as well as sellers can freely limit orders (bids or asks) and accept asks or bids entered by others. Daniel Freidman [23] reports on his experiments where traders, even with imperfect information, consistently achieve highly efficiently allocations and prices. The Continuous Double Auction (CDA) is a continuous-time variant of a DA, where the market clears continuously [24]. The prominent of the CDA is its highly allocative efficiency [25].

For task t_j, the budget interval is defined as $[\underline{c_j}, \overline{c_j}]$, where $\underline{c_j}$ is the valuation of the resource and $\overline{c_j}$ is the highest price that consumer sc_j can accept. However, in case of resource insufficiency, the eventual transaction price is not necessarily less than or equal to $\overline{c_j}$. In order to choose the winner among consumers, the price c_j^* reported by buyer sc_j shows the urgency of his current task. That is, the higher of price, the more urgent of the task. In some cases, the price is even greater than the highest price that the consumer can accept, i.e., $c_j^* > \overline{c_j}$. The winner need to satisfy:

$$\frac{c_j^* - \underline{c_j}}{\underline{c_j}} = max\left\{\frac{c_1^* - \underline{c_1}}{\underline{c_1}}, ..., \frac{c_m^* - \underline{c_m}}{\underline{c_m}}\right\}. \tag{14}$$

After the winner is selected. He/She gets the chance to execute his/her task. Meanwhile, the advantage of providers, if $c_j^* > \overline{c_j}$, is that more providers are able to join this competition.

Algorithm 2 Consumer selection in CDA Mechanism

Input: A set of consumers $\{sc_1, ..., sc_m\}$
Output: The winner and the maximum price that the winner accepts

```
max ← 0;
winner ← 0;
// determining the winner of the bidders
for i ← 1 to m do
    if (c_i^* − c_i)/c_i > max then
        max ← (c_i^* − c_i)/c_i;
        winner ← i;
    end if
end for
return winner and max
```

For providers, we invoke the aforementioned MVA mechanism. However, different from our MVA mechanism, it has a limitation that the price has the highest price c_j^* known to all the providers. Even so, the CDA mechanism also obtains Theorem 1.

Selecting the winner from the consumers is denoted as Algorithm 2. Obviously, the time complexity of this algorithm is $O(m)$.

Different from MVA mechanism, in CDA mechanism, the consumers need to compete with each other to get the resource. With the existence of monetary transfer, the consumers can't unrestrainedly improve their reported prices. Someone may think that since the price reported by the consumer is not the final transaction price, the consumer can report an untruthful price. However, they ignore the environment where the resource of the system is insufficient. The reason why these idle resources are left is due to their high price. So the transaction price is a little less than the reported price.

Naturally, there is a problem, i.e., whether is the CDA mechanism suitable for the case of resource sufficiency or not? In Section 6, our experiments show that the answer is no. The reason is that when resources are sufficient, though the consumer reports a high price, the final transaction price may be lower. This causes that the consumer gets the resource by cheat. Therefore, it is not appropriate to use the CDA mechanism when resources are sufficient.

5 Cloud planner module

As the aforementioned, the cloud planner is a middleware that interacts with consumers and providers, respectively, and responsible for choosing a proper resource from providers for consumers. In this section, we specify the workflow of the cloud planner.

Considering a scenario where there are lots of consumers who want to use the resources in self-organizing cloud, while the specifications and prices of resources are not uniform, it is challenging for consumers to select an appropriate resource from providers. Therefore, the selection of providers should be automatic and in our mechanisms, the cloud planner is responsible for this.

When the resources in self-organizing cloud are sufficient, the cloud planner invokes the FCFS rule to arrange the order of consumers' tasks. On the other hand, when the resources are insufficient, the workflow of the cloud planner for the winner among consumers is shown in Fig. 2. The consumers send required resource specifications to the cloud planner, as well as the highest prices that they could afford. Then, the cloud planner choose the winner among the consumers according to the CDA mechanism. Finally, the result will be sent to the consumers by the cloud planner.

Once the winner among the consumers is chosen, the next step of the cloud planner is shown in Fig. 3. The selection of the mechanisms is determined by the case whether the resources are sufficient or not. The request of the winner is sent to all the providers by the cloud planner. Then, the providers return their bids to the cloud planner. According to our mechanism, the winner among the providers will be chosen by cloud planner. And the result will be sent to the consumer and the responding payment is transferred to the providers. At last, the resource is scheduled from the provider to the consumer.

It is worth emphasizing that the whole process is transparent for both consumers and providers. More specifically, consumers don't need to worry about the selection of resources and only need to report their specifications and prices of required resources. As a result, the advantage is that it significantly reduces the burden of the consumers. Meanwhile, this mechanism adopted by the cloud planner

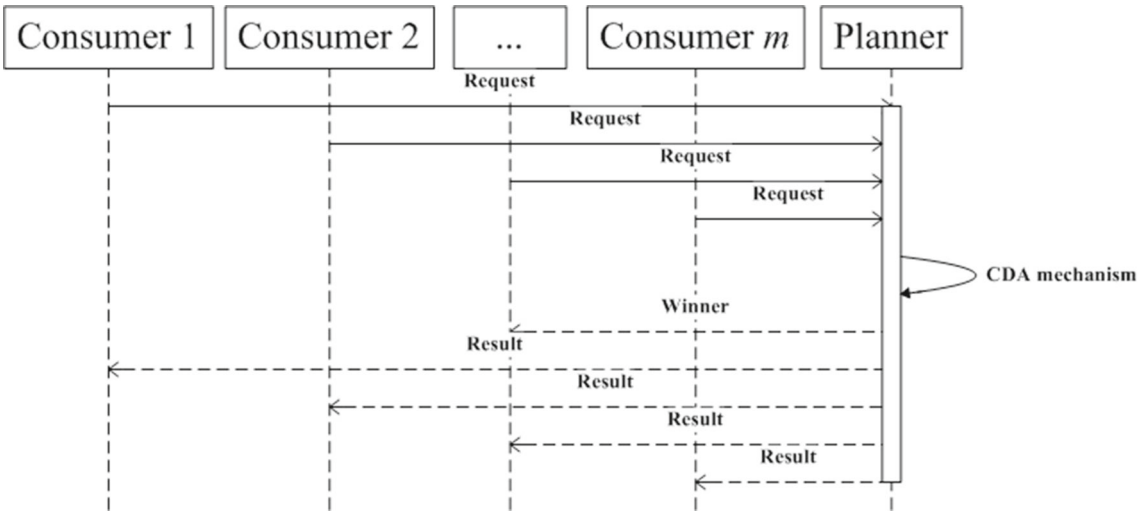

Fig. 2 The winner among consumers chosen by cloud planner

ensures that the consumers gain the appropriate resource with the minimal cost. In addition, when the number of resources or providers changes, our mechanisms is still effective because it has nothing to do with the number. On the other hand, essentially, our mechanism is a *sealed-bid auction* where all bidders simultaneously submit sealed bids to the auctioneer, so that no bidder know how much the other auction participants have bid [26]. The existence of the cloud planner guarantees the impartiality of auctions. In summary, with the above work, a scalable and automatic mechanism for resource allocation in self-organizing cloud is completed.

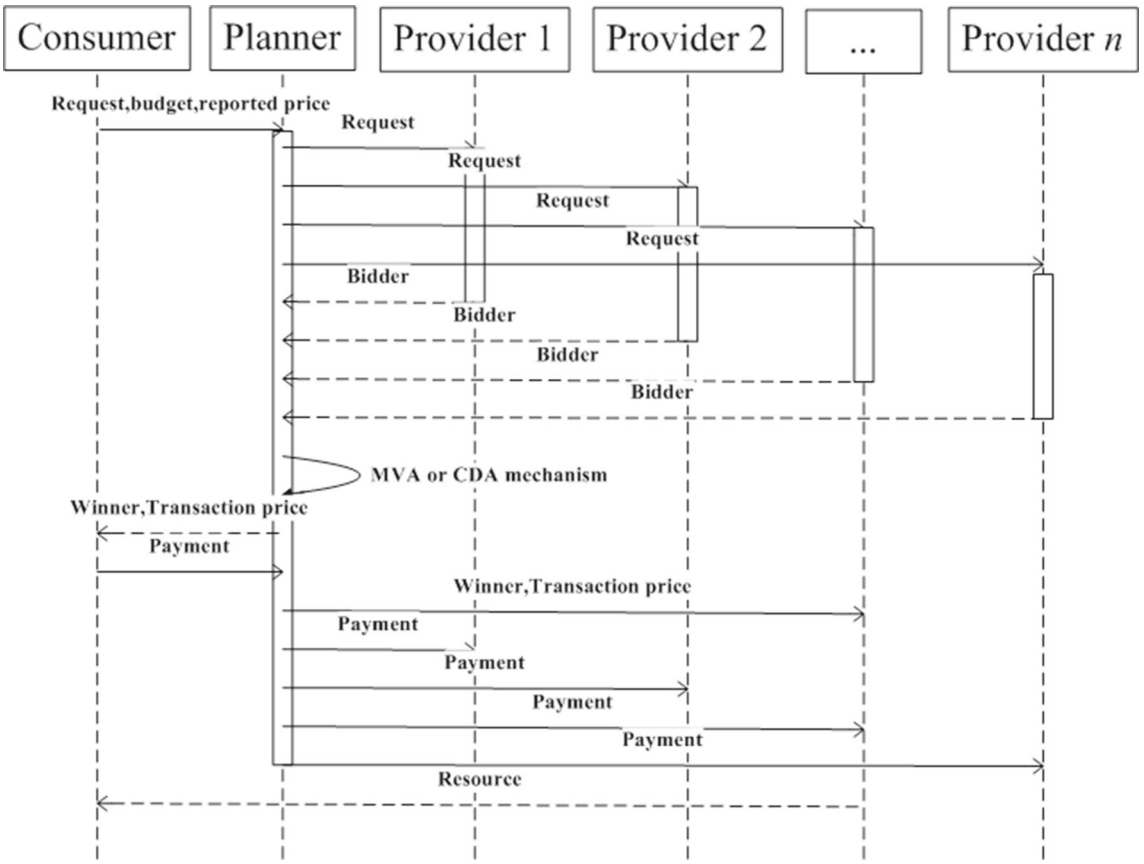

Fig. 3 The resource allocation by cloud planner

6 Numerical result

Mas-Colell et al. [27] point out that if vendors follow different price distributions, the winner determination and procurement cost computation is not optimal by using either first price auction or Vickrey auction. Here, we consider this theory is not suitable for self-organizing cloud, since price of vendors follows the same price distributions with a price-setting mechanism. What's more, one goal of the mechanisms proposed by this paper is to make providers tell the truth about the price of resources.

Similar to [17], we cannot enforce truthfulness simply by using auctions. Therefore, truthfulness cannot be measured. Hence, there is no other baseline to compare our models.

The main challenge of experiments in this paper is the simulation of price of providers and consumers, where information about their prices is limited. Thereby, in this paper, the price is simulated by the data generator. The simulated data is from *generatedata* [28]. Two scenarios are involved in our evaluation. Scenario 1 is the case where the resource is sufficient, while in Scenario 2 the resource is insufficient.

For a given task in Scenario 1, we simulate that there are 100 providers. Their valuations of the task are normally distributed with mean ($\mu = 100.00$) and variance ($\sigma^2 = 15.00$). To avoid error and occasionality, this process is repeated 10 times. Its result is shown in Fig. 4.

In Fig. 4, the x-axis scale with one unit length representing 10 providers; the y-axis scale is the procurement cost. It is intuitive that the cost made by the consumer decreases with the increase of the number of the providers. The reason is that the competition increases as the increment of providers.

Next, we consider Scenario 2. To simulate the case where the resource is insufficient, for simplicity, we still assume the tasks submitted by consumers have the same type as Scenario 1. We set that there are 100 providers and 100 consumers. Their valuations of the task are normally distributed with mean ($\mu = 100.00$) and variance ($\sigma^2 = 15.00$). The urgency urg_j of the task t_j is uniformly distributed in the interval [0, 1]. The execution sequence seq_j is valued by a set 0.01, 0.02, ..., 1. Then, $urg_j * seq_j$ is denoted as *execution-efficiency*. The higher the value of execution efficiency, the worse the execution effect of the task. The high value of execution efficiency implies that either the execution sequence is later, or the task is urgent. simulation result of MVA mechanism and CDA Mechanism based on this case are shown in Figs. 5 and 6.

The main observations in Figs. 5 and 6 are listed as follows. In Fig. 5, execution efficiency of the majority of tasks is in the interval [0,0.5] and there are a small number of tasks whose execution efficiency is in the interval [0.5,1]. It shows that many urgent tasks may be not executed early due to their poor efficiency. Since many urgent tasks are not performed in time, it is possible that consumers will drop out the platform. Compared to Fig. 5, the points in Fig. 6 are converged on the interval [0,0.3]. This means that all tasks have high efficiency. This implies the tradeoff between the urgency of the task and execution sequence. Some points have a lower procurement cost and a higher execution efficiency, which shows that many urgent tasks pay the low cost. This phenomenon indirectly demonstrates that it is not appropriate for CDA mechanism to be adopted in the environment where the resource is sufficient. That is, if CDA mechanism is executed when resources are sufficient, though the consumer reports a high price, the final transaction price may be lower. This causes that

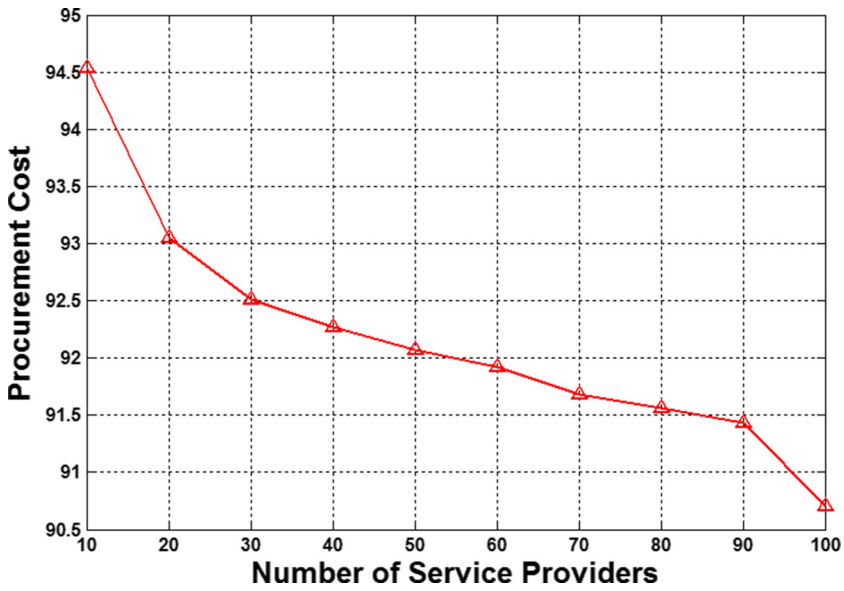

Fig. 4 Procurement cost of MVA in Scenario 1

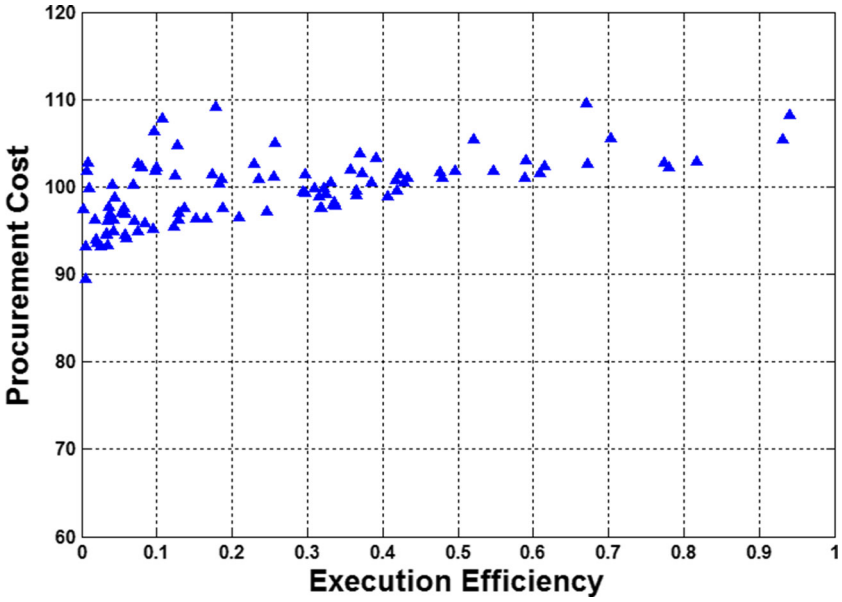

Fig. 5 Procurement cost and execution efficiency of MVA in Scenario 2

the consumer gets the resource by cheat. As a result, CDA mechanism is fit for the environment where the resource is insufficient.

7 Related work

Unlike the traditional business cloud computing platforms [3–5], there are only a few business self-organizing cloud platforms. Enomaly's SpotCloud [8] is the leading customer-provided platform. Wang et al. [6] are the first to investigate SpotCloud, through extensive measurements. They found that SpotCloud is not yet ready to serve long-term applications or some CPU sensitive tasks. Instead, it provides very flexible choices and is very friendly to individual customers when they seek to run short-term tasks at minimum costs. In addition, an instance recommendation model, which facilitates cloud providers to recommend a set of short-listed instances to the customers was proposed in [6]. Furthermore, to mitigate the severe bottleneck to provide reliable cloud service to support long-term tasks, Wang et al. [10] presented an optimal resource provisioning algorithm that ensures service availability with minimized lease costs. Di et al. [9] studied the resource query and allocation problems in a Self-Organizing Cloud, where host machines are connected by a peer-to-peer overlay network on the Internet. Besides, a resource discovery protocol, namely Proactive Index-Diffusion CAN (PID-CAN), was proposed to diffuse resource indexes over the nodes and randomly route query messages among them. Di and Wang

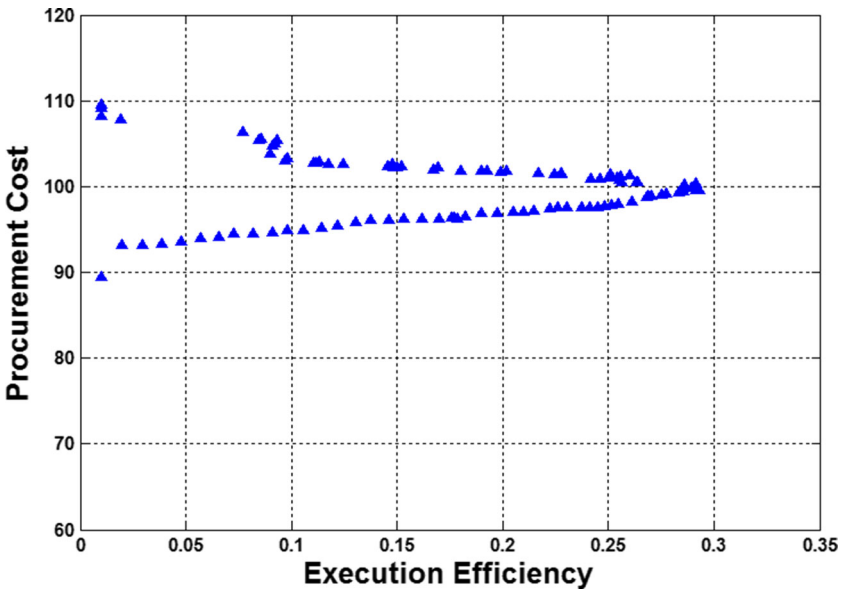

Fig. 6 Procurement cost and execution efficiency of CDA in Scenario 2

[7] envisioned a gigantic self-organizing cloud, which is formed to reap the huge potential of untapped commodity computing power over the Internet. In [7], they proposed a fully distributed, VM-multiplexing resource allocation scheme to manage decentralized resources. Compared with these existing works, we proposed a scalable and automatic mechanism for resource allocation in self-organizing cloud.

Self-organizing cloud is different from the traditional cloud model [29, 30] in the consumers' resource utilization manner. However, there is a paradigm, the federated cloud [31, 32], which is similar to self-organizing cloud. More specifically, both of them have multiple resource providers, and consumers need to consider how to select the appropriate provider. In order to increase scalability and reliability [19], a federated cloud aims to integrate different types of cloud resources from different providers. Early approaches to modeling a cloud federation were in [33, 34]. Nancy Samaan [35] modeled the interactions among the cloud providers as a repeated game among selfish players that aim at maximizing their profit by selling their unused capacity in the spot market but are uncertain of future workload fluctuations. A cloud resource procurement approach was presented in [12, 17], which not only automates the selection of an appropriate cloud vendor but also implements dynamic pricing. Toosi et al. [36] proposed policies (i.e., $NFTI$, $FAOO$, $FAPO$) that help in the decision-making process to increase resources utilization and profit. The mechanisms [37, 38] mostly promoted fairness and ensured mutual benefits for parties involved in the federation. In these works, the focus is mainly on dynamic pricing and increasing the profit of the providers. However, Compared to a federated cloud, self-organizing cloud is concerned with not only dynamic pricing for providers but also resource allocation for consumers.

Resource allocation is an important and challenging task in today's Internet, especially in large distributed systems like Grid, cloud, and so on [17]. Economic models are widespread for resource allocation in different systems. Buyya et al. [39] presented a computational economy-driven Grid system called Nimrod-G, providing an economic incentive for resource owners to share their resources and resource users to trade-off between their deadline and budget. Xhafa and Kolodziej [40] not only presented a survey of the relevant research proposals to use game-based models for the resource allocation problems but also proposed their solution based on metaheuristic methods. Zaman and Grosu [41] designed an auction-based mechanism for dynamic VM provisioning and allocation that takes into account the user demand. The high-performance resource utilization strategies presented in [42] can be used by market participants without requiring dramatic changes to the allocation protocol. In mobile cloud computing system, a game theoretic resource allocation for overall energy minimization was proposed [43]. The above works make use of existing economic models, such as Vickrey auction and game theory. In this paper, we also utilized these models to accomplish resource allocation.

8 Conclusion and future work

In self-organizing cloud, given the large and varying multitude of providers, it is inefficient and tedious to select the most appropriate one manually. Therefore, it is necessary to propose a scalable and automatic mechanism to accomplish resource allocation.

To address the problem, based on mechanism design and auctions, we presented two mechanisms, MVA and CDA, which implement dynamic pricing and resource allocation. MVA mechanism is applied to the situation where the resource is sufficient. On the other hand, CDA mechanism is fit for the environment where the resource is insufficient. Consumers have to compete with each other to obtain the chance to access the resource. Meanwhile, Both of them offer incentive for the providers to tell the truth. Moreover, the whole process is transparent to participants in self-organizing cloud.

Finally, there are still many issues that should be further explored. One critical challenge is how to offer incentive for consumers to contribute their resources or to utilize others' resources. Another challenge is how to guarantee the quality of service proposed by resource providers. Unfortunately, there are few studies about the above problems. Therefore, the design of a mechanism with better incentive and high availability is still an open issue for self-organizing cloud.

Acknowledgments This paper is partly supported by project National Science Foundation of China under Grant 91318301.

Appendix A: Proof of Theorem 1

We introduce [15] and prove as follows.

Proof First of all, we consider bidder 1. His value is v_{1j} and bid is b_{1j}. The other bidders have bids $b_{2j}, ..., b_{nj}$ and valuations $v_{2j}, ..., v_{nj}$. Due to the reverse auction, the utility value of bidder 1 is $b_{1j} - v_{1j}$. We consider the following cases.

Case 1: $v_{1j} \leq min(b_{2j}, ..., b_{nj})$. There are two sub-cases here: $b_{1j} \leq min(b_{2j}, ...b_{nj})$ and $b_{1j} > min(b_{2j}, ...b_{nj}))$.

Case 2: $v_{1j} > min(b_{2j}, ..., b_{nj})$. There are two sub-cases here: $b_{1j} \leq min(b_{2j}, ...b_{nj})$ and $b_{1j} > min(b_{2j}, ...b_{nj}))$.

We analyze these cases separately below.

Case 1: $v_{1j} \leq min(b_{2j}, ..., b_{nj})$.

- Let $b_{1j} \leq min(b_{2j}, ...b_{nj})$. This implies that bidder 1 is the winner, which refers that $u_{1j} = min(b_{2j}, ...b_{nj}) - v_{1j} \geq 0$.
- Let $b_{1j} > min(b_{2j}, ...b_{nj})$. This means that bidder 1 is not the winner, which in turn means that $u_{1j} = 0$.
- Let $b_{1j} = v_{1j}$, then since $v_{1j} \leq min(b_{2j}, ..., b_{nj})$, we have $u_{1j} = min(b_{2j}, ...b_{nj}) - v_{1j}$.

Thus, if $b_{1j} = v_{1j}$, the utility u_{1j} is greater than or equal to the maximum utility obtainable. Thus, whatever the values of $b_{2j}, ...b_{nj}$, it is a best response for player 1 to bid v_{1j}. Thus, $b_{1j} = v_{1j}$ is a weakly dominant strategy for bidder 1.

Case 2: $v_{1j} > min(b_{2j}, ..., b_{nj})$.

- Let $b_{1j} \leq min(b_{2j}, ...b_{nj})$. This implies that bidder 1 is the winner, and the payoff is given by $u_{1j} = min(b_{2j}, ...b_{nj}) - v_{1j} < 0$.
- Let $b_{1j} > min(b_{2j}, ...b_{nj})$. This means that bidder 1 is not the winner. Therefore $u_{1j} = 0$.
- Let $b_{1j} = v_{1j}$, then bidder 1 is not the winner and thus $u_{1j} = 0$.

From the above analysis, it is clear that $b_{1j} = v_{1j}$ is a best response strategy for player 1 in Case 2 also. Combining our analysis of Case 1 and Case 2, we have that

$$u_{1j}(v_{1j}, b_{2j}, ..., b_{nj}) \geq u_{1j}(\widehat{b_{1j}}, b_{2j}, ..., b_{nj})$$

where $\widehat{b_{1j}} \in \Theta_{1j}, \forall b_{2j} \in \Theta_{2j}, ..., b_{nj} \in \Theta_{nj}$.

Also, we can show that, for any $b'_{1j} \neq v_{1j}$, we can always find that for $\forall b_{2j} \in \Theta_{2j}, ..., b_{nj} \in \Theta_{nj}$, such that

$$u_{1j}(v_{1j}, b_{2j}, ..., b_{nj}) \geq u_{1j}(b'_{1j}, b_{2j}, ..., b_{nj})$$

Thus $b_{1j} = v_{1j}$ is a weakly dominant strategy for bidder 1. Using almost similar arguments, we can show that $b_{ij} = v_{ij}$ is a weakly dominant strategy for bidder i where $i = 2, ..., n$. Therefor $v_{1j}, ..., v_{nj}$ is a weakly dominant strategy equilibrium. □

References

1. Mell P, Grance T (2011) The nist definition of cloud computing (draft), NIST special publication, [Online]. Available: http://www.nist.gov/itl/cloud/upload/cloud-defv15.pdf
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, et al. (2010) "A view of cloud computing," Commun ACM 53(4):50–58
3. Amazon Web Service. [Online]. Available: http://aws.amazon.com/
4. Microsoft Windows Azure. [Online]. Available: http://www.microsoft.com/
5. Google AppEngine. [Online]. Available: http://code.google.com/appengine/
6. Wang H, Wang F, Liu J, Groen J (2012) Measurement and utilization of customer-provided resources for cloud computing. In: Proceedings IEEE INFOCOM 2012
7. Di S, Wang CL (2013) Dynamic optimization of multiattribute resource allocation in self-organizing clouds. Parallel Distrib Syst IEEE Trans 24(3):464–478
8. SpotCloud. [Online]. Available: http://www.spotcloud.com
9. Di S, Wang CL, Zhang W, Cheng L (2011) Probabilistic best-fit multi-dimensional range query in self-organizing cloud. In: 2011 International conference on parallel processing (ICPP)
10. Wang H, Wang F, Liu J, Xu K, Wu D, Lin Q (2013) Resource provisioning on customer-provided clouds: Optimization of service availability. In: IEEE International conference on communications (ICC) 2013
11. Di S, Wang C, Cheng L, Chen L (2011) Social-optimized win-win resource allocation for self-organizing cloud. In: International conference on cloud and service Computing (CSC) 2011
12. Mihailescu M, Teo YM (2010) Dynamic resource pricing on federated clouds. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing (CCGrid) 2010
13. Bichler M, Kalagnanam J, Katircioglu K, King AJ, Lawrence RD, Lee HS, Lin GY, Lu Y (2002) Applications of flexible pricing in business-to-business electronic commerce. IBM Syst J 41 (2):287–302
14. McAfee RP, McMillan J (1987) Auctions and bidding. J Econ Lit 25(2):699–738
15. Narahari Y, Garg D, Narayanam R, Prakash H (2009) Game theoretic problems in network economics and mechanism design solutions. Springer, New York
16. Jackson MO (2000) The encyclopedia of life support systems
17. Prasad A, Rao S (2014) A mechanism design approach to resource procurement in cloud computing. Comput IEEE Trans 63(1):17–30
18. Kong Z, Xu CZ, Guo M (2011) Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems In: IEEE International Conference on Cloud Computing (CLOUD) 2011, pp 614–621
19. Mihailescu M, Teo YM (2010) Strategy-proof dynamic resource pricing of multiple resource types on federated clouds. In: Algorithms and architectures for parallel processing. Springer, pp 337–350
20. Gibbard A (1973) Manipulation of voting schemes: a general result. Econometrica: J Econ Soc 587–601
21. Satterthwaite MA (1975) Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. J Econ Theory 10(2):187–217
22. Myerson RB (1981) Optimal auction design. Math Oper Res 6(1):58–73
23. Friedman D (1993) The double auction market institution: a survey. Double Auction Market. Inst Theor Evid 14:3–25
24. Klemperer P (1999) Auction theory: a guide to the literature. J Econ Surv 13(3):227–286

25. Gode DK, Sunder S (1993) Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. J Polit Econ 101(1):119–137
26. Leininger W, Linhart PB, Radner R (1989) Equilibria of the sealed-bid mechanism for bargaining with incomplete information. J Econ Theor 48(1):63–106
27. Mas-Colell A, Whinston M, Green J (1995) Microeconomics
28. Generatedata. [Online]. Available:. http://www.generatedata.com/
29. Rimal BP, Choi E, Lumb I (2009) A taxonomy and survey of cloud computing systems. In: 5th International Joint Conference on INC, IMS and IDC, 2009. NCM'09
30. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst 25(6):599–616
31. Rochwerger B, Breitgand D, Levy E, Galis A, Nagin K, Llorente IM, Montero R, Wolfsthal Y, Elmroth E, Cáceres J, et al. (2009) The reservoir model and architecture for open federated cloud computing. IBM J Res Dev 53(4):1–11
32. Bermbach D, Klems M, Tai S, Menzel M (2011) Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs. In: IEEE International conference on cloud computing (CLOUD) 2011
33. Rochwerger B, Breitgand D, Epstein A, Hadas D, Loy I, Nagin K, Tordsson J, Ragusa C, Villari M, Clayman S, et al. (2011) Reservoirwhen one cloud is not enough. IEEE Comput 44(3,):44–51
34. Avetisyan A, Campbell R, Gupta I, Heath M, et al. (2010) Open cirrus: A global cloud computing testbed. Comput 43(4):35–43
35. Samaan N (2014) A novel economic sharing model in a federation of selfish cloud providers. Parallel Distrib Syst IEEE Trans 25(1):12–21
36. Toosi AN, Calheiros RN, Thulasiram RK, Buyya R (2011) Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In: 13th International conference on high performance computing and communications (HPCC), 2011 IEEE
37. Gomes ER, Vo QB, Kowalczyk R (2012) Pure exchange markets for resource sharing in federated clouds. Concurr Comput Prac Experience 24(9):977–991
38. Song B, Hassan MM, Huh EN (2009) A novel cloud market infrastructure for trading service. In: International conference on computational science and its applications, ICCSA'09
39. Buyya R, Abramson D, Giddy J, Stockinger H (2002) Economic models for resource management and scheduling in grid computing. Concurr Comput Pract Experience 14(13-15):1507–1542
40. Xhafa F, Kolodziej J. (2010) Game-theoretic, market and metaheuristics approaches for modelling scheduling and resource allocation in grid systems. In: 2010 International conference on P2P, parallel, grid, cloud and internet computing(3PGCIC)
41. Zaman S, Grosu D (2013) A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds. Cloud Comput IEEE Trans 1(2):129–141
42. Chard K, Bubendorfer K (2013) High performance resource allocation strategies for computational economies. Parallel Distrib Syst IEEE Trans 24(1):72–84
43. Ge Y, Zhang Y, Qiu Q, Y.-H. Lu (2012) A game theoretic resource allocation for overall energy minimization in, mobile cloud computing system. In: Proceedings of the 2012 ACM/IEEE International symposium on low power electronics and design

Xiaotong Wu is currently working towards the PhD degree at the Department of Computer Science and Technology, Nanjing University, China. He has received his Bachelor's and Master's degree in Software Engineering from Central South University and Dep. of Computer Science and Technology from Nanjing University of China, respectively. His research interests include cloud computing, resource allocation, pricing.

Meng Liu is currently working towards the PhD degree at the Department of Computer Science and Technology, Nanjing University, China. He has received his Master's and Bachelor's degree in Software Engineering from Xidian University and Dep. of Computer Science and Technology from Nanjing University of China, respectively. His research interests include cloud computing, privacy and security.

WanChun Dou received his PhD degree in Mechanical and Electronic Engineering from Nanjing University of Science and Technolo-gy, China, in 2001. From Apr. 2001 to Dec. 2002, he did his postdoctoral research in the Department of Computer Science and Technology, Nanjing University, China. Now, he is a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University, China. From Apr. 2005 to Jun. 2005 and from Nov. 2008 to Feb. 2009, he respectively visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, as a visiting scholar. Up to now, he has chaired three NSFC projects and published more than 60 research papers in international journals and international conferences. His research interests include workflow, cloud computing and service computing.

Longxiang Gao received his PhD in Computer Science from Deakin University, Australia. He is currently a Lecturer in Computer Networks at School of Information Technology, Deakin University. Before he joined Deakin University, he was a post-doctoral research fellow at IBM Research Australia. His research interests include mobile social networks, delay tolerant networks and data privacy. His research works have been published in many international journals and conferences, such as IEEE Transactions on Mobile Computing. He received 2012 Chinese Government Award for Outstanding Students Abroad (Ranked No.1 in Victoria and Tasmania consular districts). He is a member of IEEE and ACS (Australian Computer Society).

Shui Yu received the B.Eng. and M.Eng. degrees from University of Electronic Science and Technology of China, Chengdu, P. R. China, in 1993 and 1999, respectively, and the Ph.D. degree from Deakin University, Victoria, Australia, in 2004. He is currently a Senior Lecturer with the School of Information Technology, Deakin University, Victoria, Australia. He has published nearly 100 peer review papers, including top journals and top conferences, such as IEEE TPDS, IEEE TIFS, IEEE TFS, IEEE TMC, and IEEE INFOCOM. His research interests include networking theory, network security, and mathematical modeling.

Dr. Yu actively servers his research communities in various roles, which include the editorial boards of IEEE Transactions on Parallel and Distributed Systems, and three other International journals, IEEE INFOCOM TPC members, symposium co-chairs of IEEE ICC 2014, IEEE ICNC 2013 and 2104, and many different roles of international conference organizing committees. He is a senior member of IEEE, and a member of AAAS.