Data-Driven Auction Mechanism Design in IaaS Cloud Computing

Chunxiao Jiang, *Member, IEEE*, Yan Chen, *Senior Member, IEEE*, Qi Wang, and K.J. Ray Liu, *Fellow, IEEE*

Abstract—With the emergence of big data computing and analysis, cloud computing services become more and more popular, which has recently drawn researchers' great attentions to develop various new applications and mechanisms. In this paper, we consider the on-demand mechanism design in the infrastructure as a service (IaaS), including resource allocation and pricing issues under dynamic scenarios. Most of existing works on mechanism design assumed static and independent individual utility, while the cloud computing services are provided in a dynamic environment. To solve such problems, we start with analyzing the Google cluster-usage dataset to draw the statistical and stochastic characteristics of the IaaS consumers and providers. Based on the characteristics mined from real data, we propose a stochastic matching algorithm with Markov Decision Process (MDP), which aims at optimizing the long-term system efficiency, with its online version using Q-learning method to address the imperfect model estimation problem. We further design an efficient (EF), incentive compatible (IC), individual rational (IR) auction mechanism, which is an extension of traditional Vickrey-Clarke-Groves (VCG) mechanism. The proposed mechanism is studied under two application scenario: quality sensitive services, where unilateral MDP-VCG auction is implemented; and quality insensitive services, where MDP-VCG double auction is implemented. To verify the performance of our proposed mechanism, we conduct experiment using the Google dataset and show that the proposed MDP-based VCG auction mechanism can achieve EF, IC and IR properties simultaneously.

Index Terms—Cloud computing service, IaaS, Markov decision process, Q-learning, Vickrey-Clarke-Groves, mechanism design

1 Introduction

1.1 Background

The rapid growth of demand for big data and large scale data processing lead the cloud computing into a booming era. Cloud computing services consists of three layers, i.e., infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [1], where IaaS provides the hardware foundation and is our focus in this paper. Lots of IaaS providers have emerged, e.g., Amazon CloudFormation, Google Compute Engine, HP Cloud etc. Meanwhile, open-source cloud service platforms like Hadoop [2] and Eucalyptus [3] are becoming more and more reliable and scalable, making it easier and faster to deploy computing clusters. This trend further encourages companies and institutions with large or small scale datacenters to join the profitable cloud service market.

One of the most important concepts in IaaS is utility computing [4]. In IaaS, computing resource is treated as a special kind of utility, just like electricity or water. For instances, a cloud service billing/pricing system was proposed to fulfill the computing service transactions in [5]; while a Nash equilibrium between the IaaS providers and SaaS providers regarding service provisioning was proposed in [6]. Meanwhile, the competition and cooperation among cloud providers were investigated in [7] and a IaaS provider's revenue maximization scheme using optimization method was proposed in [8]. Based on the pay-as-you-go market model, IaaS makes it possible for small institutions to perform large scale computations with reasonable cost. With the growing number of service providers and customers, *auction* becomes a natural choice for pricing in the cloud computing services. For example, the pricing policy of Amazon, "spot instances", allows users to bid for unused capacities. However, due to the large number of potential IaaS providers and consumers, it is difficult to design an effective auction mechanism that can efficiently utilize the computing resource. Recently, a new idea called "volunteer cloud" has been proposed [9], [10], [11], where researchers made some early attempt to combine volunteer computing with cloud service such that every personal computer can become a potential IaaS provider. This makes the auction mechanism design even more challenging, and thus calls for new solutions to the bilateral trading between IaaS providers and consumers.

1.2 Motivations

In the literature, many auction mechanisms have been proposed, including first-price, second-price, English and Dutch auctions [12]. Most of the existing works on auction mechanism design assumed static and independent private value. Meanwhile, the value function of the users was constructed in a myopic way, without considering the continuous changes of the system state. However, in the cloud computing service provision, the value model is different from traditional setting in two perspectives.

- *C. Jiang is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mail: jchx@tsinghua.edu.cn.*
- *Y. Chen, Q. Wang, and K.J.R. Liu are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742. E-mail: {yan, qwang37, kjrliu}@umd.edu.*

Manuscript received 28 Jan. 2015; revised 3 June 2015; accepted 2 Aug. 2014. Date of publication 5 Aug. 2015; date of current version 5 Oct. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2464810

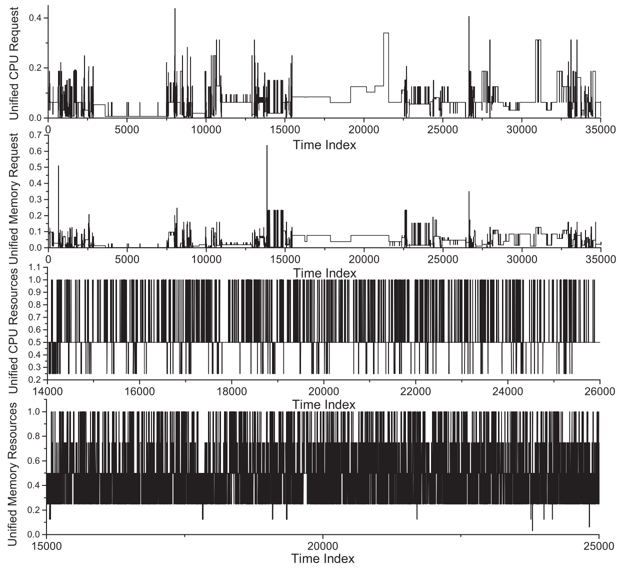


Fig. 1. Google cluster-usage traces: consumers with dynamic CPU requests and memory requests (first and second), providers with dynamic CPU and memory servers (third and fourth), all unified within [0,1].

1) The value of an IaaS consumer may vary according to the specific IaaS provider's computing resources. Different matchings between different consumers and providers may result in extremely different user experiences (values). For example, as shown in Fig. 1 drawn from the Google cluster-usage traces [13], matching an IaaS consumer who requests 0.2 CPU resource (unified within [0,1] by Google) with an IaaS machine who can provide 0.9 CPU resource would be over-satisfied for the consumer but under-utilized for the provider, and vice versa.
2) The cloud computing service is provided in a dynamic environment. In Fig. 1, we can see that different IaaS consumers requesting different CPU and memory resources dynamically arrive at the system. Meanwhile, IaaS machines providing different resources, i.e., CPU and memory, are also dynamically added, removed or updated. Apparently, both service consumers and providers appear in a temporal basis and stay in the system for a period time. Under such circumstances, it is natural for the system to consider the opportunity cost in the near future instead of one-shot or immediate reward.

The aforementioned two distinct problems require us to customize an *on-demand auction mechanism* for the cloud computing services according to their specific statistical and stochastic characteristics. While one fundamental problem is how we can find the statistical characteristics of the IaaS providers and consumers. As a matter of fact, those characteristics are hidden in the real big data generated by the practical services, just as the Google cluster-usage traces. In this paper, we start with analyzing the characteristics of IaaS consumers and providers using the Google data traces. Then, based on the characteristics mined from the real data, we propose a practical MDP-based dynamic VCG auction mechanism for the cloud computing services. The proposed mechanism is studied under two application scenarios: quality sensitive services, where unilateral MDP-VCG is implemented; and quality insensitive services, where MDP-VCG double auction is implemented.

1.3 Related Works

The major goals for auction mechanism design are efficiency, incentive compatibility (IC) and individual rationality (IR). The Vickrey-Clarke-Groves (VCG) auction mechanism [14], [15], [16] is able to achieve all the three properties simultaneously. While the problem with VCG mechanism is that it often incurs low revenue for the sellers, which can cause serious budget balance (BB) problem in bilateral trading. In terms of bilateral trading, the seminal work of Myerson [17] provided the fundamental theory that it is impossible to achieve efficiency, incentive compatibility, individual rationality, and budget balance simultaneously. Nevertheless, continual effort is being made to leverage these criteria and find the optimal tradeoff. The most widely adopted bilateral trading mechanism is the McAfee mechanism [18], since it is easy to implement and has the dominant-strategy incentive compatible and asymptotically efficient properties. In [19], the authors designed a competitive truthful double auction mechanism, with the goal of achieving the (approximate) optimal revenue for the auctioneer. Fudenberg et al. in [20] analyzed existence of pure strategy equilibrium in double auction markets and argued that the equilibrium is close to truth bidding as the market size grows, while Zhao et al. in [21] proposed a matching mechanism extending the traditional VCG mechanism to the double auction scenario.

In the cloud computing literature, there are some existing works on the auction-based cloud service provision model [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. In [22], [23], [24], [25], [26], unilateral auction-based models were employed in studying the resource allocation and pricing issues, e.g., the combinatorial auction model and the reverse auction model. Double auction models were also studied in the cloud computing market to ensure both providers' and users' truthful computing resource trading [27], [28], [29]. Meanwhile, some dynamic auction models were proposed in [31], [32], [33], [34] to deal with the dynamically fluctuating cloud computing resource demands and multi-round auction scenario. However, these existing works only designed the auction mechanism to maximize the immediate system value while ignoring the long-term expected utility, which is also of importance since the computing resource trading between the service consumers and providers are long-term instead of one-shot. Although the authors in [33] have used the MDP model to consider the opportunity cost in the near future, they focused on the scenario of single service provider and only considered the unilateral auction model, which is substantially different from our work where we focus on the stochastic matching between multiple providers and consumers and investigate both unilateral and double auction models. Last but not the least, in this paper, we start with the real service data analysis and the mechanism is built on the characteristics drawn from the real data, which is the fundamental difference between our work and the existing works.

1.4 Organizations

The rest of the paper is organized as follows. Section 2 analyzes the statistical characteristics from the Google cluster-usage dataset. Based on this, Section 3 proposes a learning strategy of optimal IaaS consumer-provider matching. Then, Section 4 proposes the MDP-VCG mechanism in terms of both unilateral and double auctions with the theoretic proves of efficiency, incentive compatibility, individual rationality. In Section 5, experiments are conducted to verify the properties of the proposed mechanism. Finally, Section 6 concludes the paper.

2 Mining Statistical Characteristics

The Google cluster-usage traces dataset contains data from an 12k-machine cell over about a month-long period in May 2011 with size of approximately 40 GB. A cell is a set of machines, typically all in a single cluster, that share a common cluster-management system which is in charge of matching work to machines. Work arrives at a cell in the form of jobs and each job is comprised of one or more tasks. Each task represents a Linux program, possibly consisting of multiple processes, to be run on (matched to) a single machine. There is a set of resource requirements accompanied with each task, which is used for matching the task to some specific machine. In this paper, we regard machines and tasks as IaaS providers (sellers) and IaaS consumers (buyers), respectively.

In the dataset, there are task-event tables containing the time stamp and status of each task, as well as the resource request for CPU cores by each task. There are nine kinds of status definitions for each task: *submit, schedule, evict, fail,finish, kill, lost, update_pending and update_running*. With these status definitions, “a new task is submitted” can be treated as a new buyer arriving, while “an existing task is completed” can be regarded as a buyer leaving the system. In such a case, with the time stamp, we can extract the buyers’ arrival and leaving intervals to estimate their traffic characteristic, i.e., arrival rate and leaving rate. Besides, there are also machine-event tables containing the time stamp and status of each machine, as well as the CPU resource each machine can provide. There are three kinds of status definitions for each machine: *add, remove, update*. In such a case, we can regard adding a new machine as a new seller’s arrival, and removing an existing machine as a seller leaving the system.[1]

When performing matching between consumers’ tasks and providers’ machines, perfect matching considering each individual consumer’s requests and each individual provider’s resources would be preferable. However, such perfect matchings would incur extremely high computational cost due to the numerous computations and dynamic environment. In practical scenarios, we have to categorize the consumers and providers into discretized types, and perform matches between different types of consumers and different types of providers. This would reduce the system complexity to a large extent and ensure the practicality. Let us take CPU resource as an example. As defined in the dataset, the CPU resource is normalized to [0,1] by scaling to the largest capacity of the resource on any machine in the trace. Thus, we can define the types of IaaS buyers and sellers according to the quantized CPU resource levels as follows.

1. In this paper, we use service seller/buyer and service provider/consumer alternatively.

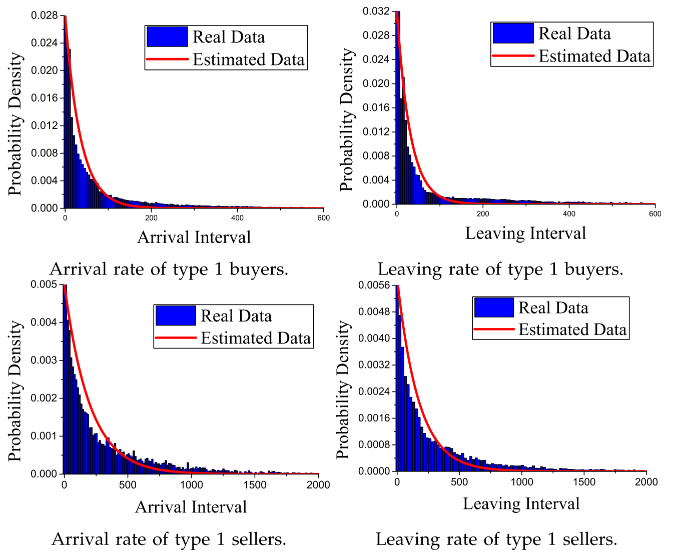


Fig. 2. Intensities of different types of buyers/sellers.

- Type 1 sellers: sellers with CPU resources belonging to interval [0, 0.5];
- Type 2 sellers: sellers with CPU resources belonging to interval (0.5, 1].

For the types of IaaS consumers, according to the Google dataset, since almost all the CPU resource requests are below 0.5, we can define the types of buyers as

- Type 1 buyers: buyers requesting CPU resources within [0, 1/6];
- Type 2 buyers: buyers requesting CPU resources within (1/6, 1/3];
- Type 3 buyers: buyers requesting CPU resources within (1/3, 0.5].

The motivation of categorizing the resource sellers into two types and resource buyers into three types comes from the analysis of the real Google cluster-usage traces. As shown in Fig. 1, the service provides’ CPU resources can be easily categorized into two types, i.e., larger than 0.5 or smaller than (equal with) 0.5, and the CPU resource requests can be divided into three types, i.e., [0,1/6), [1/6,1/3), and [1/3,0.5]. Note that the proposed services matching algorithm is independent of the categorizations and thus will work for any categorizations.

Based on the quantized types, we can further estimate the statistic characteristics of different types of service sellers and buyers, including the arrival rate λ and leaving rate μ. Through analyzing the Google dataset, we find that the service buyers and sellers arrive at and leave the system with Poisson process, as shown in Fig. 2 which illustrates the probability density of arriving/leaving intervals for type 1 buyers and type 1 sellers. From Fig. 2, we can see that the empirical distribution matches well with the exponential distribution, and the estimated probability density using Poisson modeling matches well with the empirical probability density from real data. Note that the estimation results in Fig. 2 is obtained by using minimum mean square error (MMSE) estimator and the estimated Poisson parameters are listed in Table 1. Based on those statistic characteristics, we will discuss how to perform online matching and dynamic auction in the following sections. Note that the

TABLE 1
Traffic Intensity Parameters

	Arrival Rate λ	Leaving Rate μ
Type 1 Buyers	$\lambda_{b1} = 0.028$	$\mu_{b1} = 0.032$
Type 2 Buyers	$\lambda_{b2} = 0.025$	$\mu_{b2} = 0.030$
Type 3 Buyers	$\lambda_{b3} = 0.020$	$\mu_{b3} = 0.026$
Type 1 Sellers	$\lambda_{s1} = 0.005$	$\mu_{s1} = 0.0058$
Type 2 Sellers	$\lambda_{s2} = 0.005$	$\mu_{s2} = 0.0060$

peoposed matching and auction mechanisms are not fully relied on the statistics discovered from the Google dataset. As long as the IaaS sellers and buyers are independent and memoryless, i.e., satisfying Poisson process, our proposed approaches can be applied. On one hand, one prominent change in cloud scenario recently is the volunteer cloud, where all users are unknown with each other and quite independent and memoryless. On the other hand, even the Poisson characteristic is not satisfied, our approach can also be easily extended to the general scenario by modifying the state definition and transitions.

3 Stochastic Matching Service

In this section, we study the resource matching problem between the IaaS consumers and IaaS providers. Unlike the traditional heuristic and/or myopic matching rule, we propose a stochastic scheduling rule based on MDP to achieve long-term efficiency. We have noticed that there was an MDP-based online mechanism in the literature proposed in [35] targeting at the matching of individual consumer and provider. The major drawback is that it is not scalable when the number of users grows. To reduce the complexity and make the mechanism more practical, we classify users into performance clusters and consider the matching among clusters, i.e., considering matching between different types of IaaS consumers and IaaS providers as mentioned in the previous section. Moreover, considering the system model can be varying with time or even unknown to the auctioneer, we further propose a Q-learning algorithm, the implementation of which does not rely on any system model parameters.

3.1 Stochastic Matching Using MDP

In our formulation of the cloud services provision, we regard IaaS consumers as buyers, which can consist of SaaS providers who rent the hardware to provide software service for its own customers, and individual users who use the rented hardware for their own purpose. The IaaS providers are regarded as sellers, which can be either large-scale datacenter managers or small hardware owners. When a certain buyer i is matched to a certain seller j, the buyer gets a value of z_{ij}, which is dependent on both his/her intrinsic valuation of the service and the service quality provided by seller j. Suppose the task arrival rate at the buyer i is λ_i and the deadline requirement of buyer i is T_i; the service rate of service provider j is μ_j, where both arrivals and services are assumed to follow Poisson process. Let us denote the matching between all buyers and sellers as X. The value of buyer i can also be denoted as a function $z_i(X) = \lambda_i\big(1 - e^{-(\mu_i-\lambda_i)T_i}\big)$. Correspondingly, the cost for a seller to provide his/her resources is denoted as

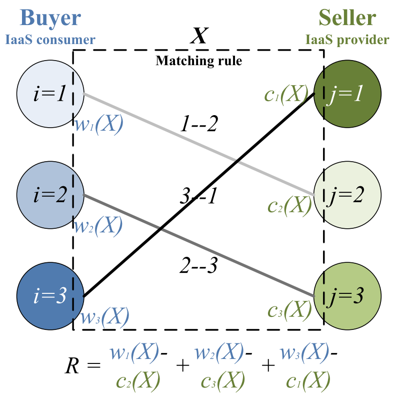


Fig. 3. Example of the matching.

$c_j(X) = \eta\mu_j$ if seller j is successfully matched to some buyer by X, otherwise 0. Note that once the seller j is matched to some buyer, the cost is only dependent on the fixed quality of service (QoS) but not the specific buyer he/she is matched to. In other words, the dependence on X in $c_j(X)$ is only about whether the seller is matched or not. The immediate value of the system at time t can be defined as the buyer's total value minus the seller's total cost:

$$R^{(t)} = \sum_{i,j} z_i(X^{(t)}) - c_j(X^{(t)}), \tag{1}$$

where i, j are matched by matching rule $X^{(t)}$. Fig. 3 illustrates an example of the matching between three IaaS consumers and three IaaS providers, where the matching rule X is buyer 1 being matched with seller 2, buyer 2 being matched with seller 3, and buyer 3 being matched with seller 1. Under such system settings, the immediate value of the system is $R = \sum_{i=1}^{3} z_i(X) - \sum_{j=1}^{3} c_j(X)$, as shown at the bottom of Fig. 3.

As discussed in Section 1, when designing the matching rule X, most of existing works only considered the immediate value $R^{(t)}$. However, the myopic rule may not be the optimal rule since the system dynamic is not taken into account. To further improve the system performance, the long-term optimized matching rule is a more favorable solution under the time-varying scenario. Markov decision process (MDP) model can find such a long-term optimal matching rule by analyzing the system state transitions and optimizing the expected long-term value function. Moreover, as shown in Section 2, the statistical characteristics of service buyers/sellers follow Poisson process, which makes the system state transition following Markov property. For numerical tractability, similar to classical MDP formulation, we introduce a discounting factor δ when calculating the long-term system value. In such a case, supposing that current system state is s and current matching rule is X, the long-term system value, $Q(s, X)$, can be calculated by

$$Q(s, X) = E\left[\sum_{t=0}^{\infty} \delta^{(t)} R^{(t)}\right], \tag{2}$$

where state s consists of the types (λ_i, T_i, μ_j) of all buyers and sellers currently in the system.

According to MDP, the optimal long-term value function is defined as follows:

$$V^*(s) = \max_X Q(s, X). \tag{3}$$

Therefore, the optimal system value and matching rule can be iteratively computed using [36]

$$V^*(s) = R(s, X^*(s)) + \delta \sum_{s'} P(s'|s, X^*(s))V^*(s'), \tag{4}$$

$$X^*(s) = \arg\max_X \left\{ R(s, X) + \delta \sum_{s'} P(s'|s, X)V^*(s') \right\}, \tag{5}$$

where $R(s, X)$ is the immediate value of the system with current system state s and matching rule X, and $P(s'|s, X)$ is the state transition probability from s to s', with the matching rule X. To achieve system efficiency, the auctioneer needs to implement the optimal matching rule $X^*(s)$ at every system state s, which can be found using value iteration method [36].

3.2 Low-Complexity Cluster Matching

Suppose there are N buyers and M sellers in the system. The state of the system (λ_i, T_i, μ_j) gives rise to a scale of $M \times M \times N = NM^2$ dimensional space that is clearly computationally intractable as N and M grows mildly, no matter how coarse we quantize the value on each dimension (the range and granularity of the value/cost). Even if we parameterize the buyers' values as being determined by a single intrinsic parameter (only λ_i or T_i), the space complexity will be of $N \times M$. Moreover, since each service seller can be matched to multiple buyers and all sellers are independent, i.e., each seller has N possible matches, the total number of possible matchings X will be of the order $O(N^M)$, which is also unacceptable. Therefore, it is necessary to use a more concise representation of the state space.

An opportunity is that in the bidding system for cloud computing services, users tend to choose among a few predefined options, rather than actually constructing random options by themselves. For example, a SaaS server can rate the value of the service according to its own customer arrival rate (customers of the software service, different from customers of the infrastructure service), and classify it into three levels as high, medium or low. An individual user can rate the value of the service as important, medium or non-important. An IaaS provider can rate its QoS as high, medium or low. In other words, we can quantize the type of buyers/sellers into several coarse levels, and classify the buyers/sellers into clusters. Since users belonging to the same cluster are with similar types, the scheduling of cloud services within a cluster can be random and has little effect on the overall performance. In such a case, we only need to consider the cluster matching instead of individual matching by describing the system state as the number of traders of each type. Suppose the buyers and sellers each have three possible types, and denote n_k as the number of buyers with type k and m_l as the number of seller with type l, then a state of the system can simply be described using a six-dimensional vector as $s = (n_1, n_2, n_3, m_1, m_2, m_3)$.

With such a representation of the state space, the matching space is also much simpler. Instead of specifying all

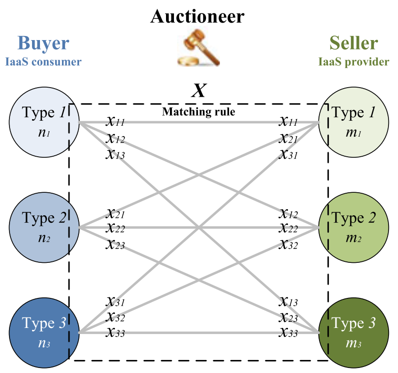

Auctioneer
Buyer
IaaS consumer
Seller
IaaS provider
X
Matching rule
Type 1
n_1
Type 2
n_2
Type 3
n_3
Type 1
m_1
Type 2
m_2
Type 3
m_3
x_{11} x_{12} x_{13}
x_{21} x_{22} x_{23}
x_{31} x_{32} x_{33}
x_{11} x_{21} x_{31}
x_{12} x_{22} x_{32}
x_{13} x_{23} x_{33}

Fig. 4. Example of the matching between three types of buyers and sellers.

matching explicitly, we now only need to specify the number of traders to be matched for each pair of types. Let the number of matches between type k buyers and type l sellers be x_{kl}, then we have

$$\sum_{l=1}^{3} x_{kl} \le n_k, \quad \sum_{k=1}^{3} x_{kl} \le m_l, \qquad x_{kl} \in \mathbb{N}. \tag{6}$$

Fig. 4 illustrates the example of matching between three types of buyers and sellers. In such a case, the matching rule X can be represented by a nine-dimensional vector as

$$X = (x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{13}, x_{23}, x_{33}). \tag{7}$$

If the system capacity of each type is $s_{max} = (N_1, N_2, N_3, M_1, M_2, M_3)$, then it is easy to see that the size of matching space would be less than $\binom{N_1+3}{3}\binom{N_2+3}{3}\binom{N_3+3}{3}$, which is the total number of non-negative integer solutions of (6). The matching space grows polynomially with respect to system capacity and exponentially with respect to the number of level type partitions. Therefore, keeping the type partition coarse makes the value iteration (4), (5) computationally tractable.

The concise representation of the state space also makes it easier to describe the statistical model of the system. We model the system as a queuing model, where the buyer/sellers' arrivals and departures follow a standard birth-death process. At each time slot, the arrival rates of buyers/sellers with type k/l are denoted as $\lambda_{bk}/\lambda_{sl}$. Accordingly, the departure rates of buyers/sellers currently in the system are $n_k\mu_{bk}/m_l\mu_{sl}$, where μ_{bk} and μ_{sl} are the "death rate" of the corresponding traders currently in the system. With sufficiently small time slot, the probability of two or more simultaneous arrivals or leavings is tiny and thus can be negligible. This model is known as "sampled-time approximation to a Markov process" [37], where the Poisson arriving and leaving processes can be approximated by Bernoulli processes. During each time slot, a single user arrives with probability λ or a single user leaves with probability μ. Therefore, given current system state s and matching rule X, the state transition probability can be summarized in (8). Due to the local-connecting characteristic of the state transition diagram, the transition probabilities $P(s'|s, X)$ do not need to be stored in a huge matrix. Instead, the required probability can be computed online with little overhead.

State transition probability:

$$P\{s'|s=(n_1,n_2,n_3,m_1,m_2,m_3),X=(x_{11},x_{12},x_{13},x_{21},x_{22},x_{23},x_{13},x_{23},x_{33})\}=$$
$$\begin{cases}\lambda_{b1}\text{ or }n'_1\mu_{b1}, & \text{if } s'=\left(n'_1+1,n'_2,n'_3,m'_1,m'_2,m'_3\right)\text{ or }\left(n'_1-1,n'_2,n'_3,m'_1,m'_2,m'_3\right);\\ \lambda_{b2}\text{ or }n'_2\mu_{b2}, & \text{if } s'=\left(n'_1,n'_2+1,n'_3,m'_1,m'_2,m'_3\right)\text{ or }\left(n'_1,n'_2-1,n'_3,m'_1,m'_2,m'_3\right);\\ \lambda_{b3}\text{ or }n'_3\mu_{b3}, & \text{if } s'=\left(n'_1,n'_2,n'_3+1,m'_1,m'_2,m'_3\right)\text{ or }\left(n'_1,n'_2,n'_3-1,m'_1,m'_2,m'_3\right);\\ \lambda_{s1}\text{ or }m'_1\mu_{s1}, & \text{if } s'=\left(n'_1,n'_2,n'_3,m'_1+1,m'_2,m'_3\right)\text{ or }\left(n'_1,n'_2,n'_3,m'_1-1,m'_2,m'_3\right);\\ \lambda_{s2}\text{ or }m'_2\mu_{s2}, & \text{if } s'=\left(n'_1,n'_2,n'_3,m'_1,m'_2+1,m'_3\right)\text{ or }\left(n'_1,n'_2,n'_3,m'_1,m'_2-1,m'_3\right);\\ \lambda_{s3}\text{ or }m'_3\mu_{s3}, & \text{if } s'=\left(n'_1,n'_2,n'_3,m'_1,m'_2,m'_3+1\right)\text{ or }\left(n'_1,n'_2,n'_3,m'_1,m'_2,m'_3-1\right);\\ 1-\sum_{k=1}^{3}\left(\lambda_{bk}+n'_k\mu_{bk}\right)-\sum_{l=1}^{3}\left(\lambda_{sl}+m'_l\mu_{sl}\right), & \text{if } s'=\left(n'_1,n'_2,n'_3,m'_1,m'_2,m'_3\right);\\ 0, & \text{otherwise};\end{cases} \tag{8}$$

$$n'_1=n_1-\sum_{l=1}^{3}x_{1l},\quad n'_2=n_2-\sum_{l=1}^{3}x_{2l},\quad n'_3=n_3-\sum_{l=1}^{3}x_{3l},$$
$$m'_1=m_1-\sum_{k=1}^{3}x_{k1},\quad m'_2=m_2-\sum_{k=1}^{3}x_{k2},\quad m'_3=m_3-\sum_{k=1}^{3}x_{k3}.$$

Up to now, we can summarize the four elements of our proposed MDP model: system state, action, state transition probability and value function, as follows:

- *System state*: the combination of the number of different types of buyers and sellers $s=(n_1, n_2,n_3,m_1,m_2,m_3)$.
- *Action*: the matching rule $X(s)=(x_{11},x_{12},x_{13},x_{21}, x_{22},x_{23},x_{13},x_{23},x_{33})$, where x_{kl} should satisfy (6).
- *State transition probability*: as defined in (8).
- *Value function*: as defined in (4).

Given the transition probability of the random process $P(s'|s,X)$, we can simply initialize $V^*(s)$ with any arbitrary value and run value iteration until it converges, as summarized in Algorithm 1.

Algorithm 1. Value Iteration for MDP-based Matching

```
1: /********* Initialization *********/
2: Initialize $V^{(0)}(s)$ for all system states $s$.
3: Initialize matching rule $X^{(0)}(s)$ for all system state $s$.
4: Setup the discount rate $\sigma$ and tolerance $\xi$.
5: while $\sum_s\left(V^{(t)}(s)-V^{(t-1)}(s)\right)^2\geq\xi$ do
6:     for each state $s$ do
7:         /***** Determine current matching rule *****/
8:         Calculate current optimal matching rule $X^{(t)}(s)$
9:         by solving
               $X^{(t)}(s)=\arg\max_X V^{(t-1)}(s)$.
10:        Determine the state transition probability using (8).
11:    end for
12:    for each state $s$ do
13:        /***** Update value function *****/
14:        Update $V^{(t)}(s)$ by computing
             $V^{(t)}(s)=R^{(t)}(s,X^{(t)}(s))$
                 $+\delta\sum_{s'}P(s'|s,X^{(t)}(s))V^{(t-1)}(s')$.
15:    end for
16:    Calculate $\sum_s\left(V^{(t)}(s)-V^{(t-1)}(s)\right)^2$.
17:    $t=t+1$.
18: end while
19: /********* Output *********/
20: $V^*(s)=V^{(t)}(s)$.
21: $X^*(s)=X^{(t)}(s)$.
```

3.3 Online Algorithm for Imperfect Model Estimation

The value iteration algorithm is designed for the scenario where the model is known perfectly and the system is stationary. When the model is imperfect, or slowly varying with time, the learned matching rule derived by value iteration will be sub-optimal. A solution to solve this problem is to use Q-learning [38], which is essentially a Monte-Carlo method for MDP. The updating formula of standard Q-learning can be written as follows:

$$\begin{aligned}Q^{(t)}(s,X)=&(1-\alpha_t)Q^{(t-1)}(s,X)\\&+\alpha_t\Big(R^{(t)}+\max_Y Q^{(t-1)}(s',Y)\Big).\end{aligned}\tag{9}$$

Here, different from the value iteration method, $R^{(t)}$ is the *observed* immediate system value at time slot t, and s' is the *observed* next state, where the transition from s to s' is driven by the underlying true system model. The α_t is the learning rate parameter which controls to what extent the learner relies on previous learning result. After the Q-function converges to $Q^*(s,X)$, the optimal value function and optimal matching are simply

$$V^*(s)=\max_X Q^*(s,X),\tag{10}$$

$$X^*(s)=\arg\max_X Q^*(s,X).\tag{11}$$

The Q-learning algorithm is summarized in Algorithm 2. We can see that the most significant advantage of Q-learning is that it does not require any knowledge about the model. The effect of the model is not explicitly shown in the updating formula. Instead, it is implicitly implemented in the underlying model. In such a case, we can avoid the model error caused by the imprecise estimation of the model parameters. Moreover, the algorithm can be easily modified to an online version by performing the optimal matching based on current estimate of the Q function. In this way, the optimal matching strategy can automatically adapt to the slow change of the underlying model.

Algorithm 2. Q-learning for MDP-based Matching

```
1: /********* Initialization *********/
2: Initialize $Q^{(0)}(s, X)$ for all system states $s$ and matching rules $X$.
3: Initialize optimal matching rule $X^{*(0)}(s)$ for all system state $s$.
4: Setup the exploration rate $\epsilon$.
5: for $t = 1, 2, 3, \ldots, t_{max}$ do
6:    /******* Determine current matching rule *******/
7:    Observe current system state $s^{(t)}$.
8:    Generate random number $\epsilon^{(t)}$ uniformly between $[0, 1]$.
9:    if $\epsilon^{(t)} \geq \epsilon$ then
10:      Determine current matching rule as $X^{*(t)}(s^{(t)})$.
11:   else
12:      Randomly determine current matching rule.
13:   end if
14:   The system state will transit to a new state $s^{(t+1)}$.
15:   /********* Update Q-function *********/
16:   Setup current learning rate $\alpha_t$.
17:   Observe the next system state $s^{(t+1)}$ after matching.
18:   Observe the immediate system vale $R^{(t)}$.
19:   Update $Q^{(t)}(s, X)$ for all $s$ and $X$ with
         $Q^{(t)}(s, X) = (1 - \alpha_t)Q^{(t-1)}(s, X) + \alpha_t\big(R^{(t)} + \max_Y Q^{(t-1)}(s, Y)\big)$.
20:   Compute
         $V^{*(t)}(s) = \max_X Q^{(t)}(s, X)$,
      and
         $X^{*(t)}(s) = \arg\max_X Q^{(t)}(s, X)$.
21: end for
22: /********* Output *********/
23: $X^*(s) = X^{*(t_{max})}(s)$.
24: $V^*(s) = V^{*(t_{max})}(s)$.
```

It is worth mentioning that there are some conditions needed to hold to guarantee the convergence of the algorithm. According to [38], a valid combination of these conditions are: 1) the rule of matching at each time slot ensures that each state is visited infinitely often (such as the ϵ-exploration rule); 2) the sequence of α_t satisfies that

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{12}$$

Intuitively, the parameter ϵ controls the trade-off between exploration and exploitation in online learning, while the parameter α_t controls the learning rate, which can be understood as a measure of "stubbornness" about previous knowledge or "curiosity" towards new information. Being too stubborn makes the learning slow, while being too curious makes the learning unstable. A good learner should properly trade-off between these two. Since one of the major reasons of using Q-learning in our system is to adapt with the model change, a constant learning rate rather than a decaying sequence as specified by (12) should be used. Although such a setting generally leads to a sub-optimal matching decisions along time, it generally give satisfactory solution that is adaptive to the environment.

4 Dynamic VCG Auction Mechanism

When it comes to the auction mechanism design, four key properties have to be involved to evaluate the performance of the mechanism, which are as follows.

- *Efficiency (EF)*: resources are distributed to users that value them most.
- *Incentive compatibility*: a user cannot do better by unilaterally misreport his/her value.
- *Individual rationality*: users always expect non-negative value from the auction.
- *Budget balance*: auctioneer do not lose money in the auction.

Vickrey-Clarke-Groves is the only family of auction mechanisms that can simultaneously achieve efficiency, incentive compatibility and individual rationality [14], [15], [16]. It is often used in applications where multiple items are traded among multiple traders in one shot. However, it is obvious that the auction associated with the cloud computing service is a dynamic multi-shot process. To tackle this challenge, we develop a dynamic VCG mechanism with the help of MDP, which considers the long-term expected value.

4.1 Dynamic VCG Auction for Quality Sensitive Services

In this section, we discuss the auction mechanism design under the quality sensitive services scenario, where the buyers are sensitive to the QoS provided by the sellers. We first introduce some notations and definitions for the MDP-based auction mechanism that are counterparts of traditional VCG auction. We assume that the value for a buyer can vary when matched to different sellers, and this variation can be characterized by a single parameter (type) w_i. For example, when the buyers are SaaS providers, the type can be the traffic intensity of the corresponding buyer; while when the buyers are individual users, the type can be the computing resources requested by the corresponding buyer. Note that the type of buyer here is equivalent with the type defined in the previous section when defining the system state of the MDP model. After a new arriving buyer reports his/her type, the auctioneer can observe the system state s according to the reported types and make the matching decisions. However, due to the selfishness of rational users, a buyer may intentionally report his/her type as r_i, which is different from the true type w_i, if this kind of misreporting can gain more utilities for the buyer. For example, a buyer may exaggerate his/her computing resource requirement as some value which far outweighs his/her true requirement, in order to be matched to sellers with higher computing capability. Since the auctioneer has no knowledge about the true type of each buyer, his/her observation of the system state has to purely depend on the buyers' reported type. As discussed at the beginning of this section, an auction mechanism should effectively prevent users from misreporting their types, i.e., the incentive compatibility.

Let $v_i(w_i, X(s))$ be the expected (discounted) long-term value obtained by buyer i with his/her true type w_i, when the reported system state is s and the matching policy is $X(s)$. Note that if all buyers report their types truthfully, $v_i(w_i, X(s))$ only depends on s. As an individual value function, $v_i(w_i, X(s))$ satisfies similar MDP formulation as the system value function as follows:

$$v_i(w_i, X(s)) = g_i(r_i) + \delta \sum_{s'} P(s'|s, X) v_i(w_i, X(s')), \tag{13}$$

where we use $g_i(r_i)$ to denote the *apparent* benefit received by buyer i when buyer i leaves the system. Note that $g_i(r_i)$ is inferred by the auctioneer according to the reported type r_i, and becomes the true benefit if and only if $r_i = w_i$.

A mechanism for the dynamic auction of cloud computing services consists of a matching policy X and a pricing rule q, both of which are functions of system state s. To formulate the price in the Bayesian setting, let $q_i(r_i, X(s))$ denote the *expected* payment of buyer i with reported type r_i when the system state is s and the matching policy is $X(s)$. With the above notation definitions, we show our proposed MDP-based VCG mechanism in the following definition.

Definition 1 (MDP-based VCG mechanism). *The MDP-based VCG mechanism (MDP-VCG) is a mechanism with the following matching and pricing rule.*

- Matching rule: *The matching function $X(s)$ is the optimal matching policy calculated from the MDP formulation (4) and (5), implemented using either value iteration or Q-learning:*

$$V^*(s) = R(s, X^*(s)) + \delta \sum_{s'} P(s'|s, X^*(s)) V^*(s'),$$
$$X^*(s) = \arg\max_X \left\{ R(s, X) + \delta \sum_{s'} P(s'|s, X) V^*(s') \right\}.$$

  Note that the MDP formulation is based on the auctioneer's observed types of the buyers, i.e., their reported types. Therefore, s may not be the true system state.
- Pricing Rule: *The pricing rule is to collect payment*

$$q_i(r_i, X^*(s)) = g_i(r_i) - V^*(s) + V^*\big(s - \{r_i\}\big), \tag{14}$$

  from buyer i when he/she leaves the system. Here s is the system state when the buyer first joins the system, $s - \{r_i\}$ is the system state by excluding buyer i. For example, if the current system state is $(n_1, n_2, n_3, m_1, m_2, m_3)$ and buyer i is of type 2, then $s - \{r_i\} = (n_1, n_2 - 1, n_3, m_1, m_2, m_3)$.

In the following, we will theoretically show that our proposed MDP-VCG mechanism is Bayesian efficient (BE), Bayesian incentive compatible and Bayesian individual rational (BIR). Before that, let us first define the Bayesian efficiency, Bayesian incentive compatibility and Bayesian individual rationality for the dynamic auction.

Definition 2 (Bayesian Efficiency, BEF). *A mechanism for dynamic auction is Bayesian efficient when the long-term expected system value $V(s)$ is maximized for all system state s, i.e.,*

$$\max V(s) = R(s, X^*(s)) + \delta \sum_{s'} P(s'|s, X^*(s)) V^*(s'). \tag{15}$$

Definition 3 (Bayesian incentive compatibility, BIC). *Let s be the observed system state when all buyers truthfully report their types, and $s^{(i)}$ be the observed system state when buyer i misreports the type while all other buyers truthfully report their types. A mechanism for dynamic auction is Bayesian Incentive Compatible when*

$$\begin{aligned} &v_i\big(w_i, X(s)\big) - q_i\big(w_i, X(s)\big) \geq \\ &\quad v_i\big(w_i, X(s^{(i)})\big) - q_i\big(r_i, X(s^{(i)})\big), \forall i, \end{aligned} \tag{16}$$

where w_i is the true type and r_i is the misreported type.

Definition 4 (Bayesian individual rationality). *A (incentive compatible) mechanism for dynamic auction is Bayesian Individual Rational when*

$$v_i\big(w_i, X(s)\big) - q_i\big(w_i, X(s)\big) \geq 0, \forall i, \tag{17}$$

where w_i is the true type.

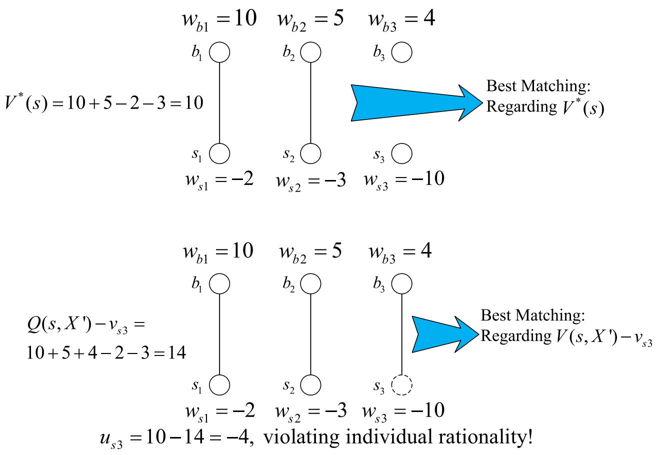
Theorem 1. *The proposed MDP-VCG is Bayesian efficient, Bayesian incentive compatible and Bayesian individual rational.*

Proof. Due to the page limit, we have put the detailed proof in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/10.1109/TSC.2015.2464810. □

4.2 Dynamic VCG Double Auction for Quality Insensitive Services

In this section, we discuss the dynamic auction mechanism under the quality insensitive services scenario, where the buyers are insensitive to the QoS provided by the sellers. In such a case, both buyers and sellers can misreport their types and the proposed MDP-VCG mechanism need to be extended to the double auction to enforce truth telling of both sides. The formulation of double auction is similar to that of unilateral auction discussed in the previous section. The only difference is that in a double auction w_i can denote the type of either a buyer or a seller (QoS), and $v_i(w_i, X^*(s))$ can denote the value function for either a buyer or a seller. For a matched seller, $v_i(w_i, X^*(s))$ taking a negative value represents the cost of the seller for providing the service. The form of the mechanism stays the same and the proof of EF, IC and IR does not change. However, the budget balance cannot be guaranteed in double auction, which means that the total payment made by buyers may be less than the total payment to the sellers, making the auctioneer lose money. Fortunately, such loss is typically small as shown later in the evaluations.

Note that the pricing rule (14) in our proposed mechanism is different from the traditional Clarke Pivot Rule in [39]. If Clarke Pivot Rule is used in the mechanism, the IR property will not hold for double auction, as illustrated by the simple example in Fig. 5. In the figure, we have three buyers with values 10, 5 and 4, and three sellers with costs 2, 3 and 10. We are trying to determine the price (payment) for seller 3. If Clarke Pivot Rule is used, we should first determine the best matching and system value for all six traders, as shown in the upper half of the figure, where $V^*(s) = 10$. Then we should find the matching X' (still over all six traders) that maximizes the value $Q(s, X') - v_{s3}$,



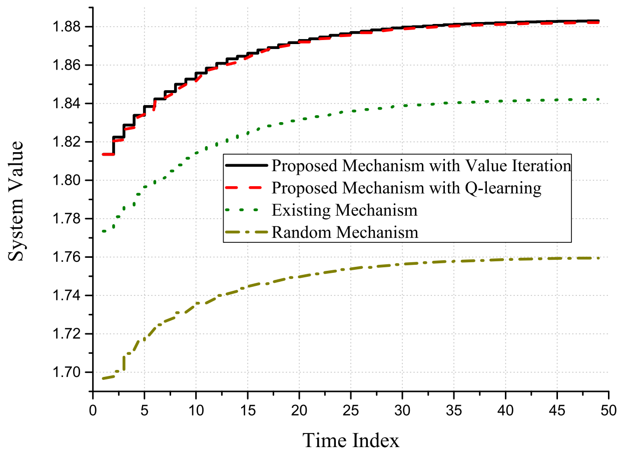
$w_{b1}=10$ $w_{b2}=5$ $w_{b3}=4$
b_1 b_2 b_3
$V^*(s)=10+5-2-3=10$
Best Matching: Regarding $V^*(s)$
s_1 s_2 s_3
$w_{s1}=-2$ $w_{s2}=-3$ $w_{s3}=-10$
$w_{b1}=10$ $w_{b2}=5$ $w_{b3}=4$
$Q(s,X')-v_{s3}=$
$10+5+4-2-3=14$
Best Matching: Regarding $V(s,X')-v_{s3}$
$w_{s1}=-2$ $w_{s2}=-3$ $w_{s3}=-10$
$u_{s3}=10-14=-4$, violating individual rationality!

Fig. 5. Problem with Clarke Pivot Rule.

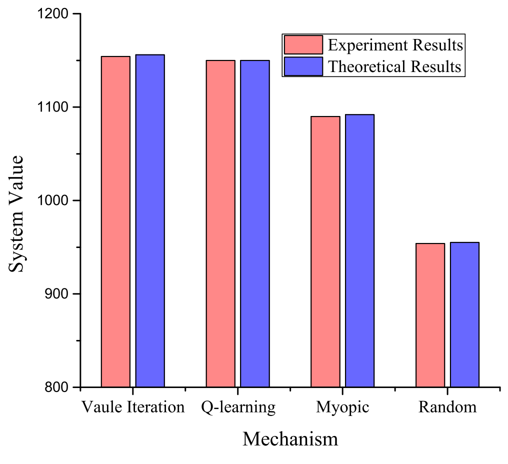
resulting in $Q(s,X')-v_{s3}=14$. The payment of seller 3 will then be determined as $v_{s3}-V^*(s)+(Q(s,X')-v_{s3})$ and the net value is $V^*(s)-(Q(s,X')-v_{s3})=-4$, violating individual rationality.

5 Evaluation

In this section, we conduct experiments to verify the efficiency, incentive compatibility and individual rationality properties of our proposed mechanism. At the same time, we also analyze the budget balance issue. In the evaluations, we use the real-world traces collected from Google cluster to compare the proposed mechanism with the existing works in terms of efficiency, and show the IC, IR and BB properties. The traffics and types of the IaaS providers and consumers discussed in Section 2 are also used for performance evaluation.

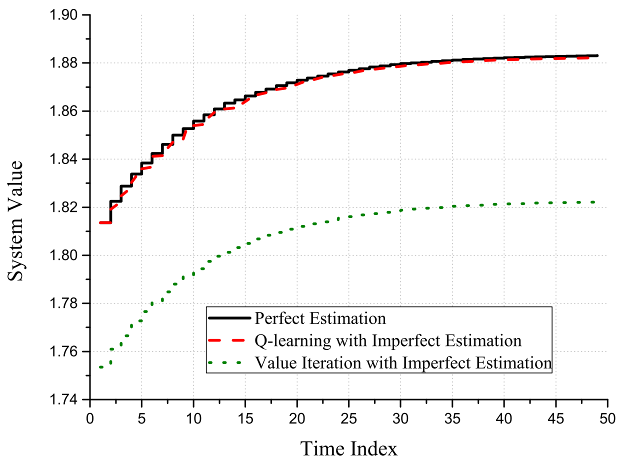


(a) System value for one instantiation.

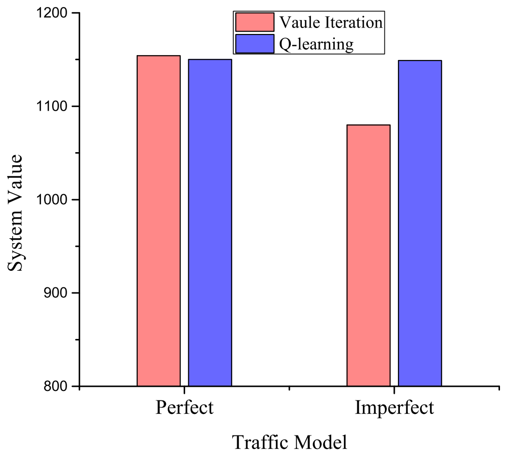


(b) Accumulated system value.

Fig. 6. Efficiency comparison with perfect traffic model.



(a) System value for one initial system state.



(b) Accumulated system value.

Fig. 7. Efficiency comparison with imperfect traffic model.

5.1 Efficiency Verification

In the experiments, we consider the application of allocating IaaS resources to individual users, i.e., the IaaS providers are sellers and individual users are buyers. In such an application, the value of the service can be characterized by the waiting time of the individual users, which is dependent on the computing capability of the specific seller. Suppose that the requested computing resources of buyer i is σ_i, and the computing capability of the corresponding seller j is ζ_j, we can define the (immediate) value function of the buyers as

$$z_i(X)=e^{-\beta\sigma_i/\zeta_j}, \tag{18}$$

and the cost of the seller as a linear function of his/her computing capacity as

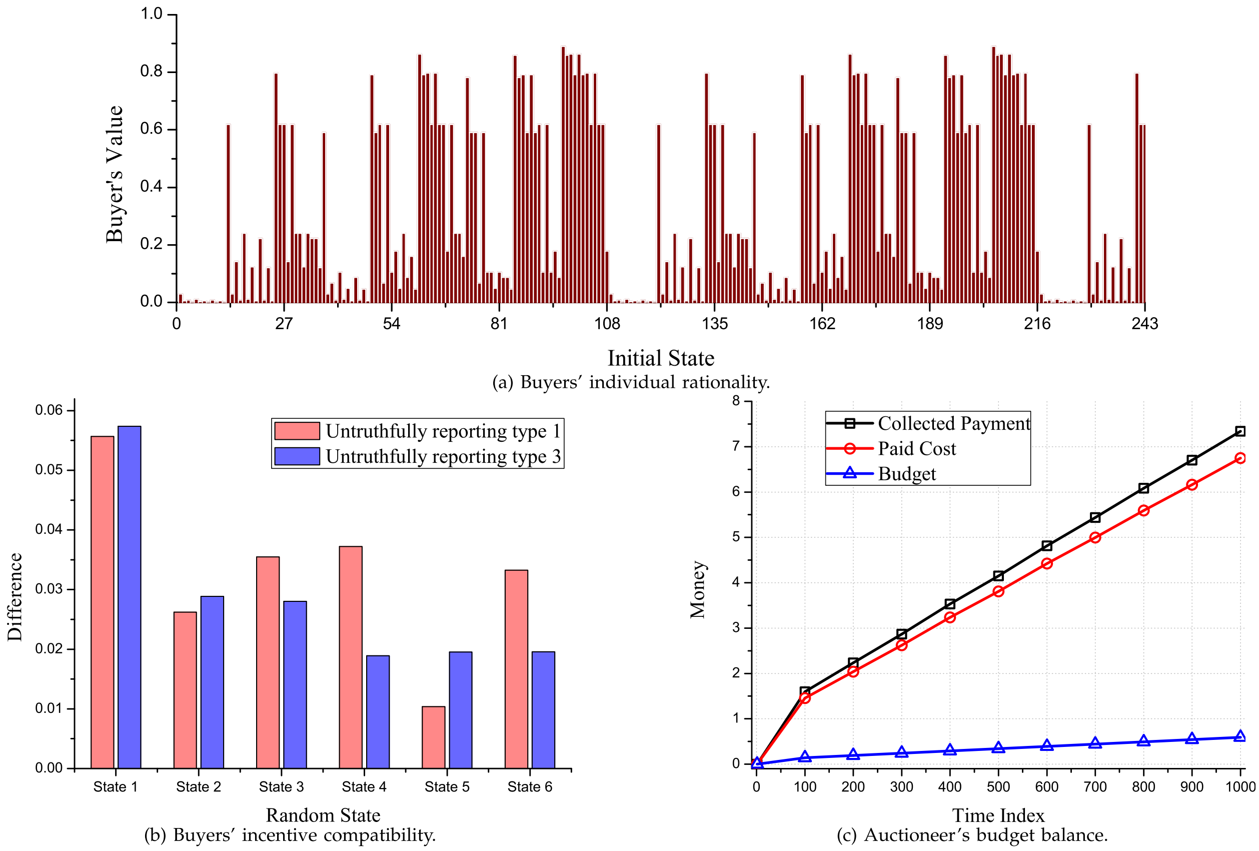


Fig. 8. IR, IC and BB verification for "quality sensitive services".

$$c_j(X) = \gamma \zeta_j, \tag{19}$$

where β and γ are positive coefficients.

Based on the value and cost functions definitions in (18) and (19), we can define different types of sellers' values and different types of sellers' cost according to the types mined from the Google dataset in Section 2. Since the types are quantized levels of computing resources, an approximation is required. Here, we adopt the mean for the approximation, e.g., for the type 1 sellers with CPU resources in [0, 0.5], their computing capability is quantized to 0.25. In such a case, we can obtain the value and cost between different pairs of types of buyers and sellers, as shown in Table 2 where $\beta = 1$, $\gamma = 0.1$ and $(\cdot,\cdot)$ means (buyer's value, seller's cost).

With the buyers' values and seller's costs defined in Table 2, we can conduct experiment to verify the efficiency of our proposed mechanism using the Google dataset. In the experiment, we compare the system value of four different methods as follows:

- Proposed mechanism with value iteration algorithm as shown in Algorithm 1,
- Proposed mechanism with Q-learning algorithm as shown in Algorithm 2,
- Existing mechanism without considering the long-term efficiency, ([23], [25], [28], [31], [32]),
- Random mechanism.

For the existing mechanism, the auctioneer determines the resource allocation through maximizing the immediate (myopic) system value R without taking into account the long-term system value in the near future, as in [23], [25], [28], [31], [32]. While with the random mechanism, a random matching rule is selected, which is considered as a comparison benchmark in the experiment. For our proposed MDP-based mechanism, the discount factor is set as $\alpha = 0.9$ and the learning rate is set as $\xi = 0.5$. Fig. 6a shows the system value comparison results, where all methods start with a same initial system state. The results are averaged over 1,000 independent experiments. From Fig. 6a, we can see that due to the discount factor α, the system values of four methods converge to some constants as time slot index goes to 50. The system values of our proposed mechanism with value iteration and Q-learning are basically the same, both of which perform better than the existing and random mechanisms. Moreover, as shown in Fig. 6b, we also compare the four methods in terms of accumulated system value, which is calculated from the summation of system values associated with 1,000 random initial system states. The experiment results are consistent with the theoretical results, which are directly calculated from value function. Similarly, we can see that our proposed mechanism can achieve the highest efficiency.

TABLE 2
Buyers' Values and Sellers' Costs

	Type 1 Sellers	Type 2 Sellers
Type 1 Buyers	(0.9355, 0.025)	(0.9780, 0.075)
Type 2 Buyers	(0.8187, 0.025)	(0.9355, 0.075)
Type 3 Buyers	(0.7165, 0.025)	(0.8948, 0.075)

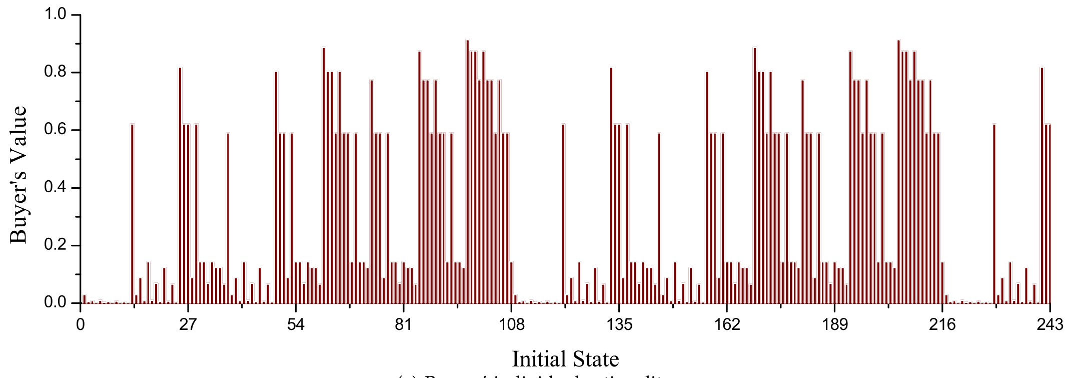
TABLE 3
Buyers' Values and Sellers' Costs for Quality Insensitive Scenario

	Type 1 Sellers	Type 2 Sellers
Type 1 Buyers	(0.9672, 0.025)	(0.9672, 0.075)
Type 2 Buyers	(0.9048, 0.025)	(0.9048, 0.075)
Type 3 Buyers	(0.8465, 0.025)	(0.8465, 0.075)

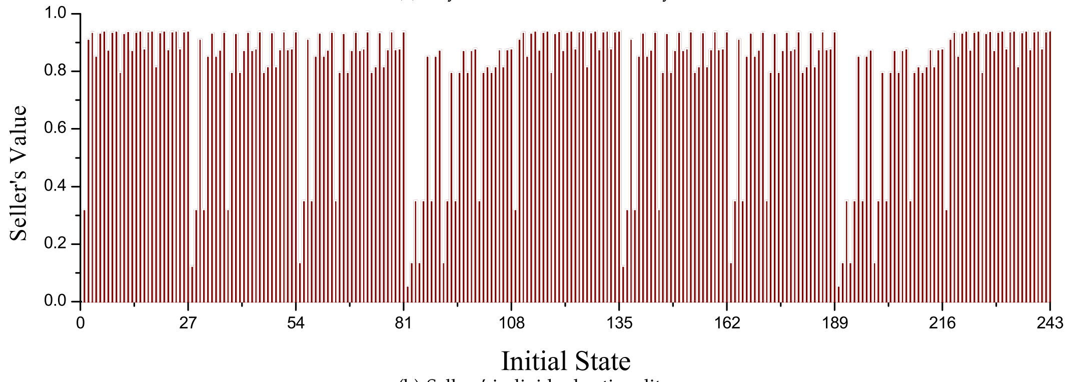
In the previous experiment, we assume that the auctioneer has perfect knowledge (estimation) about the traffic model listed in Table 1. Under such circumstance, we can see that there is no difference between value iteration algorithm and Q-learning algorithm as shown in Fig. 6. In this experiment, to compare the performance of value iteration and Q-learning, we study the case when the traffic model estimation is not precise or the traffic model is slowly varying with time. In Fig. 7a, we show the system value comparison between value iteration and Q-learning algorithms under the imperfect traffic model estimation. We can see Q-learning can adapt to the true underlying model and achieve better performance than value iteration. Therefore, Q-learning is a better candidate to decide optimal matching in real applications. Moreover, we also compare the two algorithms in terms of accumulated system value for 1,000 time slots in Fig. 7b. Similarly, we can see that there is no performance degradation when the model transits from perfect case to imperfect case.

5.2 Incentive Compatibility and Individual Rationality Verification

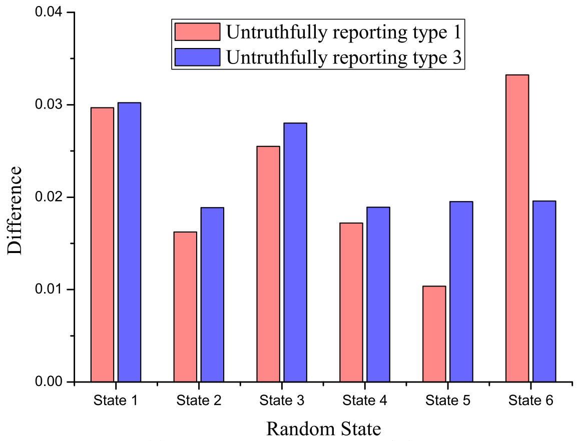
To verify IC and IR, we consider two application scenarios. The first one is "quality sensitive services", where buyers' value functions are influenced by the sellers' service quality,



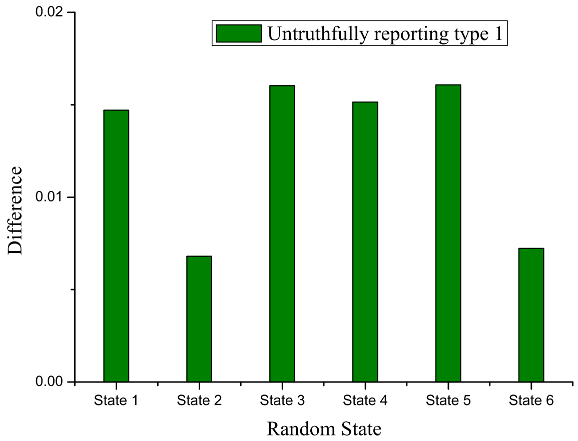
(a) Buyers' individual rationality.



(b) Sellers' individual rationality.



(c) Buyers' incentive compatibility.



(d) Sellers' incentive compatibility.

Fig. 9. IR and IC verification for "quality insensitive services".

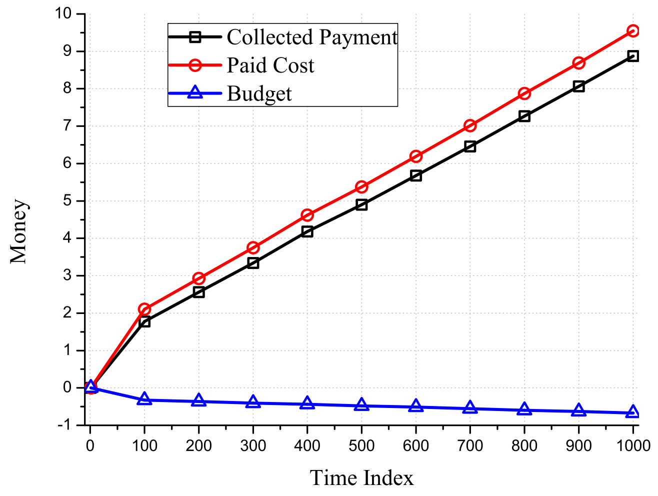


Fig. 10. Auctioneer's budget balance for "quality insensitive services".

and sellers are enforced to report truthfully. The application of allocating IaaS resources to individual users discussed in the previous section belongs to this "quality sensitive services" scenario, as shown in the definition of buyers' value function (18). Unilateral MDP-VCG mechanism is used for this kind of application scenario. The second application scenario is "quality insensitive services", where buyers' value functions are not influenced by the sellers' service quality, and neither buyers nor sellers are enforced to truthfully report their types. The double auction version of MDP-VCG is used for this "quality insensitive services" application scenario.

In the experiment of "quality sensitive services", all the system settings are same with the previous section. Firstly, in Fig. 8a, we show the average value of a buyer with different initial system states. We can see that the buyer's value is always non-negative, which means that our proposed mechanism satisfies the individual rational property. Secondly, in Fig. 8b, we show the difference of the buyer's value from bidding truthfully minus that from bidding untruthfully. The buyer is supposed to be of type 2 and untruthfully report his/her type as 1 or 3. In order to clearly show the results, we only depict several random initial state. From Fig. 8b, we can see that the differences are always non-negative, which means that our proposed mechanism satisfy the incentive compatible property. Finally, in Fig. 8c, we show the budget of the auctioneer, which is the total payment from the buyers minus the total cost of the sellers. From Fig. 8c, we can see that the budget is always non-negative, i.e., the budget balance is also achieved.

In the experiment of "quality insensitive services", where the buyers' values are not influenced by the computation capacity of sellers, we modify the system settings discussed in the previous section, especially the buyer's value function. In this case, since the buyers only care about whether the task is done regardless how the task is implemented, we can re-define the buyer's value function as

$$z_i'(X) = e^{-\beta\sigma_i/\zeta}, \tag{20}$$

where $\zeta = 0.5$ is a constant. In such a case, the modified buyers' values and sellers' costs can be calculated as Table 3. Other system settings, including the sellers' cost function, the types and traffic model of sellers and buyers, are still same with those in the previous section.

Under such settings, sellers are allowed to misreport their types (their costs). In this experiment, a buyer reports his/her value of service and a seller reports a single scalar indicating its cost. The double auction version of MDP-VCG is then implemented. We show the IC and IR verification results in Fig. 9, from which it can be seen that similar to the "quality sensitive services" scenario, both the buyers' and sellers' incentive compatibility and individual rationality are satisfied. However, from Fig. 10, we can see that different from the "quality sensitive services" scenario, the budget balance is not satisfied in the "quality insensitive services" scenario with double auction. Fortunately, the violation of budget balance is not very severe. One possible practical solution to the budget balance is through external advertisement. Just as Amazon and Alibaba, while organizing an auction platform to coordinate the sellers and buyers, the auctioneer can earn additional profits through external advertisement on the platform to achieve the budget balance.

6 Conclusion

In this paper, we discussed the resource allocation and pricing issue in the cloud services provision. We first analyze the Google cluster-usage dataset to obtain the statistical characteristics of the IaaS consumers and providers. Based on the characteristics mined from real data, we proposed a stochastic matching algorithm using MDP model, which aims at optimizing long-term system efficiency. To reduce the complexity of the MDP-based algorithm, we classify the buyers/sellers into different clusters and consider the cluster matching instead of individual matching. We then proposed its online version using Q-learning to address the imperfect model/slowly changing model problem. Based on the MDP formulation, we designed an efficient, incentive compatible, individual rational auction mechanism that is an extension of traditional VCG mechanism. The proposed mechanism were discussed under two application scenarios: quality sensitive services, where unilateral MDP-VCG is implemented; and quality insensitive services, where MD-VCG double auction is implemented. Finally, we conducted experiments using Google cluster-usage traces to verify the properties of the proposed mechanism.

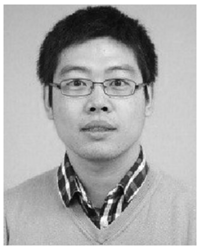
Acknowledgments

This research was supported by NSFC China under projects 61371079 and by a Postdoctoral Science Foundation funded project.

References

[1] K. M. Sim, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, 4th Quarter 2012.
[2] T. White, *Hadoop: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, 2012.
[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 124–131.
[4] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Apr.–Jun. 2012.

[5] K.-W. Park, J. Han, J. Chung, and K. H. Park, "THEMIS: A mutually verifiable billing system for the cloud computing environment," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 300–313, Jul.–Sep. 2013.
[6] D. Ardagna, B. Panicucci, and M. Passacantando, "Generalized nash equilibria for the service provisioning problem in cloud systems," *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 429–442, Oct.–Dec. 2013.
[7] T. Truong-Huu and C.-K. Tham, "A novel model for competition and cooperation among cloud providers," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 251–265, Jul. 2014.
[8] A. N. Toosi, K. Vanmechelen, K. Ramamohanarao, and R. Buyya, "Revenue maximization with optimal capacity control in infrastructure as a service cloud markets," *to appear in IEEE Trans. on Cloud Comput. 10.1109/TCC.2014.2382119*, 2015.
[9] V. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, "Volunteer computing and desktop cloud: The cloud@ home paradigm," in *Proc. 8th IEEE Int. Symp. Netw. Comput. Appl.*, 2009, pp. 134–139.
[10] A. Chandra and J. Weissman, "Nebulas: Using distributed voluntary resources to build clouds," in *Proc. Conf. Hot Topics Cloud Comput.*, 2009, pp. 2–2.
[11] A. Marosi, J. Kovács, and P. Kacsuk, "Towards a volunteer cloud system," *Future Gener. Comput. Syst.*, vol. 29, pp. 1442–1451, 2012.
[12] V. Krishna, *Auction Theory*. San Francisco, CA, USA: Academic, 2009.
[13] J. Wilkes and C. Reiss. (2011). Google Cluster-usage traces [Online]. Available: http://code.google.com/p/googleclusterdata
[14] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *J. Finance*, vol. 16, no. 1, pp. 8–37, 2012.
[15] E. Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, no. 1, pp. 17–33, 1971.
[16] T. Groves, "Incentives in teams," *Econometrica: J. Econometric Soc.*, vol. 41, pp. 617–631, 1973.
[17] R. Myerson and M. Satterthwaite, "Efficient mechanisms for bilateral trading," *J. Econ. Theory*, vol. 29, no. 2, pp. 265–281, 1983.
[18] R. McAfee, "A dominant strategy double auction," *J. Econ. Theory*, vol. 56, no. 2, pp. 434–450, 1992.
[19] K. Deshmukh, A. Goldberg, J. Hartline, and A. Karlin, "Truthful and competitive double auctions," in *Proc. 10th Annu. Eur. Symp. Algorithms*, 2002, pp. 127–130.
[20] D. Fudenberg, M. Mobius, and A. Szeidl, "Existence of equilibrium in large double auctions," *J. Econ. Theory*, vol. 133, no. 1, pp. 550–567, 2007.
[21] D. Zhao, D. Zhang, and L. Perrussel, "Mechanism design for double auctions with temporal constraints," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, vol. 1, pp. 472–477.
[22] S. Zaman and D. Grosu, "Combinatorial Auction-based mechanisms for VM provisioning and allocation in clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 107–114.
[23] X. Wang, J. Sun, M. Huang, C. Wu, and X. Wang, "A resource auction based allocation mechanism in the cloud computing environment," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, 2012, pp. 2111–2115.
[24] Q. Wang, K. Ren, and X. Meng, "When cloud meets eBay: Towards effective pricing for cloud computing," in *Proc. IEEE INFOCOM*, 2012, pp. 936–944.
[25] C.-C. Chang, K.-C. Lai, and C.-T. Yang, "Auction-based resource provisioning with SLA consideration on Multi-cloud systems," in *Proc. IEEE Annu. Comput. Softw. Appl. Conf. Workshops*, 2013, pp. 445–450.
[26] M. Anisetti, C. A. Ardagna, E. Damiani, P. A. Bonatti, M. Faella, C. Galdi, and L. Sauro, "E-auctions for multi-cloud service provisioning," in *Proc. IEEE Int. Conf. Services Comput.* 2014, pp. 35–42.
[27] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Proc. Annu. Int. Symp. Appl. Internet*, 2010, pp. 7–14.
[28] S. Shang, J. Jiang, Y. Wu, Z. Huang, G. Yang, and W. Zheng, "DABGPM: A double auction Bayesian game-based pricing model in cloud market," *Netw. Parallel Comput.*, vol. 6289, pp. 155–164, 2010.
[29] R. Prodan, MarekWieczorek, and H. M. Fard, "Double auction-based scheduling of scientific applications in distributed grid and cloud environments," *J. Grid Comput.*, vol. 9, no. 4, pp. 531–548, 2011.
[30] W. Wang, B. Liang, and B. Li, "Designing truthful spectrum double auctions with local markets," *IEEE Trans. Mobile Comput.*, vol. 13, no. 1, pp. 75–88, Jan. 2014.
[31] W.-Y. Lin, G.-Y. Lin, and H.-Y. Wei, "Dynamic auction mechanism for cloud resource allocation," in *Proc. IEEE 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, 2010, pp. 591–592.
[32] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proc. IEEE 4th Int. Conf. Utility Cloud Comput.*, 2011, pp. 178–185.
[33] W. Wang, B. Liang, and B. Li, "Revenue maximization with dynamic auctions in IaaS cloud markets," in *Proc. IEEE/ACM 21st Int. Symp. Quality Service*, 2013, pp. 1–6.
[34] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," in *Proc. IEEE INFOCOM*, 2013, pp. 1510–1518.
[35] D. Parkes and S. Singh, "An MDP-based approach to online mechanism design," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16, 2004, pp. 791–798.
[36] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 1994.
[37] R. G. Gallager, *Draft of Discrete Stochastic Processes*. Cambridge, MA: MIT Press, 2013.
[38] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[39] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2007.



Chunxiao Jiang (S'09-M'13) received the BS degree from Beihang University in 2008 and the PhD degree from Tsinghua University (THU) in 2013, both with the highest honors. He is currently a postdoc at the Department of Electronic Engineering, THU, with research interests including data-driven analysis on wireless and social networks. He received the Best Paper Award from IEEE GLOBECOM in 2013. He is a member of the IEEE.

Yan Chen (S'06-M'11-SM'14) received the bachelor's degree from the University of Science and Technology of China in 2004, the MPhil degree from Hong Kong University of Science and Technology (HKUST) in 2007, and the PhD degree from the University of Maryland College Park in 2011. His current research interests are in data science, network science, game theory, social learning and networking, as well as signal processing and wireless communications. He received multiple honors and awards including best paper award from IEEE GLOBECOM in 2013, Future Faculty Fellowship and Distinguished Dissertation Fellowship Honorable Mention from Department of Electrical and Computer Engineering in 2010 and 2011, respectively, Finalist of Dean's Doctoral Research Award from A. James Clark School of Engineering at the University of Maryland in 2011, and Chinese Government Award for outstanding students abroad in 2011. He is a senior member of the IEEE.

Qi Wang received the BS degree in electrical and information engineering from Xidian University, Xi'an, China, in 2011, and the MS degree from the University of Maryland, College Park, in 2014. He is currently an algorithm engineer at Douban Inc., China. His current research interests include recommendation systems and architectural aspects of machine learning systems.

K.J. Ray Liu (F'03) was named a distinguished scholar-teacher in the University of Maryland, College Park, in 2007, where he is a Christine Kim Eminent professor of information technology. He leads the Maryland Signals and Information Group conducting research encompassing broad areas of information and communications technology with recent focus on future wireless technologies, network science, and information forensics and security. He received the 2016 IEEE Leon K. Kirchmayer Technical Field Award on graduate teaching and mentoring, IEEE Signal Processing Society 2014 Society Award, and IEEE Signal Processing Society 2009 Technical Achievement Award. Recognized by Thomson Reuters as a Highly Cited Researcher, he is a Fellow of IEEE and AAAS. He is a director-elect of the IEEE Board of Director. He was the president of the IEEE Signal Processing Society, where he has served as a vice president – Publications and Board of Governor. He has also served as the editor-in-chief of the IEEE Signal Processing Magazine. He also received teaching and research recognitions from the University of Maryland including university-level Invention of the Year Award; and college-level Poole and Kent Senior Faculty Teaching Award, Outstanding Faculty Research Award, and Outstanding Faculty Service Award, all from A. James Clark School of Engineering. He is a fellow of the IEEE.