Data-Driven Stochastic Scheduling and Dynamic Auction in IaaS

Chunxiao Jiang*†, Yan Chen*, Qi Wang*, and K. J. Ray Liu*
*Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742, USA
†Department of Electronic Engineering, Tsinghua University, Beijing, 100084, P. R. China
E-mail: jchx@tsinghua.edu.cn, {yan, qwang37, kjrliu}@umd.edu

***Abstract*—With the emergence of large scale data processing systems and big data analysis, cloud computing has become more and more popular. In this paper, we focus on the mechanism design in the infrastructure as a service (IaaS) cloud computing service market. Most of existing works on mechanism design assume static and independent individual utility, while in practice the cloud service is provided in a dynamic environment. To solve such problems, we propose a stochastic matching algorithm based on Markov Decision Process (MDP), which aims at optimizing the long-term system efficiency by considering the opportunity cost in the future. Based on the MDP formulation, we further design an efficient (EF), incentive compatible (IC), individual rational (IR) auction mechanism. Finally, we conduct experiment using Google cluster-usage traces dataset and show that the proposed MDP-based VCG auction mechanism can achieve EF, IC and IR properties simultaneously.**

***Index Terms*—Cloud computing, Markov Decision Process, Vickrey-Clarke-Groves, efficient, incentive compatibility, individual rationality.**

I. Introduction

One of the most important concepts in Infrastructure as a Service (IaaS) is utility computing [1]. With 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 users and providers, auction becomes a natural choice for pricing in cloud computing. 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 users in the could computing market, it is difficult to design an effective auction mechanism that can efficiently utilize the computing resource.

In the cloud computing literature, there are some existing works on the auction-based cloud computing market model [2]-[11]. In [2]-[5], 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 [6]-[8]. Meanwhile, some dynamic auction models were proposed in [9]-[11] 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 buyers and sellers are long-term instead of one-shot. Considering these problems, in this paper, we propose a practical MDP-based VCG auction mechanism for the cloud computing market. Although the authors in [11] have used the MDP model to consider the opportunity cost in the near future, they focused on the scenario of single service provider (seller), which is substantially different from our work where we focus on the stochastic matching between multiple sellers and buyers.

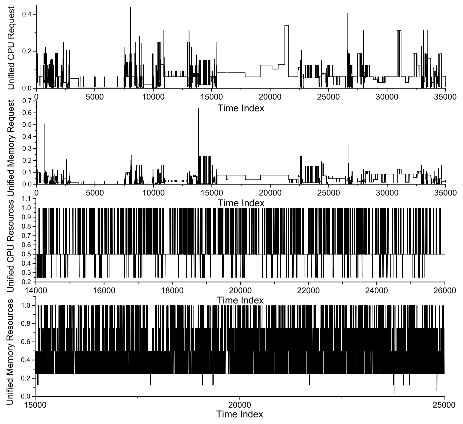


Fig. 1. Google cluster-usage traces: consumers with dynamic CPU requests and memory requests (first and second), providers with dynamic CPU and memory servers (third and fourth), all unified within [0,1].

Most of the existing works on auction mechanism design assume 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 folds. Firstly, the value of a IaaS consumer may vary according to the specific IaaS provider's computing resources. As shown in Fig. 1 drawn from the Google cluster-usage traces [12], matching a IaaS consumer who requests 0.2 CPU resource (unified within [0,1] by Google) with a 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. Secondly, The cloud computing service is provided in a dynamic environment. In Fig. 1, 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 in Section II. Then, based on the characteristics mined from the real data, we propose a learning strategy of optimal IaaS consumer-provider matching in Section III, and a practical MDP-based dynamic VCG auction mechanism in Section IV. In Section V, we conduct experiments to verify the performance. Finally, Section VI concludes our discussion.

II. 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 40GB. 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. A new task is submitted can be treated as a new buyer arrives while an existing task is completed can be regarded as a buyer leaves the system. 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 3 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.

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:

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

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 sells 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 I. Based on those statistic characteristics, we will discuss how to perform online matching and dynamic auction in the following.

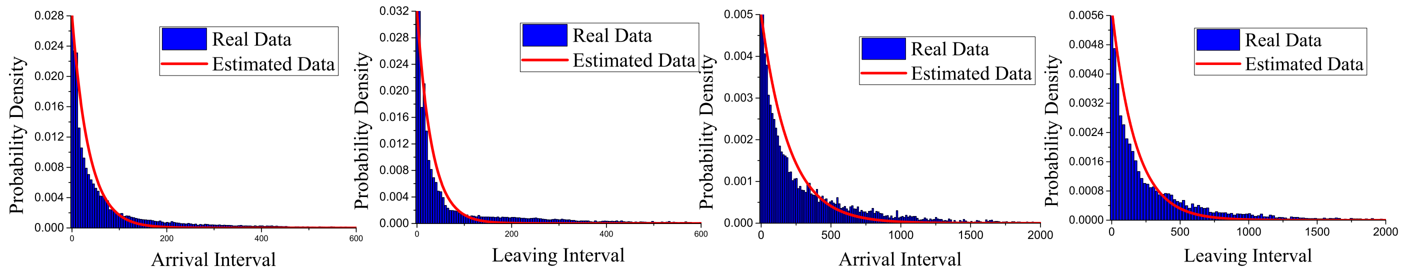
TABLE I
Traffic intensity parameters.

	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	$\lambda_{s2} = 0.005$	$\mu_{s2} = 0.0060$

III. Learning the Best Stochastic Matching

A. Stochastic Matching Using MDP

In our formulation of the cloud computing market, 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.



Arrival rate of type 1 buyers. Leaving rate of type 1 buyers. Arrival rate of type 1 sellers. Leaving rate of type 1 sellers.

Fig. 2. Traffic intensities of different types of buyers and sellers extracted from dataset.

Let us denote the matching between all buyers and sellers as X, the value of i can also be denoted as a function $z_i(X)$. Correspondingly, the cost for a seller to provide his/her resources is denoted as $c_j(X)$. 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:

$$R^{(t)} = \sum_{\substack{i,j:\\ i,j \text{ are matched by} \\ \text{matching rule } X^{(t)}}} z_i(X^{(t)}) - c_j(X^{(t)}) \tag{1}$$

As discussed in Section II, 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, which involves the opportunity cost in the future. 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, suppose that current system state is s and current matching rule is X, the long-term system value, $Q(s, X)$, can be calculated by

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

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 the corresponding optimal matching rule can be iteratively computed using [13]

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

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

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 [13].

B. *Construction of Concise State Space*

Suppose there are N buyers and M sellers in the system. To characterize the state of the system, we need a vector of length M to describe the potential values of a buyer and another scalar to describe the potential cost of a seller. This gives rise to a scale of $N \times M^2$ 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 and the matching rule, the space complexity will be of $N \times M$. Moreover, in such case 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 pre-defined options, rather than actually picking an exact number 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 a buyer/seller into a few coarse levels, and describe the 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 6-dimensional vector as

$$s = (n_1, n_2, n_3, m_1, m_2, m_3). \tag{7}$$

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

$$\sum_{l=1}^{3} x_{kl} \leq n_k, \quad \sum_{k=1}^{3} x_{kl} \leq m_l. \tag{8}$$

In such a case, the matching rule X can be represented by a 9-dimensional vector as

$$X = (x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{13}, x_{23}, x_{33}). \tag{9}$$

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 (8). The matching space grows polynomially with respect to system capacity and exponentially with respect to the number of level type partitions.

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 λ_{bk}/λ_{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. Consider small time intervals, the transition probability due to more than one arrival or departure can be neglected. Therefore, given current system state s and matching rule X, the state transition probability can be summarized in (6) below. 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.

IV. VCG AUCTION BASED ON MDP

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

$$v_i(w_i, X(s)) = g_i(r_i) + \delta \sum\nolimits_{s'} P(s'|s, X) v_i(w_i, X(s')), \tag{10}$$

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

$$P\left\{s'|s=(n_1,n_2,n_3,m_1,m_2,m_3), X=(x_{11},x_{12},x_{13},x_{21},x_{22},x_{23},x_{13},x_{23},x_{33})\right\} =$$
$$\begin{cases} \lambda_{b1} \text{ or } n'_1\mu_{b1}, & \text{if } s' = (n'_1+1, n'_2, n'_3, m'_1, m'_2, m'_3) \text{ or } (n'_1-1, n'_2, n'_3, m'_1, m'_2, m'_3); \\ \lambda_{b2} \text{ or } n'_2\mu_{b2}, & \text{if } s' = (n'_1, n'_2+1, n'_3, m'_1, m'_2, m'_3) \text{ or } (n'_1, n'_2-1, n'_3, m'_1, m'_2, m'_3); \\ \lambda_{b3} \text{ or } n'_3\mu_{b3}, & \text{if } s' = (n'_1, n'_2, n'_3+1, m'_1, m'_2, m'_3) \text{ or } (n'_1, n'_2, n'_3-1, m'_1, m'_2, m'_3); \\ \lambda_{s1} \text{ or } m'_1\mu_{s1}, & \text{if } s' = (n'_1, n'_2, n'_3, m'_1+1, m'_2, m'_3) \text{ or } (n'_1, n'_2, n'_3, m'_1-1, m'_2, m'_3); \\ \lambda_{s2} \text{ or } m'_2\mu_{s2}, & \text{if } s' = (n'_1, n'_2, n'_3, m'_1, m'_2+1, m'_3) \text{ or } (n'_1, n'_2, n'_3, m'_1, m'_2-1, m'_3); \\ \lambda_{s3} \text{ or } m'_3\mu_{s3}, & \text{if } s' = (n'_1, n'_2, n'_3, m'_1, m'_2, m'_3+1) \text{ or } (n'_1, n'_2, n'_3, m'_1, m'_2, m'_3-1); \\ 1 - \sum\limits_{k=1}^{3} (\lambda_{bk} + n'_k\mu_{bk}) - \sum\limits_{l=1}^{3} (\lambda_{sl} + m'_l\mu_{sl}), & \text{if } s' = (n'_1, n'_2, n'_3, m'_1, m'_2, m'_3); \\ 0, & \text{otherwise}; \end{cases} \tag{6}$$
$$n'_1 = n_1 - \sum_{l=1}^{3} x_{1l},\ n'_2 = n_2 - \sum_{l=1}^{3} x_{2l},\ n'_3 = n_3 - \sum_{l=1}^{3} x_{3l},\ m'_1 = m_1 - \sum_{k=1}^{3} x_{k1},\ m'_2 = m_2 - \sum_{k=1}^{3} x_{k2},\ m'_3 = m_3 - \sum_{k=1}^{3} x_{k3}.$$

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 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).
- *Pricing Rule:* The pricing rule is to collect payment

$$q_i(r_i, X^*(s)) = g_i(r_i) - V^*(s) + V^*\big(s - \{r_i\}\big), \quad (11)$$

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.

It is obvious that the MDP-VCG mechanism is efficient, since the matching rule solves the MDP condition for optimal system value function. In the following, we will theoretically show that our proposed MDP-VCG mechanism is Bayesian incentive compatible and Bayesian individual rational.

Theorem 1: The proposed MDP-VCG is Bayesian incentive compatible.

Proof: Let s be the true system state. According to the MDP-VCG mechanism, if buyer i reports the true type, then we have:

$$\begin{aligned} & v_i(w_i, X^*(s)) - q_i(w_i, X^*(s)) \\ &= v_i(w_i, X^*(s)) - E\left[g_i(w_i) - V^*(s) + V^*(s - \{w_i\})\right] \\ &= v_i(w_i, X^*(s)) - v_i(w_i, X^*(s)) + V^*(s) - V^*(s - \{w_i\}) \\ &= V^*(s) - V^*(s - \{w_i\}). \end{aligned} \quad (12)$$

On the other hand, if buy i cheats and does not report the true type, then

$$\begin{aligned} & v_i(w_i, X^*(s^{(i)})) - q_i(r_i, X^*(s^{(i)})) \\ &= v_i(w_i, X^*(s^{(i)})) - E\left[g_i(r_i) - V^*(s^{(i)}) + V^*(s^{(i)} - \{r_i\})\right] \\ &= v_i(w_i, X^*(s^{(i)})) - v_i(r_i, X^*(s^{(i)})) + V^*(s^{(i)}) \\ &\quad - V^*(s^{(i)} - \{r_i\}). \end{aligned} \quad (13)$$

According to the definition of the Q function

$$\begin{aligned} V^*(s^{(i)}) &= Q(s^{(i)}, X^*(s^{(i)})) \\ &= \sum_{n \in \{buyers\}, n \neq i} v_n(w_n, X^*(s^{(i)})) + \\ &\quad v_i(r_i, X^*(s^{(i)})) - \sum_{j \in \{sellers\}} c_j(X^*(s^{(i)})). \end{aligned} \quad (14)$$

Therefore, we can re-write (13) as

$$\begin{aligned} & v_i(w_i, X^*(s^{(i)})) - q_i(r_i, X^*(s^{(i)})) \\ &= \sum_{n \in \{buyers\}} v_n(w_n, X^*(s^{(i)})) - \sum_{j \in \{sellers\}} c_j(X^*(s^{(i)})) - V^*(s^{(i)} - \{r_i\}) \\ &= Q(s, X^*(s^{(i)})) - V^*(s^{(i)} - \{r_i\}) \\ &= Q(s, X^*(s^{(i)})) - V^*(s - \{w_i\}). \end{aligned} \quad (15)$$

Since the matching rule in MDP-VCG optimizes the system value function, we have

$$Q(s, X^*(s^{(i)})) \leq Q(s, X^*(s)) = V^*(s) \quad (16)$$

Therefore, according to (12), (15), and (16), we have

$$v_i\big(w_i, X(s)\big) - q_i\big(w_i, X(s)\big) \geq v_i\big(w_i, X(s^{(i)})\big) - q_i\big(r_i, X(s^{(i)})\big). \quad (17)$$

■

Theorem 2: The proposed MDP-VCG mechanism is Bayesian individual rational.

Proof: According to the definition of individual rationality in **Definition 3**, we need to show that

$$v_i(w_i, X^*(s)) - q_i(w_i, X^*(s)) = V^*(s) - V^*(s - \{w_i\}) \geq 0. \quad (18)$$

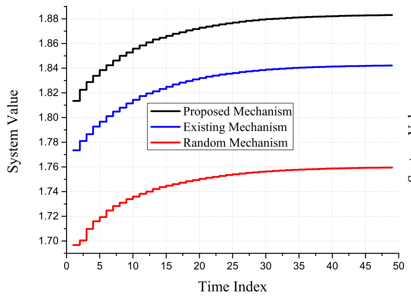
The inequality can be proved by contradiction. Let us assume $V^*(s) - V^*(s - \{w_i\}) < 0$. For state s, we can construct a matching rule by using $X^*(s - \{w_i\})$ on the buyer set, i.e., excluding buyer i in the matching process. In other words, we completely ignore buyer i. This will generate a system value of $V^*(s - \{w_i\})$. Therefore, we have $V^*(s - w_i) \leq V^*(s)$, which contradicts the assumption. ■

V. Evaluations

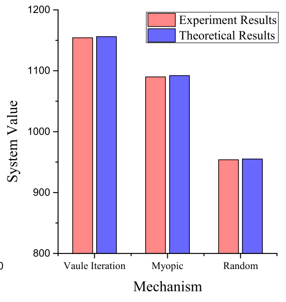
A. EF Verification

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 $z_i(X) = e^{-\beta\sigma_i/\zeta_j}$, and the cost of the seller as a linear function of his/her computing capacity as $c_j(X) = \gamma\zeta_j$. Then, we can define different types of sellers' values and different types of sellers' cost according to the types mined from the Google dataset in the previous subsection. 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 the experiment, we compare the system value of three different methods: our proposed mechanism, existing mechanism without considering the long-term efficiency ([3], [5], [7], [9], [10]), and random mechanism as a benchmark. 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 [3], [5], [7], [9], [10]. 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 $\alpha = 0.9$. Fig. 3-(a) shows the system value comparison



(a) System value for one instantiation.



(b) Accumulated system value.

Fig. 3. Efficiency comparison with perfect traffic model.

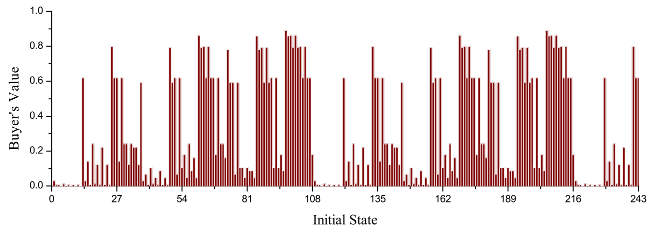
results, where all methods start with a same initial system state. The results are averaged over 1000 independent experiments. From Fig. 3-(a), we can see that due to the discount factor α, the system values of three methods converge to some constants as time slot index goes to 50. Apparently, our proposed mechanism performs better than the existing and random mechanisms. Moreover, as shown in Fig. 3-(b), we also compare the four methods in terms of accumulated system value, which is calculated from the summation of system values associated with 1000 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.

B. IC and IR Verification

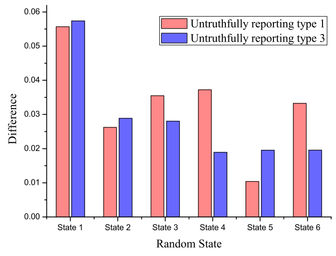
In the experiment of IC, IR verification, all the system settings are same with the previous subsection. Firstly, in Fig. 4-(a), we show the average value of a buyer with different initial system states. We can see that the buyer's value is always non-negative, which means that our proposed mechanism satisfies the individual rational property. Secondly, in Fig. 4-(b), we show the difference between the buyer's value from bidding truthfully and 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. 4-(b), we can see that the differences are always non-negative, which means that our proposed mechanism satisfy the incentive compatible property. Finally, in Fig. 4-(c), we show the budget of the auctioneer, which is the total payment from the buyers minus the total cost of the sellers. From Fig. 4-(c), we can see that the budget is always non-negative, i.e., the budget balance is also achieved.

VI. Conclusion

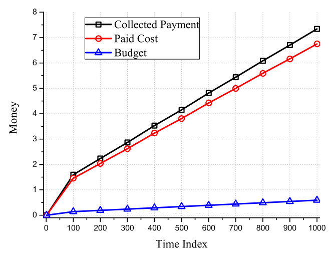
In this paper, we discussed the resource allocation and pricing issue in the cloud computing service market. We first propose a stochastic matching algorithm based on MDP model, then designed an efficient, incentive compatible, individual rational auction mechanism. Finally, we conducted experiments using real dataset to verify that the proposed mechanism can satisfy EF, IC and IR simultaneously.



(a) Buyers' individual rationality.



(b) Buyers' incentive compatibility.



(c) Auctioneer's budget balance.

Fig. 4. IR, IC and BB verification.

VII. Acknowledgement

This research was supported by NSFC China under projects 61371079, 61271267 and 91338203.

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.
[2] S. Zaman and D. Grosu, "Combinatorial auction-based mechanisms for vm provisioning and allocation in clouds," in *IEEE CloudCom*. IEEE, 2011, pp. 107–114.
[3] X. Wang, J. Sun, M. Huang, C. Wu, and X. Wang, "A resource auction based allocation mechanism in the cloud computing environment," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 2012, pp. 2111–2115.
[4] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *Proceedings of IEEE INFOCOM*. IEEE, 2012, pp. 936–944.
[5] C.-C. Chang, K.-C. Lai, and C.-T. Yang, "Auction-based resource provisioning with sla consideration on multi-cloud systems," in *Proceedings of IEEE Annual Computer Software and Applications Conference Workshops (COMPSACW)*. IEEE, 2013, pp. 445–450.
[6] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Proceedings of Annual International Symposium on Applications and the Internet (SAINT)*. IEEE, 2010, pp. 7–14.
[7] 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," *Network and Parallel Computing*, vol. 6289, pp. 155–164, 2010.
[8] R. Prodan, MarekWieczorek, and H. M. Fard, "Double auction-based scheduling of scientific applications in distributed grid and cloud environments," *J. Grid Computing*, vol. 9, no. 4, pp. 531–548, 2011.
[9] W.-Y. Lin, G.-Y. Lin, and H.-Y. Wei, "Dynamic auction mechanism for cloud resource allocation," in *IEEE CCGrid*. IEEE, 2010, pp. 591–592.
[10] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proceedings of IEEE Utility and Cloud Computing (UCC)*. IEEE, 2011, pp. 178–185.
[11] W. Wang, B. Liang, and B. Li, "Revenue maximization with dynamic auctions in iaas cloud markets," in *IEEE/ACM IWQoS*. IEEE, 2013, pp. 1–6.
[12] J. Wilkes and C. Reiss, "Google cluster-usage traces," [Online]. Available: http://code.google.com/p/googleclusterdata.
[13] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.