Cluster Computing (2021) 24:1855–1879
https://doi.org/10.1007/s10586-020-03225-9

Truthful online double auction based dynamic resource provisioning for multi-objective trade-offs in IaaS clouds

Yashwant Singh Patel[1] · Zahra Malwi[2] · Animesh Nighojkar[3] · Rajiv Misra[1]

Received: 15 July 2020 / Revised: 11 December 2020 / Accepted: 15 December 2020 / Published online: 11 January 2021

Abstract
Auction designs have recently been adopted for static and dynamic resource provisioning in IaaS clouds, such as Microsoft Azure and Amazon EC2. However, the existing mechanisms are mostly restricted to simple auctions, single-objective, offline setting, one-sided interactions either among cloud users or cloud service providers (CSPs), and possible misreports of cloud user's private information. This paper proposes a more realistic scenario of online auctioning for IaaS clouds, with the unique characteristics of elasticity for time-varying arrival of cloud user requests under the time-based server maintenance in cloud data centers. We propose an online truthful double auction technique for balancing the multi-objective trade-offs between energy, revenue, and performance in IaaS clouds, consisting of a weighted bipartite matching based winning-bid determination algorithm for resource allocation and a Vickrey–Clarke–Groves (VCG) driven algorithm for payment calculation of winning bids. Through rigorous theoretical analysis and extensive trace-driven simulation studies exploiting Google cluster workload traces, we demonstrate that our mechanism significantly improves the performance while promising truthfulness, heterogeneity, economic efficiency, individual rationality, and has a polynomial-time computational complexity.

Keywords Virtual machine · Double auction · Truthful · VCG-auction · Approximation · Energy-efficiency.

1 Introduction

The global cloud infrastructure services market is rapidly proliferating enterprise IT. According to a study by Mckinsey, public cloud expenditure is expected to grow six times the rate of general IT expenditure by 2020. The Infrastructure-as-a-Service (IaaS) cloud model is increasingly being adopted across several industry verticals to meet cloud user's real-time demands through virtualization technologies to virtually access computing resources, i.e., storage, networking-related services, or processing power at cloud service providers (CSPs) such as Rackspace, Amazon EC2, VMware, Salesforce, Google cloud, and Microsoft Azure platform. Hypervisor-based virtualization technologies such as VMware ESXi, XenServer, KVM, and Microsoft's Hyper-V have become very popular in recent years and massively applied in cloud infrastructures, explicitly, in IaaS. The key benefits of virtualization include availability, hardware independence, isolation, and security. CSPs offer cloud resources using virtual machines (VMs) to run applications (workload/services). Several other cloud services, such as Platform-as-a-Service (PaaS), Container-as-a-Service (CaaS) and Software-as-a-Service (SaaS), are most often deployed on top of an IaaS platform with all their applications running inside the VMs. However, in the data center, resources are often severely underutilized; and substantial energy saving can be achieved

✉ Yashwant Singh Patel
yashwant.pcs17@iitp.ac.in

Zahra Malwi
zahra.malwi1209@gmail.com

Animesh Nighojkar
anighojkar@usf.edu

Rajiv Misra
rajivm@iitp.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology Patna, Bihar, India

[2] ValueLabs LLP, Indore, India

[3] Department of Computer Science and Engineering, University of South Florida, 4202 E Fowler Ave, Tampa, FL 33620, USA

through efficient resource management strategies, i.e., resource allocation, scheduling, and consolidation [1–4].

This paper addresses the resource allocation problem in IaaS clouds. Efficient resource allocation problem is very challenging to IaaS cloud because, to satisfy the cloud users' requests of desired resources (i.e, Memory, CPU, network bandwidth, and storage), it has to balance multi-objective trade-offs between CSP's revenue and energy cost minimization while maintaining the performance as shown in Fig. 1. To maximize revenue, CSP prefers to allocate as many VMs as possible. However, it will drastically increase the power consumption due to a large number of active physical machines (PMs) [5].

In cloud trading markets, major IaaS providers today typically sell cloud resources via pre-configured VM instances of fixed types, while cloud customers acquire resources for executing tasks. Every participant in such a cloud computing market focuses on maximizing its revenues. Thus pricing is another critical factor for balancing the benefits of cloud users as well as CSPs. In trading, an efficient pricing strategy is required to maximize the individual profits of each participant. Cloud computing providers offer multiple pricing strategies, e.g., Microsoft Azure [6] provides resellers pricing, and Pay-As-You-Go (PAYG) models. Amazon EC2 [7] offers on-demand, reserve, spot pricing, and savings models. Most of the CSPs have adopted a fixed price strategy, i.e., charging customers a fixed amount for each pre-configured VM, where the total payment is calculated by the length of time, the total number of purchased VMs and the unit price. Despite the apparent simplicity in implementation, fixed pricing methods have failed to reflect the dynamic demand and supply relationship of the cloud market. As a result, it lacks efficiency and market agility compromising the CSP's profit and customers' utility. The adoption of different pricing models by multiple service providers poses the challenge to deliver cloud services at a reasonable price to users. Hence, in such a a federated cloud environment, we need a fair resource allocation technique to effectively allocate CSP's resources to those cloud users who can value them the most and without any market manipulation.

In economics, the auction [8] is become one of the popular trading forms as it maintains efficiency and fairly distributes the resources to those service users who will value them most and without any market manipulation. Unfortunately, we have observed that only simple auction mechanisms have been implemented by leading CSPs. Such simple auctions have significant limitations as follows:

1. One-sided interactions: In auctioning, one-sided interaction means an auction can either be performed by a buyer-side (i.e., cloud users) or the seller-side (CSPs). For example many e-commerce platforms following one-to-many negotiations [9]. In a buyer-side auctioning, a buyer procures asks from multiple sellers and determines a winner to buy the commodity. While in a seller-side auctioning, a single seller offers its distributed commodities to multiple buyers.
2. Offline setting: In an offline setting, all users' demands and bids are known in advance without considering the dynamics of an elastic model in the time-variant environment such as the arrival of users at some future time for any specified time interval, and time-based

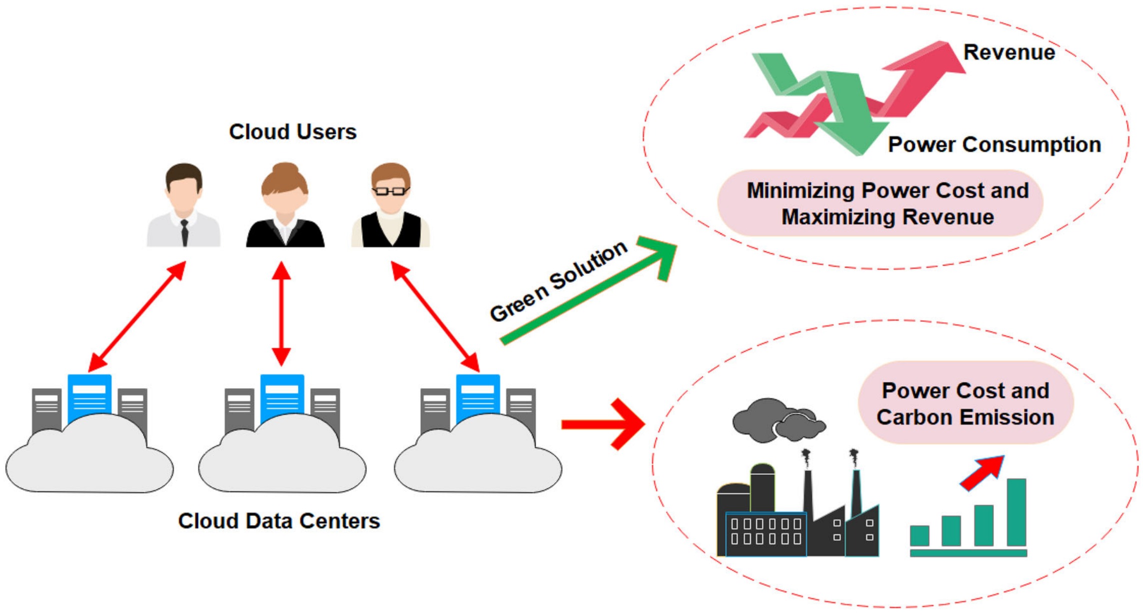
Cloud Users
Green Solution
Revenue
Power Consumption
Minimizing Power Cost and Maximizing Revenue
Cloud Data Centers
Power Cost and Carbon Emission

Fig. 1 Green cloud computing

server maintenance in data centers. This scenario gives the advantage of acquiring higher social welfare, however the cloud service users have to wait for some period of time prior to the actual resources are assigned. Also, in the time-varying resource allocation users' resource demands can not be consistent over time. For example, amid the COVID-19 lockdown, there is a sudden surge in traffic for video streaming platforms compared with the regular days. Another example of the Cab booking App server, which may not require many resources for data processing at night. This is because there are more users on the network during the peak period than the off-peak period.

3. Homogeneous task model: Existing double-auction techniques such as McAfee [10] was originally designed for homogeneous task model and not suitable for IaaS clouds having heterogeneous distribution of resources. Also, most of the studies have considered only a single type of VM, while the IaaS CSPs such as Microsoft Azure [6] and Amazon EC2 [7] offer heterogeneous types of VM instances.
4. One-to-one matching: Existing auction models such as in [11], restricted the final outcome of auction setting with one-to-one matching, where one buyer is matched to at most one seller and each seller will be alloted to at most one buyer.
5. Selfish strategies: Selfish strategies followed by cloud users or potential misreports of private information, such as the time-window of resource requirements and also the valuation, to obtain higher utilities. For example, in the mobile device cloud application model [12], some of the selfish mobile devices intended to maximize their utility by manipulating the claimed cost. Also, in the federated cloud environment, each selfish cloud provider desire to gain higher profit and total social welfare by declaring false information about the resource types and quantities [13].

To overcome the aforementioned drawbacks, we propose an **O**nline **T**ruthful **D**ouble **A**uction **M**echanism called 'OTDAM' for efficient resource allocation. OTDAM presents a more realistic case of the online double auction for IaaS clouds, with the unique characteristics of revenue maximization, energy cost minimization while performance is maintained. To overcome all the limitations of the current auction mechanisms, we consider to use a double auction or two-sided cloud market over a single-sided auction. The reason behind the selection is, a single-sided auction could be either a seller's side that compete to sell their goods or the buyer's side that compete to obtain a finite resource. But in reality, a CSP can sell IaaS resources to multiple service users, and users can purchase VMs of various types together from CSPs. As the CSP dynamically packs the requested VMs with numerous types of resources on heterogeneous PMs (cloud servers), it is more difficult to find an optimum provisioning decision under different server operational costs.

To solve this problem, we merge together a seller's side as well as a buyer's side auction design to devise a two-sided auction market framework, commonly referred to as 'double auction' for many-to-many negotiations. A double auction strategy is an efficient trading technique for organizing trade in two-sided market actively involving service users to bid and CSPs to ask with their individual demands. For example, the leading foreign exchange (FX), New York Stock Exchange (NYSE), and NASDAQ employ double auctions variants [14]. Due to the participation of multiple sellers (servers) and buyers (cloud users), our proposed scenario ideally matches to double auction market. Thus, we model the problem of resource provisioning in cloud as a truthful online double auction mechanism, where our action obtains a maximal matching between the set of cloud servers and the service users in IaaS clouds. This work proposes a completely different approach of online auctioning for IaaS resource trading with the unique properties of elasticity for time-varying arrival of cloud users' demands requesting for heterogeneous VMs and modeling the operational cost of servers under time-based maintenance in data centers. The auction mechanism design consists of a winning bid algorithm to find matching between cloud service users and servers and a payment algorithm applying price function to determine how much the cloud providers charge for compute instances of each cloud user.

1.1 Motivation

As demonstrated in Fig. 2, we focus on a two-sided IaaS cloud environment with cloud service users and service providers. Our study is driven by realistic cloud users demand and service provisioning scenarios where both cloud users and service providers are dynamic. It means the cloud service users may arrive and leave the system dynamically. Also, the physical machines (PMs) or servers in cloud data centers may have to undergo periodic maintenance, which may interrupt the services. Given the arrival-departure of cloud users and maintenance duration of the PMs, we need to perform matching of service demands, i.e., VMs to the PMs. After the arrival, service users submit their demands in the form of bids comprising start-time of VM, end-time of VM, resource requirement, and unit valuation for acquiring one-unit capacity in one-time slot.

As shown in the example, the service user is requesting a VM from 5:00 pm to 7:00 pm. The unit valuation is \$2.00. Once the cloud service provider receives a bid, first, it determines the type of allocated VMs based on

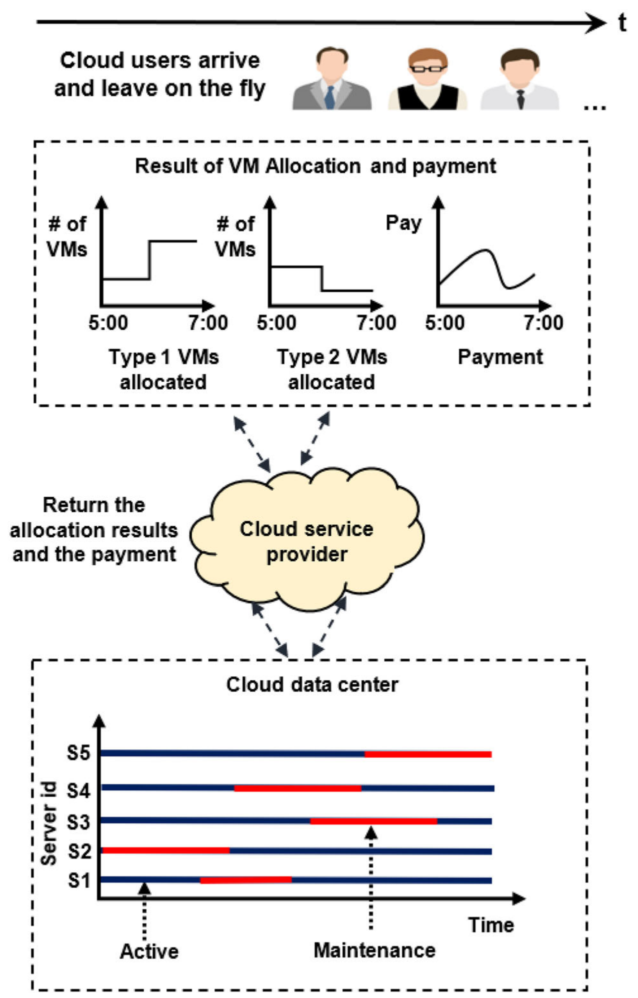

Fig. 2 Illustration of two-sided online interactions between cloud users and service providers

computation, memory, and storage capability. It then performs matching of user-requested VMs to the PMs while considering the time-varying arrival of demands and maintenance-duration of servers.

1.2 Contributions

The main technical contributions of the proposed work are as follows:

– We model the problem of dynamic resource allocation in IaaS clouds as an online double auction that offers an elastic framework for time-varying cloud user demands and availability of physical machines (PMs) due to periodic, or time-based maintenance in cloud data centers.
– We propose an online truthful double auction mechanism (OTDAM) consisting of a weighted bipartite matching based winning-bid algorithm for resource provisioning and a Vickrey–Clarke–Groves (VCG) driven algorithm for payment calculation of winning bids.
– Through rigorous theoretical analysis, we prove that OTDAM achieves truthfulness, economic-efficiency, individual rationality, and polynomial-time computational complexity.
– Based on the extensive trace-driven simulation study, we show that OTDAM maximizes revenue and minimizes energy cost while maintaining high performance in cloud data centers compared with the existing auction approaches.

1.3 Organization

The rest of the paper is structured as follows: In Sect. 2, we provide related works on double auction and resource allocation in IaaS clouds. In Sect. 3, we discuss the system model and problem formulation for dynamic resource provisioning problem. In Sect. 4, we present working of OTDAM, algorithms, an illustrative example, and also provide the proofs of desirable auction properties. We evaluate the approaches through extensive simulation studies in Sect. 5. Finally, Sect. 6 provides conclusions with some of the directions for future research.

2 Related works

This section presents some related works on double auction techniques and resource allocation approaches in IaaS clouds.

2.1 Double auction

In [15], Kumar et al. presented a comprehensive survey of double-auction approaches and investigate the benefits of applying double-auction in cloud markets for the trading of computing resources. Kumar et al. [16] further proposed a truthful combinatorial double-auction technique referred to as ’TCMDAC’ for pricing and resource allocation of computing resources in a cloud computing market. Farajian et al. [17] provided a continuous double-auction strategy for efficient resource allocation in the cloud market. In economics literature, McAfee [10], and Vickrey–Clarke–Groves (VCG) [18–20] are two well-known double-auction models. McAfee is based on the Trade-Reduction (TR) method, and it can achieve the properties of individual rationality, budget-balance, and truthfulness (incentive compatibility). While, the VCG based double-auction model can fulfill the property of individual rationality [11], and it can also satisfy the truthfulness property [21].

The McAfee [10] auction method can be illustrated as follows: Firstly, the auctioneer sorts the bids of buyers in non-increasing order, where $B_1 \geq B_2 \geq B_3 ... \geq B_m$. The ask of sellers' are sorted in non-decreasing order: $A_1 \leq A_2 \leq A_3 \ldots \leq A_n$. Here, the efficient number of trades is the number $k \leq min\{n, m\}$, where $B_k \geq A_k$ and $B_{k+1} < A_{k+1}$. Finally, the median can be defined as $p_0 = \frac{A_{k+1}+B_{k+1}}{2}$. The McAfee needs a budget balancer to operate. If $A_k \leq p_0 \leq B_k$, then all efficient k buyers and sellers trade for the resource at price p_0. Otherwise, only $k-1$ buyers and sellers trade for $k-1$. In this case buyers pay B_k and the sellers recieve ask A_k.

Generalized VCG double auction mechanism has also received popularity as they are efficient and strategy-proof. Without the budget-balance problem, the VCG payment scheme supports efficient, individual rationality, and strategy-proof exchange. VCG optimizes social welfare while achieving truthfulness property. It achieves budget-balance property by distributing "surplus" to each agent [21]. During VCG based auctioning, if $B_i \leq A_j$, then no trade is performed. If $B_i > A_j$, then only the trading is performed, and the buyer pays the ask A_j, and the seller receives the bid B_i. The McAfee and VCG auction mechanisms can not be directly applied for the proposed scenario due to the following limitations:

(i) McAfee provides a simple exchange environment for single-unit resources only, where sellers and buyers trade for single-units of the same good, and consequently it is not applicable for IaaS clouds having multi-unit resources.
(ii) McAfee's mechanism is weakly budget-balanced.
(iii) The McAfee [10] double-auction technique was originally designed for homogeneous commodities where a buyer can be assigned to one seller only. But it is not suitable for IaaS clouds having heterogeneous commodities and need multiple assignments of a seller.
(iv) VCG payment scheme is the most efficient truthful mechanism. However, it does not ensure the criteria of budget-balancing. Subsequently, the auctioneer actually has to subsidize the trade.

Huang et al. [22] designed a generalized model of McAfee [10] for a multi-unit trading environment, which is proven to be budget-balanced, truthful, and individually rational. Babaioff et al. [23] proposed an extended Trade Reduction (TR) technique for single unit commodities. It performs the bidding of buyers in bundles, and each seller sells one unit of each commodity. Chu and Shen [24] developed a multi-stage and truthful double auction approach for the consumer to consumer (C2C) market environment. The authors have assumed the single-unit market where a consumer bids for a bundle of commodities, and the seller sells a single unit for a single commodity. The mechanisms are proven to be truthful, budget-balanced, individually rational, and asymptotic efficient. Chu and Shen [25] also designed 2-incentive compatible double auction-based multi-stage mechanisms named as buyer-competition, and modified-buyer-competition). The authors have assumed a single-unit trading market where a buyer acquires the commodities in bundles with a seller selling a unit capacity for a single type of commodity. The approaches are further proven to be budget-balance and incentive compatibility.

Chu [26] provided a padding approach in a multi-unit environment, which is highly efficient than the single-unit schemes. The double auction mechanisms are proven to be budget balanced, incentive compatible, and asymptotic efficient. Mishra et al. [27] designed a descending price Dutch auction mechanism for multiple heterogeneous items. Demange et al. [28] proposed two dynamic auction mechanisms to achieve equilibrium and degree of accuracy. Ausubel et al. [29] presented a dynamic auction approach for multiple heterogeneous commodities. Charles et al. [30] devised a multiple unit double-auction that enables numerous units or block trades in the double-auction framework. Yang et al. [11] designed TASC auctioning for cooperative communication, where the wireless node can participate in relay services trading. The overall comparison of the aforementioned double-auction mechanisms is shown in Table 1.

2.2 Resource allocation in IaaS clouds

Jin et al. [38] provided cloudlet based resource sharing auction schemes for mobile cloud computing. However, these auction techniques are only applicable for homogeneous task models, and resultant matching for auctioning is one-to-one. Wang et al. also [12] proposed two truthful auctioning schemes for task based allocation in mobile clouds. They presented a VCG-driven truthful auction approach in the homogeneous cloud environment, and also a winning-bid algorithm for heterogeneous cloud models. In [31–35], and [37] CSPs adopted the auction-based resource allocation model as an economic paradigm to provide VM resources to the worthy users. In this design, the cloud users submit their resource demands with their values to the CSP. Then the CSP determines the allocation of VMs to the users for maximizing revenue. In [32], Nejad et al. given a VCG-based truthful auction scheme for revenue maximization, and prove that it is an NP-hard optimization problem.

However, to deal with the dynamic environment, Mashayekhy et al. [31] and Zhang et al. [34] developed an online truthful scheme that is invoked during the arrival of user requests. Zhang et al. [35] further enhance the online mechanism by giving more flexibility for user bidding. It

Table 1 Comparison of existing double auction approaches in the economics literature

Algorithm	Double auction	Auction properties				Heterogeneous	Dimension
		TF	IR	BB	EE		
Huang et al. [22]	✔	✔	✔	×	×	×	Multi-item
Babaioff et al. [23]	✔	✔	✔	✔	×	✔	Single-item
Chu et al. [24]	✔	✔	✔	✔	×	✔	Single-item
Chu et al. [25]	✔	✔	✔	✔	×	✔	Single-item
Chu et al. [26]	✔	✔	✔	✔	×	–	Multi-item
Mishra et al. [27]	×	–	–	–	–	✔	Multi-item
Demange et al. [28]	×	✔	–	–	–	✔	Multi-item
Ausubel et al. [29]	×	–	–	–	–	✔	Multi-item
Charles et al. [30]	✔	×	×	×	×	×	Multi-item
Yang et al. [11]	✔	✔	✔	✔	×	✔	Multi-item

TF truthful, *IR* individually rational, *BB* budget balance, *EE* economic efficient

means if the required resources are accessing during a particular time slot, then a user request can be accommodated. Wang et al. [36] consider an efficient VM allocation problem for reducing energy consumption. Zaman et al. [37] presented the VM allocation problem as a combinatorial auction problem. They have provided combinatorial auction-based approaches and compare with fixed-price allocation approach. Jin et al. [38] designed a truthful incentive mechanism (TIM) for auctioning of resources between the cloudlets and mobile devices. Lu et al. [39] suggested a truthful double auction scheme for bridging the gap between cloud users' task demands and service providers' resources for two-sided markets.

Zhang et al. [40] provided an online auction method for time-varying assignment and pricing of multidimensional resources in the cloud environment. Middya et al. [41] developed 'TARA', a multi-unit double auction approach for resource provisioning in a federated cloud environment. The authors have considered a multi-buyer and multi-seller double auction design for heterogeneous cloud resources. Patel et al. [42] designed 'TDAM', a truthful double auction scheme to provision VMs in cloud data centers. Jin et al. [43] developed 'ICAM' an incentive-compatible auction approach to assign cloudlets in order to fulfill the service requirements of mobile devices.

However, the main problem with the reported works is the lack of efficient pricing strategies for minimizing the cost of energy consumption and maximum revenue realized by satisfying the arriving user's demands. In comparison to the existing techniques, the auctioning problem studied in this work mainly differs in the following aspects: (i) heterogeneity of cloud environment where VMs have different resource requirements and PMs with different resource availability; (ii) double auction mechanism to consider the dual preferences where VMs act as *buyers* and PMs act as *sellers*; (iii) truthful auctioning while considering the fundamental research issues of revenue maximization and energy cost minimization, both are critical objectives for VM allocation scheme; (iv) dynamics of an elastic model in the time-variant environment such as arrival and departure of users and time-based server maintenance in data centers. Table 2 presents a comparison between the proposed approach and the closely related works in 12 different aspects.

3 System model and problem formulation

3.1 Auction model

We model the dynamic resource provisioning problem as an online double auction model, as illustrated in Fig. 3.

The auction model comprises of two-sided interactions between cloud service users and CSP. The auctioneer handles the auction process and acts as a trusted third party between cloud service users called buyers, and servers of CSP called sellers. Note that we will use buyers interchangeably with cloud service users and sellers with servers of a cloud service provider. We assume that the cloud service users request for VM instances. CSP offers a wide variety of instance types including varying combinations of memory, CPU, networking capacity, and storage. The CSP allocates the requested resources to the servers or physical machines (PMs). Cloud service users may arrive and leave the system dynamically. They submit their demand to reserve the resources for future use. A cloud service user submits a resource request in the form of a bid consisting of the service-time window i.e, start and finish times to use VMs, resource requirement, and the valuation or minimum price per unit time for getting one unit capacity. All the participants submit bids and asks to the cloud auctioneer in such a manner so that no auction participant has any idea about a realistic bids/asks of any other participants. With the help of received bids and asks values, an auctioneer

Table 2 Comparison of existing auction techniques in cloud literature

Algorithm	Auction type	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}
Wang et al. [12]	VCG	✔	×	×	Simple	HO&HE	Not supported	Offline	✔	✔	×	✔
Mashayekhy et al. [31]	VCG	✔	×	×	Simple	HE	Not supported	Offline	✔	×	×	✔
Nejad et al. [32]	VCG	✔	×	×	Simple	HE	Not supported	Offline	✔	×	×	×
Grosu et al. [33]	VCG	✔	×	×	Simple	HE	Not supported	Online	✔	×	×	×
Zhang et al. [34]	VCG	✔	×	×	Simple	HE	Not supported	Online	✔	×	×	✔
Jiang et al. [35]	VCG	✔	×	×	Simple	HE	Not supported	Online	✔	×	×	✔
Wu et al. [36]	–	×	✔	×	Simple	HE	Supported	Online	×	×	×	✔
Zaman et al. [37]	–	✔	×	×	Simple	HE	Not supported	Offline	✔	×	×	×
Jin et al. [38]	–	✔	×	×	Double	HE	Not supported	Offline	✔	✔	×	✔
Lu et al. [39]	–	✔	×	×	Double	HE	Not supported	-	✔	✔	×	✔
Zhang et al. [40]	–	✔	×	×	Simple	HE	Supported	Online	✔	✔	×	✔
Middya et al. [41]	McAfee	✔	×	×	Double	HE	Not supported	Offline	✔	✔	×	✔
Patel et al. [42]	VCG	✔	✔	×	Double	HE	Not Supported	Offline	✔	✔	✔	×
Jin et al. [43]	–	✔	×	×	Double	HE	Not Supported	Offline	✔	✔	×	✔
Proposed	VCG	✔	✔	✔	Double	HE	Supported	Online	✔	✔	✔	✔

F_1 maximize revenue, F_2 minimize energy, F_3 maximize performance, F_4 auction:simple/double F_5 environment:HO:homogeneous/HE:heterogeneous, F_6time-varying VM allocation, F_7 online/offline, F_8 truthful, F_9 individually rational, F_{10} economic efficient, F_{11}computational efficiency

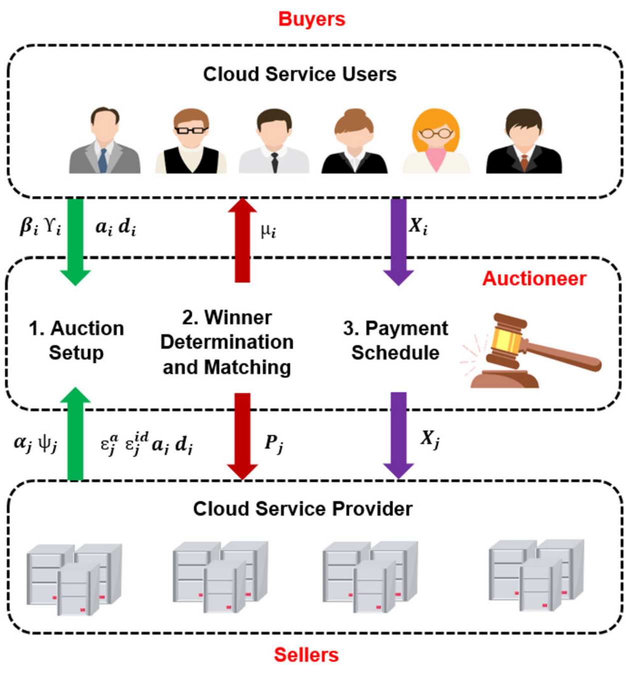

Fig. 3 Auction model

discovers mapping between the PMs of CSP and the cloud service users. The auctioneer also determines the hammer price (clearing-price) and the final payment. In the cloud market, a CSP can deliver the cloud services to users if and only if the auctioneer discovers a suitable matching between that particular CSP and the service user. Now, let us understand the entities involved in the auction process.

3.1.1 Sellers (i.e., Physical machines (PMs))

Cloud service providers offer κ servers or PMs to fulfill the demands of η VM instances in a cloud data center. A VM instance must be placed to at most one cloud server. Also, it cannot operate on multiple resources provided by multiple servers. Each seller can be expressed as a 11-tuple $S_j = <\alpha_j, \sum_j, \psi_j, \phi_j, \tau_j, a_j, d_j, \chi_j, \ \epsilon_j^{id}, \epsilon_j^{a}, v_j>$, where α_j represents the ask-per-resource submitted by S_j, $\sum_j$ indicates the list of hosted VMs at the particular cloud server, ψ_j implies the total resources available at the server, ϕ_j denotes the available resources, τ_j refers the response time of server, χ_j represents the payment the seller will gain when the auction is completed, $a_j \in T$ and $d_j \in T$ denote the arrival and departure time respectively and v_i represents the valuation of the ask raised by the Seller. ϵ_j^a indicates the energy consumption of the fully utilized server and, ϵ_j^{id} denotes the energy consumption of the fully idle server.

3.1.2 Buyers (i.e., The cloud users)

We assume a set of η buyers $B = \{B_1, B_2, \ldots, B_\eta\}$ requesting resources from κ servers $S = \{S_1, S_2, \ldots, S_\kappa\}$. A user request $C \in B$ arrives at slot t and sends a bid to the auctioneer. A user specifies the VM requirement according to its own needs and hence will report honestly. A server after joining the system dynamically will raise an ask per resource demanded by the buyer. Each buyer can be expressed as 7-tuple, $B_i = <\beta_i, \mu_i, \Upsilon_i^m, \chi_i, a_i, d_i, v_i>$ where β_i denotes the bid per resource which B_i places, μ_i

represents the server allocated to B_i, Υ_i^m is the type m VM resource requirement and χ_i denotes the total payment the buyer pays after auction is over, $a_i \in T$ and $d_i \in T$ indicate the start-time and finish-time of the buyer respectively, v_i denotes the valuation of the bid raised by buyer.

3.1.3 Auctioneer

The cloud auctioneer acts as a trustworthy third-party, which decides the mapping between the dynamic set of winning buyers $\mathbb{B}(t) \in B(t)$ and the set of winning sellers $\mathbb{S}(t) \in S(t)$ at time t, i.e, $\sigma : \{i : B_i \in \mathbb{B}\} \rightarrow \{j : S_j \in \mathbb{S}\}$. The auctioneer carries the task in slots and hence the set $T = \{0,\ 1,....,\ \mathrm{t}\}$. The auction process is performed into three phases, as follows:

– *Platform setup* Cloud users (buyers) and servers (sellers) details are taken as input and arranged accordingly.
– *Winner-determination and matching* To assign the winning cloud servers according to their asking and energy, response-time values to the winning cloud users as per their bids.
– *Payment schedule* Using a VCG-based payment strategy, we determine the payment ϕ_i^B the winning buyer $\mathbb{B}(t) \in B$ is charged and also the payment ϕ_j^S the winning seller $\mathbb{S}(t) \in S$ is rewarded per-unit of resource.

3.2 Assumptions

We make some practical assumptions for online auction model as follows:

1. Buyers & sellers have bounded patience measured as K slots of waiting time, *i.e.* $a_i + K \geq d_i$.
2. The auctioneer holds the resource allocation and forwards it to the buyer when the buyer departs. It also contains the payment to the server and pays it when it leaves.
3. The cloud users and servers have the same bids and asks respectively for the same type in the heterogeneous environment.

In Table 3, we have summarized some important mathematical notations used in this paper.

3.3 Utility functions

Definition 31 (*Buyer Utility Function*) For a buyer (cloud service user) $B_i \in \mathbb{B}$ the utility represents the difference between the true valuation of the bid raised and price paid to the auctioneer.

$$\mathbb{U}_i^B = \left\{ \begin{array}{cc} \Upsilon_i v_i - \chi_i, & \text{if } B_i \in \mathbb{B} \\ 0, & \text{otherwise} \end{array} \right\} \tag{1}$$

where v_i is the valuation of the buyer's instance. v_i states the buyer's true cost who makes willingness for payment of a single resource usage.

Definition 32 (*Seller Utility Function*) For a seller (PM) $S_i \in \mathbb{S}$ the utility can be characterized as the difference between payment received from the auctioneer and actual cost to perform resource allocation.

$$\mathbb{U}_j^S = \left\{ \begin{array}{cc} \chi_j - (\phi_j - \psi_j)v_j, & \text{if } S_j \in \mathbb{S} \\ 0, & \text{otherwise} \end{array} \right\} \tag{2}$$

where v_j is the valuation of the seller's instance. v_j tells the seller's true cost who has to sustain to provide a single resource to the buyer.

Definition 33 (*Auctioneer Utility Function*) The utility of a matched pair of a cloud service user to a server in a particular time slot is the payment difference between the buyer to the seller.

$$\mathbb{U}_i^A = \left\{ \begin{array}{cc} \chi_i - \chi_j, & \text{if } \beta_i \neq \alpha_i \\ 0, & \beta_i = \alpha_i \end{array} \right\} \tag{3}$$

where $\mathbb{U}_i^A$ is utility of auctioneer.

3.4 Problem formulation

Our objective is to maximize the revenue through the allocation of VM instances and minimize the energy consumption of cloud servers by examining the energy performance trade-off under the allocation and resource constraints, while considering the time-varying arrival of the cloud service users during the server maintenance period. Also, we aim to achieve the desired double auction features, such as truthfulness, economic efficiency, and individual rationality. Next, we mathematically describe the optimization problem and then introduce the properties of double auction.

Definition 34 (*Revenue Maximization*) Let β_t is the set of bids received by time $\tilde{t}$, $\Gamma_t = \{\beta_t | t_i \leq \tilde{t}\}$. Then, the total number of sold type-m VMs till time slot $\tilde{t}$ is:

$$\sum_{i=1}^{\eta-1} \Upsilon_i^m p_i(t) + \Upsilon_\eta^m \beta_\eta(t), \tilde{t} \leq t \leq T. \tag{4}$$

Next, we formulate the revenue maximization objective as follows:

$$maximize \sum_{t=1}^{T} \left[\sum_{i=1}^{\eta-1} \Upsilon_i^m p_i(t) + \Upsilon_\eta^m \beta_\eta(t) \right], \forall m \in M, \forall t \in [1, T] \tag{5}$$

where,

Table 3 Main notations and their descriptions

Notations	Description
η	Number of VMs
κ	Number of PMs/servers
T	Total time slots
B_i	An instance of buyer
S_i	An instance of seller
k_b	Bounded patience time for buyer
k_s	Bounded patience time for seller
β_i	*ith* buyer's bid per instance
μ_i	The server allocated to the *ith* buyer
Υ_i^m	Type *m* VM resource requirement of the *ith* buyer
χ_i	Total payment to be paid by the *ith* buyer
a_i	*ith* buyer arrival time
d_i	*ith* buyer departure time
v_i	Valuation of the bid raised by the *ith* buyer
α_j	*jth* seller's ask per instance
$\sum_j$	List of hosted VMs at *jth* seller
ψ_j	Total resources at the *jth* seller
ϕ_j	Resources available at the *jth* seller
χ_j	Total Payment to be paid to the *jth* seller
a_j	Start-time of server *j*
d_j	End-time of server *j*
τ_j	Response time of *jth* server
ϵ_j^a	Energy consumed when *jth* seller is fully active
ϵ_j^{id}	Energy consumed when *jth* seller is fully idle
$\mathbb{B}$	Winning buyers set
$\mathbb{S}$	Winning sellers set
$\Pi_j(t)$	Server utilization of the *jth* server at time *t*
B	Set of all the buyers
S	Set of all the sellers
Θ	Ordered list of buyers with respect to bid-per-instance
$\oint_j$	Energy consumption difference value
σ_j	Seller desirability
Ω	Ordered list of sellers with respect to their σ-values
∇	Asymptotic approximation ratio
$\oint^{id}$	Minimum idle energy consumption
$\oint^{a}$	Minimum active energy consumption
$\oint_{total}$	Total energy consumption
$\oint_{opt}$	Optimal energy consumption

$$p_i = \begin{Bmatrix} \alpha_{\mu_i}, & \text{if } \alpha_{\mu_i} \geq \beta_{\theta+1} \\ \beta_{\theta+1}, & \text{otherwise} \end{Bmatrix} \tag{6}$$

Here θ represents the position of buyer B_i in Θ, under the following constraints -

1. *Resource capacity constraint* The resource capacity constraint indicates that the allocated capacity of the total number of hosted type *m* VMs at a cloud server *j* must be less than or equal to the total number of available resources at server *j* of the data center. Implicitly, it is guaranteed that the allocation of resources for any type *m* VM will not reach beyond the capacity of server *j*.

$$\sum_{k \in \sum_j} \Upsilon_k^m \leq \psi_j, \forall 1 \leq j \leq \kappa, t \in T \tag{7}$$

2. *Placement constraint* The placement constraint ensures that the requested type *m* VM can be provisioned on at most one server at the cloud data center.

$$\sum_{t \in T} \sum_{\mu \in S_\kappa} y_{\eta t \mu} \leq 1, \forall B_\eta(t) \in \mathbb{B}, \mu \in [1, \kappa], t \in T \tag{8}$$

Revenue can be maximized by increasing the cardinality of winning buyers set denoted by ($\mathbb{B}$).

$$maximize|\mathbb{B}| : \mathbb{B} \subseteq B \tag{9}$$

The fundamental idea behind the maximization of VM allocation is to always place the seller's costly resources to the buyers having a higher valuation, who can afford to pay. Consequently, it will leave the economic seller resources for the buyers having a low valuation.

Definition 35 (*Energy Minimization*) The CSP's operational cost mainly depends on the energy consumption cost. In fact, the power consumption of a data center increases linearly as the resource utilization increases. The total energy cost at time slot *t* can be stated as:

$$minimize \sum_{j=1}^{\kappa} (\epsilon_j^a \Pi_j(t) + \epsilon_j^{id}(1 - \Pi_j(t))) \tag{10}$$

where Π represents the server utilization denoted as -

$$\Pi_j(t) = \frac{\psi_j(t) - \phi_j(t)}{\psi_j(t)}, t \in T \tag{11}$$

Definition 36 (*Performance Maximization*) As suggested by Gupta et al. [44] "the ERP (Energy and Response-timeratio. Here the performance can Product) can be applied to acquire the trade-off between the performance and energy parameter". By minimizing ERP, we can maximize the "performance-per-watt (PPW)" ratio. Here the performance can be denoted as the inverse of run-time or response–time denoted by *R*. The ERP value is calculated by Eq. 12:

$$ERP = \delta w_1 E \times \delta w_2 R \tag{12}$$

where δw_1 and δw_2 represent the domination factors for *E* and *R* respectively [44, 45]. Hence the overall objective function for performance maximization can be written as:

$$minimize(ERP) \tag{13}$$

3.5 NP-hardness

Theorem 1 *The proposed online VM allocation problem is NP-Hard.*

Proof To prove that the online VM allocation problem is NP-Hard, let's examine the problem of bin-packing. Assume the set of n items with different weights $w_1, w_2, w_3, \ldots, w_n$ and bins each of with capacity c. Our goal is to place each item to a bin such that the number of total used bins is minimized. Next, we examine the proposed VM allocation problem. Assume a set of η VM instances requiring heterogeneous resources $\Upsilon_1^m, \Upsilon_2^m, \Upsilon_3^m, \ldots, \Upsilon_\eta^m$ and κ servers with $\psi_1, \psi_2, \psi_3, \ldots, \psi_\kappa$ capacities. The objective is to allocate each VM instance to a cloud server in such a way that the total number of allocated VM instances is maximized and the overall energy consumption of cloud servers is minimized. Now to prove a given problem is NP-Hard, it is sufficient to prove that its sub-problems are NP-Hard. So, our proposed problem can be further classified into two sub-problems as follows:

1. Maximize the number of assigned VMs: We can consider two cases of this problem as follows:
   - *Case 1* When each cloud server is having unlimited amount of resources: VM instances can be allocated to the server by using the criteria of least ask-per-instance, let's assume S_j, and now we require to pick the buyers satisfying the constraint $\beta_i \geq \alpha_j$. Then we compute the final buyers' payment χ_i based on their corresponding bid-per-instance value, which can be performed with polynomial-time complexity.
   - *Case 2* When each cloud server is having limited amount of resources: VM instances can be allocated to the server by applying the Algorithm 2 that is equivalent to assigning η VM instances on κ cloud servers in such a way that the total number of allocated VM instances are maximized. It also ensures the total number of active cloud servers are minimized, satisfying the allocation constraint and resource constraint. Thus, this problem is equivalent to the problem of Bin-Packing, which is NP-Hard. Afterward, we compute the final payment of buyers' and sellers' denoted as χ_i and χ_j respectively by applying the Algorithm 3, which can be solved with polynomial-time complexity.
2. Minimize the overall energy consumption of cloud servers: We can examine two cases of this problem as follows:
   - *Case 1* When each cloud server is having unlimited amount of resources: VM instances can be allocated to the server having the least energy consumption, let's assume S_j, and now we require to pick the buyers satisfying the constraint $\beta_i \geq \alpha_j$. Then we compute the buyers' payment χ_i based on their corresponding bid-per-instance value, that would be performed with polynomial-time complexity.
   - *Case 2* When each cloud server is having limited amount of resources: This case is just the same as the case 2 of the first sub-problem, except that we require to allocate the VMs to the cloud servers while considering the servers' energy consumption which is equivalent to assigning η VM instances to the κ cloud servers in such a way that the overall value of $\sum_{j=1}^{\kappa} \oint$ is minimized satisfying the allocation constraint and resource constraint. Hence, this problem is equivalent to the decision version of the Bin-Packing problem, which is also a NP-Hard problem. Afterward, we can compute the final payment buyers' and sellers' denoted as χ_i and χ_j by applying the Algorithm 3, which can be executed with polynomial-time complexity.

Since both these sub-problems are of the form of Bin-Packing problem, we can infer that proposed online VM allocation problem can be reduced in polynomial time to a Bin-Packing problem. Since it is known that the Bin-Packing is a NP-Hard problem, the online VM allocation problem is also NP-hard.

□

3.6 Double auction properties

Designing an efficient double auction technique for allocating cloud resources of CSP to the cloud service users is expected to satisfy four desired properties as follows:

1. *Truthfulness* A truthful approach implies incentive compatibility, which means that the bid value of each buyer and the ask value of each seller must be equal to the actual true valuation and true cost of the resource, respectively. A property where the dominant strategy is bidding one's true valuation, is called strategy proofs. A mechanism is truthful if a buyer or seller obtains the largest utility in a trade if and only if it truthfully represents its ask or bid information i.e.

$$\mathbb{U}_i^B \geq \mathbb{U}_i^{\hat{B}}, \mathbb{U}_i^S \geq \mathbb{U}_i^{\hat{S}}, \forall \hat{B}, \hat{S} \in \Omega, \tag{14}$$

where Ω represents arbitrary information about an ask or a bid. *Remark:* We assume that the units (buyer/

seller) cannot manipulate other task information, and truthful working only the bids and asks are considered. The unit (buyer/seller) cannot gain extra profit by reporting false information.

2. *Individual Rationality:* An auction design is individually rational if it does not incur a loss on joining the auction. This means a unit winning in the auction should have a non-negative utility and should gain profit, i.e.

$$\begin{aligned}&\mathbb{U}_i^B \geq 0, \mathbb{U}_i^S \geq 0, \chi_i \leq \beta_i\\&\chi_j \geq \alpha_i, \forall B_i \in \mathbb{B}, S_j \in \mathbb{S},\end{aligned} \tag{15}$$

3. *Budget-Balance* An auction mechanism follows the budget-balance property if, for a matched pair of a buyer and seller, the auctioneer charges buyer no less than what it pays to the seller. It deals with the difference in payment between buyer and seller i.e., the utility of the platform should be non-negative.

$$\mathbb{U}_i^A \geq 0 \tag{16}$$

It can be divided into two types:

(a) Strong Budget-Balance - The auctioneer should not lose or gain money. Transactions should strictly happen between buyer and seller i.e.

$$\sum_{i=1}^{\eta} \chi_i = \sum_{j=1}^{\eta} \chi_j \tag{17}$$

(b) Weak Budget-Balance - The auctioneer must not pay for the transaction to be successful, in other words it can gain profit but not suffer any loss i.e.

$$\sum_{i=1}^{\eta} \chi_i \geq \sum_{j=1}^{\eta} \chi_j \tag{18}$$

4. *Economic efficiency* Economic efficiency implies that the winner of the auction should be the one who deserves or values the auction most after the trades have been completed.

However, the Myerson–Satterthwaite (MS) impossibility theorem states that, it is always impossible to accomplish all four of these desirable auction properties simultaneously.

4 Proposed approach

The detailed algorithm for proposed auction mechanism (OTDAM) is given by Algorithm 1. OTDAM works in two modules: winner-determination and matching and VCG based payment calculation. OTDAM calls Winner_Determination(S, B, η, κ, T) to obtain the set of winner candidates (as shown in Algorithm 2). After winner determination stage, OTDAM performs final payment calculation using VCG_Payment_Calculation $(A, \mathbb{B}, \mathbb{S}, \Theta, \Omega)$ (as shown in Algorithm 3). We next describe the winner-determination and the VCG based payment calculation algorithms in detail.

Algorithm 1 OTDAM

```
Input: Cloud servers set $S$, Cloud users set $B$, Bid vector $\beta$, Ask vector $\alpha$, Set of time slots $T$, Resource requirement $\Upsilon^m$
Output: Winning buyers set $\mathbb{B}$, Winning sellers set $\mathbb{S}$, $\chi_w^b$, $\chi_w^s$
1: $Auction\_Setup(B, S, \beta, \alpha, T, \Upsilon^m)$
2: $Winner\_Determination(S, B, \eta, \kappa, T)$
3: $VCG\_Payment\_Calculation(A, \mathbb{B}, \mathbb{S}, \Theta, \Omega)$
```

4.1 Winner determination and matching

To perform online double auction, first we form list **B** and **S** of the bids and asks of the buyers and sellers respectively, we get η and κ as :

$$\eta = |B|, \kappa = |S| \tag{19}$$

Then we set the initial values of the parameters for buyer and seller. With each time slot, we check the condition of the buyer and seller availability for a particular time slot. We sort all the bids per resource of buyers in non-increasing sequence, so that buyer having the highest bid goes first to satisfy the resource request. A function σ is a 'seller desirability' or usability factor which denotes the ratio of the *ask-per-resource*(α_j) to the product of *fully-utilized-energy-consumption* (ϵ_j^a), when server is fully utilized, and *response-time of the seller in milliseconds*.

$$\sigma = \frac{\alpha_j}{\epsilon_j^a * \tau_j} \tag{20}$$

where τ_j is the response time. Therefore we have to sort all the sellers according to their σ values in non-increasing order. So that we can allocate maximum number of possible VM instances to the cloud server having least active energy consumption value, and the cloud server with highest ask-per-resource considering the VM instances with the highest bid value are granted first. Thus we iterate over both the lists within a particular time slot to assign maximum possible VM instances to the cloud server.

Algorithm 2 Winner_Determination(S, B, η, κ, T)

```
    Input: Sellers set $S$, Buyers set $B$, Set of Time Slots $T$ = {0,1,....,t}
    Output: $\mathbb{B}$ and $\mathbb{S}$
    Phase 1: Initialization
 1: Create a set $H \leftarrow \rho$ of vertices corresponding to $B$
 2: Create a set $A \leftarrow \rho$ of vertices corresponding to $S$
 3: Input Buyer and Seller resources , $\Gamma_i$ resource requirement of Buyer ,$\psi_j$ resource availability of Seller
 4: Set $\kappa$ = number of sellers and $\eta$ = number of buyers
 5: for all buyer $B_i$ in input do
 6:     Initialize $\mu_i \leftarrow null, \chi_i \leftarrow 0$
 7:     Add $H \leftarrow B_i$
 8: for all seller $S_j$ in the input list do
 9:     Initialize $P_j \leftarrow null, \phi_j \leftarrow \psi_j, \chi_j \leftarrow 0$
10:     Add $A \leftarrow S_j$
11: Set Bounded Patience time $k_b$ and $k_s$ for buyers and sellers respectively
12: Set $\lambda \leftarrow$ 0
13: Set $t \leftarrow$ 0
14: for all buyers and sellers in input do
15:     set $(a_i, d_i)$ and $(a_j, d_j)$     ▷ Setting arrival and departure time of buyer and seller respectively
16: while $t \leq T$ do
17:     for all buyer in input do
18:         if $a_i \geq t \geq d_i$ then
19:             $T_b \leftarrow B_i$
20:     for all seller in input do
21:         if $a_j \geq t \geq d_j$ then
22:             $T_s \leftarrow S_j$
23:     Sort all buyers in $T_c$ to get an ordered list of $\Theta = [B_1, B_2, \dots, B_\eta]$
24:     Such that $\beta_1 \geq \beta_2 \geq \dots \geq \beta_\eta$
25:     Sort all sellers in $T_s$ to get an ordered list of $\Omega = [S_1, S_2, \dots, S_\kappa]$
26:     Such that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_\eta$     ▷ Where $\sigma$ is the Seller Desirability or User sell ability
    Phase 2: Resource Provisioning
27:     for all buyer $B_i \in \Theta$ do
28:         while $\lambda \leq$ length of $\Omega$ and $B_i$ is not in $P_j$ do
29:             Set $\mathbb{W} \leftarrow \Omega$ of $\lambda$
30:             if $\alpha_{\mathbb{W}} \leq \beta_{B_i}$ then then
31:                 if resource requirement of $\beta_i$ i.e. $\Gamma_i \leq$
32: resource availability at seller i.e. $\psi_{\mathbb{W}}$ then
33:                     $\mu_i \leftarrow \mathbb{W}$
34:                     Add $B_i$ to the client of $\mathbb{W}$ i.e.$P_{\mathbb{W}} \leftarrow B_i$
    Phase 3: Updation
35:                     Set $\phi_{\mathbb{W}} \leftarrow \phi_{\mathbb{W}} - \Gamma_i$     ▷ Decrease availability of resources at the particular seller
36:                     Add $B_i$ and $S_{\mathbb{W}}$ to $\mathbb{B}$ and $\mathbb{S}$ their winning set respectively
37:
            else Increment $\lambda$
38:     Increment $t$
39: return $\mathbb{B}, \mathbb{S}$
```

We check for the availability of resources at the server and then allocate keeping in priority that the highest ask-per resource of the seller must be less than or equal to the buyer's having lowest bid value. With each matching pair, we add the corresponding winner buyer and seller to their winning set $\mathbb{B}$ and $\mathbb{S}$, respectively.

4.2 VCG based pricing scheme

Vickrey–Clarke–Groves (VCG) pricing scheme is a generic truthful approach for achieving a socially-optimal solution. It is one of the most commonly used scheme for payment determination. Our payment schedule does not follow VCG completely, as VCG lacks in providing budget-balance over individual rationality. Next, we iterate the list of cloud users and find the final payment value for each user in the same time slot before it increments. Payment value is a product of resource requirement of the minimum value required to claim that cloud server, whose value is equal to the greater of the two i.e., ask-per-resource and the second-highest bid.

Algorithm 3 VCG_Payment_Calculation($A, \mathbb{B}, \mathbb{S}, \Theta, \Omega$)

```
    Input: A, $\mathbb{B}, \mathbb{S}, \Theta, \Omega$
    Output: $\chi_w^b, \chi_w^s$
 1: while $t \leftarrow 0$ do     ▷ Processing the Winner determination algorithm and then executing the Payment calculation for particular time period
 2:     Sort all the sellers in $\boldsymbol{A}$ to get an ordered list $\boldsymbol{A}^\alpha = [S_1, S_2, \dots, S_\kappa]$ such that $\alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_\kappa$
 3:     for all seller $S_j \in \mathbb{N}$ do
 4:         for all buyer $B_i \in$ users list of $S_j$ do
    Phase 1: Buyer's Payment
 5:             if $B_i$ is not the last buyer in the list of $\Theta$ then
 6:                 Set $\chi_i \leftarrow \Upsilon_i * max(\alpha_j, \beta_{\delta+1})$ where $\vartheta$ is the position of $B_i$ in $\Theta$
 7:             else
 8:                 Set $\chi_i \leftarrow \Upsilon_i * \beta_i$
 9:             Set $\chi_w^b \leftarrow \chi_w^b \cup \chi_i$
    Phase 2: Revision of Seller's Payment
10:             if $S_j$ is not the last seller in the list of $\Gamma^\alpha$ then
11:                 Set $\chi_j \leftarrow \chi_j + (\Upsilon_i * min(\beta_i, \alpha_{\varepsilon+1}))$ where $\varepsilon$ denotes the $S_j$'s position in $\Gamma^\alpha$
12:             else
13:                 Set $\chi_j \leftarrow \chi_j + \chi_i$
14:             Set $\chi_w^s \leftarrow \chi_w^s \cup \chi_j$
15:     return $\chi_w^b, \chi_w^s$
```

Considering ϑ as the current buyer's position in the Θ ordering list and the particular buyer placed at cloud server j.

$$\chi_i = \begin{Bmatrix} \Upsilon_i \alpha_j, & \text{if } \alpha_j \geq \beta_{\vartheta+1} \\ \Upsilon_i \beta_{\vartheta+1}, & \text{otherwise} \end{Bmatrix} \tag{21}$$

We calculate payment the server will receive from buyer B_i, by computing minimum value from all the buyer's payment ensuring that the designated buyer is hosted at next economical cloud server. Taking ε as the current seller selected from the ordered list A^α, we obtain

$$\chi_j^i = \begin{Bmatrix} \Upsilon_i \beta_j, & \text{if } \beta_i \leq \alpha_{\varepsilon+1} \\ \Upsilon_i \alpha_{\varepsilon+1}, & \text{otherwise} \end{Bmatrix} \tag{22}$$

The final payment to S_j is the sum of all payments to S_j received from all its buyers.

$$\chi_j = \sum_{i=1}^{|P|} \chi_j^i \tag{23}$$

4.3 Illustrative example

We consider a random input of user requests and servers for a randomly generalized distribution of parameters assuming a valid time window and payment schedule. Table 4 shows the bid matrix that contains bids and resource requirements of 10 buyers. The ask vector for sellers is shown in Table 5. It contains ask, resource availability, active and idle energy consumption of 3 servers. The values of bids, asks, resource requirements, total resource availability, energy consumption, and response time have been generated between [2.0, 12.0], [1.0, 10.0], [3, 10], [10, 30], [150.0, 350.0], and [0.5, 2.0] respectively. Next, we illustrate the OTDAM steps to derive the auction outcome as follows:

- **Step 1** First, we form the ordered list Θ of buyers, where all the buyers are sorted based on their per-resource bids in non-increasing order as displayed in Table 6.
- **Step 2** Next, the sellers are sorted in the non-increasing sequence of their seller desirability values σ, as shown in Table 7.
- **Step 3** Iterate the ordered list of buyers and sellers and allocate the VMs to the servers efficiently (Table 8).
- **Step 4** Create the ordered list A^{α} for payment calculation (Table 9)
- **Step 5** Calculate the payment for the buyers based on the resource demand and the minimum value required to claim that cloud server (Table 10).
- **Step 6** Calculate the payment for the sellers (Table 11).
- **Step 7** Now, we examine the individual rationality and truthfulness for the given example.

  1. Buyers' Individual Rationality: As shown in Fig. 4, the actual payment a buyer has to give is never greater than the expected payment as per its bid.
  2. Sellers' Individual Rationality: As represented in Fig. 5, the seller's actual payment is never lesser than the expected payment as per its ask.
  3. Buyers' Truthfulness: We maintain the auction's truthfulness by ensuring that the buyer's utility is maximum only when its bid equals the truthful cost. To analyze truthfulness, we select a random buyer, say buyer 8 and analyze its utility as it places bids different from its true value. Figure 6 shows that the utility increases when you get close to the truthful cost, is maximum at the truthful cost, and decreases from there. This is just an instance involving buyer 8 and the graph will not be the exact same for other buyers but will follow the same pattern due to the way the algorithm is designed (as shown in lines 23 and 24 of Algorithm 2).

- **Step 8** The allocation done by OTDAM for a random input can be illustrated by a bipartite graph, as shown in Fig. 7. The vertices in the first row show VMs and the vertices in the second-row show servers. The randomly generated bid and resource requirement values are indicated above the buyer vertices, and the ask and total resource availability are indicated below the seller vertices. The numbers along the edge show the payment receiving from the buyers set and the payment obtained by the sellers set. Since we show this graph only for illustration of allocation, energy consumption values have not been stated. However, they are taken into account as displayed by the seller with ask value of 9 coming before the seller with ask value of 13.

4.4 Energy approximation

Definition 37 For any algorithm $A(B, S)$, the asymptotic approximation ratio (∇) can be expressed as follows:

$$\nabla_A = \frac{A(B,S)}{OPT(B,S)} \tag{24}$$

The total energy consumption can be calculated as:

Table 4 Bid matrix of 10 buyers

i	β	Υ	a	d
B				
1	7.48	5	2	3
2	11.14	5	0	2
3	2.15	4	2	5
4	8.75	8	1	4
5	6.05	4	2	5
6	2.45	8	3	5
7	3.41	4	4	7
8	5.49	9	5	7
9	6.04	6	3	6
10	4.47	5	6	8

Table 5 Ask vector of 3 sellers

j	α	ψ	ϵ^a	ϵ^{id}	τ	a	d
S							
1	9.89	24	239.51	205.51	0.7	0	3
2	4.15	25	274.27	186.64	1.6	2	8
3	7.74	12	299.38	232.25	1.1	1	4

Table 6 Build ordered list of buyers

i	β	Υ	a	d
Θ				
2	11.14	5	0	2
4	8.75	8	1	4
1	7.48	5	2	3
5	6.05	4	2	5
9	6.04	6	3	6
8	5.49	9	5	7
10	4.47	5	6	8
7	3.41	4	4	7
6	2.45	8	3	5
3	2.15	4	2	5

Table 7 Ordered list of sellers based on their seller desirability values σ

j	α	ψ	ϵ^a	ϵ^{id}	τ	σ	a	d
Ω								
1	9.89	24	239.51	205.51	0.7	0.0589	0	3
3	7.74	12	299.38	232.25	1.1	0.0235	1	4
2	4.15	25	274.27	186.64	1.6	0.0094	2	8

Table 8 Allocation of VMs to the servers

j	$\sum$	ψ	ϕ
S			
1	[2]	24	19
2	[1, 5, 9, 8]	25	1
3	[4]	12	4

$$\oint_{total} = \sum_{j=1}^{\kappa} (\epsilon_j^a \Pi_j(t) + \epsilon_j^{id}(1 - \Pi_j(t))) \tag{25}$$

For an ideal condition, an energy-optimal algorithm (EOA) does not consider the cloud server's asking price, and try to assign VM instances in such a way that it consumes minimum energy. Hence, the overall energy consumption in this scenario can be written as:

Table 9 Build A^α for payment calculation

j	$\sum$	ψ	ϵ^a	ϵ^{id}	τ
A^α					
2	[1, 5, 9, 8]	25	274.27	186.64	1.6
3	[4]	12	299.38	232.25	1.1
1	[2]	24	239.51	205.51	0.7

Table 10 Calculate buyers payment

i	β	Υ	μ	χ
B				
1	7.48	5	2	30.26
2	11.14	5	1	49.46
3	2.15	4	–	0.0
4	8.75	8	3	61.91
5	6.05	4	2	24.16
6	2.45	8	–	0.0
7	3.41	4	–	0.0
8	5.49	9	2	40.20
9	6.04	6	2	32.97
10	4.47	5	–	0.0

Table 11 Calculate sellers payment

j	α	ψ	ϵ^a	ϵ^{id}	τ	$\sum$	ϕ	χ
S								
1	9.89	24	239.51	205.51	0.7	[2]	19	49.46
2	4.15	25	274.27	186.64	1.6	[1, 5, 9, 8]	1	147.30
3	7.74	12	299.38	232.25	1.1	[4]	4	70.02

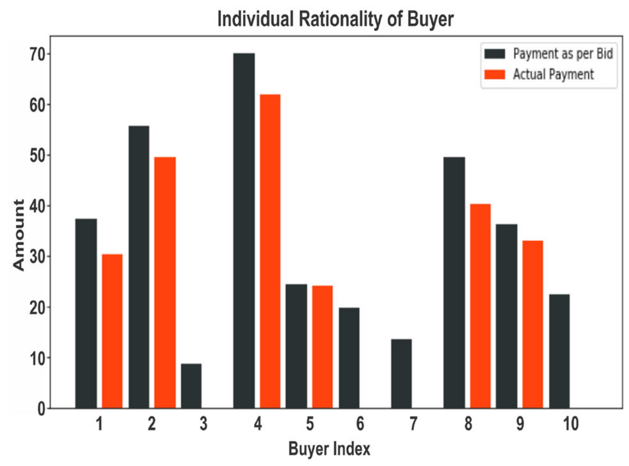

Fig. 4 Buyers' individual rationality

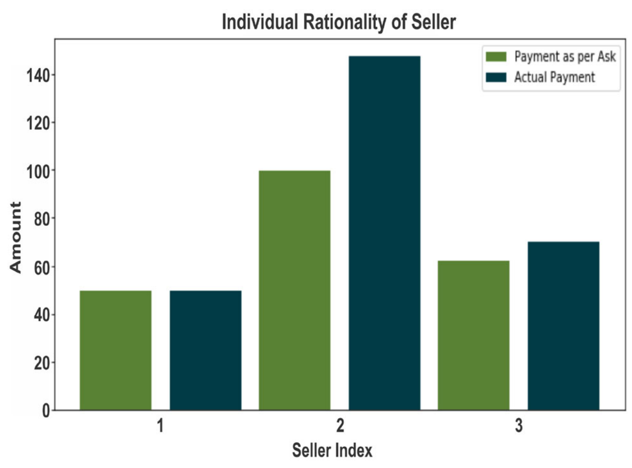

Fig. 5 Sellers' individual rationality

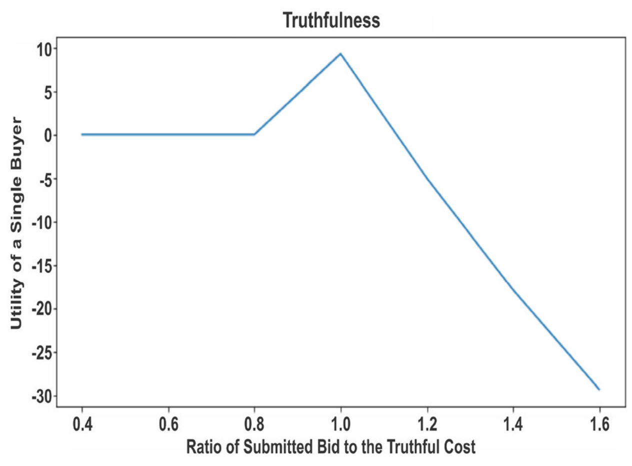

Fig. 6 Buyer's truthfulness

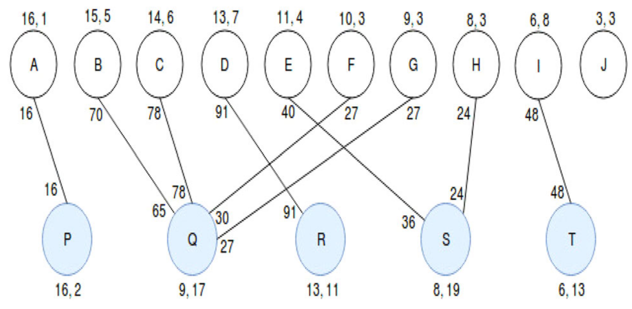

Fig. 7 Bipartite graph representation of final VM allocation

$$\oint_{OPT} = \sum_{j=1}^{\kappa_a} \epsilon_j^a + \sum_{j=\kappa_a+1}^{\kappa} \epsilon_j^{id} \tag{26}$$

where κ_a represents the total active cloud servers. Here, we suppose that the cloud servers are positioned in an increasing sequence of their corresponding active energy consumption value and also in the decreasing sequence of their corresponding idle energy consumption value. Hence the $\oint_{OPT}$ can be expressed mathematically as follows:

$$\oint_{OPT} = \oint^a + \oint^{id} \tag{27}$$

where $\oint^{id}$ denotes the minimum idle energy consumption, and $\oint^a$ represents the minimum active energy consumption.

Note that $\oint^a$ and $\oint^{id}$ are ideal values which more often are not achieved in the real world scenario. So, our algorithm will yield an energy consumption $\oint_{OTDAM}$ which will be calculated by taking the server utilization values ψ, into consideration.

Theorem 2 *The maximum energy consumed by OTDAM, when $\Pi_j(t) = 1$, $\forall 1 \le j \le \kappa$ is calculated by*:

$$\oint^a + \oint^{id} + \sum_{j=\kappa_a+1}^{\kappa} \oint_j$$

Proof From Equation 10, it is known that

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} (\epsilon_j^a \Pi_j(t) + \epsilon_j^{id}(1 - \Pi_j(t))) \tag{28}$$

By expanding, we get

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} (\epsilon_j^a \Pi_j(t) + \epsilon_j^{id} - \epsilon_j^{id} \Pi_j(t)) \tag{29}$$

Next, we characterized a new function '$\oint$', for the *Difference Value of Energy Consumption*, whose value is always positive. Since it is evident that the active energy consumption of a cloud server will always be higher than the value of its idle energy consumption.

$$\oint = \epsilon^a - \epsilon^{id} \tag{30}$$

By substituting Eq. 30 in the Eq. 29, we obtain,

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} \Pi_j(t) \oint_j + \sum_{j=1}^{\kappa} \epsilon_j^{id} \tag{31}$$

Here, we consider the κ_a denotes the active servers in the ideal condition. Thus, it is evident that, $1 \le \kappa_a \le \kappa$. We can note that it can not be implied that only κ_a cloud servers would be in active mode in the practical run-time of our approach. Using κ_a in the Equation 31,

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} \Pi_j(t) \oint_j + \sum_{j=1}^{\kappa_a} \epsilon_j^{id} + \sum_{j=\kappa_a+1}^{\kappa} \epsilon_j^{id} \tag{32}$$

Let's introduce a new term $\sum_{j=1}^{\kappa_a} \epsilon_j^a$ for Equation 32,

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} \Pi_j(t) \oint_j + \sum_{j=1}^{\kappa_a} \epsilon_j^{id} - \sum_{j=1}^{\kappa_a} \epsilon_j^{a} + \sum_{j=\kappa_a+1}^{\kappa} \epsilon_j^{id} + \sum_{j=1}^{\kappa_a} \epsilon_j^{a} \tag{33}$$

By applying Eq. 30 and Eq. 26 in the Eq. 33, we obtain:

$$\oint_{OTDAM} = \sum_{j=1}^{\kappa} \Pi_j(t) \oint_j - \sum_{j=1}^{\kappa_a} \oint_j + \oint_{OPT} \tag{34}$$

As the ideal and active energy consumption costs of all the cloud servers are already known and constant. Hence, the final value of $\oint_{OTDAM}$ relies primarily on the cost of $\Pi_j(t)\ \forall\ 1 \leq j \leq \kappa$. After all, $\Pi_j(t)$ represents the utilization of cloud server, it is known that $\forall\ 1 \leq j \leq \kappa$, $0 \leq \Pi_j(t) \leq 1$. Thus by applying the Eq. 27,

$$\oint_{OTDAM} \leq \sum_{j=\kappa_a+1}^{\kappa} \oint_j + \oint^{a} + \oint^{id} \tag{35}$$

□

Corollary 1 *For the proposed algorithm, the maximum value of the asymptotic approximation ratio (∇_{OTDAM}) is:*

$$\frac{\oint^a + \oint^{id} + \sum_{j=\kappa_a+1}^{\kappa} \oint_j}{\oint^a + \oint^{id}}$$

Proof By applying Eq. 24 and Theorem 2, we can find:

$$\nabla_A = \frac{\oint_{OTDAM}}{\oint_{OPT}} \leq 1 + \frac{\sum_{j=\kappa_a+1}^{\kappa} \oint_j}{\oint^a + \oint^{id}} \tag{36}$$

□

4.5 Time complexity analysis

Theorem 3 *The time complexity of Algorithm 2 is $\mathcal{O}(\kappa \log \kappa + \eta \log \eta + 5\eta\kappa + 2\kappa + 4\eta)$*

Proof The second algorithm is about determining the winning bids and allocating the VM instances to the cloud servers. We consider the mergesort algorithm for sorting, whose complexity is $\mathcal{O}(\eta \log \eta)$. We have two allocation steps in the algorithm with complexities of 2η and 2κ, respectively. We then have two sorting steps in the algorithm with time complexities of $\mathcal{O}(\eta \log \eta)$ and $\mathcal{O}(\kappa \log \kappa)$ separately. The for-loop in the line no. 27 contains a while-loop and another for-loop within that while-loop. We have two candidate worst cases here - first, where the value of $\lambda = \kappa$, thus making the contents of the for-loop to iterate only once and giving a time complexity of $\mathcal{O}(\kappa + 5)$ and second, where the value of $\lambda = 1$, thus making the contents of the for-loop to iterate maximum κ times and giving a time complexity of $5\kappa + 2$. We'll consider the second case as our worst-case as its time complexity is higher. So, the time complexity of the for-loop is $\mathcal{O}(\eta(5\kappa + 2))$ and thus, the total time complexity of Algorithm 2 is $\mathcal{O}(\kappa \log \kappa + \eta \log \eta + 5\eta\kappa + 2\kappa + 4\eta)$.

□

Theorem 4 *The time complexity of Algorithm 3 is $\mathcal{O}(\eta(\log \eta + \log \kappa) + \kappa \log \kappa)$*

Proof The third algorithm is about calculating the payment to be given by each buyer and to be received by each seller. We assume the searching algorithm to be binary search, whose complexity is $\mathcal{O}(\log n)$. We have a sorting step in the algorithm with a time complexity of $\mathcal{O}(\kappa \log \kappa)$. The for-loop in the third line contains another for-loop. The worst case is where the VMs are allocated such that every server becomes a winning seller, and every buyer becomes a winning buyer, thus making the for-loop in the third line iterate κ times, and the for-loop in the fourth line iterate a total of η times. The complexity of binary search (in sixth and eleventh statements) is $\mathcal{O}(\log n)$. So, the total complexity of the statements within the loops is $\mathcal{O}(\log \eta + \log \kappa)$, and the time complexity of whole algorithm is $\mathcal{O}(\eta(\log \eta + \log \kappa) + \kappa \log \kappa)$. □

Theorem 5 *The time complexity of Algorithm 1 is $\mathcal{O}(2\eta \log \eta + \eta \log \kappa + 5\eta\kappa)$.*

Proof Algorithm 1 time complexity can be obtained as the sum of complexities of Algorithm 2 and Algorithm 3. Considering that $\eta > > \kappa$, from Theorem 3 and Theorem 4, we can calculate the overall time complexity of Algorithm 1 and can be written as:

$$\begin{aligned} &\mathcal{O}[(2\eta + 2\kappa) + (\kappa \log \kappa + \eta \log \eta + 5\eta\kappa + 2\eta) \\ &\quad + (\eta \log \eta + \kappa \log \kappa + \eta \log \kappa)] \\ &\quad \approx \mathcal{O}[2\eta \log \eta + \eta \log \kappa + 5\eta\kappa] \end{aligned} \tag{37}$$

□

4.6 Properties of OTDAM

Theorem 6 *OTDAM follows individual rationality.*

Proof Any auction design is individually rational if and only if the buyer's payment is less than or equal to its bidding value, and the seller also receives greater than or equal to its asking value.

As, it is known that $\forall\ B_i$ in B, there can be two cases:

1. B_i is not a winning buyer. Thus, it is not necessary to calculate its rationality.
2. B_i is a winning buyer. We have two more situations as described in Algorithm 3 -
   (a) B_i is not the last buyer of Θ. It is known that $\alpha_j \leq \beta_i$, and it is also a major criteria for B_i to be

declared as a winning bid. Also, it is known that $\beta_{\delta+1} \leq \beta_i$ since the buyers in Θ are positioned in non-increasing sequence of their bid-per-instance value. Hence, $\chi_i \leq \Upsilon_i \beta_i$.

(b) B_i is the last buyer of Θ. $\chi_i = \Upsilon_i \beta_i$.

As, we know that $\forall$ S_j in S, there can be two cases:

1. S_j is not a winning seller. Hence, it is not necessary to calculate its rationality.
2. S_j is a winning seller. Thus, we have two more cases as described in Algorithm 3 -

   (a) S_j is not the last seller of A^{α}. As, it is known that $\beta_i \geq \alpha_j$, and it is also a major criteria for B_i to be declared as a winning bid. Also, it is known that $\alpha_{\omega+1} \geq \alpha_j$ since the sellers in Ω are positioned in non-decreasing order based on their ask-per-instance value. Thus, $\chi_j \geq (\psi_j - \phi_j)\alpha_j$.

   (b) S_j is the last buyer of Ω. $\chi_j = (\psi_j - \phi_j)\alpha_j$.

The Individual Rationality of OTDAM has also been illustrated in Figs. 4 and 5. □

Lemma 1 *OTDAM is always truthful for buyers.*

Proof Let β_i be the true valuation bid of the buyer B_i and β'_i be the false bid buyer B_i places in the greed of a better utility. Now, we have two possible cases

– Case 1 : $\beta'_i < \beta_i$ If the buyer places a bid that is less than its true valuation, it might jeopardize its chances of allocating resources at a more energy-efficient server. Also, in some cases, there is a chance that the buyer might not be allocated at all since the buyer won't know the asking prices as this is a sealed-bid auction. So, the buyer might suffer a loss if it places a bid less than its true valuation.

– Case 2 : $\beta'_i > \beta_i$ If the buyer places a bid whose value is greater than its true valuation, it might end up being allocated to a higher-ask seller, thus reducing its utility. Also, the payment mechanism is such that if the buyer's bid is significantly higher than its true valuation, it might have to pay an amount greater than its valuation, which is undesirable. So, a buyer stands a huge chance of suffering a loss if it places a bid higher than its true valuation.

So, in either of the two cases, the best policy for a buyer is to place a bid equal to its true valuation. □

Lemma 2 *OTDAM is always truthful for sellers.*

Proof Let α_j be the true cost ask of the buyer S_j and α'_j be the false ask seller S_j places in the greed of a better utility. Now, we have two possible cases

– Case 1 : $\alpha'_j < \alpha_j$ If the seller places an ask less than its true cost, it might end up jeopardizing its chances of winning the auction since the seller won't know the bidding prices as this is a sealed-bid auction. Also, the payment mechanism is such that if the seller's ask is significantly lower than its true cost, it might get paid less than its cost, which is undesirable. So, a seller stands a huge chance of suffering a loss if it places an ask lower than its true cost.

– Case 2 : $\alpha'_j > \alpha_j$ If the seller places an ask higher than its true cost, there is a chance that no buyer might be able to match its asking price, and the seller might lose the auction.

So, in either of the two cases, the best strategy for a seller is to place an ask equal to its true cost. □

Theorem 7 *OTDAM is truthful.*

Proof By applying Lemma 1 and Lemma 2 simultaneously prove that OTDAM is truthful. The truthfulness of OTDAM has also been illustrated in Fig. 6. □

Theorem 8 *OTDAM is also economically efficient.*

Proof OTDAM always prefers a buyer bidding high bid-per-instance. Hence the costly resources of sellers will always be given to those buyers who are willing to pay more. Thus, OTDAM is economically-efficient. □

5 Performance evaluation

We have conducted a set of simulation experiments to examine the efficacy of OTDAM while allocating resources in IaaS clouds. Subsection 5.1 illustrates the experimental environment and the benchmark. The evaluation methods are explained in Subsection 5.2. We also compare the performance of these approaches and analyze the experimental results in Subsection 5.3.

5.1 Experimental environment

To investigate the overall performance of proposed auction mechanisms, we have applied trace-driven simulations, utilizing Google cluster-usage data [46]. The dataset captures the activity of a production cluster comprising a total of 12,583 machines with a heterogeneous configuration for a period of 29 days. Machines are categorized into three different classes and ten distinct architectures with available CPU and memory capacity, as shown in Table 12. In the table, the memory and CPU units are linearly scaled in such a way that the maximum resource capacity is 1.

To simulate a cloud data center with servers of heterogeneous resource capacities, we have configured the hosts

Table 12 Configuration of machines in the Google cluster

Number of machines	Platform	CPUs	Memory
1	B	0.50	0.06
3	C	1.00	0.50
5	B	0.50	0.97
5	B	0.50	0.03
52	B	0.50	0.12
126	A	0.25	0.25
795	C	1.00	1.00
1001	B	0.50	0.75
3863	B	0.50	0.25
6732	B	0.50	0.50

based on the distribution of cloud server configurations reported in the Google cluster data, also shown in Table 13. The hardware specification and energy consumption values for the hosts were accumulated from the well-known SPECpower [47] benchmarks. Since the disk configurations of servers are not available in the Google dataset, we randomly set the capacity of disk storage for servers within [320, 800](in GB).

Resource tables are broadly classified into three major categories: Machines, Jobs and Tasks, and CPU Usage. Each class has one or more than one table containing multiple features. The dataset comprises approximately 672,074 incoming real-time jobs, and an individual user creates each job out of total 925 users. In Google cluster data, each job consists of multiple tasks that unite to approximately 24,281,242 different tasks. Each task is having three types of resource requirements, i.e., RAM, CPU, and disk. The usage of every type of resource is collected at five minutes intervals. It also provides the arrival time for each job and comprises task duration, i.e., start-time and finish-time of each task present in a particular job. The dataset illustrates that most of the tasks only run for short durations. There are also some of the long-running tasks that would yield some weeks to finish their execution. Our experiment comprises of six VM types of Amazon's instance classes, as shown in Table 14.

Table 13 Host characteristics for Google's cloud

Host characteristics			
Model of CPU	Xeon 2695	Xeon 2670	Xeon 2670
Speed(GHz)	2.3	2.5	2.6
ECUs	32.2	25	20.8
No. of cores	14	10	8
Memory (GiB)	768	768	384
P_{max} (W)	120	115	105
P_{idle} (W)	70	65	55

Each VM instances is randomly allocated a workload trace from the data set, ensuring the demands extracted from the cluster traces at time t should be less than the total capacity of VM. We set the start-time and finish-time of VM usage based on the corresponding tasks' latest time slot. The value of T is set to 3000 and each of duration 10 seconds. The assumption of other values for the following variables lie in range: $\beta \in [4.0,14.0)$, $\alpha \in [2.0,20.0)$, $\tau \in [3,10)$ (milliseconds). We have implemented the auction mechanism in the python programming language using an HPC cluster. The cluster contains one management server, and four compute servers. Table 15 shows the details, including hardware and software specifications. Each compute node is connected with the InfiniBand network (56 Gbps) and ethernet to the management server.

5.2 Evaluation methods

To analyze the performance of OTDAM, we compare its experimental results with three other existing double auction approaches: TASC [11], TDAM [42], and ICAM [43]. Each experiment is run over 100 times and averaged as the final result. The auction mechanisms are evaluated by using the following criterion: (i) Successful trades, (ii) Revenue, (iii) Energy-performance, (iv) Time-slots, (v) Volatility, and (vi) Running time. We have also discussed a theoretical comparison among auction mechanisms in Table 16.

5.3 Analysis of results

We first investigate the impact of different performance parameters and analyze the results. Afterward, we discuss the theoretical comparison of OTDAM results with three existing double auction-based mechanisms.

– *Number of successful trades* In Fig. 8, we examine the successful trades of OTDAM and compare its performance with the other three approaches. In this experiment, we have varied the buyers (users) from 0 to 14,000 and sellers (servers) from 0 to 7000 by increasing 2000 numbers of buyers and 1000 sellers respectively at each step. From Fig. 8, it can be observed that the plot for successful trades is a linearly increasing graph for all the four algorithms used for comparison. TASC wins over by giving maximum successful trades with an increase of servers and VMs, as illustrated in Fig. 8. OTDAM initially provides a good amount of successful trades in comparison with TDAM and ICAM. Still, as the number of VMs and servers increase respectively in a particular slot, it becomes difficult for servers to allocate more VMs within that specific period. For example, in Fig. 8, it is depicted that when the number of buyers is 14,000, the

Table 14 Types and characteristics of Amazon EC2 instances

Types and characteristics of Amazon EC2 instances					
Instance	Speed (GHz)	vCPUs	ECUs	Memory (GiB)	Storage (GB)
m3.medium	3.0	1	3	3.75	4
m1.medium	2.0	1	2	3.75	410
m1.small	1.0	1	1	1.7	160
t2.micro	1.0	1	1	0.613	1
t2.micro	1.0	1	1	1	1
t2.nano	1.0	1	1	0.5	1

Table 15 Hardware and software specification of HPC Cluster

HPC cluster details					
	Master node	Compute node 1	Compute node 2	Compute node 3	Compute node 4
Processor	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
RAM	256GB	128GB	128GB	128GB	128GB
Hard disk	4TB	1TB	1TB	1TB	1TB
CPU sockets	2	2	2	2	2
Cores	$2\times8 = 16$	$2\times8 = 16$	$2\times8 = 16$	$2\times8 = 16$	$2\times8 = 16$
OS	VMware ESXi 6.0	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04

Table 16 Comparison between TASC, TDAM, ICAM, and OTDAM

Property	TASC [11]	TDAM [42]	ICAM [43]	OTDAM
Auction type	McAfee	VCG	McAfee	VCG
Truthfulness for buyers	Not guaranteed	Guaranteed	Guaranteed	Guaranteed
Truthfulness for sellers	Guaranteed	Guaranteed	Guaranteed	Guaranteed
Individual rationality	Guaranteed	Guaranteed	Guaranteed	Guaranteed
Computational Complexity	Polynomial	Polynomial	Polynomial	Polynomial
Buyers utility	Lesser than TASC, TDAM, ICAM, and OTDAM	Lesser than OTDAM and higher than TASC, TDAM, ICAM	Greater than TASC but less than OTDAM	Greater than TASC, TDAM, ICAM

OTDAM realizes approximately 11,000 successful trades, and the TASC achieves around 15,500 successful trades. The interesting thing in our proposed auction approach is, it compares the bid and ask values at each period, and the winning seller obtains a final payment not less than its ask price, and the buyers are not charged more than its bid value.

– *Revenue maximization* In Fig. 9, we illustrate the revenue generated by OTDAM and compare its performance with the other approaches. To verify its performance, we pick a fixed number of buyers and sellers. From Fig. 9, it can be observed that among the four algorithms, the revenue generation is maximum for OTDAM after a certain point, i.e., 7000 buyers. The point where the revenue generated is increasing when buyers and sellers are in the range of thousands, and it takes time in minutes to complete the execution, as shown in Fig. 9. It is also observed that, the TASC results in worst performance than TDAM. The reason behind the maximum revenue generation is the seller

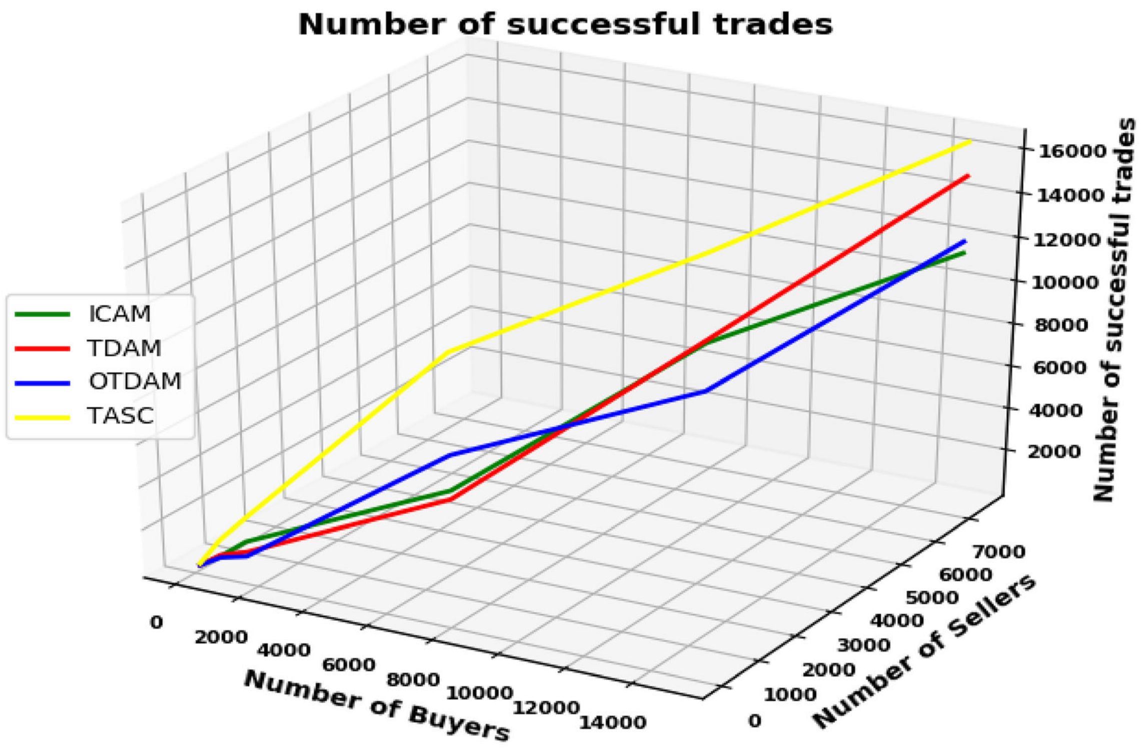

Fig. 8 Comparison of successful trades

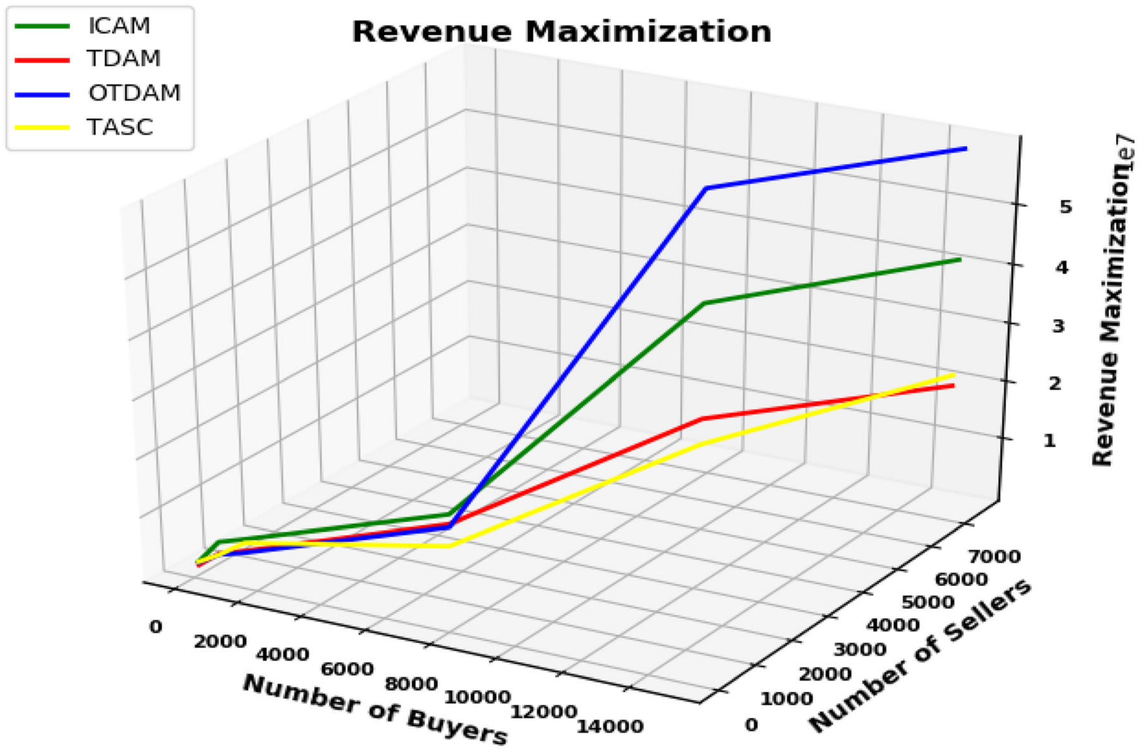

Fig. 9 Analysis of revenue maximization

desirability factor since the VM with the highest bid is allocated first to the servers with the highest ask-per-resource value. Thus, it maximizes the VM allocation by always placing the costly resources of the seller to a buyer having a high valuation, and being able to pay. As a result, it maximizes the overall revenue for a larger set of buyers and sellers, respectively.

– *Energy-performance analysis* Figure 10 compares the energy performance of the four approaches. For this experiment, we have varied the buyers from 200 to 1000 and sellers from 100 to 500 by increasing 200 numbers of buyers and 100 numbers of sellers respectively at each step. From Fig. 10, it can be identified that the energy consumption of the VMs is increasing linearly for the ICAM algorithm, which is the only one under-performing than OTDAM. But considering the general case, the total energy consumption of OTDAM is minimum for maximum successful trades. The reason behind the minimum energy consumption is the seller-desirability value, which always allocates the VM instances to the cloud server having the least active energy consumption.
– *Time slot analysis* Figure 11 examines the number of successful trades each algorithm experiences with the increase in time slots. For the time-slots versus successful trade analysis, we have varied the number of time-slots from 10 to 40 by increasing five numbers of time-slots respectively at each step. From Fig. 11, it can be observed that the OTDAM experiences a drop initially, but after 15-time slots, it maximizes the trades with an increase in time slots. It can also be noticed that when the time-slots are 35, the OTDAM achieves more than 1000 successful trades while the TDAM gives fewer successful trades.
– *Volatility analysis* Figure 12 investigates the impact of market volatility on the allocation efficiency by taking out the standard deviation of the valid price schedules. For this experiment, we have ranged from 10 to 35 time-slots and the number of trades from 0 to 1400 by increasing five numbers of time-slots and 200 numbers of trades respectively at each step. From Fig. 12, it can be observed that when the market is at low volatility trades are reasonably low. For example, when the trades are 200 at 10th time-slots, the volatility value is around 0.63 for OTDAM. As the volatility increases, the allocation efficiency becomes better. For example, at 23rd time-slots, OTDAM achieves approximately 1.5 volatility value, which is higher than the TASC approach, which produces only 1.0 volatility. However, the TDAM and ICAM approaches perform well and experience stable changes.

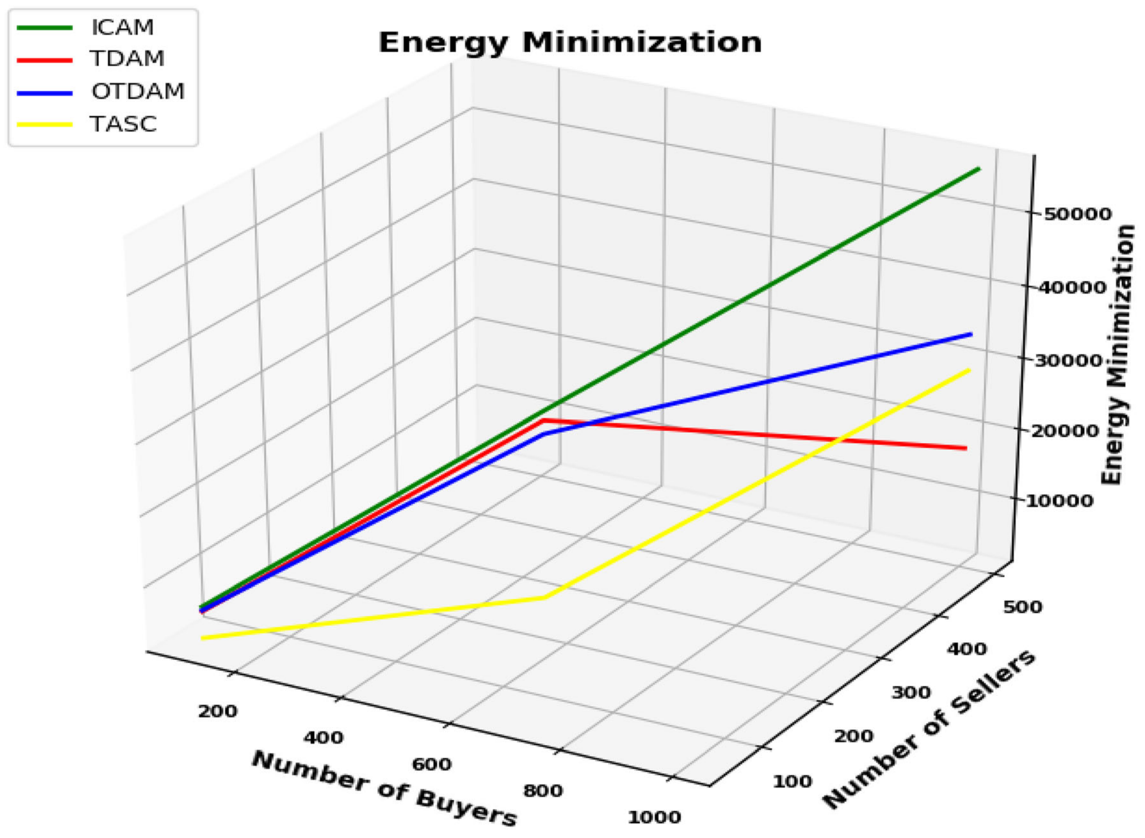

Fig. 10 Energy-performance analysis

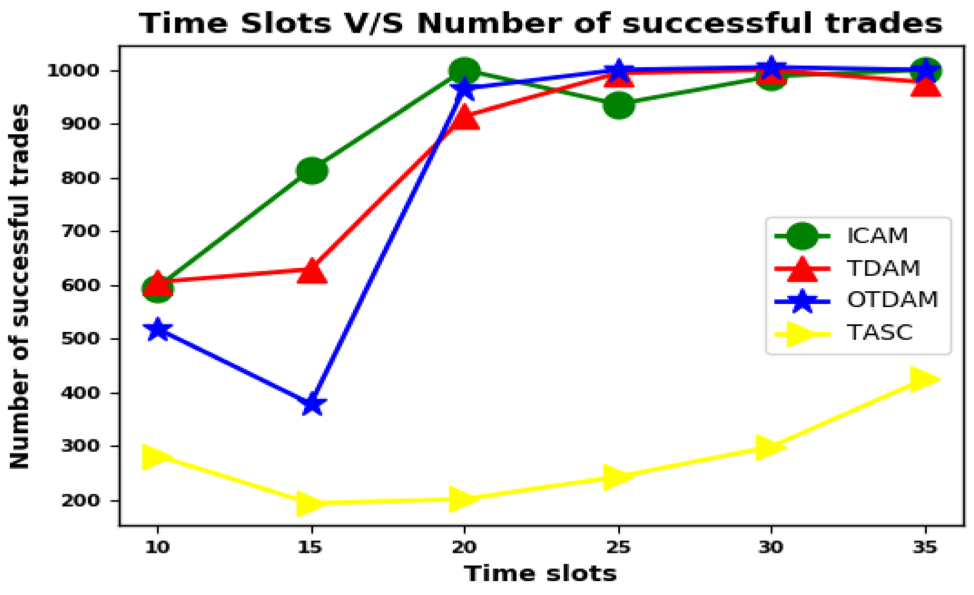

Fig. 11 Time slots versus Successful trades

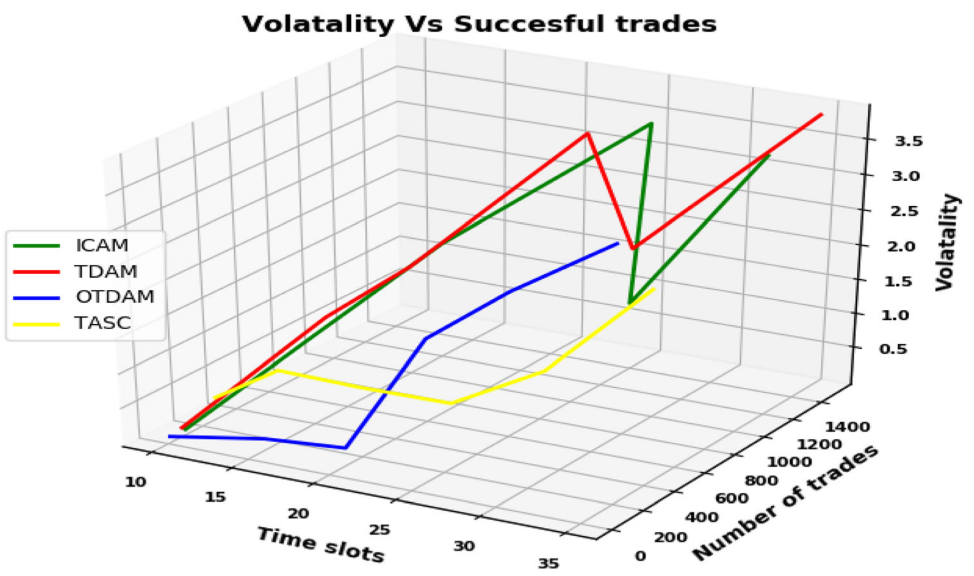

Fig. 12 Volatility analysis

– *Running-time analysis* Figure 13 compares the running time of all four algorithms. For the running-time analysis, we have varied the buyers from 0 to 14,000 and sellers from 0 to 7000 by increasing 2000 numbers of buyers and 1000 numbers of sellers respectively at each step. From Fig. 13, it can be identified that all the algorithms have a related functioning for a small amount of VM instances and servers. Still, as the number increases, the TASC mechanism takes more time to complete its allocation, i.e., taking more than 600 seconds to satisfy 14000 number of buyers. The OTDAM clearly shows its optimized technique saving upon running time even with a large set of of buyers and sellers, as depic in Fig. 13. For instance, when the buyers are 14,000, OTDAM takes less than 400 seconds to fulfill the request of VMs, which is much less than TASC and TDAM taking over 600 and 400 seconds, respectively. It can also be identified that the running-time of OTDAM increases with respect to the rise in the number of participating buyers and sellers separately. Thus, OTDAM is subjected to a polynomial computation time concerning the increasing number of buyers and sellers.

– *Theoretical comparison* In Table 16, we present a theoretical comparison between the four auction approaches: TASC, TDAM, ICAM, and OTDAM. The auction mechanisms are evaluated using the following important criterion: (i) Auction type, (ii) Truthfulness for buyers, (iii) Truthfulness for sellers, (iv) Individual rationality, (v) Buyers utility, and (vi)

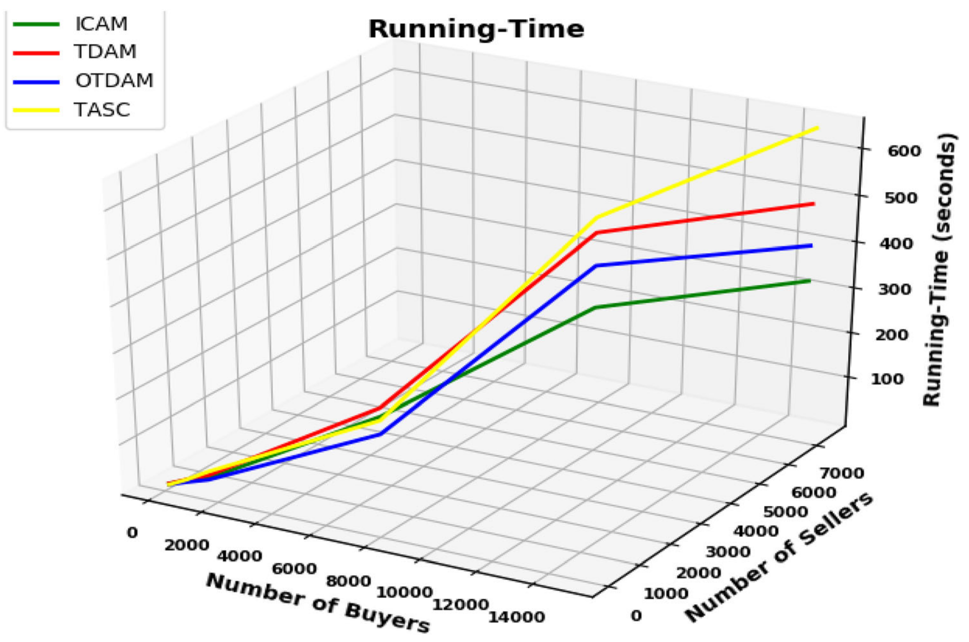

Fig. 13 Running time analysis

Computational complexity. In summary, OTDAM promises truthfulness of buyers and sellers considering the utility cannot be enhanced by bidding or asking untruthfully. It also guarantees individual rationality and higher utility values for buyers while taking polynomial computation time corresponding to the more significant number of buyers and sellers.

6 Conclusion

This paper studies the limitations of existing auction techniques for dynamic resource allocation in IaaS clouds. We have observed that the current auction techniques are mainly one-sided or offline without considering the dynamics of an elastic model in a time-variant environment. To address the issues of the current auction approaches of VM allocation, we have devised an online truthful double auction approach 'OTDAM' with the unique features of truthfulness, elasticity, heterogeneity, and multi-objective trade-offs for dynamic resource allocation in IaaS clouds. OTDAM has two main algorithms first one is weighted bipartite matching based winning bids determination, and the second is a VCG based algorithm for payment calculation. OTDAM can impressively distribute the cloud resources between the cloud users to meet their instance demands while ensuring the desirable auction features such as individual rationality, truthfulness, economic-efficiency, and polynomial-time computational complexity. Further, we have provided rigorous proofs on the OTDAM auction properties and validate the analysis with extensive trace-driven simulations showing improvements over performance, revenue, energy consumption, and running-time.

As a future study, we plan to investigate the emerging constraint challenges in distributed data centers, such as anti-affinity, scalability, network resource utilization, and availability etc., and apply these constraints to online resource provisioning algorithms to evaluate its performance in a distributed environment.

Acknowledgements The work of Yashwant Singh Patel is supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India MEITY-PHD-2525. This work was also partially supported by the Department of Science and Technology (DST), Government of India under ICPS Programme through the Project No. DST/ICPS/CPS-Individual/2018/403(G), "Low-cost Energy-Efficient Cloud for Cyber-Physical Disaster Management Systems".

References

1. Beloglazov, A., Buyya, R., Lee, Y.C., Zomaya, A.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. Adv. Comput. **82**, 47–111 (2011)
2. Azizi, S., Zandsalimi, M., Li, D.: An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Clust. Comput. **2020**, 1–15 (2020)
3. Rasouli, N., Razavi, R., Faragardi, H.R.: EPBLA: energy-efficient consolidation of virtual machines using learning automata in cloud data centers. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03066-6
4. Belgacem, A., Beghdad-Bey, K., Nacer, H., et al.: Efficient dynamic resource allocation method for cloud computing environment. Clust. Comput. **23**, 2871–2889 (2020)
5. Amoon, M., El-obely, T.E.: A Green energy-efficient scheduler for cloud data centers. Clust. Comput. **22**, 3247–3259 (2019)

6. Microsoft Azure: Pricing overview - how azure pricing works.https://azure.microsoft.com/en-us/pricing/. Accessed 15 Feb 2020
7. Amazon: EC2 instance pricing - Amazon web service (AWS). https://aws.amazon.com/ec2/pricing. Accessed 15 Feb 2020
8. Krishna, V.: Auction Theory, 2nd edn. Academic Press, Cambridge (2009)
9. K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang and M. Shen: Online combinatorial double auction for mobile cloud computing markets. In: Proceedings of 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC), pp. 1-8. Astin, TX (2014)
10. McAfee, R.P.: A dominant strategy double auction. J. Econ. Theory. **56**, 434–450 (1992)
11. Yang, D., Fang, X., Xue, G.: Truthful auction for cooperative communications. In: Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '11). Association for Computing Machinery, New York, NY, USA, Article 9. pp. 1–10 (2011)
12. X. Wang, X. Chen and W. Wu: Towards truthful auction mechanisms for task assignment in mobile device clouds. In: Proceedings of 2017 IEEE Conference on Computer Communications (INFOCOM), pp. 1-9. Alanta, GA (2017)
13. Li, H., Wu, C., Li, Z., Lau, F.C.M.: Virtual machine trading in a federation of clouds: individual profit and social welfare maximization. IEEE/ACM Trans. Netw. **24**(3), 1827–1840 (2016)
14. Vytelingum, P., Cliff, D., Jennings, N.R.: Strategic bidding in continuous double auctions. Artif. Intell. **172**(14), 1700–1729 (2008)
15. Kumar, D., Baranwal, G., Raza, Z., Vidyarthi, D.P.: A systematic study of double auction mechanisms in cloud computing. J. Syst. Softw. 234–255 (2017)
16. Kumar, D., Baranwal, G., Raza, Z., Vidyarthi, D.P.: A truthful combinatorial double auction-based marketplace mechanism for cloud computing. J. Syst. Softw. 91–108 (2018)
17. Farajian, N., Zamanifar, K.: Market-driven continuous double auction method for service allocation in cloud computing. In: Proceedings of the International Conference on Advances in Computing, Communication and Control, pp. 14–24. Springer (2013)
18. Vickrey, W.: Counter speculation, auctions, and competitive sealed tenders. J. Fin. **16**, 8–37 (1961)
19. Clarke, E.H.: Multipart pricing of public goods. Public Choice. **11**, 17–33 (1971)
20. Groves, T.: Incentives in teams. Econometrica. **41**(4), 617–631 (1973)
21. Parkes, D., Kalagnanam, J., Eso, M.: Achieving budget-balance with Vickrey-based payment schemes in exchanges. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1161–1168 (2001)
22. Huang, P., Scheller-Wolf, A., Sycara, K.: Design of a multi-unit double auction e-market. Comput. Intell. **18**, 596–617 (2002)
23. Babaioff, M., Walsh, W.E.: Incentive-compatible, budget-balanced, yet highlyvefficient auctions for supply chain formation. Decis. Support Syst. **39**, 123–149 (2005)
24. Chu, L.Y., Shen, Z.-J.M.: Agent competition double-auction mechanism. Manag. Sci. **52**, 1215–1222 (2006)
25. Chu, L.Y., Shen, Z.-J.M.: Truthful double auction mechanisms. Oper. Res. **56**, 102–120 (2008)
26. Chu, L.Y.: Truthful bundle/multiunit double auctions. Manag. Sci. **55**, 1184–1198 (2009)
27. Mishra, D., Garg, R.: Descending price multi-item auctions. J. Math. Econ. **42**(2), 161–179 (2006)
28. Demange, G., Gale, D., Sotomayor, M.: Multi-item auctions. J. Polit Econ. **94**(4), 863–872 (1986)
29. Ausubel, M., Binmore, K., Cramton, P., Ledyard, J., Milgrom, P., Morrill, T., Stacchetti, E.: An efficient dynamic auction for heterogeneous commodities. The American Economic Review, 602–629 (2006)
30. Plott, C.R., Gray, P.: The multiple unit double auction. J. Econ. Behav. Organ. **13**(2), 245–258 (1990)
31. Mashayekhy, L., Nejad, M.M., Grosu, D.: A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources. IEEE Trans. Parallel Distrib. Syst. **26**(9), 2386–2399 (2015)
32. Nejad, M.M., Mashayekhy, L., Grosu, D.: Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. IEEE Trans. Parallel Distrib. Syst. **26**(2), 594–603 (2015)
33. L. Mashayekhy, M. M. Nejad, D. Grosu and A. V. Vasilakos: Incentive-Compatible Online Mechanisms for Resource Provisioning and Allocation in Clouds. In: Proceedings of 2014 IEEE 7th International Conference on Cloud Computing, pp. 312-319. Achorage, AK (2014)
34. L. Zhang, Z. Li and C. Wu: Dynamic resource provisioning in cloud computing: A randomized auction approach. In: Proceedings of 2014 IEEE Conference on Computer Communications, pp. 433–41. Toranto, ON (2014)
35. Zhang, H., Jiang, H., Li, B., Liu, F., Vasilakos, A.V., Liu, J.: A framework for truthful online auctions in cloud computing with heterogeneous user demands. IEEE Trans. Comput. **65**(3), 805–818 (2016)
36. Wang, W., Jiang, Y., Wu, W.: Multiagent-based resource allocation for energy minimization in cloud computing systems. IEEE Trans. Syst. Man Cybern. Syst. **47**(2), 205–220 (2017)
37. S. Zaman and D. Grosu: Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds. In: Proceedings of 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 127-134. Indianapolis IN (2010)
38. Jin, L., Song, W., Wang, P., Niyato, D., Ju, P.: Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. IEEE Trans. Serv. Comput. **9**(6), 895–909 (2016)
39. Lu, L., Yu, J., Zhu, Y., Li, M.: A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets. IEEE Trans. Parallel Distrib. Syst.. **29**(4), 720–733 (2018)
40. Zhang, J., Yang, X., Xie, N., Zhang, X., Vasilakos, A.V., Li, W.: An online auction mechanism for time-varying multidimensional resource allocation in clouds. Future Gener. Comput. Syst. **111**, 27–38 (2020)
41. Middya, A.I., Ray, B., Roy, S.: Auction based resource allocation mechanism in federated cloud environment: TARA. IEEE Transactions on Services Computing (2019)
42. Patel, Y.S., Nighojkar, A., Misra, R.: Truthful double auction based vm allocation for revenue-energy trade-off in cloud data centers. In: Proceedings of the 2019 National Conference on Communications (NCC), Bangalore, India, pp. 1–6 (2019)
43. Jin, A., Song, W., Zhuang, W.: Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Transactions on Emerging Topics in Computing (2015)
44. Gupta, V.: Stochastic models and analysis for resource management in server farms. Ph.D. dissertation. Intel Corporation (2011)
45. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.: Energy efficient dynamic capacity provisioning in server farms. School of Computer Science, Carnegie Mellon University, Tech. Rep. CMU-CS-10-108 (2010)
46. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: Format + schema. Google Inc., Mountain View, CA, USA, Tech. Rep. ClusterData2011_2, Nov. 2011. http://code.google.com/p/googleclusterdata/wiki/Trace Version2. Accessed 15 Aug 2019

47. SPEC. https://www.spec.org/power_ssj2008/. Accessed 15 Aug 2019

Yashwant Singh Patel is currently pursuing his Ph.D. in Computer Science and Engineering at the Indian Institute of Technology Patna, India. His Ph.D. work is supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India <MEITY-PHD-2525>. He received B.E. degree in Computer Science and Engineering from Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India, in 2011 and M.Tech. degree in Computer Science and Engineering from KIIT University, Bhubaneswar, India, in 2014. His research interests include Cloud Computing, Edge Computing, Distributed Algorithms, and Grid Computing.

Zahra Malwi is currently working as a software developer in ValueLabs LLP Indore, India. She has a Bachelor's degree in Computer Science Engineering from Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India in 2019. She exhibits keen interest in areas including Machine Learning, Cloud Computing and Data Science . Zahra has been an active volunteer of the International Conference on Advance Computing Networking and Informatics 2018, hosted by Medicaps Institute of technology and Management Indore during Februrary 22-24 , 2018 and Women in Data Science (WiDs) hosted under the aegis of Women in Data Science, in association with Stanford University, 2019 Indore. She was a summer intern at the Indian Institute of Technology Patna in 2019 during which she successfully worked on the Dynamic Resource Allocation in Cloud Data Centers.

Animesh Nighojkar is currently pursuing his Ph.D. in Computer Science at the University of South Florida, Tampa, FL, USA. He received B.E in Computer Science and Engineering from Rajiv Gandhi Technical University Bhopal, India, in 2019. His research interests include Algorithms, Natural Language Processing, and Computational Linguistics. He is employed as a TA at the University of South Florida, USA. This work is a part of his research internship at the Indian Institute of Technology Patna, India.

Rajiv Misra is currently working as an Associate Professor in the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Patna, India. He received M.Tech. degree in Computer Science and Engineering from IIT Bombay and Ph.D. degree in the area of mobile computing from IIT Kharagpur. His research interests include Distributed Systems, Cloud Computing, Big Data Computing, Consensus in Blockchain, Cloud IoT Edge Computing, Adhoc Networks and Sensor Networks. He has contributed significantly to these areas and published more than 90 papers in high quality journals and conferences. His h-index is 13 with more than 800 citations. He has authored papers in IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems etc. He is a senior member of IEEE.