An Online Auction Mechanism for Dynamic Virtual Cluster Provisioning in Geo-Distributed Clouds

Weijie Shi, *IEEE Student Member,* Chuan Wu, *IEEE Senior Member,* and Zongpeng Li, *IEEE Senior Member*

Abstract—It is common for cloud users to require clusters of inter-connected virtual machines (VMs) in a geo-distributed IaaS cloud, to run their services. Compared to isolated VMs, key challenges on dynamic virtual cluster (VC) provisioning (computation + communication resources) lie in two folds: (1) optimal placement of VCs and inter-VM traffic routing involve NP-hard problems, which are non-trivial to solve offline, not to mention if an online efficient algorithm is sought; (2) an efficient pricing mechanism is missing, which charges a market-driven price for each VC as a whole upon request, while maximizing system efficiency or provider revenue over the entire span. This paper proposes efficient online auction mechanisms to address the above challenges. We first design SWMOA, a novel online algorithm for dynamic VC provisioning and pricing, achieving truthfulness, individual rationality, computation efficiency, and $(1 + 2\log\mu)$-competitiveness in social welfare, where μ is related to the problem size. Next, applying a randomized reduction technique, we convert the social welfare maximizing auction into a revenue maximizing online auction, PRMOA, achieving $O(\log\mu)$-competitiveness in provider revenue, as well as truthfulness, individual rationality and computation efficiency. We investigate auction design in different cases of resource cost functions in the system. We validate the efficacy of the mechanisms through solid theoretical analysis and trace-driven simulations.

Index Terms—Cloud Computing, Auction Mechanism Design, Online Algorithm, Resource Allocation

1 Introduction

With the proliferation of cloud computing, more and more individuals and businesses are resorting to cloud platforms for deploying services and running jobs. Besides purchasing individual virtual machines (VMs), significant demands arise on renting a collection of VMs and network bandwidth in-between, to create a virtual cluster (VC) with an inter-connecting virtual private network. Prominent examples include cloud CDNs built on top of virtual clusters across geo-distributed cloud data centers, *e.g.*, Netflix and Comcast on AWS cloud [1]. Enabling technologies such as network virtualization have been studied in the past years [2][3][4][5]. Virtual cluster/network services have been provided by IaaS providers (*e.g.*, Amazon Virtual Private Cloud [6]), where a user defines a VC containing several VMs and the bandwidth demand among them, and the cloud provider provisions the cluster (or virtual private cloud) with exclusive resources and bandwidth guarantee. For example, Comcast is exploiting Amazon VPC to provide interactive entertainment on demand [7].

The provisioning of a virtual cluster involves VM placement, *i.e.*, assigning each VM to a location (*e.g.*, a data center), and inter-VM traffic routing, *i.e.*, finding path(s) with available bandwidth to send the traffic from one VM to another. There is not yet an efficient resource allocation algorithm for VC provisioning even in the offline case with all user VC requests known, due to the NP hard nature of the underlying problem. In practice, user requests arrive dynamically over time; an efficient online algorithm is in need, which allocates resources for VC provisioning on the spot, while guaranteeing long-term optimality in resource utilization and user satisfaction. The focus of recent studies on VC provisioning (a.k.a. virtual network embedding in some literature) has been on heuristic offline or online algorithm design to approximate the optimal solutions, with no analytical performance guarantee. We provide a detailed summary of the existing work in Sec. 2.

Moreover, an efficient pricing mechanism is missing, to charge users for the VCs on the go. The current practice is to charge an aggregate price of VMs and bandwidth usage in a virtual cluster, based on the fixed unit prices of the computation and communication resources [8]. Such a pricing method lacks market agility to adapt to supply-and-demand changes, risking the provider's revenue as well as social welfare. As a representative market-driven mechanism, auctions have been studied in cloud computing [9][10][23][25]. Compared with fixed pricing, an auction mechanism enables appropriate prices that take real-time demand and supply into consideration, avoiding overpricing or underpricing and achieving revenue or social welfare maximization. The existing cloud auctions focus on allocation of separate VMs [9], or VM bundles which demand computational resources (CPU, RAM, storage) only (but not inter-VM bandwidths) [10]. It is in fact common to

- *Weijie Shi is with the Department of Computer Science, the University of Hong Kong. E-mail: swj05652@gmail.com*
- *Chuan Wu is with the Department of Computer Science, the University of Hong Kong. E-mail: cwu@cs.hku.hk*
- *Zongpeng Li is with the Department of Computer Science, University of Calgary. E-mail: zongpeng@ucalgary.ca*

The project was supported in part by grants from Hong Kong RGC under the contracts HKU 718513, 17204715, 17225516 and C7036-15G (CRF), and by NSERC as well as MITACS.

send traffic between VMs in a VC, *e.g.*, large replication traffic between VMs in a distributed cloud storage system, requesting bandwidths to be allocated between the VMs as well.

To the best of our knowledge, online auction of an entire virtual cluster, including VMs and the network in-between, has not been studied. The difficulty mainly lies in the NP-hard nature of the resource allocation problem for VC provisioning, which hinders exact solutions as typically needed in designing truthful and social welfare maximizing auction mechanisms, even in the offline case. The challenge escalates when we practically target an online auction requiring timely allocation and pricing decisions on the go, while maximizing social welfare or provider revenue over the entire system span.

This paper designs efficient and competitive online auctions for on-demand provisioning and pricing of VCs deployed over geo-distributed cloud data centers with linear or non-linear operational costs. Users arrive dynamically, specifying potential VCs to deploy with tailor-made VMs in different data centers, as well as the traffic in between, at different willingness-to-pay prices. Two online auction mechanisms are designed for social welfare maximization and provider revenue maximization, respectively. The mechanisms are designed based on an online auction framework which dynamically maintains a cost for each type of resources in each data center. Based on which payments of potential VCs are computed, the best VM placement scheme for each user is selected, and acceptance/rejection decisions are made. Our detailed contributions are summarized as follows.

First, in the design of the social welfare maximizing auction, *SWMOA*, we set a dynamic unit cost for each type of computation and communication resources at and across the data centers, which increases with the depletion of the corresponding type of resource. We find the best VC provisioning scheme for each user among the schemes indicated in his bid, which maximizes his utility, by formulating a VC provisioning linear program (LP). The LP can be reduced to a minimum cost multicommodity flow problem, which is efficiently solvable using existing algorithms. We then compute the overall cost of the obtained VC provisioning scheme and compare the cost with the willingness-to-pay from the user. A user is accepted if the VC acquired provides positive utility. In the case of linear resource costs, we show that *SWMOA* achieves truthfulness, individual rationality, computation efficiency, and $(1 + 2\log\mu)$-competitiveness in social welfare, where μ is related to the problem size. In the case of non-linear costs, we also show that our auction algorithm still achieves truthfulness, individual rationality, computation efficiency, and $O(\log\mu)$-competitiveness in social welfare.

Second, the revenue maximizing online auction, *PRMOA*, is built on the basis of *SWMOA*. We use *SWMOA* to first obtain a tentative VC allocation and a payment for each user, and then re-examine each tentatively accepted bid with a randomized boosted payment to improve provider revenue. The randomized payment is carefully designed to be still below the user's corresponding true valuation with high probability, without the knowledge of the actual true valuation. In this way, the provider is able to extract almost the largest possible revenue with high probability. PRMOA achieves $O(\log\mu)$-competitiveness in provider revenue in case of linear resource costs, as well as truthfulness, individual rationality and computation efficiency.

In the rest of the paper, we discuss related work in Sec. 2 and present the system model in Sec. 3. Sec. 4 presents the social welfare maximizing online auction, *SWMOA*, in the case of linear resource costs. Sec. 5 presents the revenue maximizing online auction, *PRMOA*, in the case of linear resource costs. Sec. 6 extends the model and auction design to the non-linear operational cost cases. Sec. 7 presents the trace-driven simulation studies and Sec. 8 concludes the paper.

2 Related Work

The virtual cluster provisioning (a.k.a. virtual network embedding/mapping) problem has attracted substantial research interest in recent years. Li *et al.* [11] formulate the VM placement problem and consider the traffic cost and physical machine utilization cost. Zhang *et al.* [12] study how to map VMs to servers to minimize the failure probability of the user's virtual data center (*i.e.*, maximize the reliability). Heuristic algorithms are proposed to calculate the failure probability and minimize it efficiently. Ballani *et al.* [13] propose a pricing mechanism for VMs based on a user's bottleneck resource consumption in VM placement. Esposito *et al.* [14] solve the VC mapping problem using primal and dual decomposition. Chowdhury *et al.* [15] reduce VC mapping to link mapping which is formulated as an integer programming (IP) program, and solve the latter whenever a VC request arrives using LP relaxation and deterministic/randomized rounding. All these work on offline solutions of VC provisioning.

For handling online VC provisioning, Grandl *et al.* [16] present a scheduling algorithm to assign tasks to machines, which is essentially a multidimensional bin packing algorithm. Even *et al.* [17] [18] study the online multicommodity-flow routing problem and the online VC mapping algorithm. Allowing violation of capacity constraints, a near-optimal online algorithm is proposed. Cai *et al.* [19] propose a quadratic IP formulation for the VC provisioning problem. Compared to these work, we propose the pricing rule to stimulate users to report their real valuation, and we use solid theoretical analysis to guarantee the performance of our algorithms in the worst case.

Auction mechanisms have been widely applied for efficient resource allocation in various networking systems. To design a truthful auction with an NP-hard underlying allocation problem, a useful technique is to first design an approximation algorithm, and then use the critical bid rule to decide an appropriate price [20], which is an extension of the classic VCG technique [21]. Sun *et al.* [22] adopt this technique and design a dynamic spectrum auction. Wang *et al.* [9] apply the same method, and derive a collusion-resistant mechanism for cloud computing. Mashayekhy *et al.* [23] extend the method to online auctions. Another approach is to resort to the LP decomposition technique [24]. Zhang *et al.* [25] and Shi *et al.* [10] design truthful auctions for dynamic VM provisioning using this method. In addition,

Zhang *et al.* [26][27] design VM auctions following different approaches, the MIDR algorithm and the primal-dual framework. Fu *et al.* [28] apply a core-selecting technique to the VM provisioning problem, to prevent shill bidding. None of the existing cloud auctions consider the allocation of both VM computational resources and inter-VM communication resources simultaneously, which is necessary when users request VCs. In VC auctions, the underlying resource allocation problem is significantly more difficult, involving both VM placement and traffic routing decisions, calling for novel online algorithm design.

3 Problem Model

3.1 Online VC Auction

Consider an IaaS cloud with P geo-distributed data centers (DCs). Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$, where X can be different quantities. Data center $p \in [P]$ has $\hat{A}^t_{p,r}$ units of type-r computational resources at time t (such as CPU, RAM and disk), for all $r \in [R]$, where R is the number of computational resource types. The data centers are inter-connected through S gateway routers, each of which is located with one data center, and hence $S = P$. An illustration of the data center network is given in Fig. 1. Let $\mathbb{E}$ be the set of links connecting gateway routers and data centers. The bandwidth capacity of link $(w_1, w_2) \in \mathbb{E}$ is $\hat{d}^t_{w_1,w_2}$ at time t, where w_1 and w_2 can be either a router in $[S]$ or a data center in $[P]$, whose value can vary over time. Practically, we assume the bandwidth on the link connecting a data center and its colocated gateway router (*e.g.*, the link from DC 1 to Router 1 in Fig. 1) is not the bottleneck, as compared to inter-DC link bandwidth. For example in the Google data center network given in [29], there are 19 links interconnecting the geo-distributed data centers. The bandwidth capacity is known if the cloud provider owns the network in-between its data centers (*e.g.*, the case of Google data centers), or can be probed by the cloud provider otherwise. We will first consider a linear cost model: the unit operational cost of resource r at DC p at time t is $o_{p,r}$, and the unit operational cost of bandwidth on link (w_1, w_2) is o_{w_1,w_2}. We will extend our model and auction design to the non-linear cost function case in Sec. 6.

N users arrive on the fly, and request for virtual clusters (VCs). User $n \in [N]$ arrives at time T^s_n, submits a bid to demand a VC immediately, and releases the VC at time T^f_n. Let $T_n = T^f_n - T^s_n$ be the usage duration of user n's VC, and $T = \max_{n\in[N]}\{T_n\}$ denote the largest usage duration among all users. The VC required by user n includes V_n tailor-made VMs, and VM v in the VC consumes $a^n_{v,r}$ units of type-r computational resource, $\forall r \in [R]$. The bandwidth demand to send traffic from VM v_1 to VM v_2 in user n's VC is $\Gamma^n_{v_1,v_2}$, which occupies inter-datacenter bandwidth when VM v_1 and VM v_2 are located in different data centers. User n can specify several VM placement schemes with different preferences. One example of different preferences for VM placement schemes is in the application of CDN, users prefer that their content is close to the customers for better service. Let $\mathbb{B}_n$ be his set of VM placement schemes. Each scheme $\beta \in \mathbb{B}_n$ specifies the placement of VMs in his VC, represented by $z^{n,\beta}_{v,p} \in \{0, 1\}$, indicating that VM v is placed in data center p if $z^{n,\beta}_{v,p} = 1$ and not if $z^{n,\beta}_{v,p} = 0$. Together

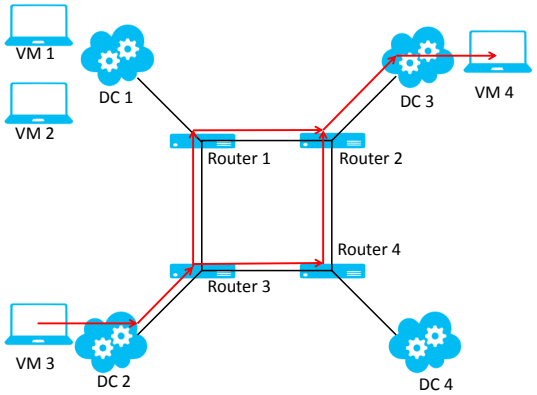


Fig. 1. An example of virtual cluster provisioning.

with each scheme, the user submits a valuation $b'_{n,\beta}$, which is his willingness-to-pay if his VC bid is successful the VMs are allocated in the data centers specified in scheme β. Different values of $b'_{n,\beta}$'s indicate his preferences among the schemes, decided by the need of his workload or services. For example, if the user is running MapReduce workloads, he may specify to place all his VMs in a selected datacenter; if the user is operating an online video service, his VMs are preferred to be located close to large population of the service users.

In summary, the bid of user n can be expressed as $(T^s_n, T^f_n, \{a^n_{v,r}\}_{v\in[V_n],r\in[R]}, \{\Gamma^n_{v_1,v_2}\}_{v_1,v_2\in[V_n]}, \{z^{n,\beta}_{v,p}\}_{\beta\in\mathbb{B}_n,v\in[V_n],p\in[P]}, \{b'_{n,\beta}\}_{\beta\in\mathbb{B}_n})$. Fig. 1 shows an example of a user requesting a VC of 4 VMs and plots one VM placement scheme.

The cloud provider acts as the auctioneer. Upon arrival of a user's bid, the provider immediately responds with whether to serve this user, which VM placement scheme to be adopted, and what price to charge this user for. The decision variables include the following: (i) $y_{n,\beta} \in \{0, 1\}, \forall n \in [N], \beta \in \mathbb{B}_n$, indicating whether user n's request is accepted according to his placement scheme β ($y_{n,\beta} = 1$) and not otherwise ($y_{n,\beta} = 0$). At most one VM placement scheme can be accepted for each bid. (ii) $\tilde{b}'_n, \forall n \in [N]$, payment of user n if his bid is accepted. (iii) $f^n_{v_1,v_2,w_1,w_2}, \forall n \in [N], v_1, v_2 \in [V_n], (w_1, w_2) \in \mathbb{E}$, indicating the routing traffic on each link, which will be illustrated in detail in Sec. 3.2.

3.2 Goals of Mechanism Design

Our online auction design targets the following properties. (i) *Truthfulness and individual rationality:* The auction mechanism is truthful if for any user n, bidding a different valuation other than $b'_{n,\beta}$ does not increase his utility, which is the difference between his valuation and his payment $b'_{n,\beta} - \tilde{b}'_n$. Truthfulness ensures that selfish buyers are stimulated to reveal their true valuations of the VCs they demand, simplifying the bidding strategy and the auction design. Individual rationality requires that any user's utility is non-negative. (ii) *Computation efficiency:* The mechanism should run in polynomial time, in order to be practically applied in an online fashion. (iii) *Competitive in social welfare or provider revenue:* The provider's revenue equals the total payment from all users in the online auction, $\sum_{n\in[N]} \tilde{b}'_n$, less the total operational cost on DCs. Since the provider's revenue and the users' payment part in the aggregate user utility

($\sum_{n\in[N],\beta\in\mathbb{B}_n} b'_{n,\beta}y_{n,\beta} - \sum_{n\in[N]} \tilde{b}'_n$) cancel each other, the social welfare is the total valuation of accepted users, *i.e.*, $\sum_{n\in[N],\beta\in\mathbb{B}_n} b_{n,\beta}y_{n,\beta}$, less the total operational cost. Let S_{online} denote the social welfare achieved under an online mechanism, and S_{opt} be the offline optimum social welfare. An online mechanism is c-competitive in social welfare if the ratio of S_{opt}/S_{online} is upper-bounded by c for any input instance. On the other hand, Let R_{online} be the total provider revenue obtained under the online auction. An online auction is c-competitive in provider revenue if the ratio between the offline optimal social welfare and the provider revenue achieved by the online auction, S_{opt}/R_{online}, is upper-bounded by c for any input instance. Here comparing to S_{opt} instead of the offline revenue in computing the competitive ratio in revenue is a general practice, since the optimal truthful auction generating largest revenue cannot be identified [30]. We also note that in fact no truthful auction can achieve a revenue at the amount of S_{opt}, which is only achievable when the users bid true valuations and the provider always charges users according to their bid prices. However, the latter leads to untruthful bidding, and hence a contradiction.

We next formulate the offline VC provisioning and winner determination problem, supposing all bids within system span are known and truthful bidding is guaranteed. The objective in (1) indicates social welfare maximization, whose optimal value is S_{opt}. It can be easily changed to revenue maximization by replacing the social welfare with the provider's revenue.

$$\text{maximize} \sum_{n\in[N],\beta\in\mathbb{B}_n} b'_{n,\beta}y_{n,\beta} - \sum_{n\in[N],v\in[V_n],p\in[P],r\in[R]} z^n_{v,p}a^n_{v,r}o_{p,r}T_n - \sum_{n\in[N],v_1,v_2\in[V_n],(w_1,w_2)\in\mathbb{E}} f^n_{v_1,v_2,w_1,w_2}o_{w_1,w_2}T_n \quad (1)$$

$$\text{s.t. } z^n_{v,p} = \sum_{\beta\in\mathbb{B}_n} y_{n,\beta}z^{n,\beta}_{v,p} \ \forall n\in[N], v\in[V_n], p\in[P] \quad (1a)$$

$$\sum_{\beta\in\mathbb{B}_n} y_{n,\beta} \le 1 \ \forall n\in[N] \quad (1b)$$

$$\sum_{n\in N_t}\sum_{v\in[V_n]} z^n_{v,p}a^n_{v,r} \le \hat{A}^t_{p,r} \ \forall p\in[P], r\in[R], t\in[T] \quad (1c)$$

$$\sum_{w:(p,w)\in\mathbb{E}} f^n_{v_1,v_2,p,w} = \Gamma^n_{v_1,v_2}z^n_{v_1,p} \ \forall p\in[P], n\in[N], v_1,v_2\in[V_n] \quad (1d)$$

$$\sum_{w:(w,p)\in\mathbb{E}} f^n_{v_1,v_2,w,p} = \Gamma^n_{v_1,v_2}z^n_{v_2,p} \ \forall p\in[P], n\in[N], v_1,v_2\in[V_n] \quad (1e)$$

$$\sum_{w:(w,\nu)\in\mathbb{E}} f^n_{v_1,v_2,w,\nu} = \sum_{w:(\nu,w)\in\mathbb{E}} f^n_{v_1,v_2,\nu,w} \quad \forall n\in[N], v_1,v_2\in[V_n], \nu\in[S] \quad (1f)$$

$$\sum_{n\in N_t}\sum_{v_1,v_2\in[V_n]} f^n_{v_1,v_2,w_1,w_2} \le \hat{d}^t_{w_1,w_2} \ \forall (w_1,w_2)\in\mathbb{E}, t\in[T] \quad (1g)$$

$$y_{n,\beta}\in\{0,1\} \ \forall n\in[N],\beta\in\mathbb{B}_n \quad (1h)$$

$$f^n_{v_1,v_2,w_1,w_2} \ge 0 \ \forall n\in[N], v_1,v_2\in[V_n], (w_1,w_2)\in\mathbb{E} \quad (1i)$$

Here $z^n_{v,p}$ is an auxiliary variable defined in the first constraint (1a), representing if data center p is selected to host VM v in user n's VC, which is 1 if user n's bid is accepted, one VM placement scheme is picked, and v is placed in p according to this scheme, and 0 otherwise. $f^n_{v_1,v_2,w_1,w_2}$ represents the allocated bandwidth on link (w_1,w_2) for the traffic flow from VM v_1 to VM v_2 in user n's VC. Let N_t be the set of active users at time t, whose VCs are in use at t: $N_t = \{n|t\in[T^s_n, T^f_n]\}$. Constraint (1b) guarantees that at most one scheme is accepted for each user. Constraint (1c) requires that at any time the allocated computational resources at each data center do not exceed their respective capacity. Constraints (1d) (1e) and (1f) model routing of user n's traffic flow from VM v_1 to VM v_2. We allow multi-path routing of each inter-VM flow. An illustration of two paths taken by the flow from VM 3 to VM 4 is given in Fig. 1. Constraints (1d) specifies that at the data center p where v_1 of user n is placed ($z^n_{v_1,p} = 1$), the total out-bound bandwidth from data center p allocated for user n's flow from v_1 to v_2 (LHS of (1d)) should equal his specified bandwidth demand ($\Gamma^n_{v_1,v_2}$). Similarly, constraint (1e) specifies the total in-bound bandwidth at data center p equal to the bandwidth demand $\Gamma^n_{v_1,v_2}$ if v_2 is placed at data center p. Constraint (1f) is the flow conservation constraint at each router for user n's flow from v_1 to v_2. Since routers are intermediate nodes, in-bound and out-bound flow rates should be equal. Constraint (1g) requires that the aggregated bandwidth allocated on each link in $\mathbb{E}$ does not exceed the link capacity.

The objective function of (1) is maximizing the social welfare, which is the total user valuation less the total operational cost. The offline optimization problem in (1) is a mixed integer linear program and can be proven NP-hard by a reduction to the knapsack problem. The proof is given in Appendix **??**.

Theorem 1. *The offline optimization problem (1) is NP-hard.*

Hence, it is very challenging even to solve the VC provisioning problem in an offline fashion, with all information known.

3.3 Preliminaries for Online Mechanism Design

To design an efficient online auction, we introduce a few concepts which will be useful later. We regard each computational resource at each data center and the bandwidth on each link as different resources. Thus there are $M = PR+E$ types of resources in total (here $E = |\mathbb{E}|$), *i.e.*, PR types of computational resources at different data centers and E bandwidth resources on different links. We use $r(n,m,t)$ to denote the amount of type m resource consumed by user n at time t, which is a value depending on VM placement and traffic routing decisions made for the user's bid. For example, let m correspond to computational resource r at data center p. If the bid is accepted and one or more VMs are allocated in data center p, then the amount of resource m consumed by user n in a $t\in T_n$ is: $r(n,m,t) = \sum_{v\in[V_n]} z^n_{v,p}a^n_{v,r}$. If m denotes the bandwidth resource on link $(w_1,w_2)\in\mathbb{E}$, then the bandwidth consumed on the link by the user during T_n is: $r(n,m,t) = \sum_{v_1,v_2\in[V_n]} f^n_{v_1,v_2,w_1,w_2}$.

For ease of presentation, we normalize the scale of the capacity of each type of resource, so that the total capacity of resource m at each time t is equal to 1, *i.e.*,, $\hat{A}^t_{p,r} = 1, \hat{d}^t_{w_1,w_2} = 1, \forall p\in[P], r\in[R], (w_1,w_2)\in\mathbb{E}$. We then divide all the demands of resource $r(n,m,t)$ by $\hat{A}^t_{p,r}$ or $\hat{d}^t_{w_1,w_2}$, respectively. Then the resource constraints (1c) and (1g) require that $\sum_{n\in[N]} r(n,m,t) \le 1$, for all $m\in[M]$ and t in $[T]$. For example, if a user needs 2 VMs in one DC, and each VM uses 4 CPUs, then the amount of CPU resource consumed by this user in this DC is 8. If there are totally 320 CPUs in that DC, then $r(n,m,t) = 1/40$ after normalization.

Such normalization does not change the resource allocation result.

Notice that the total operational cost brought by a user n's β scheme can be easily calculated, which is $o(n,\beta) = \sum_{m\in[M],t\in T_n} r(n,m,t)T_n o_m$, where o_m is the unit operational cost of type m. We define the net valuation of user n's scheme β to be his valuation less operational cost: $b_{n,\beta} = b'_{n,\beta} - o(n,\beta)$. Then the total social welfare equals the total net valuation of accepted users.

We make two assumptions. *First*, a user's net valuation is approximately proportional to the amount of resources his VC requires, *i.e.*, the per unit per time slot valuation is bounded: $1 \le \frac{b_{n,\beta}}{Mr(n,m,t)T_n} \le F_1$, where F_1 is a positive value, $\forall n,m,t$ and $r(n,m,t)\neq 0$. We also assume that the ratio between the highest and lowest valuations is bounded: $\frac{\max_{n\in[N],\beta\in\mathbb{B}_n}\{b_{n,\beta}\}}{\min_{n\in[N],\beta\in\mathbb{B}_n}\{b_{n,\beta}\}} \le F_2$, where F_2 is a positive value. Let $F = \max\{F_1, F_2\}$ and we will use F as the upper bound for both of the above inequalities in the following. Second, there is an upper bound on the amount of resources required by each VC in each time slot at each data center or link, *i.e.*, $r(n,m,t) \le \frac{1}{\log\mu}$ where $\mu = 2MTF+1$, which implies that the resource demand of each individual user is small as compared to the total capacity of each data center. Here μ (related to F) is an important parameter to appear in our competitive ratios. We summarize important notation in Table 1, for ease of reference.

TABLE 1
Key Notation

N	# of users	[X]	integer set $\{1,\dots,X\}$
P	# of data centers	$\tilde{b}'_n$	user n's payment
M	$M = PR + \lvert\mathbb{E}\rvert$	T_n	duration of user n's VC

R	# of computational resource types
T_n^s	arriving time of user n
T_n^f	departure time of user n
T	$\max_n\{T_n\}$
$b'_{n,\beta}$	user n's valuation if scheme β is accepted
$\mathbb{B}_n$	user n's set of VM placement schemes
$r(n,m,t)$	demand of resource m at t by user n
$\lambda_m(t,n)$	load factor of resource m at t before user n
F	constant related to valuation variation
μ	$2MTF+1$
$c_m(t,n)$	cost of resource m at t before user n
$C(n,\beta)$	total cost for scheme β of user n
V_n	# of VMs in user n's VC
$a^n_{v,r}$	amount of resource r required by VM v of user n
$\Gamma^n_{v,v'}$	traffic from VM v to VM v' of user n
$o_{p,r}$	unit operational cost of resource r on DC p
o_{w_1,w_2}	unit operational cost of link (w_1,w_2)
$y_{n,\beta}$	scheme β of user n is accepted or not
$z^{n,\beta}_{v,p}$	VM v is assigned to data center p or not in β of user n
$\mathbb{E}$	set of links
$\hat{d}^t_{w_1,w_2}$	bandwidth capacity of link (w_1,w_2) at time t
$\hat{A}^t_{p,r}$	capacity of type r resource at DC p at time t
$f^n_{v_1,v_2,w_1,w_2}$	traffic from v_1 to v_2 of user n on link (w_1,w_2)

4 Social Welfare Maximizing Online Auction

We now design a social welfare maximizing online mechanism, *SWMOA*.

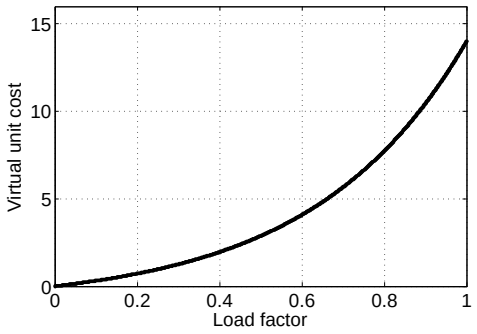


Fig. 2. An example of the virtual unit cost function $c_m(t,n)$, where $\mu = 13.5$.

Main Idea. At a high level, our strategy of finding a competitive VC allocation solution for problem (1) is to maintain a dynamic *virtual unit cost* $c_m(t,n)$ for each type of resource m at each time slot $t \in [T]$ (containing future time slots with resource reservation), before user n arrives. For any VM placement scheme β of the user, we calculate a total virtual cost $C(n,\beta)$ based on the unit costs and the amount of resources the scheme consumes. Scheme β is a candidate to be accepted if its cost is smaller than its net valuation: $C(n,\beta) \le b_{n,\beta}$.

The virtual unit cost $c_m(t,n)$ is designed to increase when the remaining available amount of resource m at time t depletes. Thus a higher virtual cost indicates that type of resource is scarce. As a result, the cost of a VC is higher if it consumes resources in shortage, reducing the likelihood that it is chosen by the online algorithm. On the other hand, by comparing the total cost with the net valuation of the VC, the online mechanism can pick users with higher net valuation compared with the amount of resources consumed.

We define the *load factor* (before user n arrives) of resource m for each time $t \in [T]$ to be the amount of already allocated (reserved) resources in the respective time slot before user n arrives: $\lambda_m(t,n) = \sum_{n'<n,n'\in[N]} r(n',m,t)$, where $n' < n$ indicates that user n' arrives earlier than user n. We define the virtual unit cost of resource m at time t, computed before user n arrives (i.e., before counting in resource consumption of user n in each time slot), as follows:

$$c_m(t,n) = \mu^{\lambda_m(t,n)} - 1, \forall t\in[T], n\in[N], m\in[M]. \quad (2)$$

$c_m(t,n)$ is designed to increase exponentially with the increase of consumed resources. This allows the user to consume resources freely when resources are abundant, since the cost is close to 0 when $\lambda_m(t,n)$ is small. However when the load factor is close to 1, the cost increases fast to a value large enough to forbid the user to use the resource in shortage. The idea of the unit cost formula comes from the study of packing/covering LPs [31]. We show an example of the virtual unit cost function in Fig. 2.

We define the total cost that user n incurs if his scheme β is accepted as the sum of the costs of all resources consumed:

$$C(n,\beta) = \sum_{m\in[M]}\sum_{t\in T_n} c_m(t,n)r(n,m,t), \forall n\in[N], \beta\in\mathbb{B}_n. \quad (3)$$

The mechanism accepts a user if there exists a scheme β with cost less than its net valuation. If there are more

than one schemes satisfying the condition, we choose the one with the largest user utility: $b_{n,\beta} - C(n,\beta)$, which is the scheme maximizing user's utility under truthful bidding when we set the payment to be $\tilde{b}'_n = C(n,\beta) + o(n,\beta)$.

Cost Computation. The total cost $C(n,\beta)$ for each VM placement scheme β includes the cost of computation resources and the cost of bandwidth consumption. Suppose $c_{r,p}(t,n)$ denote the virtual unit cost for resource r in data center p at time t. The total cost of computational resource is $\sum_{t\in T_n}\sum_{r\in[R]}\sum_{p\in[P]}\sum_{v\in[V_n]} c_{r,p}(t,n)a_{v,r}z^n_{v,p}$. The bandwidth cost differs depending on the routing plan of flows among VMs in user n's VC. We find the best traffic routing plan for VM placement scheme β, that minimizes the total bandwidth cost, by solving the LP (4).

The LP identifies the best routing paths and bandwidth allocation for flows among VMs in user n's VC (decided by $f^n_{v_1,v_2,w_1,w_2}$'s), when the VMs are placed in data centers according to scheme β. Here $c_{w_1,w_2}(t,n)$ represents the virtual unit cost for bandwidth on link (w_1,w_2) at time t, before user n submits his bid. $\sum_{v_1,v_2\in[V_n]} f^n_{v_1,v_2,w_1,w_2}$ is the overall bandwidth consumed on link (w_1,w_2) by flows among user n's VMs. p_{v_1} and p_{v_2} represent the assigned data center for VM v_1 and VM v_2, respectively. Constraints (4a) and (4b) ensure that the total out-bound/in-bound traffic from p_{v_1}/to p_{v_2} equals the traffic demand from v_1 to v_2. Constraint (4c) is the flow conservation constraint. Solving (4) gives us the optimal routing as well as the minimum total bandwidth cost under scheme β. In fact, (4) is a minimum cost multi-commodity flow problem which allows fractional flows, and can be solved using efficient algorithms [32] in $O((E+V_n^2)P)$ time.

$$\text{minimize} \sum_{(w_1,w_2)\in\mathbb{E}}\sum_{t\in T_n}(c_{w_1,w_2}(t,n)\sum_{v_1,v_2\in[V_n]} f^n_{v_1,v_2,w_1,w_2}) \quad (4)$$

$$\sum_{w:(p_{v_1},w)\in\mathbb{E}} f^n_{v_1,v_2,p_{v_1},w} = \Gamma^n_{v_1,v_2} \ \ \forall v_1,v_2\in[V_n] \ (4a)$$

$$\sum_{w:(w,p_{v_2})\in\mathbb{E}} f^n_{v_1,v_2,w,p_{v_2}} = \Gamma^n_{v_1,v_2} \ \ \forall v_1,v_2\in[V_n] \ (4b)$$

$$\sum_{w:(w,\nu)\in\mathbb{E}} f^n_{v_1,v_2,w,\nu} = \sum_{w:(\nu,w)\in\mathbb{E}} f^n_{v_1,v_2,\nu,w} \ \ \forall v_1,v_2\in[V_n],\nu\in[S] \ (4c)$$

$$f^n_{v_1,v_2,w_1,w_2} \geq 0 \ \ \forall (w_1,w_2)\in\mathbb{E}, v_1,v_2\in[V_n] \ (4d)$$

Online Auction. The social welfare maximizing online auction is summarized in Alg. 1. Upon receiving a new user's bid, we calculate the minimum total cost for each scheme β by solving (4). We choose the scheme β^* with largest user utility and accept the user if his utility is positive. Then we upgrade the virtual unit costs. We present properties achieved by *SWMOA* in Thm. 2, with proof given in Appendix **??**. Especially, the payment of each winning user is computed based on virtual unit resource costs before counting in resources required by the user. Such independence of payments from the bids guarantees truthfulness of the mechanism.

Theorem 2. *The online auction mechanism* SWMOA *(Alg. 1) is truthful and individually rational, never violates the resource constraints, runs in $O((E+V_n^2)PB)$ time in each round, and achieves a $(1+2\log\mu)$-competitive ratio in social welfare, where $B=\max_{n\in[N]}|\mathbb{B}_n|$ is the largest number of schemes submitted by any user.*

Algorithm 1 Social Welfare Maximization Online Auction *SWMOA*

if user $n\in[N]$ arrives **then**
  for all schemes $\beta\in\mathbb{B}$ **do**
    Solve the minimum cost multi-commodity flow problem (4) and find optimal routing
    Compute the total cost $C(n,\beta)$
    Calculate the utility $b_{n,\beta}-C(n,\beta)$ of scheme β
  end for
  Let β^* be the scheme with maximal utility
  if $C(n,\beta^*)\leq b_{n,\beta^*}$ **then**
    Accept user n with scheme β^*, i.e., set $y_{n,\beta^*}=1$, $y_{n,\beta}=0,\forall\beta\neq\beta^*$
    Charge user n the payment $\tilde{b}'_n = C(n,\beta^*)+o(n,\beta)$
    Update unit costs $c_m(t,n+1)$ according to (2)
  else
    Reject user n, i.e., set $y_{n,\beta}=0,\forall\beta\in\mathbb{B}_n$
  end if
end if

Dynamic Routing. Unlike VMs which are not easy to be migrated among DCs, routing plans can be dynamically adjusted. An intuitive improvement of the online auction *SWMOA* is to adjust the routing plan after each user's departure, and find a new routing plan with the lowest flow cost for each existing user. This can be done by solving the multi-commodity flow problem (4) for each user again. By decreasing the traffic on bottleneck links in the network, we may possibly decrease the users' payment, while improving the social welfare. But in some cases, which are easy to construct, routing adjustment cannot increase the total social welfare. For example, if only one user arrives during the whole time span, then any adjustment cannot improve the social welfare. So we cannot assert that the competitive ratio, which represents the worst-case performance of the online algorithm, can be improved under dynamic routing. We will verify the effect of dynamic routing in the simulations.

5 Revenue Maximizing Online Auction

We next design a revenue maximizing online auction *PRMOA*, inspired by a randomized reduction technique [30].

Main Idea. We use *SWMOA* to first obtain a tentative VC allocation and a payment for each user, and then re-examine each tentatively accepted bid with a randomized boosted payment to improve revenue. The randomized boosted payment is carefully designed to be still below the user's corresponding true valuation with high probability, without the knowledge of the true valuation.

Obviously, the ideal payment which maximizes the revenue is a value just below the user's true valuation. However, a straightforward payment rule, *e.g.*, 90% of the user's bid price, breaks the truthfulness of the mechanism, since users can gain more from under-reporting their valuation. The payment has to be independent of the users' bids in order to guarantee truthfulness. So the principle of designing a revenue maximization auction is to set the payment close to the valuation but without knowing the valuation in the decision process.

Recall the payment decision process of *SWMOA*: We calculate the cost for a scheme $C(\beta, n)$, accept and charge the user $\tilde{b}'_n = C(n,\beta) + o(n,\beta)$ if his net valuation for this scheme is larger than $C(\beta, n)$. We use the original threshold $C(n,\beta)$ as a foundation, and add a randomized value δ on it: $\tilde{b}'_n = C(\beta, n) + \delta + o(n,\beta)$. Setting δ is the key in achieving revenue competitiveness. If δ is too small, not enough revenue is extracted. On the other hand, if δ is too large, the payment may exceed the valuation, forcing our auction mechanism to reject the user, which brings great loss to the revenue. In order to upper-bound the occurrence probability of either cases, we let $\delta = 0$ with probability $1/2$, which means that our algorithm will be exactly the same as *SWMOA*, both in allocation and payment, with probability $1/2$. With the other half probability, we try to set the payment as close to the valuation as possible, using a power-of-2 rule: Let $\delta = 2^i b_{min}$ with probability $\frac{1}{2\log F}$, for $i = 1, 2, \ldots, \log F$, where $b_{min} = \min_{n\in[N],\beta\in[\mathbb{B}_n]} b_{n,\beta}$ is the minimal valuation among all users. This guarantees that with probability $\frac{1}{2\log F}$, the payment is at least half of the valuation but still less than the valuation: $b_{n,\beta}/2 \le \tilde{b}_n \le b_{n,\beta}$, which is the ideal payment we want for revenue competitiveness – this will be verified in our proof of Thm. 3.

Another point worth noting is, since we increase the payment $\tilde{b}'_n$, some users will be rejected in *PRMOA*, while accepted in *SWMOA*. In those cases, we still update the virtual unit costs as if these users were accepted. That is, the virtual unit costs are updated just like running *SWMOA*, no matter whether a user is actually accepted by *PRMOA* or not. In this way, we guarantee that the costs of the user schemes are always the same as in *SWMOA*, which is needed to show the competitiveness of our mechanism.

Online Auction. The online auction is summarized in Alg. 2, with properties given in Thm. 3. The proof of Thm. 3 is given in Appendix **??**.

Algorithm 2 Provider Revenue Maximization Online Auction *PRMOA*

if user $n \in [N]$ arrives **then**
  Set $\delta = 0$ with probability $1/2$, and $\delta = 2^i b_{min}$ with probability $\frac{1}{2\log F}$, for $i = 1, 2, \ldots, \log F$
  for all schemes $\beta \in \mathbb{B}$ **do**
    Solve the minimum cost multi-commodity flow problem (4) and find optimal routing
    Compute the total cost $C(n,\beta)$
    Calculate the utility $b_{n,\beta} - C(n,\beta)$
  end for
  Let scheme β^* be the scheme with maximal utility
  if $C(n,\beta^*) \le b_{n,\beta^*}$ **then**
    Update unit costs $c_m(t, n+1)$ according to (2), supposing user n is accepted
  end if
  Set payment $\tilde{b}_n = C(n,\beta^*) + \delta + o(n,\beta)$
  if $\tilde{b}_n \le b_{n,\beta^*}$ **then**
    Accept user n with scheme β^*. Charge the payment $\tilde{b}'_n$.
  else
    Reject user n
  end if
end if

Theorem 3. *The online auction mechanism* PRMOA *(Alg. 2) is truthful and individually rational, never violates the resource constraints, and achieves a $O(\log \mu)$-competitive ratio in provider revenue in expectation in polynomial time.*

6 The Case of Nonlinear Operational Cost

In this section, we extend the model from linear operational costs to non-linear costs, which are more representative in some scenarios. For example, when Dynamic Voltage Frequency Scaling (DVFS) [33], which adjusts the voltage of a CPU to conserve power consumption, is in place, the cost of CPU, reflecting the power consumption of CPU usage, is typically non-linear [34]. To deal with such more complex cost functions, we adopt the Fenchel duality method [35] to find a competitive online auction, which is similar to *SWMOA*. The advantage of applying Fenchel duality is that it gives us the conjugate form of the nonlinear cost functions, allowing us to obtain appropriate pricing design under nonlinear costs.

6.1 Problem Formulation

First we formulate the VC provisioning problem under nonlinear costs in a slightly different way for simplifying problem presentation. Suppose there are in total R types of resources in the system, including all computational resources at different DCs and bandwidth on all links. We define a bundle to be a VC placement scheme with a fixed routing plan. Recall that a scheme is a user specification of placement of VMs. Under a scheme, the cloud provider can adopt different routing plans resulting in different resource usage on links. So a scheme contains an infinite number of bundles. This is just for simplicity of presentation, and we will show in the algorithm design that we can find the desired bundle from many bundles using a multi-commodity flow algorithm in polynomial time. Suppose N users bid for VCs, and user n's bundle β requires $d_{n,\beta,r}(t)$ units of resource r at time t, for all r and t, and has valuation $b_{n,\beta}$. The operational cost function of resource r is defined as:

$$f_r(y_r(t)) = \begin{cases} h_r y_r(t)^{1+\nu_r} & y_r(t) \in [0, C_r] \\ +\infty & y_r(t) > C_r \end{cases} \tag{5}$$

where C_r is the capacity of resource r, $y_r(t)$ is the amount of resource used at time t, and h_r, ν_r are positive constants. The cost is super-linear with the increase of $y_r(t)$, and becomes infinite when $y_r(t)$ exceeds the capacity. Now we formulate the social welfare maximization problem:

$$\text{maximize} \quad \sum_{n\in[N]} \sum_{\beta\in\mathbb{B}} b_{n,\beta} x_{n,\beta} - \sum_{t\in[T]} \sum_{r\in[R]} f_r(y_r(t)) \tag{6}$$

$$\text{s.t.} \quad \sum_{\beta\in\mathbb{B}} x_{n,\beta} \le 1 \ \ \forall n \in [N] \tag{6a}$$

$$\sum_{n\in[N]} \sum_{\beta\in\mathbb{B}} d_{n,\beta,r}(t) x_{n,\beta} \le y_r(t) \ \ \forall r \in [R], t \in [T] \tag{6b}$$

$$x_{n,\beta} \in \{0,1\}, y_r(t) > 0 \ \ \forall n \in [N], r \in [R], t \in [T], \beta \in \mathbb{B} \tag{6c}$$

where $x_{n,\beta} = 1$ represents that user n is served with bundle β. The dual of the problem (6) is:

$$\text{minimize} \quad \sum_{n\in[N]} u_n + \sum_{t\in[T]}\sum_{r\in[R]} f_r^*(p_r(t)) \tag{7}$$

$$\text{s.t.} \quad u_n \geq b_{n,\beta} - \sum_{t\in[T]}\sum_{r\in[R]} d_{n,\beta,r}(t)p_r(t) \quad \forall n\in[N], \beta\in\mathbb{B} \tag{7a}$$

$$u_n \geq 0, p_r(t) \geq 0 \quad \forall n\in[N], r\in[R], t\in[T] \tag{7b}$$

where u_n and $p_r(t)$ are dual variables associated with constraint (6a) and (6b), and $f_r^*(p_r(t))$ is the Fenchel conjugate [35] of the cost function, whose exact form is:

$$f_r^*(p_r(t)) = \begin{cases} (p_r(t)/(1+\nu_r))^{(1+\nu_r)/\nu_r} \frac{\nu_r}{(h_r)^{1/\nu_r}} & y_r^0(t) \leq C_r \\ C_r p_r(t) - h_r C_r^{1+\nu_r} & y_r^0(t) > C_r \end{cases} \tag{8}$$

where $y_r^0(t) = (\frac{p_r(t)}{h_r(1+\nu_r)})^{1/\nu_r}$.

Compared with the model in Sec. 3.1, the formulation here is more concise since we replace the concept of schemes by bundles, and use uniform subscripts r to refer to both computational and bandwidth resources.

6.2 VC Allocation

We find the allocation of VCs, *i.e.*, the values of $x_{n,\beta}$'s, using the KKT conditions. The KKT conditions indicate that $x_{n,\beta} = 0$ unless constraint (7a) is tight for n and β. Thus, we let u_n be the maximal of 0 and the right hand side (RHS) of constraint (7a):

$$u_n \triangleq \max\{0, \max_{\beta\in\mathbb{B}}\{b_{n,\beta} - \sum_{t\in[T],r\in[R]} d_{n,\beta,r}(t)p_r(t)\}\} \tag{9}$$

Subsequently, we let $x_{n,\beta} = 0$, *i.e.*, the provider does not serve user n, when $u_n = 0$. Otherwise we serve bundle β ($x_{n,\beta} = 1$) that maximizes the RHS. The rational of the usage of KKT conditions here is as follows: we can consider $p_r(t)$ as the marginal price of resources (similar as the $c_m(t,n)$ in *SWMOA*). Then the second term of the RHS becomes the total payment of bundle β, which user n should pay for the requested resources, *i.e.*, $\hat{p}_{n,\beta} \triangleq \sum_{t\in[T],r\in[R]} d_{n,\beta,r}(t)p_r(t)$. So the RHS is the utility of bundle β if it would be served. Then the above method maximizes each user's utility, which leads to truthfulness and social welfare maximization.

6.3 Payment Design

To design the payment of each user, we only need to design the marginal price $p_r(t)$ at any amount of allocated resources $y_r(t)$. The marginal price should cover the operational cost. So intuitively, under the offline setting, the marginal price is equal to the marginal operational cost: $f_r'(y_r(t))$. However, in the online setting, the provider can only see the current demand. Therefore, our strategy is to predict the final total demand to be δ_r times the current demand, where $\delta_r = \max\{2, (1+\nu_r)^{1/\nu_r}\}$. The corresponding marginal price is: $p_r(t) = f_r'(\delta_r y_r(t))$. If the predicted demand $\delta_r y_r(t)$ is larger than the capacity C_r, then we need to set a threshold price to filter out low valuation bids. We use the state-of-the-art technique in online auctions [36], and let the marginal price increase exponentially with demand: $p_r(t) = f_r'(C_r)\exp(\theta_r(y_r(t) - C_r/\delta_r))$, where $\theta_r = \max\{\frac{\delta_r\nu_r}{C_r}, \frac{\delta_r}{C_r(\delta_r-1)}\ln(\frac{U_r}{h_r(1+\nu_r)C_r^{\nu_r}})\}$, and $U_r = \max_{n,\beta}\{\frac{b_{n,\beta}}{\sum_t d_{n,r,\beta}(t)}\}$. We show in Appendix **??** that this price function guarantees competitiveness of the online auction. In summary, the marginal price function is defined as:

$$p_r(t) = \begin{cases} f_r'(\delta_r y_r(t)) & y_r(t)\delta_r \leq C_r \\ f_r'(C_r)\exp(\theta_r(y_r(t) - C_r/\delta_r)) & y_r(t)\delta_r > C_r \end{cases} \tag{10}$$

Directed by the discussions above, we present the online algorithm in Alg. 3. Upon arrival of each user, we find the bundle with largest utility by enumerating all schemes. For each scheme, we solve the multi-commodity flow problem to find the routing plan with the lowest payment. Then we accept the user if there is a bundle with positive utility, and update the allocation of resources and marginal prices. The proof of Thm. 4 is given in Appendix **??**.

Algorithm 3 Online Auction for Nonlinear Operational Costs

```
if user n ∈ [N] arrives then
    Find the bundle β with the largest utility: b_{n,β} − p̂_{n,β}, by solving the multi-commodity flow problem (4) for each VM scheme.
    Set the value of u_n according to (9)
    if u_n = 0 then
        Reject user n, i.e., set x_{n,β} = 0, ∀β ∈ 𝔹
    else
        Serve user n with bundle β, i.e., set x_{n,β} = 1, and x_{n,β'} = 0, ∀β' ≠ β
        Charge payment p̂_{n,β}
        Update the value of y_r(t), p_r(t)
    end if
end if
```

Theorem 4. *The online auction mechanism Alg. 3 is truthful and individually rational, never violates the resource constraints, runs in polynomial time in each round, and achieves $O(\log\mu)$-competitive ratio in social welfare.*

Finally, we can still apply the revenue maximizing technique introduced in Sec. 5 to design a revenue maximizing online auction for the non-linear operational cost case based on Alg. 3, since the basic process of Alg. 3 is the same with *SWMOA*. We omit the auction design to avoid duplication.

7 Performance Evaluation

We evaluate our online auction mechanisms based on trace-driven simulations using data collected from the Google cluster [37], and run our mechanisms on a server machine in our server cluster. We translate each job containing multiple tasks, each demanding different amounts of resources, in the Google cluster data into a VC request with VMs of different resource compositions. We generate 5 to 10 schemes for each user. Each scheme indicates a VC request. To generate a VC request for a user, we randomly choose a job from the Google trace data, and decide the number of VMs and their resource composition according to the tasks the job contains and their respective resource demand. The VMs are placed at randomly chosen data centers in each scheme.

There are $R = 3$ types of computational resources: CPU (number of CPU cores), RAM and disk (hard disk space). We simulate $P = 12$ data centers inter-connected following the topology of Google data center network [29]. The default system span is $T = 100$. Users arrive following a Poisson distribution with parameter $\lambda = 50$, and their VC usage durations are randomly chosen within $[5, 10]$. The amount of resources consumed by a VM $a^n_{v,r}$ is set according to the resource demand of the corresponding task in Google cluster data. The bandwidth demand between VMs $\Gamma^n_{v,v'}$ is randomly generated within a range $[0, 1]$. The overall resource capacities, $\hat{\mathbf{A}}, \hat{\mathbf{d}}$, are set to be approximately $[0.2, 0.5]$ fraction of the respective overall user resource demand, and randomly distributed onto data centers and links. This is to create a highly competitive environment to evaluate the performance of our online mechanisms. The operational cost of resources in the linear cost model is randomly generated within range $[0.03, 0.1]$. The valuation $b_{n,\beta}$ is uniformly randomly chosen within the interval decided by the constant F, whose default value is 10. We repeat each experiment for 50 times to obtain the average results.

7.1 Evaluation of *SWMOA*

We first compare our online algorithm *SWMOA*, Alg. 1, with the offline optimum. We calculate the offline optimal social welfare by solving (1) exactly, and divide it by the social welfare achieved by *SWMOA* to obtain the *performance ratio* (*i.e.*, offline/online social welfare ratio as used in our figures), under different numbers of users and different values of F. Recall that the parameter μ in our theoretical competitive ratio is related to F, *i.e.*, $\mu = 2MTF + 1$. Fig. 3 shows the ratio is smaller for a larger number of users N. This is because the more users in the system, the less impact a bad decision for a previous user has on later users. In contrast, less users make it more difficult to achieve optimal allocation since each decision involves more resources. We can also see that for larger values of F, the performance ratio is larger. The theoretical competitive ratio, $O(MTF)$, implies such a result. Larger F represents a larger range of user valuation. In online resource allocation, serving one user means that we may need to reject some user in the future, whose valuation may be much higher when F is larger, leading to low competitiveness of the online mechanism. Fig. 4 shows that the ratio is larger when the number of data centers P is larger, which is consistent with our theoretical competitive ratio as well, since $M = PR + |\mathbb{E}|$. Intuitively, more data centers make the allocation harder since more different types of resources need to be considered.

We now evaluate the online algorithm *SWMOA* under other simulation settings. Different from the default setting where the overall resource capacities $\hat{\mathbf{A}}, \hat{\mathbf{d}}$ are set to be $k \in [0.2, 0.5]$ times of the respective overall user resource demand, we further evaluate *SWMOA* under $k \in [0.05, 0.3]$ and $k \in [0.8, 1.2]$, respectively. Fig. 5 shows that with smaller resource capacities (the case of $k \in [0.05, 0.3]$), the performance of *SWMOA* is not affected. When the resource capacities are about the overall user demands (the case of $k \in [0.8, 1.2]$), the performance of *SWMOA* is near optimal, since the majority of user demands can be satisfied.

Next we compare *SWMOA* with a heuristic adaptive algorithm TV-VNE in [38], by comparing the social welfare ratio they each achieve when comparing to the offline optimum. We cannot find any auction algorithms designed for DC provisioning. We can only find TV-VNE with a similar model. The design idea of TV-VNE is to provision VCs according to the amount of consumed resources and always try to find a load-balancing allocation. The core technique used by TV-VNE is a heuristic weighted function which achieves a tradeoff between the computational resource usage and the bandwidth usage. When a new user request arrives, TV-VNE chooses an allocation and routing plan which maximizes this function, which intends to balance the usage of all data centers and link bandwidths. This idea is similar to our online algorithm in some sense, but in a heuristic way. Fig. 6 shows that our algorithm consistently outperforms TV-VNE. We further observe that the performance of TV-VNE does not improve much for larger N, reflecting that its heuristic allocation cannot efficiently adjust resource utilization with the increase of users. In contrast, *SWMOA* performs better when the number of user is large, which makes it more suitable for cloud systems at a large scale. In Fig. 7, we prepare two special scenarios: computation intensive (CI-Case) and bandwidth intensive (BI-Case). In the computation intensive case, computation resource demands of VCs are 2 times larger than the default setting. In the bandwidth intensive scenario, bandwidth demands are 2 times larger than the default. We see that *SWMOA* outperforms TV-VNE in all cases. The advantages in the special cases are larger than in the default case, implying that our algorithm better handles extreme VC requests by dynamically adjusting to user demands.

7.2 Evaluation of *PRMOA*

We first compare *PRMOA* in Alg. 2 with the offline optimum. We divide the offline optimal social welfare by the revenue achieved with *PRMOA* to calculate the performance ratio. Fig. 8 shows the performance of *PRMOA* under different values of F. The trends are similar with those for *SWMOA*: larger N, smaller F bring better performance. This is not a coincidence because *PRMOA* uses *SWMOA* as its foundation, and in half of the time adopts the same payment and allocation. In the other half of the time, the success of the attempt to "guess" user's valuation only depends on the random variation, and is not affected by these parameters.

Next we compare the provider revenue achieved by *PRMOA* and TV-VNE. However, TV-VNE is just a VC provisioning algorithm without pricing. Fortunately, it is easy to design a payment for it according to the "unit virtual cost" used in the TV-VNE algorithm, which is the only payment rule making the mechanism truthful. Fig. 9 shows that *PRMOA* performs better in all scenarios. The advantages are even larger than what we show when comparing *SWMOA* with TV-VNE. This validates the efficiency of our algorithm specifically optimized for chasing high revenue.

7.3 Evaluation of Dynamic Routing

We evaluate the performance of dynamic routing adjustment presented in Sec. 4. In Fig. 10 we can see that the performance improvement is small. Because this intuitive improvement only optimizes the bandwidth resource allocation, but does not change the allocation of computational

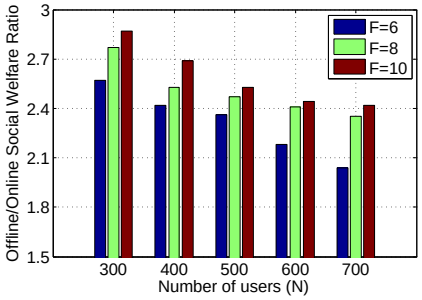


Fig. 3. Performance of *SWMOA* under different values of F

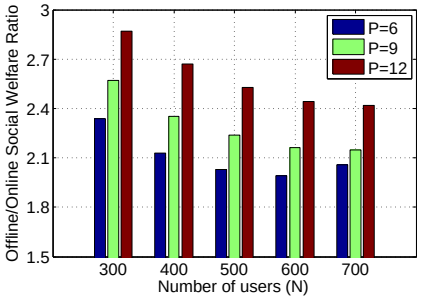


Fig. 4. Performance of *SWMOA* under different values of P

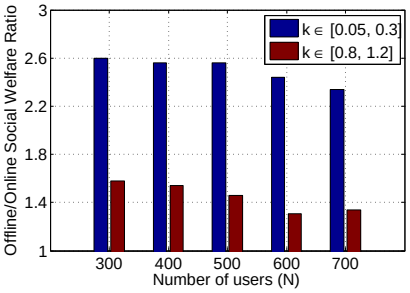


Fig. 5. Performance of *SWMOA* under different resource capacities

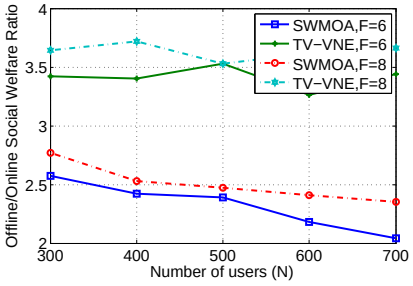


Fig. 6. Comparison between *SWMOA* and TV-VNE

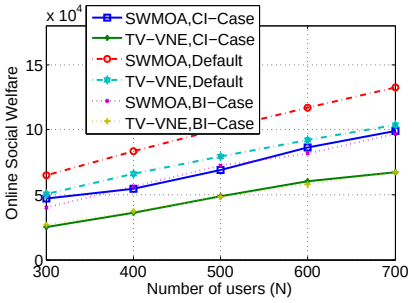


Fig. 7. Comparison between *SWMOA* and TV-VNE under special scenarios

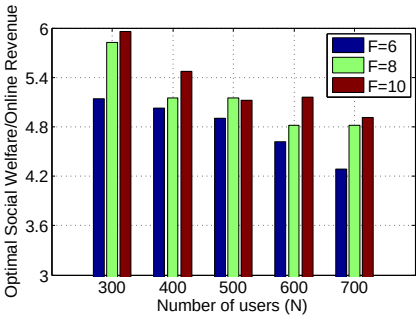


Fig. 8. Performance of *PRMOA* under different values of F

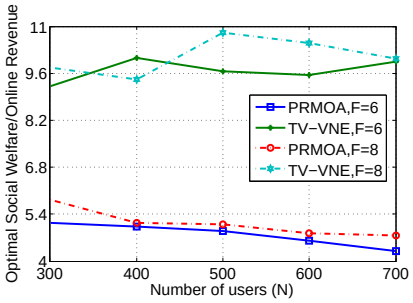


Fig. 9. Comparison between *PRMOA* and TV-VNE

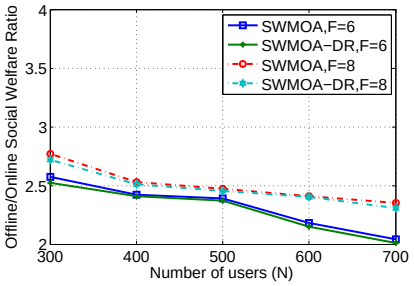


Fig. 10. Performance of dynamic routing (DR)

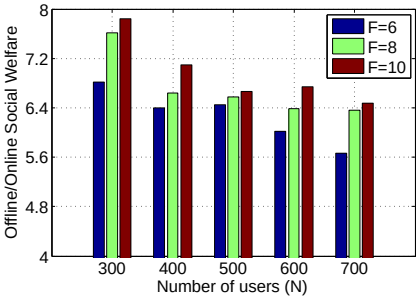


Fig. 11. Performance of *SWMOA* under quadratic costs

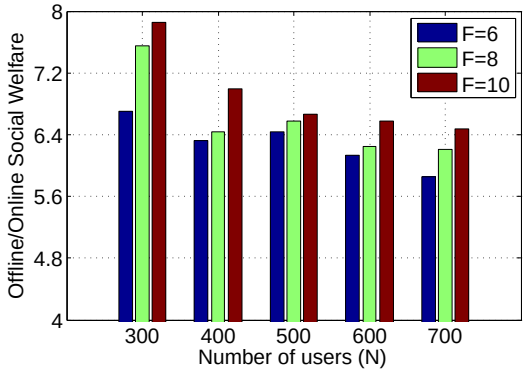


Fig. 12. Performance of *SWMOA* under cubic costs

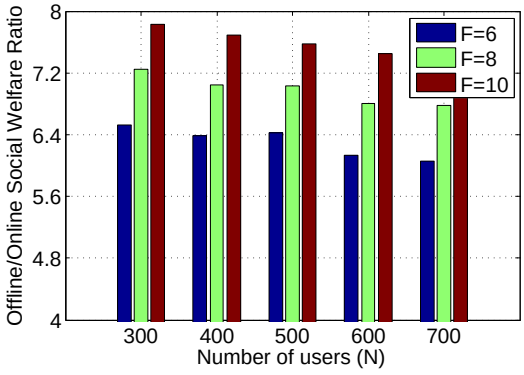


Fig. 13. Performance of *SWMOA* under larger cost coefficients

resources. Since a VC request needs both types of resources, improvement only on bandwidth does not improve the system as a whole much when the bottleneck of provisioning more VCs lies in the computational resources.

7.4 Evaluation of the Non-Linear Cost Function Case

We evaluate the performance of the online auction designed for non-linear operational costs in Sec. 6. We first use $\nu_r = 1$ (quadratic cost), and then $\nu_r = 2$ (cubic cost), with results given in Fig. 11 and Fig. 12, respectively. Coefficients h_r in the cost functions are randomly generated within range $[0.03, 0.1]$. In general, the performance under non-linear costs is not as good as the performance under linear costs. This is consistent with the intuition that non-linear cost functions are harder to deal with in online optimization. Other features are all the same with *SWMOA*, since the underlying algorithm is a similar process with *SWMOA*.

Next, we change the range of coefficients h_r to $[0.2, 0.4]$, and Fig. 13 shows the corresponding performance under quadratic cost functions. We observe that the range of these coefficients does not affect the performance of our algorithm.

7.5 Evaluation of Running Time

We evaluate the running time of our online algorithm *SWMOA* in Alg. 1 on a server machine (with 16 cores Intel Xeon E2-2650 CPU, 80 GB RAM and 500 GB hard disk). Note that *SWMOA* runs in an online fashion to handle each user's request upon its arrival. Table 2 shows the average running time of *SWMOA* to handle a user's request when the total number of users in the system differs. The results indicate that our online algorithm can be executed on common servers fast enough for practical usage.

TABLE 2
Running time of *SWMOA*

N	Running Time per User (milliseconds)
10000	0.698
20000	0.6675
30000	0.6433
100000	0.6753
200000	0.7051
300000	0.6857
1000000	0.4327

8 Conclusion

This paper presents efficient and competitive online auction mechanisms for on-demand provisioning and pricing of virtual clusters, taking both computational resources and communication resources into consideration. In the case of linear resource costs, we first design an online social welfare maximizing auction, *SWMOA*, which sets a carefully designed virtual unit cost for each type of resources on the go, and decides winners and bid-independent payments based on the total resource cost. We then design a revenue maximizing online auction, *PRMOA*, which runs *SWMOA* as a basis, and boosts payments following a carefully designed random distribution, to pursue higher revenue. The mechanisms are truthful, individually rational, time-efficient, and guarantee a $(2\log\mu + 1)$ competitive ratio in social welfare and a $O(\log\mu)$ competitive ratio in provider revenue in expectation, respectively. We further extend our auction design to the non-linear operational cost cases, and show similar competitive ratios can be achieved. Our extensive simulation studies validate our theoretical analysis, and show good performance of our mechanisms in various scenarios.

References

[1] "Case Studies and Customer Success Stories, Powered by the AWS Cloud," https://aws.amazon.com/solutions/case-studies/.

[2] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a Data Center Network Virtualization Architecture with Bandwidth Guarantees," in *Proc. of ACM CoNEXT*, 2010.

[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards Predictable Datacenter Networks," in *ACM SIGCOMM Computer Communication Review*, 2011.

[4] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks," in *WIOV*, 2011.

[5] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: a Cloud Networking Platform for Enterprise Applications," in *Proc. of ACM SOCC*, 2011.

[6] "Amazon Virtual Private Cloud (VPC)," https://aws.amazon.com/vpc/.

[7] "Comcast Case Study," https://aws.amazon.com/solutions/case-studies/comcast/.

[8] "Amazon Virtual Cluster," http://docs.aws.amazon.com/redshift/latest/mgmt/managing-clusters-vpc.html.

[9] Q. Wang, K. Ren, and X. Meng, "When Cloud meets eBay: Towards Effective Pricing for Cloud Computing," in *Proc. of IEEE INFOCOM*, 2012.

[10] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," in *Proc. of ACM SIGMETRICS*, 2014.

[11] X. Li, J. Wu, S. Tang, and S. Lu, "Let's Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers," in *Proc. of IEEE INFOCOM*, 2014.

[12] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable Virtual Data Center Embedding in Clouds," in *Proc. of IEEE INFOCOM*, 2014.

[13] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "The Price Is Right: Towards Location-Independent Costs in Datacenters," in *Proc. of ACM HotNet*, 2011.

[14] F. Esposito and I. Matta, "A Decomposition-based Architecture for Distributed Virtual Network Embedding," *ACM SIGCOMM DCC Workshop*, 2014.

[15] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in *Proc. of IEEE INFOCOM*, 2009.

[16] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-Resource Packing for Cluster Schedulers," in *Proc. of ACM SIGCOMM*, 2014.

[17] G. Even and M. Medina, "Online Multi-Commodity Flow with High Demands," in *Approximation and Online Algorithms*, 2013.

[18] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and Deterministic Embeddings of Virtual Networks," in *Distributed Computing and Networking*, 2012, pp. 106–121.

[19] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang, "Virtual Network Embedding for Evolving Networks," in *Proc. of IEEE GLOBECOM*, 2010.

[20] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth Revelation in Approximately Efficient Combinatorial Auctions," *Journal of the ACM (JACM)*, 2002.

[21] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, 1961.

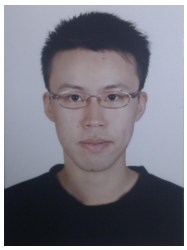
[22] Q. Sun, Q. Wang, K. Ren, and X. Jia, "Fair Pricing in the Sky: Truthful Frequency Allocation with Dynamic Spectrum Supply," in *Proc. of IEEE ICNP*, 2014.

[23] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "Incentive-Compatible Online Mechanism For Resource Provisioning and Allocation in Cloud," in *Proc. of IEEE CLOUD*, 2014.

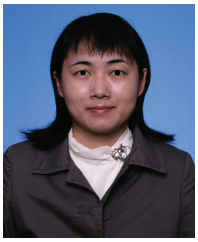
[24] R. Lavi and C. Swamy, "Truthful and near-Optimal Mechanism Design via Linear Programming," in *Proc. of FOCS*, 2005.

[25] L. Zhang, C. Wu, and Z. Li, "Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach," in *Proc. of IEEE INFOCOM*, 2014.

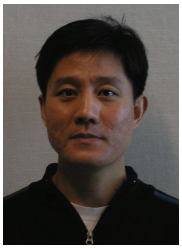
[26] X. Zhang, C. Wu, Z. Li, and F. C. Lau, "A Truthful (1-ϵ)-Optimal Mechanism for On-demand Cloud Resource Provisioning," in *Proc. of IEEE INFOCOM*, 2015.
[27] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau, "Online Auctions in IaaS Clouds: Welfare and Profit Maximization with Server Costs," in *Proc. of ACM SIGMETRICS*, 2015.
[28] H. Fu, Z. Li, C. Wu, and X. Chu, "Core-Selecting Auctions for Dynamically Allocating Heterogeneous VMs in Cloud Computing," in *Proc. of IEEE CLOUD*, 2014.
[29] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a Globally-Deployed Software Defined WAN," in *Proc. of ACM SIGCOMM*, 2013.
[30] A. Madry, "Reducing Truth-Telling Online Mechanisms to Online Optimization," in *Proc. of ACM STOC*, 2003.
[31] N. Buchbinder and J. Naor, "The Design of Competitive Online Algorithms via a Primal: Dual Approach," *Foundations and Trends in Theoretical Computer Science*, 2009.
[32] A. Madry, "Faster Approximation Schemes for Fractional Multicommodity Flow Problems via Dynamic Graph Algorithms," in *Proc. of ACM STOC*, 2010.
[33] "Xen DVFS," http://lists.xen.org/archives/html/xen-devel/2009-09/msg00585.html.
[34] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware Provisioning of Virtual Machines for Real-time Cloud Services," *Concurrency and Computation: Practice and Experience archive*, vol. 23, no. 13, pp. 1491–1505, 2011.
[35] N. R. Devanur and Z. Huang, "Primal Dual Gives Almost Optimal Energy Efficient Online Algorithms," in *Proc. of ACM SODA*, 2014.
[36] N. Buchbinder and J. Naor, "Online Primal-Dual Algorithms for Covering and Packing Problems," in *Algorithms-ESA*, 2005.
[37] "Google Cluster Data," https://github.com/google/cluster-data.
[38] L. Bo, T. Huang, Z.-K. Wang, J.-Y. Chen, Y.-J. Liu, and L. Jiang, "Adaptive Scheme Based on Status Feedback for Virtual Network Mapping," *The Journal of China Universities of Posts and Telecommunications*, 2011.
[39] A. V. Goldberg, J. D. Hartline, and A. Wright, "Competitive Auctions and Digital Goods," in *Proc. of ACM SODA*, 2001.
[40] A. V. Goldberg, J. D. Hartline, A. R. Karlin, M. Saks, and A. Wright, "Competitive Auctions," *Elsevier Games and Economic Behavior*, no. 55, pp. 242–269, 2006.



Weijie Shi received his B.E. degree in 2012, from Department of Computer Science and Technology, Tsinghua University, China. He is currently a PhD candidate in the Department of Computer Science, the University of Hong Kong. His research interests include cloud computing and mechanism design.



Chuan Wu received her B.E. and M.E. degrees in 2000 and 2002 from Department of Computer Science and Technology, Tsinghua University, China, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. She is currently an associate professor in the Department of Computer Science, The University of Hong Kong, China. Her research interests include cloud computing and online/mobile social network.



Zongpeng Li received his B.E. degree in Computer Science and Technology from Tsinghua University (Beijing) in 1999, his M.S. degree in Computer Science from University of Toronto in 2001, and his Ph.D. degree in Electrical and Computer Engineering from University of Toronto in 2005. He is currently a professor in the Department of Computer Science in the University of Calgary. His research interests are in computer networks, particularly in network optimization, multicast algorithm design, network game theory and network coding.