A Truthful and Near-optimal Mechanism for Colocation Emergency Demand Response

Jianhai Chen, Deshi Ye*(✉), Zhenguang Liu, Shouling Ji, Qinming He, *Member, IEEE* and Yang Xiang *Fellow, IEEE*

Abstract—Demand response (DR) has been widely adopted as a strategic plan of the electricity market in maintaining power grid reliability, sustainability, and stability. In a typical emergency DR (EDR) that arises in colocation data centers, participating tenants can reduce their power consumption when the supply of electricity is a shortage and be rewarded with financial compensation. In this paper, we study a mechanism design problem of motivating tenants for colocation EDR (MEDR). To solve the MEDR problem, we present a truthful Fully Polynomial-Time Approximation Scheme (FPTAS) which is theoretically proved deterministic, truthful and near-optimal, and can be approximated within $1 + \epsilon$ for any given $\epsilon > 0$, while the running time is in the polynomial of the number of tenants n and $1/\epsilon$. To speed up the calculation of the payments, we further study the Vickrey-Clarke-Groves (VCG) based mechanism. Moreover, we build a MEDR auction system (MEDRAS) and implement all mechanism algorithms for a colocation data center. Comprehensive and detailed experiments have been implemented to validate the efficiency of our proposed mechanisms.

Index Terms—Emergency Demand Response; Mechanism Design; Colocation Datacenter; Auction; FPTAS; Smart Grid

1 Introduction

Demand response (DR) is not only a technique for regulating energy consumption over time but also one of the major reliability impacts for smart grids [1]. DR is a vital means of demand-side management (DSM) which refers to the way that countries use policy measures to guide power users to reduce electricity at peak time, use electricity in a low valley, improve power supply efficiency and optimize the usage of electricity. A typical DR is the emergency demand response (EDR) for the case of emergency demand for using electricity (for example, earthquake or extremely bad weather). When EDR happens, the demand side management (DSM) will hold an auction to start an immediate response or incentive mechanism of the users' report electricity and declare the amount of electricity and the price of electricity. On the consumer side, DR is commonly utilized as a powerful tool for employing the flexibility of using electricity in response to supply-demand conditions [2]. When electricity price rises or the system reliability is threatened, the electricity supplier will firstly deliver the notice of direct compensation, of inductively reducing power load or signal of power price rise to the power consumers. Then consumers will change their intrinsic power using the mode to meet the demand of electricity supply, reducing or passing a special period of power load, ensuring the stability of the power grid and restraining the rise of electricity price.

It is worth noting that the colocation of data centers is quite popular now with the rapid development and application of cloud computing. According to the website [1], there are 3,775 colocation data centers from 112 countries. A colocation is the third-party leased placement that provides physical homes for many data centers and provides lots of services such as the fast Internet, stable power supply and cooling. The running of large-scale cloud application services requires sufficient resources and power supply to ensure its reliability especially in the case of an emergency. Many demand response programs have been used in colocation datacenters for improving the efficiency of power grids [3], [4], with attempts to adjust the demand for power instead of adjusting the supply. Though the colocation of datacenters provided a nice solution for those enterprise tenants, it consumed huge electricity. As pointed out in [5], 91 billion kilowatt-hours of electricity was consumed in the U.S. in 2013, and it emitted around 97 million metric tons of carbon pollution in that year. On the other hand, it is possible to close or migrate some tasks in a large datacenter such that some computing servers can be shut-down. This makes possible for datacenters to be a participant in demand response. In case of emergency or reaching the capacity of a grid, colocation datacenter requires implementing the EDR.

To realize the EDR, the incentive auction mechanism has been usually used to motivate power users to participate actively in responding to DR activities [6], [7], [8]. In an EDR setting, given demand for power size and a group of agent bids with bidding size and cost, the auction mechanism needs to determine a set of bids that are winners with a minimized social cost. The overall process of auction requires the truthfulness of bids and the efficiency of the winning bids' decisions. However, there are still few ef-

- *J. Chen, D. Ye, S. Ji, Q. He are with the Institute of Cyberspace Research and College of Computer Science, Zhejiang University, Hangzhou, China, 310027.*
  E-mail: chenjh919, yedeshi, sji, hqm@zju.edu.cn.
- *Z. Liu was with the National University of Singapore, Singapore. Email: liuzhenguang2008@gmail.com.*
- *S. Ji is also with Alibaba-Zhejiang University Joint Institute of Frontier Technologies.*
- *Y. Xiang is with the Swinburne University of Technology, Victoria 3122 Australia. Email: yxiang@swinburne.edu.au.*
- *D. Ye is the corresponding author.*

1. Data collected from http://www.datacentermap.com/ on Jan 7, 2016.

fective mechanisms and systems to support high-efficiency power management in the current power grid and many colocation datacenters, leading to high power cost and low efficiency [9]. For example, to reduce peak demand in a power grid, the DR is usually implemented manually by sending signals to large consumers, such as datacenters. To the best of our knowledge, Zhang et al. [2] were the first ones to study approximated truthful mechanisms for the MEDR problem. They provided a 2-approximated mechanism with truthful in expectation. The key significance of their work is to present a 2-approximated algorithm, and then convert the approximation algorithm into a mechanism with truthful in expectation, while keeping the approximation ratio of 2. The framework of their work is based on a convex decomposition technique [10], which transfers an approximation algorithm into a truthful randomized mechanism.

In this work, we target to study a **M**echanism design problem of **EDR** that arises in colocation data centers, denoted by the MEDR problem. Our task is to propose deterministic mechanisms and an auction system prototype as a holistic solution to the MEDR problem. Our work includes two main challenges. The first challenge is to design a highly efficient and truthful mechanism. The MEDR optimization generalizes a min-knapsack problem and a max-knapsack problem which are both NP-hard. This motivates us to study approximate methods, and hence we propose a truthful FPTAS (Fully Polynomial-Time Approximation Scheme) mechanism. Though our proposed FPTAS mechanism runs in polynomial time, the running time, especially, the running time for the payment is still large. Thus we consider another approach to get a tradeoff between truthfulness and computational efficiency. It is worth noting that we could apply the VCG (Vickrey-Clarke-Groves) based mechanism, a deterministic truthful mechanism to the MEDR problem [11], [12], [13]. But, to be truthful, it requires an optimal allocation algorithm in the VCG-based mechanism, and approximation algorithms for MEDR might not be truthful. Our second mechanism is called a VCG-based mechanism that could transfer an approximation algorithm into a mechanism while not sacrificing the truthfulness too much. We prove that our proposed FPTAS allocation algorithm with VCG-based payment is a ε-truthful mechanism for any given $\varepsilon > 0$, and the running time of this mechanism will outperform than the FPTAS-mechanism when the number of tenants is not in a large-scale.

The second challenge is the design and implementation of high efficient mechanism algorithms and auction system. We consider a reverse auction scenario to realize the MEDR mechanism. In order to facilitate building a high efficient auction platform for EDR in colocation data centers, we integrate our MEDR mechanism into a routine that can be independently run for practical auctions. However, to implement a realistic auction system platform we might further consider the overall function requirement of the auction and how to inter-communicate with the existing online working datacenter systems. On account of the complexity of datacenter practical EDR auction business logic, in this work, we focus on the integration of MEDR mechanism algorithms and the design of a system prototype. Actually, the mechanism in the auction includes two classes, namely winner decision, and winner payment algorithms. The winner decision considers the dynamic programming with a table that requires a large memory to store the pilot process data in the auction if the number of bids is very big. Due to the limit resource capacity [14], [15] of a server machine is running the mechanism program, the scalability and efficiency are required to be considered.

In technique, our work differs from previous in two folds. Firstly, we show that combining dynamic programming and the technique of rounding for monotone FPTAS [16] can lead to a deterministic truthful mechanism for our problem. Secondly, our designed FPTAS can also transform into a computational efficiency ε-truthful mechanism.

In summary, the main contributions of this paper are as follows:

- We propose a deterministic truthful mechanism with FPTAS approximated and theoretically prove it to be truthful and can be approximated within $1 + \varepsilon$ for any given $\varepsilon > 0$, the running time of our mechanism is in polynomial of n and $1/\varepsilon$, where n is the number of tenants in the datacenter. We further investigate VCG-based mechanisms. We show that our provided FPTAS allocation algorithm can be turned into a ε-truthful mechanism. The VCG-based mechanism will speed up the computation of the payments when the number of tenants is small.
- We provide a holistic solution for the MEDR problem, with a highly efficient MEDRAS prototype with the integration of all our truthful FPTAS mechanism algorithms. The MEDRAS is a routine that can be run independently and easily used in practical datacenter systems. We develop a simulation test tool, design a large quantity of test cases with simulation datasets generated by real datasets and perform an overall performance evaluation to demonstrate the effectiveness of our truthful FPTAS mechanism. The tool is opensourced in github (URL: https://github.com/ZJU-INCAS/medras).

The rest of this paper is organized as follows. In Section 2, we state and formalize MEDR, a mechanism design problem for EDR. In Section 3, we present an FPTAS mechanism to solve the MEDR problem. In Section 4, we depict the VCG-based mechanism. In Section 5, we propose a MEDR auction system and depict its design and implementation. In Section 6, extensive simulation experiments are conducted to evaluate the performance. The related work is presented in Section 7, and concluding remarks are given in Section 8.

2 Problem Statement

In this section we state the MEDR problem that arises in colocation datacenter Emergency Demand Response (EDR).

There are n tenants in a colocation datacenter. Each tenant $i \in \{1, 2, \ldots, n\}$ subscribes a certain amount of power supply from the colocation operator. In the event of EDR, the colocation operator is required to reduce W amount of energy, such as W kWh electric power. Given power-based contracts, tenants may not have the incentive to participate in EDR unless they are awarded. Even if some tenants are interested in EDR, their reduction may not reach the reduction target W. In case of not reaching the target, the colocation operator can use backup energy storage (BES) to fulfill the shortage of the EDR target. Let z be the amount of grid-power demand reduction due to the usage of BES, and α be the cost of BES usage per kWh. In general, the cost of generating electricity from a backup power is much higher than the cost of an ordinary power supply. The power from BES usually is produced by pre-charged batteries and diesel generators, which are very expensive and/or environment unfriendly [17]. We could assume that BES can be large enough. Because BES needs to cover the target W if there is no participant in.

Each tenant i submits a bid with two parameters (s_i, b_i), where s_i is the amount of planned energy reduction and b_i is the claimed cost due to such a reduction. However, each tenant i has its own true type (e_i, c_i), where c_i is the cost due to a reduction of e_i energy. The value e_i and c_i are only known to the tenant i. The cost c_i is an integer because it represents the money. The cost c_i comes from various sources. One case is that one has to pay for the delay of executing a task due to load reduction. It could also be the cost if one moves the task to some other computing platform, such as a rented VM. Thus c_i represents a general cost due to such load reduction.

Moreover, each tenant is *single-minded* [2], [16], [18] such that each tenant is restricted to one single bid. The single-minded bidders are only interested to get a specified scalar value if they get a whole amount of energy reduction and get zero value otherwise. The single-minded case indicates that the allocation algorithms shall output binary integral solutions, *i.e.*, either tenant i will be accepted or not.

Every tenant has the freedom to choose participation in this EDR or not. If a tenant is not willing to participate in this EDR, we can suppose its bid is $(0, 0)$. Let $B = \{(s_1, b_1), \ldots, (s_n, b_n)\}$ be the set of bids by the n tenants. Based on this bidding B, the colocation operator will pay money $P_i(B)$ to each tenant i to encourage their participation in this EDR. Let $U_i(B) = P_i(B) - c_i$ be the utility of tenant i according to the biddings of B. Clearly each tenant i attempts to maximize his/her utility. According to [2], the power consumption in colocation data center consists of both the energy consumption of tenants and also the consumption of management such as cooling. There is a ratio called Power Usage Effectiveness (PUE) γ between the total energy consumption to the energy consumed by tenants, which typically ranges from 1.1 to 2.0 [19].

A tenant is a *winner* if her/his bidding is successful. Let N be the set of winners. To meet the energy reduction target W, we require that $z + \gamma \sum_{i \in N} s_i \geq W$. The social cost of the colocation operator is $\alpha z + \sum_{i \in N} P_i(B)$. The social cost of tenants is $\sum_{i \in N}(c_i - P_i(B))$. Thus, the total social cost is equivalent to aggregate tenant cost due to energy reduction plus the operator's cost for using BES, i.e., $\alpha z + \sum_{i \in N} c_i$. The goal of the mechanism design is to minimize the total social cost, meanwhile, no tenant can benefit by proposing false bidding. The optimization version of this problem can be formulated as an integer programming. Let $x_i = 1$ if tenant i is a winner, i.e. $i \in N$, otherwise $x_i = 0$.

$$min \quad \alpha z + \textstyle\sum_{i=1}^{n} x_i c_i \tag{1}$$

TABLE 1
Summary of Notations

Notation	Description
n	total number of agents
e_i	energy reduction by agent i
c_i	the cost incurred of agent i by energy reduction
s_i	reported energy reduction by agent i
b_i	reported cost of agent i
W	energy reduction target
z	energy provided by BES
x_i	binary variable indicates whether agent i win or not
B_{-i}	the bids except tenant i
$P_i(B)$	the money paid to the agent i according to the bids B
$U_i(B)$	the utility of agent i according to the bids B
α	the cost of BES usage per kWh
γ	PUE(ratio of the total energy consumption to the energy consumed by tenants)

subject to

$$z + \gamma \textstyle\sum_{i \in N} s_i \geq W \tag{2}$$

$$x_i = \{1, 0\}, \forall i \in \{1, 2, \ldots, n\} \tag{3}$$

Table 1 summarizes the key notations in the paper.

The studied MEDR problem is closely related to knapsack auction problems. According to objective functions, we define two types of mechanism design for knapsack problems. One is the *max-knapsack* problem, in which each agent has a private valuation for having his/her objective in the knapsack. The problem is to find an allocation of the agents without exceeding the capacity of the knapsack as so to maximize the sum of each agent's value. Another one is the *min-knapsack* problem, in which each agent has a private cost for having his/her item in the knapsack. The problem is to find an allocation to cover the knapsack, while the sum of the agents' cost is minimized.

For any instance I, we define by $C(\mathcal{M}(I))$ the *social cost* of the mechanism $\mathcal{M}$, which is the total costs of tenants plus the operator's cost for using BES. A mechanism $\mathcal{M}$ is said to be ρ-approximated if $C(\mathcal{M}(I)) \leq \rho \cdot C(OPT(I))$, where OPT is an optimal algorithm.

Let $B_{-i} = \{B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_n\}$ be the bids except tenant i's bid.

Definition 1. (Truthfulness): A mechanism $\mathcal{M}$ consisting of an allocation function $\mathcal{A}$ and a payment function $\mathcal{P}$ is truthful (or strategy-proof) if for every tenant i with the true cost c_i cannot increase his/her utility by declaring any other bid (s_i, b_i) regardless of every bidding of other agents B_{-i}, i.e., it satisfies

$$U_i((e_i, c_i), B_{-i}) \geq U_i((s_i, b_i), B_{-i}).$$

This definition implies that truthful reporting is a dominant strategy for every tenant.

Definition 2. (Individual rationality): A mechanism $\mathcal{M}$ is said to be individual rationality if every agent always obtains non-negative utility with bidding of the true cost, i.e., $U_i((e_i, c_i), B_{-i}) \geq 0$ for any i and any B_{-i}.

Remark: As we know, demand response usually is due to the lack of energy, hence, in our work, we set the target W as a low bound. In some scenarios that the reduced energy shall be exactly the target W to maintain the balance of energy. All our algorithms can be easily adapted to this case. We can modify the line 6 in Algorithm 1 (line 10 in Algorithm 2) such that $y(i, c)$ equals infinity. That means we return the solution with $\gamma A(i, c) \leq W$. In this sense, in this work, we only deal with the general case that the target W is a lower bound.

3 FPTAS TRUTHFUL MECHANISM

In this section, we describe an approximated FPTAS truthful mechanism to solve the MEDR problem.

3.1 Dynamic Programming

Our dynamic programming requires to solve the min-knapsack problem as a subroutine. The min-knapsack problem consists of finding a subset of items, where each item i has a cost c_i and a size s_i, with the minimum cost such that the sum of their sizes is at least as large as a specified capacity. Based on the idea of the max-knapsack problem [20], Tauhidul [21] gave a dynamic programming for the min-knapsack problem. We adopt this dynamic programming in (4) [21] as a subroutine in the following.

Let $S(i,c)$ denote a subset of $\{1,\ldots,i\}$ whose cost is exactly c and whose total size is maximized. Let $A(i,c)$ be the size of $S(i,c)$ ($A(i,c)=0$ if no such set exists). The recursive formula of the dynamic programming is given in (4). In this formula $A(i,c)$ gives a tabular of an optimal value for each subproblem (i,c).

$$A(i,c) = \begin{cases} \max\{A(i-1,c), s_i + A(i-1,c-c_i)\}, \\ \quad if\ c_i \le c \\ A(i-1,c), \quad otherwise \end{cases} \tag{4}$$

In the following, we design dynamic programming based on the recursive function (4). The Algorithm 1 (Algorithm $DOPT(I)$) gives the details of the dynamic programming for the MEDR problem.

Algorithm 1: Algorithm $DOPT(I)$: Dynamic Programming for MEDR

```
Input: The set of tenants $I$, and demand capacity $W$.
1 Run the dynamic programming based on the
    formula (4) for the input $I$, and obtain $A(i,c)$ for
    each $(i,c)$, where $1 \le i \le n$ and $0 \le c \le \sum_i c_i$;
2 for each $(i,c)$ do
3     if $\gamma A(i,c) < W$ then
4         $y(i,c) = \alpha(W - \gamma A(i,c)) + c$
5     else
6         $y(i,c) = c$
Output: Return $\min_{(i,c)} y(i,c)$
```

Theorem 1. The dynamic programming $DOPT(I)$ produces an optimal solution for any instance of tenants I with demand request W, and unit cost of BES α. The running time is pseudo-polynomial, which is $O(n^2 c_{\max})$, where $c_{\max}$ is the largest cost.

Proof. Any optimal solution consists of two parts, one is covered by BES, and another is covered by items from I. Let p, q be the cost due to tenants I and BES, respectively. Let $c_{\max} = \max_i c_i$ be the largest cost among all tenants.

In the dynamic programming we iterate all possible (i,c), where $c \in C = \{0,1,2,\ldots,nc_{\max}\}$. Any cost due to tenants I is in C, hence $p \in C$. Note that the dynamic programming $A(i,p)$ provides the maximal size whose cost is exactly p. If $\gamma A(n,p)$ is less than W, and we require at least $W - \gamma A(n,p)$ BES to cover the knapsack in the optimal. Therefore, $p + q \ge \alpha(W - \gamma A(n,p)) + p$. If $\gamma A(n,p) \ge W$, then $q = 0$. These two cases are both covered in the dynamic programming, which implies that the dynamic programming outputs an optimal solution.

The running time of dynamic programming is $O(n^2 c_{\max})$, since $i \le n$, and $c \le nc_{\max}$, and the running time is bounded by the iterative function of (i,c). □

3.2 Monotone FPTAS Mechanism

Motivated by the truthful mechanism for max-knapsack problem [16], we propose a deterministic truthful mechanism. To keep the truthful property, the idea of our mechanism is to give a monotone algorithm. To obtain an FPTAS, we need to design a monotone algorithm whose approximation ratio is arbitrarily close to 1. The detailed algorithm is given in Algorithm 3, which iteratively calls a subroutine Algorithm 2 (Algorithm $A_r(k,I)$). The motivation of Algorithm 2 is to keep the truthfulness, in which the rounding of each item is independent of the bidding of all tenants.

Algorithm 2: Algorithm $A_r(k,I)$

```
Input: Given parameter $k$, and the instance
        $I = (s_1,c_1),\ldots,(s_n,c_n)$.
1 Let $a_k = \frac{\varepsilon 2^k}{n+1}$;
2 Let $T(k)$ be the subset of items with cost at most of
    $2^k$; We construct a new instance $I'$ based on $T(k)$,
    which is identical to $T(k)$, but the cost of each item
    $c'$ is given as below.
3 for $i \in T(k)$ do
4     $c'_i = \lfloor \frac{c_i}{a_k} \rfloor$
5 Run the dynamic programming $DOPT(I')$for the
    items in $T(k)$ with cost $c'_i$, and obtain $A(i,c')$;
6 for each $(i,c')$ do
7     if $\gamma A(i,c') < W$ then
8         $y(i,c') = \lfloor \frac{\alpha(W-\gamma A(i,c'))}{a_k} \rfloor + c'$
9     else
10        $y(i,c') = c'$
Output: Return $\min_{(i,c')} y(i,c')$
```

Algorithm 3: Monotone FPTAS A_{FPTAS}

```
Input: Given $\varepsilon > 0$, and the instance $I$.
1 Let $best \leftarrow \infty$, and $c_{\max} = \max_i c_i$.
2 for $k \leftarrow 1$ to $\log c_{\max}$ do
3     $S'(k) \leftarrow A_r(k,I)$; /* call Algorithm 2 (Algorithm
        $A_r(k,I)$) */
4     if $S'(k) < best$ then
5         $best \leftarrow S'(k)$
6         $\bar{S} \leftarrow$ the subset items that contained in the
            solution of $S'(k)$
Output: $\bar{S}$, and use BES $W - \gamma \sum_{i \in \bar{S}} s_i$
```

Lemma 1. For any $\varepsilon > 0$, Algorithm A_{FPTAS} has approximation ratio of $1+\varepsilon$, and its running time is polynomial in $1/\varepsilon, n, \log c_{max}$.

Proof. Let c_q be the largest cost among the items in an optimal algorithm to cover the knapsack. Define k^*, such that

$$2^{k^*-1} < c_q \leq 2^{k^*}.$$

Denote O^* to be the subset of items in the optimal solution. Let y^* be the size BES used in the optimal solution. Let $O^*(R) = O^* \bigcup \{R\}$, where R is a virtual item with size y^* and cost αy^*. Let OPT be the cost of the optimal solution. We have $OPT \geq c_q$.

Note that in $T(k^*)$ as denoted in the algorithm $A_r(k, I)$, we have $O^* \subseteq T(k^*)$. Let $\bar{S}$ be the subset of items returned by the algorithm $A_r(k, I)$ with k^* as the parameter, and let $(\bar{i}, \bar{c})$ be the pair of values that reaches the minimum of $A_r(k, I)$.

Let O' be the subset of items with costs rounded by 2^{k^*} from O^*. Let R' be a virtual item with size y^*/a_{k^*}.

Let ALG be the final cost incurred by the algorithm A_{FPTAS}, we can use the following inequalities to approximate the cost by the algorithm with the optimal solution.

$$\begin{aligned}
ALG &= \sum_{i\in\bar{S}} c_i + \max(\alpha(W - \gamma A(\bar{i}, \bar{c})), 0) \\
&\leq \sum_{i\in\bar{S}} c_i + \max(\lfloor \frac{\alpha(W-\gamma A(\bar{i},\bar{c}))}{a_k^*} \rfloor, 0) a_{k^*} + a_{k^*} \\
&\leq \sum_{i\in\bar{S}} (c'_i \cdot a_{k^*} + a_{k^*}) + \\
&\quad \max(\lfloor \frac{\alpha(W-\gamma A(\bar{i},\bar{c}))}{a_k^*} \rfloor, 0) \cdot a_{k^*} + a_{k^*} \\
&\leq \sum_{i\in\bar{S}} c'_i \cdot a_{k^*} + \max(\lfloor \frac{\alpha(W-\gamma A(\bar{i},\bar{c}))}{a_k^*} \rfloor, 0) \cdot a_{k^*} \\
&\quad +(n+1)a_{k^*} \\
&\leq \sum_{i\in O'\bigcup\{R'\}} c'_i \cdot a_{k^*} + (n+1)a_{k^*} \\
&\leq \sum_{i\in O^*\bigcup\{R\}} c_i + (n+1)a_{k^*} \\
&\leq OPT + \varepsilon 2^{k^*} \\
&\leq (1+2\varepsilon)OPT.
\end{aligned}$$

The running time is $poly(1/\varepsilon, n, \log c_{\max})$. In algorithm $A_r(k, I)$, the largest cost of $T(k)$ is 2^k, the running time of dynamic programming is $O(n^3/\varepsilon)$. The total running time of A_{FPTAS} is $O(\frac{1}{\varepsilon}n^3 \log c_{max})$. □

3.2.1 Monotone

A declaration $B'_i = (s'_i, b'_i)$ is said to be a *higher declaration* than the bidding $B_i = (s_i, b_i)$ if $s'_i \geq s_i$ and $b'_i \leq b_i$, i.e. $B_i \leq B'_i$. A bid (s_i, b_i) is said to be a *winner declaration* if this item is selected in the knapsack.

Definition 3. (Monotone) We say that an algorithm A is monotone if, for any bidder (s_i, b_i) is a winning declaration then any higher declaration also wins.

To design a monotone algorithm, some times we need to choose the output with minimum or maximum value among several monotone algorithms. The monotone property may not be valid anymore for such selecting, we need another property named bitonic to deal with this challenge. Bitonic was introduced by Mu'Alem and Nisan [22] for maximum problems, such as multi-unit auction, and it was generalized by Briest, Krysta, and Vöcking [16].

In this work, we apply the technique of bitonic for minimum problems.

Definition 4. (Bitonic) Given a function $f : \mathcal{A}^n \rightarrow$, a monotone algorithm A is bitonic with respect to the function f if for any agent i, the following hold:

1) If $i \in A(B)$, then $f(A(B_i, B_{-i})) \geq f(A(B'_i, B_{-i}))$ for any $B_i \leq B'_i$.
2) If $i \notin A(B)$, then $f(A(B_i, B_{-i})) \geq f(A(B'_i, B_{-i}))$ for any $B'_i \leq B_i$.

Intuitively, a monotone algorithm A is bitonic with respect to f if f is a monotone non-decreasing function of each agent's valuation while she is not selected for the solution, but becomes monotone non-increasing after she is selected for the solution. In this work, the function f is the objective function, *i.e.*, social welfare. The bitonic is indeed required to guarantee the monotone for compositions of algorithms.

Algorithm 4: $MIN(A_1, A_2)$ Operator

```
  Input: Bidding $B$
1 Run the algorithm $A_1$ and $A_2$;
2 Let $sw_{A_1}(B)$ and $sw_{A_2}(B)$ be the social welfare of
    Algorithm $A_1$ and $A_2$, respectively.
3 if $sw_{A_1}(B) \leq sw_{A_2}(B)$ then
4     return $A_1(B)$;
5 else
6     return $A_2(B)$.
```

Lemma 2. Let A_1 and A_2 be two monotone bitonic allocation algorithms. Then, $M = MIN(A_1, A_2)$ is a monotone bitonic allocation algorithm.

Proof. This can be easily extended from the proof of the Theorem 2 in [22], which was designed for the MAX operator. □

Lemma 3. Algorithm $A_r(k, I)$ is monotone and bitonic with respect to the objective function.

Proof. Algorithm $A_r(k, I)$ returns an optimal solution, if an agent reports a higher bidder, then the optimal algorithm will accept this item too. Suppose an agent i was not selected, and any lower declaration B'_i, if this item was accepted then the objective function shall be smaller, otherwise, the objective remains, and hence the objective function is non-increasing for any lower bidders. Thus the property of bitonic follows. □

Lemma 4. Algorithm A_{FPTAS} is monotone and bitonic with respect to the objective function.

Proof. The lemma follows immediately according to Lemma 2 and Lemma 3. □

3.2.2 Payment

In this section, we will provide a payment scheme for our problem. Lehmann et al. [18] provided a sufficient condition for a truthful mechanism for single-minded combinatorial auctions. The payment in [18] is called *critical payment*, which was also used for MAX-knapsack problem [16]. Our payment scheme was adapted from [16], [18], and the critical payment was defined similarly as below.

Definition 5. (Critical payment) Let algorithm A be a monotone algorithm, if we fix the declaration B_{-i}, and then for any agent i and fixed bidding s_i, there exists a unique cost θ_i^A, called *critical payment*, such that $\forall b_i \leq \theta_i^A$, b_i is a winning declaration, and $\forall b_i > \theta_i^A$ is a losing declaration.

To calculate the critical value for any agent j, we fix the other agents' bids and then use a binary search on the interval $[b_j, \max_j b_j]$ and repeatedly run the allocation algorithm A to check whether the agent j is selected.

Definition 6. The payment p^A associated with the monotone allocation algorithm A that is based on the critical value is defined by $p_j^A = \theta_j^A$ if agent j wins with allocation $Alloc_i(B) = s_i$, and $p_j^A = 0$ otherwise.

A mechanism $M_A = (A, p^A)$ is *normalized* , if its payment p^A is defined as in Definition 6, i.e. agents that are not selected pay 0. We say that algorithm A is *exact* if $Alloc_i(B) = s_i$ or $Alloc_i(B) = \emptyset$ for each declaration (s_i, b_i).

In this work, we only consider a limited type of agent called *single-minded*, the cost function ∞ if $Alloc_i(B) > s_i$ and c_i otherwise. That will force each agent does not overbid his/her size if allocation algorithm is an exact algorithm.

Theorem 2. [16] Let A be an exact and monotone algorithm for some minimization problems and single-minded agents. Then mechanism $M_A = (A, p^A)$ is truthful and individually rational.

Proof. In [16], they gave detailed proof for utilitarian problems, and thus it holds for the MAX-knapsack problem. Moreover, in their paper, it was shown that the proof is valid for minimization problems, such as the reverse single-minded multi-unit auction problem, which is equivalent to the minimum knapsack problem. For completeness, we restate the main idea as follows. Fix B_{-i}. Considering a bidding (e_i, b_i) and truthful bidding (e_i, c_i), respectively.

1) Both bidds of agent i are winners. The payments θ_i for both bidders are the same because the critical payment returns a maximum value of winner declarations. Hence, the utility is the same and non-negative.
2) Bid (e_i, b_i) wins and bid (e_i, c_i) loses. In this case $c_i > \theta_i \geq b_i$, the utility of bid (e_i, b_i) is negative, while the utility of truthful bid (e_i, c_i) gets utility zero because of losing. In this case the agent i will bid truthful and the utility is zero.
3) Bid (e_i, b_i) loses and bid (e_i, c_i) wins. We have $\theta_i \geq c_i$, and they have the same utility $\theta_i - c_i$, which is non-negative.

From the critital payment and truthfulness, we get that the mechanism is individual rationality. Because the utility of any agent i is non-negative according to the above three cases. □

Algorithm 5: Algorithm $P^{\mathcal{A}}(B)$

```
Input: The bidding $B$ of all tenants, and the
       allocation algorithm $\mathcal{A}$
1  for $i \leftarrow 1$ to $n$ do
2      Let $z_i = 1$ if the $i$th item is selected in the
       knapsack by the allocation problem $\mathcal{A}(B)$, and 0
       otherwise.
3      Let $h = \alpha\gamma s_i$ and $l = b_i$.
4      while $h - l \geq 1$ do
5          $b_i' = (h + l)/2$;
6          $z_i = \mathcal{A}(B_{-i}, (s_i, b_i'))$;
7          if $z_i == 1$ then
8              $l = b_i'$
9          else
10             $h = b_i'$
11     $P_i(B) \leftarrow l$.
Output: The payment $P_i(B)$ for each agent $i$.
```

Theorem 3. The mechanism $M_{A_{FPTAS}} = (A_{FPTAS}, p^{A_{FPTAS}})$ is truthful and it is an FPTAS mechanism, i.e. its approximation ratio is $1 + \varepsilon$ for any given $\varepsilon > 0$, and the total running time of the mechanism is polynomial in $1/\varepsilon, n, \log c_{max}$.

Proof. Algorithm $P^{\mathcal{A}}(B)$ (Algorithm 5) is a critical payment, Algorithm A_{FPTAS} is an exact algorithm, and bitonic with respect to the objective function according to Lemma 4. Thus the mechanism $M_{A_{FPTAS}}$ is truthful followed by Theorem 2. The FPTAS is achieved in Lemma 1. □

Theorem 4. The runtime complexity of algorithm FPTAS-PAY is $O(\frac{1}{\varepsilon} n^4 \log c_{max} log(\alpha\gamma s_{max}))$, where α, γ are constants, n is the number of tenant bids, c_{max} is the largest bid cost, and s_{max} is the largest bid size.

Proof. The FPTAS payment algorithm $A_{FPTAS-PAY}$ is based on FPTAS, namely, it firstly invokes the FPTAS algorithm to achieve the winner tenant items and then uses a bitonic method in Algorithm $P^{\mathcal{A}}(B)$ to set payment for each winner.

According to Algorithm $P^{\mathcal{A}}(B)$, the bitonic while-iteration invokes FPTAS algorithm repeatedly to set payment for each winner. For winner k with bid data (s_k, b_k), the runtime of each bitonic iteration T_{bk} is $log(h - l) * T_{FPTAS} = log(\alpha\gamma s_k - b_k) * T_{FPTAS}$. Let n^{**} be the number of winners, obviously, $n^{**} \leq n$, then the runtime of $A_{FPTAS-PAY}$ $T_{FPTAS-PAY}$ can be concluded by $T_{FPTAS-PAY} = T_{FPTAS} * (1 + n^{**} * \log(\alpha\gamma s_k - b_k) \leq T_{FPTAS} * (1 + n) log(\alpha\gamma s_k - b_k) \leq T_{FPTAS} * (1 + n) log(\alpha\gamma s_{max})$. So we have the runtime of $A_{FPTAS-PAY}$ $O(\frac{1}{\varepsilon} n^4 \log c_{max} \log(\alpha\gamma s_{max})$, where α, γ *and* ε are constants, n is the number of tenant bids, c_{max} is the largest cost of bid cost, and s_{max} is the largest size of bid size. □

4 VCG-based Mechanisms

One of the basic game-theoretic requirements in mechanism design is that of truthfulness, and the VCG mechanism will ensure truthfulness. However, the VCG mechanism was criticized for the computation efficiency. To get a tradeoff between truthfulness and computation efficiency, we investigate VCG-based mechanisms to solve the MEDR problem.

4.1 The VCG-based Mechanism

In fact, the well-known Vickrey-Clarke-Groves (VCG) [23] mechanism can be applied to our MEDR problem. Note that a VCG mechanism requires an optimal allocation algorithm to guarantee the truthfulness.

In detail, the VCG mechanism is defined as below.

Definition 7. (**The VCG Mechanism**) The VCG mechanism consists of an optimal assignment algorithm A^* and a payment function P, where

1) A^* returns the minimum cost of MEDR to cover the demand response requirement.
2) P_i the payment for agent i is defined as $P_i = A^*(d_{-i}) - A^*_{j\neq i}(d)$.

Here $A^*(d_{-i})$ is the minimum cost of MEDR without counting agent i, and $A^*_{j\neq i}(d)$ is the total cost of an optimal solution for all agents minus the cost of agent i.

In detail, to solve our MEDR problem, we present the VCG-MEDR Mechanism as follows.

Definition 8. (**The VCG-MEDR Mechanism**) The VCG-MEDR mechanism consists of the optimal assignment algorithm DOPT(I) as given in Algorithm 1, and the payment function VCG-PAY is given in Definition 7.

The VCG-MEDR mechanism is truthful and it provides an optimal assignment. However, the running time of VCG-MEDR might grow exponentially for large instances of MEDR. Finally, one can check that the running time of the VCG-MEDR Mechanism is $O(n^3 c_{max})$.

4.2 ε-truthful Mechanism

As we know, a truthful mechanism with VCG-based payment is only valid if the allocation algorithm is an optimal algorithm. It is not guaranteed that an approximated allocation algorithm with VCG-based payment will lead to truthful bidding. However, a VCG-based payment with approximated allocation algorithm brings great benefit in designing an allocation algorithm, because it is quite easy to compute the payment, with running time proportional to the allocation algorithm. One question remains that can we bound the maximal gain in utility that an agent can expect to obtain through non-truthful bidding? This motivated us to study a mechanism with an approximated allocation algorithm and with VCG-based payment.

For each agent i, in a bidding $B = ((s_i, b_i), B_{-i})$, we define the *ex post* regret to agent i at the bidding profile B as follows,

$$regret_i(B) = sup_{(s_i,b_i)}(U_i((s_i, b_i), B_{-i}) - U_i((e_i, c_i), B_{-i})).$$

Thus a mechanism is said to be ε-truthfulness if $|regret_i(B)| \leq \varepsilon$ for all agent i. That is to say that an agent can gain at most ε through some non-truthful strategy in a ε-truthful mechanism.

Let $C^*((e_i, c_i), B_{-i}))$ denote the minimum cost of the social welfare given the reported bidding of every agent $j \neq i$ and the true value of agent i.

Theorem 5. A VCG-based mechanism with an FPTAS allocation algorithm is $\varepsilon C^*((e_i, c_i), B_{-i}))$-truthfulness for agent i given bids B_{-i} from other agents.

Proof. Let $\bar{C}(B)$ denote the social cost generated by the approximation allocation algorithm A_{FPTAS} given the bidding B, and $\bar{C}(B_{-i})$ be the social cost generated by the allocation algorithm A_{FPTAS} without counting the agent i.

From the approximation allocation algorithm A_{FPTAS}, we know that

$$\bar{C}(B) \leq (1 + \varepsilon)C^*(B),$$

where $C^*(B)$ is the optimal cost with respect to the bids B.

Let $\bar{x}$ be the allocation implemented by the approximation algorithm A_{FPTAS}, where $\bar{x}_i = 1$ if agent i was a winner. Denote $cost_i(\bar{x})$ to be the reported cost of agent i given the allocation $\bar{x}$. Let $c_i(\bar{x})$ be the true cost of agent i, which $c_i(\bar{x}) = c_i$ if $\bar{x}_i = 1$ and $c_i(\bar{x}) = 0$ if $\bar{x}_i = 0$.

The VCG-payment of agent i is $P_i(B) = \bar{C}(B_{-i}) - \sum_{j\neq i} cost_j(\bar{x})$, and the utility of agent i with respect to VCG-payment is

$$U_i(B) = \bar{C}(B_{-i}) - \sum_{j\neq i} cost_j(\bar{x}) - c_i(\bar{x})$$

Note that $\bar{C}(B_{-i})$ is independent on agent i, and then agent i tries to maximize his/her utility by declaring some value to minimize $\sum_{j\neq i} cost_j(\bar{x}) + c_i(\bar{x})$.

The maximal benefit of agent i from reporting a non-truthful value occurs when the allocation of truthful revealing is as bad as possible, that is $(1 + \varepsilon)C^*((e_i, c_i), B_{-i}))$. Since, the agent tries to reveal a value to minimize $\sum_{j\neq i} cost_j(\bar{x}) + c_i(\bar{x})$, and the minimal value can be achieved is $C^*((e_i, c_i), B_{-i}))$. The agent i's gain in utility in comparison with truthful bidding, is

$$(1 + \varepsilon)C^*((e_i, c_i), B_{-i})) - C^*((e_i, c_i), B_{-i})) = \varepsilon C^*((e_i, c_i), B_{-i})).$$

□

4.3 Runtime Hinge

We have proposed DOPT, FPTAS, VCG-PAY, and FPTAS-PAY algorithms. We will compare the runtime performance of all the algorithms in this section. We hope to find a guideline to choose algorithms based on the instance we have.

Definition 9. (**Runtime Hinge**) Let P be the common parameter variable of the two algorithms a, b. The runtime T_a, T_b of the algorithms a, b are increasing with respect to the growing value of P. If there exists a constant C, such that $T_a = T_b$ when $n = C$ (where n is the number of tenants), then we call the constant C is the hinge of algorithms a, b based on P, denoted by $H(a, b)|P$.

By Definition 9, we know one algorithm will run faster than another when the number of tenants larger than the hinge. We consider the runtime hinge between algorithms DOPT and FPTAS, and algorithms VCG-PAY and FPTAS-PAY.

Theorem 6. The algorithm DOPT and FPTAS have a hinge C based on the number of tenant bids n, namely, $H(A_{DOPT}, A_{FPTAS})|n$.

Proof. Let k_1, k_2 be two positive constants. The runtime of algorithm DOPT (A_{DOPT}) T_{DOPT} is $O(n^2 c_{max})$, which can be denoted by $T_{DOPT} = k_1 n^2 c_{max}$. The runtime of algorithm FPTAS (A_{FPTAS}) T_{FPTAS} is $O(\frac{1}{\varepsilon} n^3 \log c_{max})$, which can be denoted by $T_{FPTAS} = k_2 \frac{1}{\varepsilon} n^3 \log c_{max}$.

Let T_{DOPT} be equal to T_{FPTAS}, we have $k_1 n^2 c_{max} = k_2 \frac{1}{\varepsilon} n^3 \log c_{max}$ and then conclude that the hinge $C = n = k_1/k_2 \varepsilon c_{max}/\log c_{max}$, where k_1 and k_2 can be determined by the fitting method through experiments. □

By Theorem 6, the algorithm *FPTAS* will run faster when $n \geq k_1/k_2 \varepsilon c_{max}/\log c_{max}$.

Theorem 7. The algorithm VCG-PAY and FPTAS-PAY have a hinge C based on the number of tenant bids n, namely, $H(A_{VCG-PAY}, A_{FPTAS-PAY})|n$.

Proof. We compare the FPTAS-PAY runtime with the VCG-PAY algorithm. Given two positive constant numbers $l_1, l_2 > 0$, then 1) the runtime of $A_{VCG-PAY} T_{VCG-PAY}$ can be denoted by $l_1 n^3 c_{max}$; 2) according to Theorem 4, the runtime of $A_{FPTAS-PAY} T_{FPTAS-PAY}$ can be denoted by $l_2 \frac{1}{\varepsilon} n^4 log c_{max} \log(\alpha\gamma s_{max})$.

Let $T_{VCG-PAY}$ be equal to $T_{FPTAS-PAY}$, then we have $l_1 \frac{1}{\varepsilon} n^4 \log c_{max} log(\alpha\gamma s_{max}) = l_2 n^3 c_{max} \implies$ the hinge $C = n = l_2/l_1 \varepsilon c_{max}/(\log c_{max} \log(\alpha\gamma s_{max}))$, where l_1 and l_2 can be determined by the fitting method through experiments. □

By Theorem 7, we conclude that the runtime of FPTAS-PAY will be larger than the runtime of VCG-PAY when the number of tenant agents $n \geq l_2/l_1 \varepsilon c_{max}/(\log c_{max} \log(\alpha\gamma s_{max}))$.

5 MEDR Auction System

In this section, we implement all the algorithms and build **MEDRAS**, an *Auction System* as a holistic solution for the MEDR problem.

5.1 Integrate Framework

As far as the system framework is concerned, we give the design of the MEDRAS in large scale colocation datacenters and the integrated framework of MEDRAS shown in Fig. 1.

The integrate framework of MEDRAS logically consists of three layers, namely, ***user client***, ***MEDRAS platform*** layer as well as ***MEDRAS service*** layer, respectively.

The ***user client*** refers to the user of MEDRAS, in which there are two types of users, i.e., the tenant users and the colocation operators. The colocation operator charges sending EDR signals, launching auctions and maintain the auction bid data information. While the tenant users who are willing to join in MEDR auction and can submit the bid data to the auction.

The ***MEDRAS platform***, briefly called ***MEDRAS_plat***, denotes a uniform operation platform for auction activity. It includes an *Auction Management Information System* (**AMIS**) and a *Power Control System* (**PCS**).

The **AMIS** is responsible for managing the elemental auction information, including the auction participant user information and bid data. In **AMIS**, the colocation operator can send EDR signals and launch a new auction, while the tenants can declare and submit bids. The **PCS** is a system used to control power supply according to the tenant's EDR

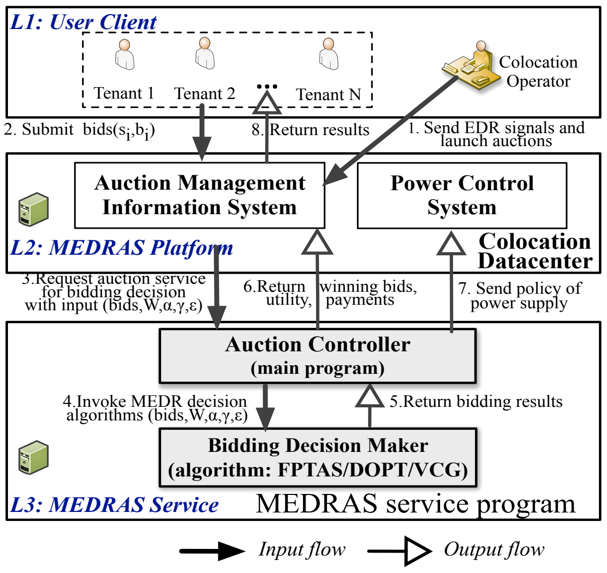

L1: User Client
Tenant 1
Tenant 2
...
Tenant N
Colocation Operator
2. Submit bids(s_i,b_i)
8. Return results
1. Send EDR signals and launch auctions
Auction Management Information System
Power Control System
L2: MEDRAS Platform
Colocation Datacenter
3.Request auction service for bidding decision with input (bids,W,α,γ,ε)
6.Return utility,
winning bids, payments
7. Send policy of power supply
Auction Controller (main program)
4.Invoke MEDR decision algorithms (bids,W,α,γ,ε)
5.Return bidding results
Bidding Decision Maker (algorithm: FPTAS/DOPT/VCG)
L3: MEDRAS Service
MEDRAS service program
Input flow
Output flow

Fig. 1. The integrate framework of MEDRAS

auction bidding results. It charges executing power supply policies for balancing power use according to the power size of bidding winners.

The ***MEDRAS service*** layer is designed as an auction service program of MEDRAS, briefly denoted by (***MEDRAS_sp***), which is realized as a special stand-alone running service program. It involves the ***auction controller (AC)*** and the ***bidding decision maker (BDM)*** functions. The **AC** is the main program of MEDRAS service. It is responsible for interaction with **AMIS**, including functions of receiving the bidding requests from **AMIS**, invoking **BDM** program to make a bidding decision, obtaining the decision results and returning back to **AMIS**. The **BDM** implements all the MEDR algorithms presented in Section 3, including the DOPT, FPTAS, VCG-PAY and FPTAS-PAY.

5.2 Working Process

Fig. 1 illustrates the basic working process against an auction. We further explain it as follows.

(1) Upon the arrival of the EDR signals, the colocation operator (Auctioneer) launches an auction by **AMIS** in the MEDRAS platform and send the EDR signal to tenants to participate this auction activity.
(2) The tenant users who are willing to participate the auction will login the MEDRAS platform and submit their own bids with a size of power and cost they are willing to accept in **AMIS**.
(3) The **AMIS** sends a request of auction service for bidding decision making to the AC in the MEDR server. Before the sending, **AMIS** must prepare the data information which are required to be used in bidding decision algorithms.
(4) The **AC** in the MEDRAS service layer firstly receives the bid information from the **AMIS**, then ask for **BDM** program to handle the bidding decision.
(5) In bidding decision, the **BDM** will firstly input all data from the **AC** and invoke a decision algorithm, such as FPTAS/VCG, and return back to **AC** the results of

winning tenants with minimized social cost, utility and payments.

(6) The **AC** returns back the decision results to **AMIS**. At the same time, a new power supply request will be transmitted to the **PCS** for carrying out power policy from winner tenants in datacenter.

(7) The winning bids information will be published in the MEDRAS platform. The tenant users can learn the result through **AMIS**.

In the end, not only will the winning tenants achieve compensation rewards from the colocation data center operator, but the new policy of power supply will be submitted to the **PCS** and be executed for the EDR.

5.3 System Prototype

We build a prototype of MEDR auction system by implementing the ***MEDRAS_plat*** and ***MEDRAS_sp*** respectively as two modules.

The ***MEDRAS_plat*** module includes **AMIS** and **PCS** which might be closely related to the practical datacenter system. The implementation arises challenges when combining with the actual datacenter system, such that it must consider all kinds of used technologies such as the database, the system architecture and associated development methods. For simplification, we only consider the basic functions to build the prototype of a simple auction system based on the web using PHP and MySQL database technology. We develop some web pages for the basic functions of **AMIS** and employ MySQL database to uniformly store the basic auction data. On the whole, the purpose of the MEDRAS platform is to collect the auction bidding information for ***MEDRAS_sp*** to make the decision of winning bids.

The ***MEDRAS_sp*** module is realized by integrating all the algorithms into a stand-alone running program. We use the C/C++ programming language to develop this program. The ***MEDRAS_sp*** is realized as a command-line parameter program with some command line parameters. When we start a bidding decision making for a given auction, we can write a shell command script and easily run in a command shell environment. The running of the bidding decision algorithm needs firstly to input the required parameter data. We prepare a standard data file to store the auction bid data and another standard file to save the result data. We set the tenant's bid data with the name of the standard data file to the ***MEDRAS_sp*** command. In ***MEDRAS_sp*** command, we set one parameter as the name of the input data file and one parameter as the name of an output data file. During the whole running, this command will automatically read bid data from the input file, invoke the MEDR bidding decision algorithms to conclude the result winning bids, and finally save the result data into the output file. After the running is completed, the results are pooled in the output file.

5.4 Some Implementation Tricks

Some small tricks can be used in the implementation of ***MEDRAS_sp***.

Note that the dynamic programming algorithm DOPT(I) needs a data structure to store the optimal tabular value for each subproblem (i, c). Given a set of tenants with the number n, the bid of tenant i, (s_i, c_i), where $1 \leq i \leq n$. The tabular can be designed as a two-array structure $A[N][NC]$, where $N > n, NC > \sum_i c_i$. However, it comes one issue that the size of tabular will be very large with the growing size of N and NC and the program will require a large size of memory to store the tabular data. The program cannot run normally if the memory is not sufficient or too small. To improve the memory efficiency, in our implementation, we adopt a new one-array tabular, such as $A[NC]$ to only store the last line elements of A[N][NC] because the last line includes all other optimal ones. We employ a *bottom-up* method in which we create each element of this tabular one by one reversely from the largest element to the first one based on the recursive function 4. Moreover, to reduce the running time, in the rounding procedure of algorithm FPTAS A_{FPTAS}, the value of k is setting to within $\log c_{\min}$ and $\log c_{\max}$.

In algorithm Monotone FPTAS A_{FPTAS}, we observe that there is no need for starting the k variable from 1 in the *for* iteration sentence. More specifically, what if, in algorithm 2, $a_k = \frac{\varepsilon 2^k}{n+1}$ and for $i \in T(k)$, we conclude $c'_i = \lfloor \frac{c_i}{a_k} \rfloor$. However, if $k = 1$, we may have $a_k < 1$, then the c'_i will be larger than c_i, resulting in that the computation scale of FPTAS will be larger than the DOPT. Thus, in realistic implementation, we improve it set the k start from the $\log c_{min}$.

The ***MEDRAS_sp*** is finally compiled by gcc/g++. The compile parameter includes -O1,-O2, or -O3, and the default compile is -O1. We found that ***MEDRAS_sp*** performs the different running times when it is compiled with different compiling parameters. For example, the runtime in parameter -O3 is only one-fifth of the parameter -O1.

6 Performance Evaluation

In this section, we perform a large number of experiments to evaluate the performance of MEDRAS and demonstrate the effectiveness of our mechanism. During the whole working process of MEDRAS, the stage of running the BDM service program is the stage that determines the performance or the one in which it has the most critical impact on performance. So in this work, we focus on evaluating the performance of mechanism associated algorithms in the MEDRAS service program. We extend the MEDRAS service program to build an automatic test program, called **BDM-sim**, select the performance metrics, design the test cases, run all the cases and get the corresponding result for analysis.

6.1 Performance Metrics

We choose four performance metrics, namely, approximation ratio, tenants' utility, social cost reductions as well as the runtime metric. The approximation ratio metric is concluded by a ratio between the social cost obtained through the FPTAS and the one obtained through the DOPT mechanism. This metric can determine that how close the solution obtained through the FPTAS is to the optimal solution. The tenants' utility and social cost reductions are used to evaluate the cost efficiency of our FPTAS mechanism. The runtime metric is used to evaluate the runtime performance in a physical server machine with a fixed hardware environment condition, such as the number of CPU cores and memory space capacity. Finally, we will explore the maximum regrets among all tenants when a VCG-based mechanism is applied.

6.2 Test Case Design

We generate auction dataset and design comprehensive test cases for performance evaluation.

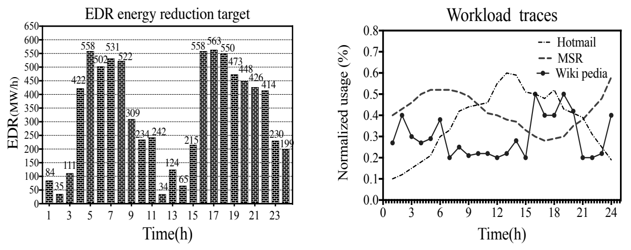
6.2.1 Auction Dataset

We prepare for several auction bid datasets. The most elemental content of the auction dataset primarily is made up of a number of basic auction data. An auction data contains an EDR reduction target W and N pairs of tenant bid (s_i, b_i), where $i = 1, 2, \ldots, N$. We combine some of real data and some of generated random data to build auction datasets for the performance test experiments.

1. Small Scale Dataset: We consider a typical auction with a small scale number of $N(N = 12)$ participating tenants (denoted as Tenant #1, Tenant #2, ...,Tenant #12) in a colocation data center. Each tenant i has $m_i (= 200,000)$ homogeneous servers. The power of each server includes $150W$ computing power and $d_0 = 100W$ idle power [24]. That is to say, if a server is turned off, we can obtain $100W$ power. If a server is running a workload, it needs $150 + 100 = 250W$ power. Hence, each tenant has a demand power at least 50MW.

The data in an auction primary include EDR reduction target W and tenant bids. For simplicity, we generate auction simulation datasets according to the real datasets including total EDR energy reduction by PJM on November 15, 2018 [25] and workload traces used in [2] which are collected from [26] ("Hotmail" and "MSR") and [27] ("Wikipedia"), as shown in Fig. 2(a) and Fig. 2(b). The EDR energy reduction data consists of 24 EDR reduction event numbers. Each number represents an energy reduction of DR that arises in an hour. Like [2], the tenants' bids are generated randomly with consideration of the workload traces and some data center factors, such as the general cost of a server or whole datacenter and the power energy price in market. We duplicate each workload for all tenants with randomness up to 20%. For the total 12 tenants, we firstly generate 24 random numbers r_i between 0.01 and 0.20, where $i = 1, 2, \ldots, 12$. Then we duplicate the workload for four tenants by Hotmail, four by MSR and four by Wikipedia, respectively. We assume that each size of the workload denotes a ratio of the total number of servers which is in running status. If we turn off some servers, then we can slash energy consumption. The total energy reduction by tenant i is $s_i = n_i.d_{0,i}.T$, where $T = 1$ hour in simulation.

We conclude the size of tenants' bid by the formula: $s_i = workloads * M * r_i * d_0 / 1,000,000(MW)$, where $M = 200,000$ and $d_0 = 100$. The size s is set in a range on $1 \sim 80$ MWh [26]. Besides, according to Zhang [2]'s work, tenants can reasonably save $6.7 \sim 13.3$ cents/kWh (depending on electricity price) power when they house servers in their datacenters [26]. Likewise, tenants can save $67 \sim 133$ \$/MWh. Hence we produce several random numbers rb between 67\$ and 133\$ for each tenant as a bid power price in which tenants are willing to take part in the auction activity. The ith tenant bid is concluded by a formula $b_i = s_i * rb_i$. Finally, all the generated data is shown in Table 2. Each tenant bid Bi has a pair of numbers (s_i, b_i), which denotes a size of energy reduction and cost for supplying its responding size of power, respectively. Besides, The table also lists the total sum of bid size $\sum_i s_i$ and bid cost b_i. Most of the hours except the Time hour 17 have the auction with the total sum of bid size larger than the given EDR W, which indicates that all EDR W will be covered by the tenants' bid size.



(a) Total EDR energy reduction by PJMon November 15, 2018. (b) Normalized workloads in distinct hour time.

Fig. 2. The source dataset for bidding decision simulation.

2. Large Scale Dataset: Besides, we also consider the large scale auction scenario with the number of tenant bids $N >= 100$ because many present colocation datacenters, such as Google, Amazon, and Aliyun cloud, etc. possess multi-tenants with a great number of tenants [28]. We generate a large scale simulation auction dataset with the number of tenant $N >= 100$ and the total sum cost of all tenant bids $>= 50,000$, because in our mechanism algorithms, the column number of tabular used in the dynamic programming and the time complexity is closely related to the total cost of all tenant bids. In each tenant bid, the size is randomly in a range from 1 to 15, and the cost is concluded by a formula $size * r$, where r is a general market power price randomly in a range from 67 to 133 per MWh. We set the number of tenant bids from 100 to $1,000$ in a growing step 100 and build a total of 10 groups of bid datasets according to the number of tenant bid. The EDR reduction target W is set the default half of the total size of all tenant bids and will vary in the later evaluation.

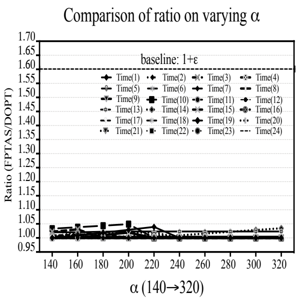
6.2.2 Test Cases

We combine the auction datasets with the FPTAS mechanism algorithm relevant parameters, such as α, γ, and ε, etc. and design a lot of test cases for all the performance metrics to evaluate the performance of our proposed mechanisms.

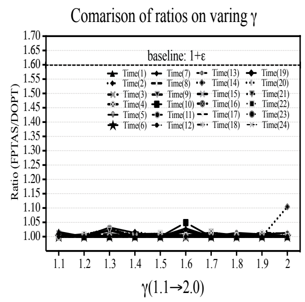
For the metrics of approximation ratio, utility and social cost compared to BES only, we build test cases based on the small scale datasets shown in Table 2 and vary the parameters α, γ, and ε. In each case, we vary one parameter and set the other two parameters to be constant. To evaluate the performance impact resulted from α parameter, we build the case by setting γ to 1.6 [19], ε to 0.6 and varying the parameter α in a range from 140 to 320 [24], [29] with an increasing step length 20. Similarly, for the parameter γ, ε is set to 0.6, α is set to 200\$, and γ changed from 1.1 to 2 with an increasing step length 0.1. For the parameter ε, α is set to 200\$, γ is set to 1.6, and ε is changed from 0.1 to 1.0 with an increasing step 0.1. In sum, we have a total of $24 * (10 + 10 + 10) = 720$ test case data. For the runtime performance metric, we use the large scale dataset to generate a lot of test cases by varying the EDR reduction target W, the number of tenant bids n and the algorithm parameters α, γ, and ε.

TABLE 2
The tenant bidding data

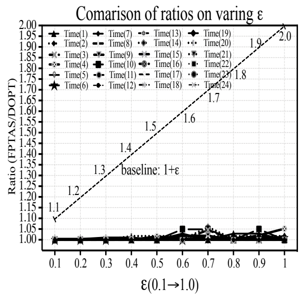
Time (h)	W	Ten.#1		Ten.#2		Ten.#3		Ten.#4		Ten.#5		Ten.#6		Ten.#7		Ten.#8		Ten.#9		Ten.#10		Ten.#11		Ten.#12		Total		$\frac{\sum s_i}{W}$
		s_1	b_1	s_2	b_2	s_3	b_3	s_4	b_1	s_5	b_5	s_6	b_6	s_7	b_7	s_8	b_8	s_9	b_9	s_{10}	b_{10}	s_{11}	b_{11}	s_{12}	b_{12}	$\sum s_i$	$\sum b_i$	
1	84	15	1,440	11	1,254	51	4,539	20	2,060	59	6,431	57	5,757	12	804	55	5,390	18	2,214	7	910	48	4,656	32	2,592	385	38,047	4.6
2	35	30	2,550	10	750	22	2,266	14	1,736	56	5,320	45	4,455	13	910	34	4,012	6	522	5	600	48	4,224	63	7,938	346	35,283	9.9
3	111	38	3,078	47	3,384	50	3,800	35	3,115	58	6,960	40	3,400	15	1,455	8	776	13	1,274	5	575	48	5,376	29	2,755	386	35,948	3.5
4	422	43	3,440	21	1,617	60	4,980	27	2,322	44	3,652	57	4,731	11	737	44	5,324	39	3,315	23	1,909	29	3,422	51	3,570	449	39,019	1.1
5	558	57	4,560	35	4,200	57	4,332	39	2,964	61	8,052	54	3,780	43	3,053	57	7,581	68	8,772	66	7,590	45	5,850	41	3,690	623	64,424	1.1
6	502	27	2,160	33	3,432	22	2,530	37	4,810	63	5,418	53	6,625	47	4,136	51	4,539	61	8,052	69	7,314	48	3,408	67	6,298	578	58,722	1.2
7	531	59	6,136	17	1,207	32	3,872	42	4,872	57	6,612	57	4,047	44	5,236	63	5,859	46	3,910	68	8,568	41	4,756	62	7,564	588	62,639	1.1
8	522	58	7,018	28	2,408	26	2,938	49	3,479	56	5,880	61	6,161	47	4,324	67	5,561	45	4,590	67	5,159	45	3,150	63	8,127	612	58,795	1.2
9	309	37	4,292	11	1,023	15	1,590	36	4,500	32	3,648	53	3,710	36	2,412	45	5,310	43	5,117	23	3,059	49	3,283	25	1,750	405	39,694	1.3
10	234	25	1,850	9	1,089	58	7,424	11	880	38	4,408	47	5,734	46	5,842	37	3,552	37	3,848	37	4,218	50	5,000	25	2,100	420	45,945	1.8
11	242	45	5,670	24	2,496	26	1,794	27	2,106	29	2,929	45	3,870	41	3,198	31	3,596	36	4,032	17	1,904	28	2,352	35	3,605	384	37,552	1.6
12	34	38	3,724	16	1,280	34	2,754	22	2,244	18	1,440	44	5,764	42	5,334	25	1,750	39	4,875	15	1,545	40	3,040	39	3,900	372	37,650	10.9
13	124	22	1,650	19	2,299	21	2,268	33	4,224	23	2,185	9	1,152	27	2,808	23	2,346	13	1,495	15	1,425	11	1,177	6	792	222	23,821	1.8
14	65	20	1,840	13	1,482	17	1,462	30	2,310	28	2,520	7	651	23	2,967	21	2,373	11	1,441	10	670	12	1,464	3	210	195	19,390	3.0
15	215	29	3,161	26	2,678	19	2,261	34	3,230	14	1,722	23	2,300	33	3,927	37	3,219	18	1,476	8	608	51	5,559	11	1,188	303	31,329	1.4
16	558	51	4,794	38	3,268	57	5,301	38	4,408	21	2,793	51	3,621	56	5,992	47	6,016	68	8,772	67	8,174	56	6,776	56	5,152	606	65,067	1.1
17	563	24	1,896	10	850	67	6,767	40	4,800	19	1,767	45	5,490	37	3,700	52	4,472	63	7,497	69	9,039	32	3,776	54	4,914	512	54,968	0.9
18	550	68	6,596	34	3,740	65	7,085	36	4,536	48	6,384	58	4,524	46	3,312	41	4,674	48	4,752	32	2,592	65	8,450	61	7,320	602	63,965	1.1
19	473	55	4,730	29	2,639	15	1,380	35	3,045	31	3,999	65	5,330	50	5,600	24	1,680	38	2,660	52	4,836	63	8,190	58	7,540	515	51,629	1.1
20	448	19	2,318	19	1,824	31	3,844	25	2,750	58	4,234	57	7,524	47	3,901	48	4,512	38	4,104	7	483	55	4,180	57	5,757	461	45,431	1.0
21	426	35	2,765	33	2,409	55	6,435	13	1,222	36	3,060	33	3,630	46	4,462	30	3,360	47	3,619	31	3,937	55	3,960	56	4,424	470	43,283	1.1
22	414	32	2,336	23	2,323	25	2,925	13	1,378	42	4,326	49	3,724	32	3,200	19	1,273	48	3,792	34	3,400	58	6,322	50	5,900	425	40,899	1.0
23	230	8	696	35	4,515	48	4,992	38	3,420	56	5,488	40	2,960	49	5,341	25	2,475	43	3,999	28	3,360	27	3,510	24	2,784	421	43,540	1.8
24	199	11	1,001	42	5,040	47	4,606	15	1,965	23	2,599	59	7,788	49	4,949	39	4,446	45	3,105	7	868	52	4,108	46	4,278	435	44,753	2.2



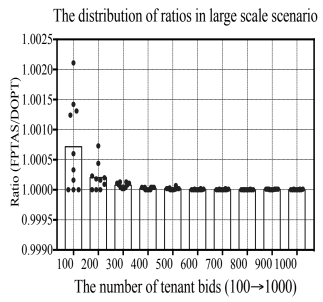
(a) The ratio between FPTAS and DOPT for 24 EDR instances on varying α, where $\gamma = 1.6$, $\varepsilon = 0.6$



(b) The ratio between FPTAS and DOPT for 24 EDR instances on varying γ, where $\alpha = 160\$$, $\varepsilon = 0.6$



(c) The ratio between FPTAS and DOPT for 24 EDR instances on varying ε, where $\alpha = 160\$$, $\gamma = 1.6$



(d) The ratio distribution of 100 auction bid cases on varying the large scale number of auction bids, where $\alpha = 180$, $\gamma = 1.6$, $\varepsilon = 0.5$.

Fig. 3. The performance evaluation on approximation ratio for distinct time of EDR instances with the number of tenant bids $N = 12$.

6.3 Experimental Result Analysis

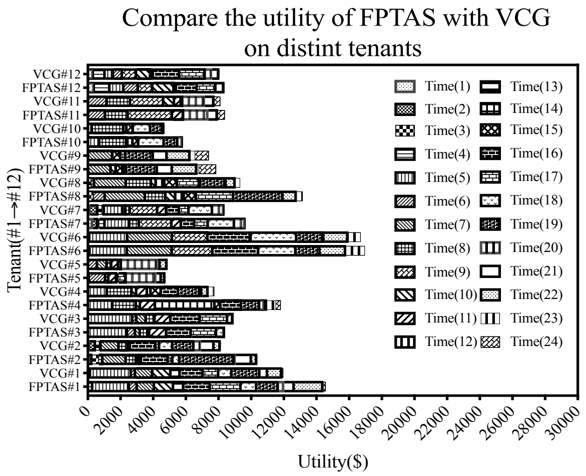
We develop an automatic simulation software tool ***MEDRAS_sim***, based on the ***MEDRAS_sp***. The tool can automatically generate the simulation datasets in batch, the test case data, run the testing and obtain results for analysis. We run all experiments in a specific physical server machine, a Dell PowerEdge R730, with dual Intel(R) Xeon(R) CPU E5-2650L v4 @ 1.70GHz (total 56 CPU cores) and 64GB memory. The server runs CentOS 7.0 Linux with GCC version 4.8.5 20150623. The simulation tool is compiled by g++ with optimization parameter -O3 for supporting better parallel efficiency. We prepare all test case data together into a standard data file that is used as input for the algorithms in the MEDRAS service program. During the test running process, our MEDRAS_sp in batches reads the data, executes all algorithms one by one, and output the performance result data to an output data file for analysis. The result information includes a list of test case result, for each case, it consists of the runtime of DOPT, FPTAS, FPTAS-PAY, and VCG-PAY algorithms, the payment, Non-negative utilities obtained by FPTAS and VCG for each winner tenant, and the social cost by BES. We analyze the experimental results in detail as follows.

6.3.1 Approximation Ratio Performance

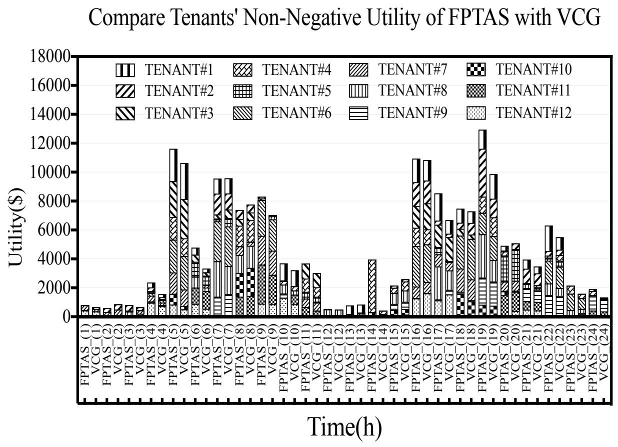
Our truthful FPTAS mechanism in MEDRAS can achieve a $1+\varepsilon$ approximation ratio in theory. The approximation ratio is concluded by a ratio between the minimized total social cost derived from the FPTAS algorithm Y(FPTAS) and the DOPT-optimal one Y(DOPT), i.e., $ratio = \frac{Y(FPTAS)}{Y(DOPT)}$.

We run all test cases for all the 24 EDR reduction instances which are shown in Table 2. The result is shown in Fig. 3. Fig. 3(a), Fig. 3(b) and Fig. 3(c) are corresponding to the performance result from three parameters of FPTAS mechanism algorithm. The Fig. 3(d) is the ratio performance result from the large scale test cases.

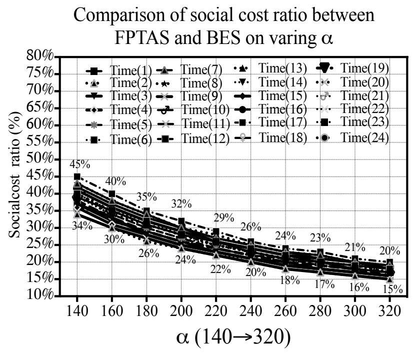
From all the figures we observe that all ratios are between 1 and $1+\varepsilon$. Moreover, the ratios are very close to 1. It means that the FPTAS solution is near-optimal. All experimental results reveal that the FPTAS solution achieves better performance than the theoretical upper bound. Among the parameters $\alpha, \gamma, \varepsilon$, the largest impact factor is the parameter



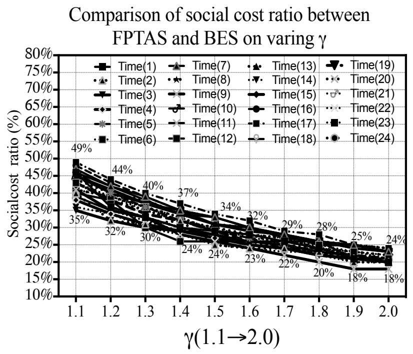
(a) Compare tenants' non-negative utilities in FPTAS with that in VCG mechanism, where $\alpha = 200, \gamma = 1.6$, and $\varepsilon = 0.6$.



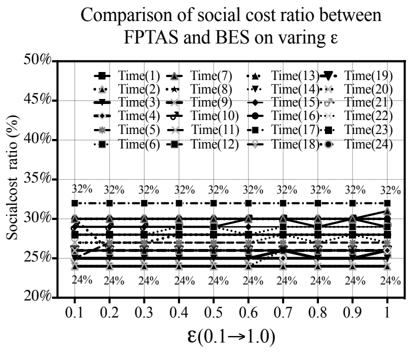
(b) Compare tenants' non-negative utilities in FPTAS with that in VCG mechanism for 24 EDR events, where $\alpha = 200, \gamma = 1.6$, and $\varepsilon = 0.6$.



(c) The normalized social cost ratio between FPTAS and BES, where $\gamma = 1.6$, $\varepsilon = 0.6$ and α varying from 140 to 320(+20).



(d) The normalized social cost ratio between FPTAS and BES, where $\alpha = 200$, $\varepsilon = 0.6$ and γ varying from 1.1 to 2.0(+0.1).



(e) Comparison of social cost ratio between FPTAS and BES, where $\alpha = 200$, $\gamma = 1.6$ and ε varying from 0.1 to 1.0(+0.1).

Fig. 4. The performance evaluation on tenants' Non-negative utilities and social costs compared to BES only for all 24 EDR events.

ε. With the increase of ε, the ratio increases a bit but still is far smaller than the theoretical upper bound $1 + \varepsilon$. These experimental results show that the parameter ε did not to be very small, we still can achieve near-optimal performance. This observation is useful, and we can reduce the running time that depends on $1/\varepsilon$ by setting ε to a relatively large number, such as $\varepsilon = 0.6$.

6.3.2 The Cost Efficiency of FPTAS Mechanism

We consider the cost efficiency of the FPTAS mechanism and the VCG mechanism. It means that how much cost the colocation datacenter operator will pay for the tenants in an auction using these mechanisms and how much cost the colocation datacenter will have if they do not carry out the auction. We adopt the agents' utilities and social cost reduction compared to BES only metrics to evaluate the cost efficiency of our mechanisms. We compare the two metrics obtained from the FPTAS and VCG mechanism.

1. Tenants' utilities: We study each tenant or agent's utility in all the experiments for the two mechanisms. Note that agents' utilities are concluded from the payment for each tenant subtracts his/her actual cost. The individual rationality of both FPTAS and VCG mechanism has been proved. And the experimental results also confirm that each tenant obtains a non-negative utility in each variant of our experiment. We only show the result in Fig. 4(a) and Fig. 4(b) by letting α=200\$, $\gamma = 1.6$ and $\varepsilon = 0.6$. In the Figure, U_FPTAS_i and U_VCG_i denote the utility of the tenant i concluded from the FPTAS and VCG mechanism, respectively.

We would like to find which mechanism will achieve more utility to encourage agents to participate in.

From the point of view of an individual agent, there is no conclusion that which mechanism might lead to more utility. According to Fig. 4(a), although most of the tenants will obtain more utilities from the FPTAS mechanism, the tenants 3 and 5 achieve more utility from the VCG mechanism.

But if we regard the total utilities obtained by all tenants, Fig. 4(b) shows that the FPTAS mechanism will achieve more than VCG mechanism. The Fig. 4(b) also indicates 1) the 24 hour times EDR reduction W have different total utilities; 2) the more EDR W and the more tenant bids will derive the more utilities.

2. Social cost reduction compared to BES only: Each winner tenant has obtained a non-negative utility, which implies that the colocation operator will pay a lot of money to these tenants. To study whether this payment is too much, we investigate the social costs compared to the one we only use BES. We use a normalized percentage (%) μ concluded from the ratio between the cost of FPTAS and that of the BES only. The social cost reduction can be concluded by a formula: $BES * (1 - \mu)$. The larger percentage means the smaller the reduction of social costs. Firstly, we know that $\mu \leq 1$, i.e., there must have social cost reduction because the mechanism will use BES instead of the tenant whose unit cost is more than α.

We compare all social costs in the 24 hour time EDR events on varying the parameters α, γ and ε in FPTAS. The results are illustrated in Fig. 4(c), Fig. 4(d) and Fig. 4(e). The results show

1) the percentages μ are declined when α or γ increases. Large α means that it is more expensive to use BES. Large γ indicates less energy reduction from tenants will still meet the target, which incurs smaller social costs as well.
2) the percentages μ are stable as ε varies. The reason behinds it is that the social costs are not changing too much as ε varies. One can see it in Fig. 3(c), the approximation ratio is close to 1, which means the social cost is close to the optimal cost that is independent of ε.
3) we can get the largest EDR reduction $1 - 15\% = 85\%$ from Fig. 4(c), and the smallest one $1 - 49\% = 51\%$ of BES only from Fig. 4(d).

In sum, the social cost given by the FPTAS mechanism is much smaller than that one of BES only, and the colocation data center needs to enable more tenants to attend the DR activities.

6.3.3 Runtime Performance

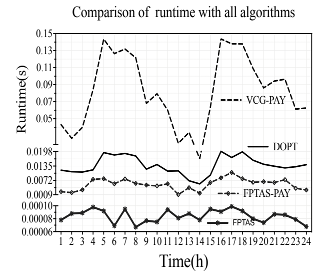
We evaluate the runtime performance in the small scale case of the 24-hours time EDR auction instances. We then take a holistic evaluation of the performance influence of the associated parameters in our mechanism algorithms. In our MEDR model, the relevant parameters include the EDR reduction target W and tenant bid (size, cost) in an auction, the number of tenant bids n, α, γ and ε. We design a lot of cases for each parameter performance test. All the results are shown in Fig. 5.

1. Small scale case: The small scale case is prepared for the 24-hour time EDR auction instances. The result is shown in Fig. 5(a). We observe that all MEDR algorithms seem running very quickly because every algorithm in each hour's time EDR instance is completed in less than 0.15 seconds. The runtime of all algorithms in the 24-hour time EDR instances is different and presents a similar distribution and changing trend. In all EDR instances, the FPTAS algorithm outperforms the fastest runtime while the VCG-PAY takes the longest runtime. What's more, in the FPTAS even the slowest runtime that happened in the $17th$-hour time EDR instance is still less than 0.000, 1s. The runtime of the FPTAS-PAY algorithm performs about 10 times longer than the runtime of FPTAS. The slowest runtime point at the $17th$-hour time EDR instance does not reach 0.01s. The runtime of the DOPT algorithm gets a range between 0.006s and 0.02s. It exists a long distance from the runtime of the FPTAS. The runtime of the VCG-PAY algorithm is naturally longer than the DOPT because it depends on the sum of the DOPT run times with the same number of auction winner tenants. After all, this experiment test manifests that FPTAS and FPTAS-PAY are high-efficiently adaptable to be used in small scale case EDR auction.

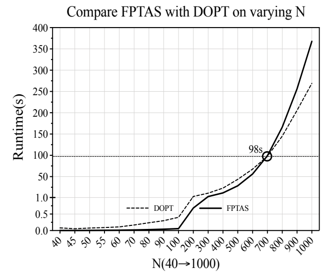
2. Runtime changing trend on varying the number of tenant bids: According to the runtime complexity analysis, it is clear that an auction with a larger number of tenant bids may incur the longer runtime of all algorithms. We design a series of auction test cases for distinct number scales, which is changed from a smaller number to a larger one. Besides the number scale, all cases are provided with the same randomly generated uniform distribution bid data and EDR reduction target W. The α, γ, and ε parameters are set the same fixed numerical value. We compare the runtime of FPTAS with DOPT and the one of FPTAS-PAY with VCG-PAY in two figures, respectively. After running all the cases, the results are shown in Fig. 5(b) and Fig. 5(c). We vary the number of tenant bids from 40 to 1000 for each case. The α is set $160\$/MWh$, γ set 1.6 and ε set 0.6. The EDR W is set 360, satisfying the total bid size $\tau * \gamma \geq W$ with a guarantee of sufficient size of bids competing for the limited EDR resources.

Both the two figures depict the similar growing trend of the runtime with the increasing number of tenant bids. In each figure, the two running time curves plot different growth rates. Due to the FPTAS and DOPT algorithm or the FPTAS-PAY and VCG-PAY algorithm have distinct runtime complexity, we observe that 1) initially, in Fig. 5(b) the FPTAS has the smaller runtime than DOPT and in Fig. 5(c) the FPTAS-PAY smaller than VCG-PAY; 2) Then, as the scale number of tenant bid n gets larger and larger reaching a hinge at some scale number C, in which both the FPTAS and DOPT as well as the FPTAS-PAY and VCG-PAY have the same runtime; 3) when the number $n > C$ FPTAS begins to get bigger than the DOPT and the FPTAS-PAY bigger than the VCG-PAY in runtime. In our cases, the runtime of FPTAS and DOPT have a hinge C_1 700 in Fig. 5(b) and FPTAS-PAY and VCG-PAY have a hinge C_2 60 in Fig. 5(c). From Fig. 5(c), We further find that especially the runtime of FPTAS-PAY possesses a much more sharply growing rate than the VCG-PAY when the scale number n across after the hinge ($n > 700$) and increases continuously. When the scale number $n = 500$, the FPTAS-PAY reaches the runtime more than $8,000s$, while VCG-PAY does not get $8,000s$ even the scale number n reaches $1,000$. In the case with the number $n = 1,000$, our real runtime of FPTAS-PAY is $122,933.096s$ about $34h$, more than one day. It indicates that the FPTAS-PAY can only be used for the type of long-term auction when the number of scales is very large. Comparatively, the VCG-PAY may be better. But for short term auction with a small number scale such as $n < 60$, FPTAS will be the first choice method.

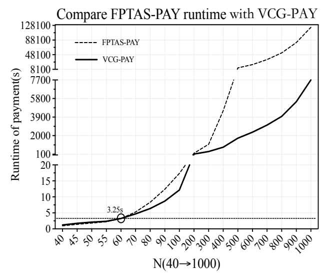
3. Runtime changing trend on varying the bid cost: We design 10 test cases for this scenario. We generate auction bid data by randomizing the size s in a range $1 - S$, and S is set $10, 20, \ldots, 100$ ten times, respectively. We use a simulated price of μ randomized in a real electricity market price range $67 - 133\$$. The cost of each bid is finally generated by $\mu * s$. The EDR reduction target W is set half of the sum total size of all bids in order to the number of the winner bids remains close the same. We generate ten auction cases with



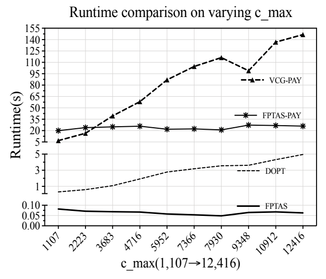
(a) Comparison of the runtime of FPTAS, DOPT, FPTAS-PAY, and VCG-PAY for the 24 hour time EDR auction instances with 12 tenant bids, where $\gamma = 1.6$, $\varepsilon = 0.6$ and $\alpha = 200$.



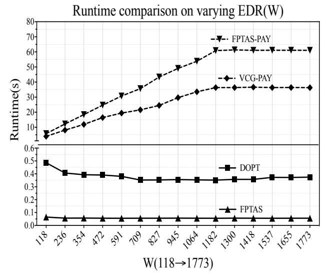
(b) Comparison of the FPTAS runtime with DOPT for EDR instances with distinct number of tenant bids, where $\gamma = 1.6$, $\varepsilon = 0.6$, $\alpha = 160$, and EDR W=360.



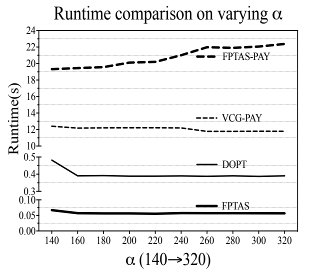
(c) Comparison of FPTAS-PAY runtime with VCG-PAY on varying the number of tenants, where EDR $W = 360$, $\gamma = 1.6$, $\varepsilon = 0.6$ and $\alpha = 160$



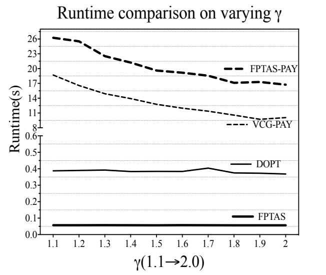
(d) Compare the runtime of all algorithms on varying the c_{max}, where $\gamma = 1.6$, $\varepsilon = 0.6$, $\alpha = 160$, and the number of tenant $N = 100$.



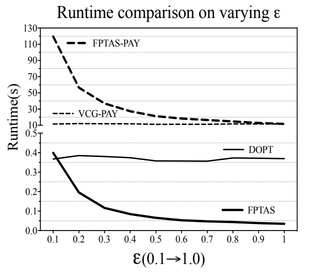
(e) The runtime comparison on varying the EDR W, where $\gamma = 1.6$, $\varepsilon = 0.6$, $\alpha = 160$, the number of tenant $N = 100$, the total bid size $S = 740$ and EDR W varying from 118 to 1773



(f) The runtime comparison on varying α, where the number of tenant bids N, $N = 100$, $\gamma = 1.6$, $\varepsilon = 0.6$ and EDR target $W = 370(0.5)$.



(g) The runtime comparison on varying γ, where the number of tenant bids N, $N = 100$, $\alpha = 200$, $\varepsilon = 0.6$ and EDR target $W = 370(0.5)$.



(h) The runtime comparison on varing γ, where $\varepsilon = 0.6$ and $\alpha = 160$.

Fig. 5. The performance evaluation on runtime.

distinct S corresponding to the number from 10 to 100. Each auction case has 100 tenant bids. Other parameters are set default, i.e., the α is set 160\$/MWh, γ set 1.6 and ε set 0.6. We consider the max of cost c_{max} in each case and its impact on the runtime of all algorithms. Obviously, the c_{max} of $10-100$ case is increasing gradually. The result is shown in Fig. 5(d).

It is observed that the runtime of VCG-PAY and DOPT increases while there is not much more difference in the runtime of FPTAS and FPTAS-PAY with the growing amount of c_{max}. On the whole, comparing with the VCG-PAY, FPTAS-PAY performs nearly no influence on the runtime in c_{max}, and the result is as same as the FPTAS and DOPT.

4. Runtime changing trend on varying the EDR reduction target W: In this scenario, we consider the EDR reduction target W and the total sum of tenant bid size τ. The larger EDR reduction target W will require the more winner items if an auction has sufficient tenant bids, namely, $\tau * \gamma > W$. In our payment algorithm, the more winner items require more iterations to setting payment for the winners. If the EDR reduction target $W > \tau * \gamma$ then all tenant bid items will be winners and the runtime of payment will be constant. We design several cases to test the relation between the EDR target W and the runtime performance for all algorithms. Given an auction with N tenant bids, the total size of bids is $\mathbb{S}$, we provide a ratio ω, which is respectively set from 0.1 to 1.5 with a growing step +0.1, and use a formula $\tau * \gamma * \omega$ to generate 15 EDR reduction target cases. The result is shown in Fig. 5(e). The figure reveals that the runtime of the two payment algorithms FPTAS-PAY and VCG-PAY gets a growing trend with the increasing value of EDR reduction target. When the EDR W ¿ $\tau * \gamma$, the runtime remains the same because all participated tenant bids are winners and these cases iteratively perform the same time iteration to set payment for the winners by invoking FPTAS in FPTAS-PAY or DOPT in DOPT-PAY.

5. Runtime changing trend on varying the α, γ and ε: Ultimately we evaluate the impact of α, γ, and ε parameters on the runtime performance of FPTAS algorithm, and analyze the changing trend of the runtime as these parameters are changed regularly. We choose a large scale test case in which the auction has 100 tenant bids. The results are shown in Fig. 5(f), Fig. 5(g) and Fig. 5(h). Fig. 5(f) shows that the α parameter has little impact on the FPTAS, DOPT and VCG-PAY algorithms and the runtime of FPTAS-PAY takes a slow growth with the increasing of α from 140 to 320 []. According to the Fig. 5(g) we find that the varying γ parameter has little difference in runtime performance of DOPT and FPTAS but the runtime of FPTAS-PAY and VCG-PAY algorithm is changing faster and faster as the γ gets bigger and bigger. We can see from Fig. 5(h) that the ε parameter does not impact on the runtime performance of DOPT and VCG-PAY algorithm, while the runtime of FPTAS and FPTAS-PAY algorithm will become faster and faster as the ε parameter becomes bigger and bigger.

6.3.4 Regret Analysis for ϵ-truthfulness Mechanism

We study the regret performance of each agent in a VCG-based mechanism. To generate simulation data, we suppose that the given bidding is truthful. Then, we fix bids B_{-i}, and calculate the maximum regret of agent i by deviating the bid of (s_i, c_i) (varies s_i and or c_i). Briefly, for each agent i, we fix s_i and vary the c_i by a scale, $c_i = \delta * s_i$, in which δ is changing from 1 to α with an increasing step 4. In each bidding, we conclude the regret result and obtain the max regret. We use the dataset of 24-hours auction settings. In each setting, we get the ratio between the max regret of all agents and the social cost. The results are shown in Table 3. As shown in the table, most of the regret of agents is 0. Some non zero regrets are less than the given ϵ. The results demonstrate that every agent has little regret in the VCG mechanism, which implies that there is no need for an agent to report bids untruthfully.

TABLE 3
The ratio between maximal regret and minicost

ϵ / Time(h)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Time(1)	0	0	0	0	0	0.053	0	0.053	0.053	0
Time(2)	0	0	0	0	0	0	0	0	0	0
Time(3)	0	0	0	0	0	0	0	0	0	0
Time(4)	0	0	0	0	0	0	0	0	0	0.009
Time(5)	0	0	0	0	0	0	0	0	0	0
Time(6)	0	0	0	0	0	0	0	0	0	0
Time(7)	0	0	0	0	0	0	0	0	0	0
Time(8)	0	0	0	0	0	0	0	0	0	0
Time(9)	0	0	0	0	0	0	0	0	0	0
Time(10)	0	0	0	0	0	0	0	0	0	0
Time(11)	0	0	0	0	0	0	0	0	0	0
Time(12)	0	0	0	0	0	0	0	0	0	0
Time(13)	0	0	0.007	0.007	0	0.013	0.010	0.007	0	0
Time(14)	0	0	0	0	0	0	0.057	0.003	0.003	0.003
Time(15)	0	0	0	0	0	0	0.012	0	0	0
Time(16)	0	0	0	0	0	0	0	0	0	0
Time(17)	0	0	0	0	0	0	0	0	0.165	0.174
Time(18)	0	0	0	0	0	0	0	0	0	0
Time(19)	0	0	0	0	0	0	0	0	0	0
Time(20)	0	0	0	0.024	0.024	0.024	0.024	0.004	0.004	0.021
Time(21)	0	0	0	0	0	0.008	0	0.008	0	0
Time(22)	0	0	0	0	0	0	0	0	0	0
Time(23)	0	0	0	0	0	0	0	0	0	0
Time(24)	0	0	0	0	0	0	0	0	0	0

7 RELATED WORK

There are several existing research efforts on mechanism design for DR. For example, Ren and Islam [24] studied the mechanism design for colocation demand response, but their mechanism is not truthful and may not meet the target of EDR. Chen et al. [30] studied the green colocation data center by designing a pricing mechanism to fulfill the energy reduction requirement for EDR. The energy reduction from tenants is calculated by the price-taking and price-anticipating equilibrium. Zhou et al. [7] studied demand response on the geo-distributed cloud through the VCG-based mechanism, in which the utility of each agent depends highly on its interactive workload. Incentive mechanism for emergency demand response in geo-distributed colocation data centers has also been studied in [31]. Sun et al. [32] considered fairness among the mechanism design and provided an online mechanism with a competitive ratio of 3.2 in expectation. Ahmed et al. [33] proposed a contract-based mechanism, in which the colocation operator offers a set of contracts (i.e., a pair of energy reduction and rewards) to tenants and tenants can voluntarily select none or one of the contracts to accept, while the objective is to minimize the operator's cost, the sum of rewards plus the cost of BES. Islam et al. [34] reduced the operator's cost by learning the tenants' response to reward. Tran et al. [35], [36] used a two-stage Stackelberg game to model the economic demand response where the operator can adjust an elastic energy reduction target. Recently the authors [37] also studied the incentivization of energy reduction for EDR in Multi-Tenant Mixed-Use buildings. They did not consider the truthful feature of proposed mechanisms.

Another line of closely related work concerns DR in smart grids. Zhou et al. [5] studied the mechanism design on DR in smart grids. Let $\alpha = 1$, and there is an upper bound on the BES, i.e. $y \leq z_{\max}$. A randomized FPTAS mechanism was given in [5]. Their idea is to combine with smooth analysis and randomize auction. Actually, Dough and Roughgarden [38] showed that if there exists an FPTAS approximation, then this algorithm can be transformed into a truthful in expectation mechanism that retains the FPTAS property. The work in [38] does not require the existence of FPTAS. However, it still remains open whether there exists a deterministic FPTAS for this problem. Zhou et al. [39] proposed a truthful online mechanism for location-aware tasks in mobile crowd sensing, an effective incentive mechanism for mobile crowd sensing, stimulating the participation of smartphone users. Gao et al. [40] addressed a truthful incentive mechanism design for a vehicle-based, nondeterministic crowdsensing system. Our problem is deterministic and not online.

A vast amount of work has been done for mechanism design on multi-unit auction problem, in which there is a set of identical items among bidders, and every bidder has a private valuation function on the number of items, and the problem is to find an allocation of the items to the bidders so as to maximize the sum of bidders' valuations [41], [42], [43]. Our FPTAS mechanism was used for single-unit auction. Briest et al. [16] presented a truthful FPTAS for the max-knapsack problem. Our problem differs from the min-knapsack in which we have BES such that the capacity of the knapsack we need to cover is soft.

8 CONCLUSIONS

In this paper, we have studied MEDR, a mechanism design problem for EDR in colocation data centers. To solve MEDR, we have proposed a near-optimal deterministic truthful mechanism, which is a $1+\varepsilon$ approximation ratio for a reverse auction of EDR in colocation data centers. We also presented a VCG-based mechanism. We have implemented MEDRAS, an auction system combining our mechanism algorithms. The codes for the system are open-sourced. We have also developed MEDRAS_sim, a bidding decision tool for simulation experiments. The experimental results demonstrated the effectiveness of our methods. In the future, we plan to study the scalability of the FPTAS mechanism which will be more challenging and interesting work. We also plan a meaningful work to explore the parallel methods to speed up the runtime of the key algorithms in the large scale number case. Another direction is to study the mechanism design for multi-minded agents because many open problems arise in the area of DR. Our provided technique allows us to deal with single-minded agents, wheres both the size of energy and the cost are private information.

Acknowledgment

This research was in part supported by the National Key R&D Program of China (2017YFB1401304), the Zhejiang Provincial Key R&D Project Grant (2019C01055), NSFC (11671355, 60970125, 61272303, 61772466, U1936215, and U1836202), the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, the Zhejiang Provincial Natural Science Foundation under No. LSY19H180011, the Ant Financial Research Funding, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. We thank all the reviewers for giving us a lot of valuable suggestions.

References

[1] W. Wei, F. Liu, and S. Mei, "Energy pricing and dispatch for smart grid retailers under demand response and market price uncertainty," *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1364–1374, 2017.

[2] L. Zhang, S. Ren, C. Wu, and Z. Li, "A truthful incentive mechanism for emergency demand response in colocation data centers," in *Proc. of IEEE INFOCOM*, 2015.

[3] F. Kamyab, M. Amini, S. Sheykhha, M. Hasanpour, and M. M. Jalali, "Demand response program in smart grid using supply function bidding mechanism," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1277–1284, 2016.

[4] M. H. Y. Moghaddam, A. Leon-Garcia, and M. Moghaddassian, "On the performance of distributed and cloud-based demand response in smart grid," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2017.

[5] R. Zhou, Z. Li, C. Wu, and M. Chen, "Demand response in smart grids: A randomized auction approach," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2540–2553, 2015.

[6] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 71–83, 2014.

[7] Z. Zhou, F. Liu, Z. Li, and H. Jin, "When smart grid meets geo-distributed cloud: An auction approach to datacenter demand response," in *Proc. of IEEE INFOCOM*, 2015.

[8] S. M. Errapotu, J. Loveless, R. Yu, S. Ren, M. Pan, and Z. Han, "Privacy preserving clock auction for emergency demand response in colocation data centers," in *Proceedings of 2017 IEEE International Conference on Communications (ICC 2017)*. IEEE, 2017, pp. 1–6.

[9] P. Jacquot, O. Beaude, S. Gaubert, and N. Oudjane, "Demand side management in the smart grid: An efficiency and fairness tradeoff," in *IEEE Pes Innovative Smart Grid Technologies Conference Europe*, 2018, pp. 1–6.

[10] R. Lavi and C. Swamy, "Truthful and near-optimal mechanism design via linear programming," *Journal of the ACM (JACM)*, vol. 58, no. 6, p. 25, 2011.

[11] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, vol. 16, no. 1, pp. 8–37, 1961.

[12] E. H. Clarke, "Multipart pricing of public goods," *Public choice*, vol. 11, no. 1, pp. 17–33, 1971.

[13] T. Groves, "Incentives in teams," *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.

[14] J. Chen, Q. He, D. Ye, W. Chen, X. Yang, K. Chiew, and L. Zhu, "Joint affinity aware grouping and virtual machine placement," *Microprocessors & Microsystems*, vol. 52, p. S0141933116304136, 2016.

[15] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A ptas mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386–2399, 2015.

[16] P. Briest, P. Krysta, and B. Vöcking, "Approximation techniques for utilitarian mechanism design," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1587–1622, 2011.

[17] M. Bortolini, M. Gamberi, F. Pilati, and A. Regattieri, "Design and management of renewable smart energy systems: An optimization model and italian case study," in *International Conference on Engineering Optimization*. Springer, 2018, pp. 1340–1352.

[18] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM (JACM)*, vol. 49, no. 5, pp. 577–602, 2002.

[19] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[20] E. L. Lawler, "Fast approximation algorithms for knapsack problems," *Mathematics of Operations Research*, vol. 4, no. 4, pp. 339–356, 1979.

[21] I. M. Tauhidul, "Approximation algorithms for minimum knapsack problem," *Master's degree Thesis, university of lethbridge*, 2009.

[22] A. Mu'Alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions," *Games and Economic Behavior*, vol. 64, no. 2, pp. 612–631, 2008.

[23] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[24] S. Ren and M. Islam, "Colocation demand response: Why do i turn off my servers?" in *Proc. of 11th International Conference on Autonomic Computing (ICAC 2014)*. USENIX Association, 2014.

[25] url, "www.pjm.com." [Online]. Available: https://dataminer2.pjm.com/feed/reg_zone_prelim_bill/definition

[26] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1378–1391, 2013.

[27] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, and Gibson, *Safe and effective fine-grained TCP retransmissions for datacenter communication*. ACM, 2009.

[28] url, "www.datacenterknowledge.com." [Online]. Available: https://www.datacenterknowledge.com/archives/2017/03/16/google-data-center-faq

[29] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 2011, pp. 221–232.

[30] N. Chen, X. Ren, S. Ren, and A. Wierman, "Greening multi-tenant data center demand response," *Performance Evaluation*, vol. 91, pp. 229–254, 2015.

[31] L. Zhang, S. Ren, C. Wu, and Z. Li, "A truthful incentive mechanism for emergency demand response in geo-distributed colocation data centers," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, pp. 1–23, 2016.

[32] Q. Sun, C. Wu, S. Ren, and Z. Li, "Fair rewarding in colocation data centers: Truthful mechanism for emergency demand response," in *Proceedings of IEEE 23rd International Symposium on Quality of Service (IWQoS)*, 2015, pp. 363–372.

[33] K. Ahmed, M. A. Islam, and S. Ren, "A contract design approach for colocation data center demand response," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2015, pp. 635–640.

[34] M. Islam, H. Mahmud, S. Ren, X. Wang *et al.*, "Paying to save: Reducing cost of colocation data center via rewards," in *Proc. of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA 2015)*. IEEE, 2015, pp. 235–245.

[35] N. H. Tran, C. T. Do, S. Ren, Z. Han, and C. S. Hong, "Incentive mechanisms for economic and emergency demand responses of colocation datacenters," *Selected Areas in Communications, IEEE Journal on*, vol. 33, no. 12, pp. 2892–2905, 2015.

[36] N. H. Tran, C. Pham, S. Ren, Z. Han, and C. S. Hong, "Coordinated power reduction in multi-tenant colocation datacenter: An emergency demand response study," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

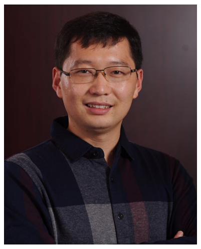
[37] N. H. Tran, C. Pham, M. N. Nguyen, S. Ren, and C. S. Hong, "Incentivizing energy reduction for emergency demand response in multi-tenant mixed-use buildings," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3701–3715, 2018.

[38] S. Dughmi and T. Roughgarden, "Black-box randomized reductions in algorithmic mechanism design," *SIAM Journal on Computing*, vol. 43, no. 1, pp. 312–336, 2014.
[39] R. Zhou, Z. Li, and C. Wu, "A truthful online mechanism for location-aware tasks in mobile crowd sensing," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1737–1749, 2018.
[40] G. Gao, M. Xiao, J. Wu, L. Huang, and C. Hu, "Truthful incentive mechanism for nondeterministic crowdsensing with vehicles," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2982–2997, 2018.
[41] J. Lessan and S. Karabatı, "A preference-based, multi-unit auction for pricing and capacity allocation," *Computers & Operations Research*, vol. 91, pp. 237–246, 2018.
[42] E. Anderson and P. Holmberg, "Price instability in multi-unit auctions," *Journal of Economic Theory*, vol. 175, pp. 318–341, 2018.
[43] Y. Zhang, Y. Gu, M. Pan, N. H. Tran, Z. Dawy, and Z. Han, "Multidimensional incentive mechanism in mobile crowdsourcing with moral hazard," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 604–616, 2018.

Jianhai Chen Jianhai Chen is currently an Associate Professor of College of Computer Science and Technology at Zhejiang University (ZJU). He is the director of ZJU SuperComputing Team, and the director of ZJU Intelligent Computing Innovation and Entrepreneurship laboratory (ICE-lab). He received his M.S. and PhD degrees in Computer Science and Technology at Zhejiang University. His research interests include blockchain system security, cloud computing scheduling algorithms and game theory, supercomputing application optimization, and data mining. He is a member of CCF, IEEE and ACM.

Deshi Ye is currently an Associate Professor of Computer Science and Technology at Zhejiang University (ZJU). He received his Ph.D. from Zhejiang University in Mar. 2005. His research interests include online algorithms and approximation algorithms, scheduling and bin packing problems, wireless networks and mobile computing, and algorithmic Game theory. He is a member of CCF, IEEE.

Zhenguang Liu had been worked as a research fellow in National University of Singapore (NUS) and Singapopre Agency for Science, Technology and Research (A* STAR) for three years. He is now with Zhejiang Gongshang University. He respectively received his Ph.D. and B.E. degrees from Zhejiang University and Shandong University, China, in 2010 and 2015. His research interests include data mining, distributed system, and multimedia data analysis. Various parts of his work have been published in first-tier venues including AAAI, ACM MM, NIPS, TIP, TMM, TOMM, etc. He has served as the program committee member for conferences such as ACM MM and MMM, and the reviewer for IEEE Transactions on visualization and computer graphics (TVCG), Journal of Parallel and Distributed Computing (JPDC), ACM MM, IEEE Transactions on Multimedia, Multimedia Tools and Applications, etc.



Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology. He received a Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology, a Ph.D. in Computer Science from Georgia State University. His current research interests include AI Security, Data-driven Security, Privacy and Data Analytics. He is a member of IEEE and ACM and was the Membership Chair of the IEEE Student Branch at Georgia State (2012-2013).

Qinming He is currently a Professor in College of Computer Science & Technology at Zhejiang University. He received his BS, MS and Ph.D. degrees in Computer Science from Zhejiang University, P. R. China in 1985, 1988 and 2000 respectively. His research interests include data mining and blockchain system security.

Yang Xiang Professor Yang Xiang received his PhD in Computer Science from Deakin University, Australia. He is currently a full professor and the Dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia. His research interests include cyber security, which covers network and system security, data analytics, distributed systems, and networking. He is also leading the Blockchain initiatives at Swinburne. In the past 20 years, he has been working in the broad area of cyber security, which covers network and system security, AI, data analytics, and networking. He has published more than 300 research papers in many international journals and conferences. He is the Editor-in-Chief of the SpringerBriefs on Cyber Security Systems and Networks. He serves as the Associate Editor of IEEE Transactions on Dependable and Secure Computing and IEEE Internet of Things Journal, and the Editor of Journal of Network and Computer Applications. He served as the Associate Editor of IEEE Transactions on Computers and IEEE Transactions on Parallel and Distributed Systems. He is the Coordinator, Asia for IEEE Computer Society Technical Committee on Distributed Processing (TCDP). He is a Fellow of the IEEE.