EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Emergency demand response in edge computing

Zhaoyan Song[1], Ruiting Zhou[1,2*], Shihan Zhao[1], Shixin Qin[1], John C.S. Lui[2] and Zongpeng Li[1]

*Correspondence:
ruitingzhou@whu.edu.cn
[1]Wuhan University, Bayi Road, Wuchang District, 430070 Wuhan, China
[2]Chinese University of Hong Kong, Shatin, NT, 999077 Hong Kong, China

Abstract
A cloudlet is a small-scale cloud datacenter deployed at the network edge to support mobile applications in proximity with low latency. While an individual cloudlet operates on moderate power, cloudlet clusters are well-suited candidates for emergency demand response (EDR) scenarios due to substantial electricity consumption and job elasticity: mobile workloads in the edge often exhibit elasticity in their execution. To efficiently carry out edge EDR via cloudlet cluster control, two fundamental problems need to be addressed: how to incentivize the participation of cloudlet clusters and how to schedule and allocate workloads in each cluster to satisfy EDR requirements. We propose a two-stage control scheme, consisting of (i) an auction mechanism to motivate clusters' voluntary energy reduction and select participants with the minimum social cost and (ii) an online task scheduling algorithm for chosen clusters to dispatch workloads to guarantee target EDR power reduction. Using the primal-dual optimization theory, we prove that our control scheme is truthful, individually rational, runs in polynomial time, and achieves near-optimal performance. Large-scale simulation studies based on real-world data also confirm the efficiency and superiority of our scheme over state-of-the-art algorithms.

Keywords: Emergency demand response, Edge computing, Auction mechanism, Online scheduling, Primal-dual optimization

1 Introduction

Cloudlet, in the form of a small datacenter, is a new computing paradigm that extends today's cloud architecture. As the middle tier of a 3-tier hierarchy: mobile or IoT device—cloudlet—cloud, cloudlet is often placed at the edge of the network to provide low-latency and high bandwidth services for nearby mobile or IoT devices [1]. A cloud service provider often deploys a cluster of cloudlets to serve mobile users, since the computing power of an individual cloudlet is limited. The wide distribution of cloudlets not only increases the edge network's capacity and coverage but also brings flexibility in workload management [2, 3].

It is quintessential for a power network to be stable and reliable. When an emergency happens (e.g., extreme weather conditions), supply scarcity needs to adjust immediately to avoid involuntary service interruptions [4]. Besides clouds, cloudlet clusters

Springer Open

© The Author(s). 2020 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

now serve as an important force in emergency demand response (EDR). While individual cloudlet uses a moderate amount of electricity, a cluster of cloudlets consumes substantial electricity. Furthermore, edge computing tasks (e.g., video surveillance and analysis) are often elastic [5]. Hence, the task execution is flexible, which means that a task can tolerate a certain level of delay. The above features make cloudlet clusters well suited to participate in EDR programs to stabilize the power grid by reducing and temporally shifting peak loads. To realize edge EDR via cloudlet cluster control, two fundamental challenges need to be addressed. First, cloudlet clusters are often operated by different service providers at their own cost. *How to incentivize them to voluntarily participate in EDR* is a challenging problem. An efficient market mechanism must be created to select cost-conscious participants and reward them accordingly. Second, for a chosen cloudlet cluster which receives task execution requests from mobile users online, the main issue is *how to make decisions on accepting/declining tasks and schedule accepted tasks, such that the target energy reduction is satisfied and its utility is maximized?*

Many efforts have been made on incentive mechanism design for demand response and task scheduling. However, they cannot be applied to edge EDR directly. Previous work either only focuses on electricity procurement [6, 7] or does not consider the electricity consumption in scheduling algorithm design [8, 9]. More discussions can be found in Section 2. The goal of this work is to design an online control scheme for cloud clusters to participate in EDR. We propose a two-stage approach to address the above two problems. We first propose an auction mechanism to stimulate the participation of cloudlet clusters, such that (i) the auction is computationally efficient; (ii) the auction is truthful and individually rational—all participants receive non-negative and maximum utility by biding their true cost; (iii) the power reduction goal in EDR is met with minimum possible social cost. We then design an online task scheduling algorithm, tailored for a participant cloudlet cluster, to dispatch workloads to guarantee target EDR power consumption. Our algorithm has the following goals: (i) the algorithm is time efficient and makes task admission and scheduling decisions immediately upon the arrival of each task, (ii) the total power consumption in the specified EDR time window plus the local electricity generation can satisfy the EDR requirement, (iii) the utility of the cluster, which is completed tasks' value minus the cluster's operation cost, is maximized. Many edge-computing tasks have flexibility in both placement and job completion time, so their workload can be distributed across different cloudlets. In addition, they allow a certain level of delay, and the task's value depends on the degree of deadline violation. The operating cost primarily comprises of server maintenance cost and local generation cost. Our contributions are listed as follows:

First, on the first stage of EDR, the smart grid acts as the auctioneer to elicit bids from cloudlet clusters. Each cluster specifies its energy reduction and corresponding remuneration it asks for. We formulate the social cost minimization problem as an integer linear program (ILP), with a constraint to satisfy the EDR reduction target. The problem is proven to be NP-hard. To solve the problem, we convert it to an equivalent ILP and adopt the primal-dual method to obtain the solution based on its dual problem. We show that our algorithm runs in polynomial time and achieves 2-approximation in social cost. We further propose a payment rule to determine winning clusters' reward, which guarantees truthfulness as well as individual rationality.

Second, we proceed to consider the second stage, where the winning cluster manages its tasks scheduling to satisfy the EDR power consumption requirement. We design an online scheduling algorithm to determine whether a task should be accepted or not, when and where the accepted task should be processed, and the amount of local electricity generation. The cluster's utility maximization problem is formulated as a convex problem. To eliminate the non-linear constraints that capture the task's temporal demand, we introduce a set of new variables for each task to represent feasible schedules. Although the new formulation has an exponential size of variables, we demonstrate that it can be solved in polynomial time. A primal-dual method is applied to the new formulation and its dual problem. Two dual variables can be interpreted as the unit workload price and the unit local generation cost, respectively. They are used as the threshold to compute the best schedule with the maximum utility for each task. The proof for feasibility, efficiency, and a good competitive ratio are conducted in our theoretical analysis.

Third, we conduct large-scale simulations based on the real-world circumstances. On the first stage of EDR, our algorithm performs better than the theoretical 2-approximation worst case, achieving nearly 1.05 in approximation ratio at a 50 clusters scale and 1.025 with 400 clusters. On the second stage of EDR, we also obtain several noticeable results: (i) our algorithm achieves a low competitive ratio (< 1.6); (ii) our algorithm beats two benchmark algorithms, a greedy algorithm based on the idea of [10] and a first-come, first-served algorithm applying the scheme in [11], in terms of cluster utility, either in different problem scales or time slots; (iii) our algorithm can control the peak usage of electricity and save up to 49.8%, 22.4% of the local generation, compared with other two algorithms, respectively; and (iv) the execution time of our algorithm grows mildly as problem scale increases, proving that our algorithm runs in polynomial time. Through extensive simulations in both theoretical and practical circumstances, we demonstrate the superiority of our method.

In the rest of this paper, we discuss related work in Section 2 and introduce the system model in Section 3. Algorithm designs for the first and second stage of edge EDR are presented in Sections 4 and 5, respectively. Section 6 evaluates the performance of proposed algorithms and Section 7 concludes the paper.

2 Related work

Demand response and edge EDR. Previous analyses for EDR tend to investigate the mutual impact between service providers and customers [12, 13], without fully considering energy provisioning. A series of recent studies exist on the reduction goal and social welfare maximization in demand response. For instance, in [14], a storage-assisted system is considered, with batteries and plug-in vehicles helping balance between supply and user demand. Demand response in residential power allocation is investigated by Ma et al. [15], in which two types of electricity applications are analyzed. Sun et al. [16] study an eco-friendly objective for reducing diesel generation. Chen et al. [17] argue that edge computing represents a natural subject of EDR. Their work differs from ours in that they fix the execution time slot of each workload and assume a simplified linear cost for deadline violation; our model is comparatively more practical. There are also several studies on online task allocation in EDR, including [18] and [19]. The former only maximizes the operator's cost while the latter ignores local generator consumption. Our work presents a general scenario in edge computing where auction and scheduling take place, targeting EDR in the provision of electricity supply and scheduling tasks online with more flexibility.

Datacenter EDR. Existing studies mainly focus on EDR with colocation data centers. Each colocation tenant submits a bid, including its energy reduction and cost, to the operator controller who provisions electricity and other services to them. Ren and Islam [20] are the first to study "split incentive" in a demand response scenario. They propose a mechanism, *iCODE*, based on a reverse auction. More efforts can be seen in the work of Zhang et al. [6], which designs an efficient truthful mechanism to achieve 2-approximation in colocation social cost. Zhou et al. [21] investigate a decentralized method in a geo-distributed data center. Zhou et al. [22] and Chen et al. [7] both study the FPTAS auction to design a truthful and energy-saving scheme. However, their assumptions of allocation in response to demand are barely offline situations. In our circumstance, the challenge escalates as a series of tasks arrive at the cloudlet stochastically over time.

Online Scheduling. Online scheduling is fundamental in cloud computing, where computing and storage resources are limited for task processing. Zhang et al. [8] propose a *primal-dual* style auction to dynamically allocate tasks into different VMs, in which the time windows of users' bids are fixed. Subsequently, Zhou et al. [9] develop the compact exponential method to handle hard and soft deadline constraints for job execution, showing more elastic than [8]. Scheduling jobs online is also studied in a general way to suit every aspect of daily life. Specifically, Garg et al. [23] and Devanur et al. [24] study online scheduling jobs on unrelated machines, with the first paying attention to weighted flow time and the latter considering arbitrary power functions of the machine. Their researches share a *primal-dual* based framework with our model, but their problems are different from ours. Agrawal et al. [25] introduce a general version to solve online problems with a concave objective and convex constraints. However, they assume the inputs are independent and identically distributed. We study edge computing, targeting not only truthfulness and efficiency for the online mechanism appeared in cloud computing, but also the flexibility to schedule tasks in heterogeneous clusters.

3 System model

3.1 System overview

We consider a community where a smart grid provisions electricity to multiple heterogeneous cloudlet clusters. Each cluster signs a contract with the grid, specifying the electricity demand in a certain time period and the corresponding payment. While the grid is only in charge of power supply, the cloudlet cluster is responsible for their own operation, i.e., considering how to schedule computing tasks, such that its utility is maximized under limited energy. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. The cloudlet clusters are denoted as $[I] = \{1, 2, \ldots, I\}$. Cluster i consists of heterogeneous cloudlets, denoted by $[L_i] = \{1, 2, \ldots, L_i\}$, $\forall i \in [I]$. A set of tasks $[J_i] = \{1, 2, \ldots, J_i\}$, $\forall i \in [I]$ run in cluster i where they reside, receiving their workloads at end users via access points and transmitting them to the particular service providers.

3.2 Emergency demand response via cloudlet clusters

EDR process. According to an agreement signed by the cloudlet operators and the smart grid, when an emergency event takes place, the smart grid acts as an auctioneer and sends signals to cloudlet clusters in the community to specify the total energy reduction target E_{all} in the following T time slots. In response, each cloudlet cluster voluntarily submits a bid. Recently, in edge computing, the tasks are mainly involved with video surveillance, whose power consumption usually varies little [26]. Based on records in the past, the clusters estimate an energy reduction when the emergency takes place. Cluster i submits a bid (c_i, E_i), where E_i is the amount of power consumption it is willing to shed and c_i is the corresponding remuneration asked for. After receiving bids from cloudlet clusters, the smart grid determines which clusters are chosen for the EDR, as well as how much it should pay for the selected clusters. Figure 1 illustrates the first stage in the EDR event.

EDR decision variable. We introduce a binary variable z_i for each bid. If $z_i = 1$, the bid submitted by cluster i is successfully chosen for EDR and earns a reward P_i from the grid; otherwise, the bid is not selected for EDR.

Truthful auction. Let v_i denote the true cost of E_i. v_i can be obtained from the previous data, usually proportional to the workload and duration. P_i is the reward for winning the bid in EDR. Cluster i's utility with bidding price c_i is:

$$u_i(c_i) = \begin{cases} P_i - v_i, & \text{if} \quad z_i = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Each cluster is assumed to be selfish yet rational and aims to maximize its own utility. Cluster i may lie about its cost, i.e., $c_i \neq v_i$, if doing so leads to a higher utility. Since our goal is to minimize the social cost of the community, it is important to elicit truthful bids from clusters.

Definition 1 *(Truthful auction): An auction is a truthful auction if for any cluster i, its dominant strategy is to bid with its true cost, and its utility is maximized, i.e.,* $u_i(v_i) \geq u_i(c_i), \forall c_i \neq v_i$.

Definition 2 *(Individual rationality): Clusters always obtain non-negative utility, i.e.,* $u_i(c_i) \geq 0$.

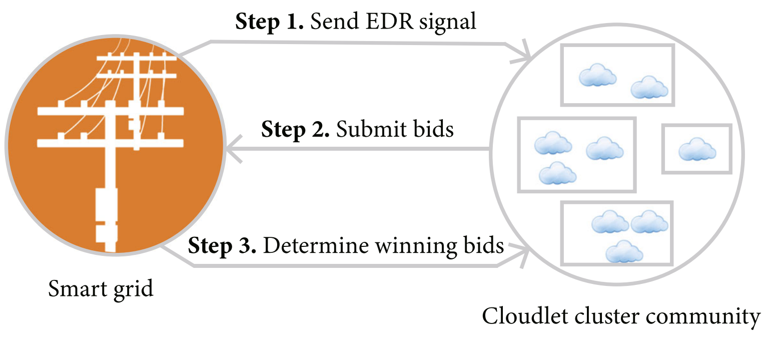

Step 1. Send EDR signal
Step 2. Submit bids
Step 3. Determine winning bids
Smart grid
Cloudlet cluster community

Fig. 1 A demonstration of EDR via cloudlet clusters control

Definition 3 *(Social welfare, social cost): The social welfare is the sum of the grid's utility* $-\sum_{i\in[I]} P_i$ *and clusters' utility* $\sum_{i\in[I]}(P_i - v_i z)$*. Maximizing the social welfare is equivalent to minimizing the social cost* $\sum_{i\in[I]} v_i z_i$*, since payments cancel themselves.*

EDR problem formulation. Our goal is to minimize the social cost in the community while ensuring the total reduction meets the EDR requirement E_{all}. The social cost minimization problem under truthful bidding ($c_i = v_i$) can be formulated into the following integer linear program (ILP):

$$\text{minimize} \sum_{i\in[I]} c_i z_i \tag{2}$$

subject to:

$$\sum_{i\in[I]} E_i z_i \geq E_{all}. \tag{2a}$$

$$z_i \in \{0,1\}, \forall i \in [I]. \tag{2b}$$

3.3 Scheduling in cloudlet clusters

Computing task information. Assume that cluster i is chosen at EDR, where a total number of L_i cloudlets are in the cluster. Each cloudlet $l \in [L_i]$ can process at most R_l workloads. Let $[J_i]$ denote the task set for i. Each task j in $[J_i]$ is expressed by a tuple: $\Gamma_{ij} = \{b_{ij}, a_{ij}, d_{ij}, w_{ij}, \lambda_{ij}, f_{ij}(\tau_{ij})\}$, where b_{ij} is the value of task j if it completes before its deadline. a_{ij} and d_{ij} are the arrival time and the deadline for task j, w_{ij} is the total number of time slots required to complete the task. We assume that the deadline for each task should be no longer than the end of the EDR period; otherwise, auctions must be executed again for those tasks. The workload in one time slot is λ_{ij}, so the total workload of task j is $w_{ij}\lambda_{ij}$. τ_{ij} refers to the level of deadline violation, whose penalty function is denoted by $f_{ij}(\tau_{ij})$. $f_{c_{ij}}(\tau_{ij})$ is a piecewise function of $f_{ij}(\tau_{ij})$, $f_{ij}(\tau_{ij})$ is a non-decreasing function, and $f_{ij}(0) = 0$, defined as:

$$f_{ij}(\tau_{ij}) = \begin{cases} f_{c_{ij}}(\tau_{ij}), & \tau_{ij} \in [0, T - d_{ij}] \\ +\infty, & \text{otherwise.} \end{cases} \tag{3}$$

Decision variable. As shown in Fig. 2, in the second stage, every cloudlet cluster operates individually and schedules computing tasks to satisfy EDR energy consumption. For simplicity, we omit i in the element of the tuple for cluster i. We introduce two additional binaries, x_{ij} and $y_{ijl}(t)$, to indicate whether task j in cluster i is scheduled and whether task j is scheduled at cloudlet l at time slot t. Important notations are summarized in Table 1 for ease of reference.

Problem formulation. Cluster i aims to maximize its own utility, i.e., the sum of its reward and value minus the sum of the delay penalty, local generation cost, and main-

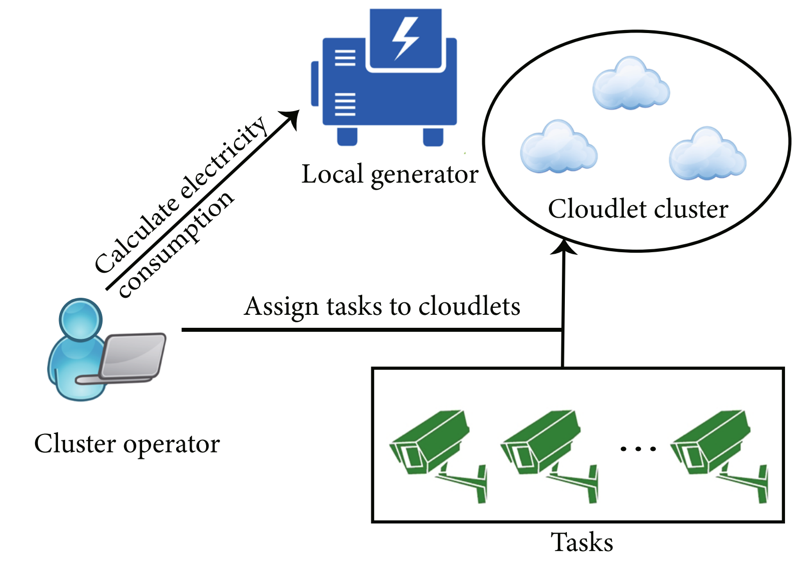


Fig. 2 Task scheduling in the cluster

tenance cost. The objective is to maximize $P_i + \sum_{j\in[J]} b_{ij}x_{ij} - \sum_{j\in[J]} f_{ij}(\tau_{ij}) - pu_g - \theta_i$, where P_i indicates its reward, θ_i is its maintenance cost. Since P_i and θ_i are constants, we can omit them for simplicity in our integer program. p is the per-unit local generation cost, and u_g is the amount of local generation. The optimization problem is formulated as follows:

Table 1 Major notations

I	# of cloud clusters
T	# of time slots
L_i	# of cloudlets in cluster *i*
J_i	# of tasks in cluster *i*
D_i	Amount of electricity that *i* purchased from the grid
c_i	Asking price of E_i
E_i	Electricity reduction that cluster *i* can offer
P_i	Cluster *i*'s reward earned from the smart grid
z_i	Cluster *i*'s bid is accepted (1) or not (0)
E_{all}	EDR target
b_{ij}	Value of task *j* in cluster *i*
a_{ij}	The arrival time of task *j* in cluster *i*
d_{ij}	The deadline of task *j* in cluster *i*
w_{ij}	# of time slots required for task *j* in cluster *i*
λ_{ij}	The workload of task *j* in cluster *i* in one slot
τ_{ij}	# of slots that pass the deadline for task *j* in *i*
$Z_l(t)$	Marginal price of unit workload at cloudlet *l* at *t*
$R_l(t)$	The amount of allocated resources in cloudlet *l* at *t*
θ_i	Maintenance cost for cluster *i*
p	Local generation cost per unit
e_l^t	Energy consumption of cloudlet *l* at *t*
u_g	Total local generation
x_{ij}	Task *j* in cluster *i* is accepted (1) or not (0)
$y_{ijl}(t)$	Task *j* is allocated in cluster *i*'s *l* at *t* (1) or not (0)

$$\text{maximize} \sum_{j\in[J]} b_{ij}x_{ij} - \sum_{j\in[J]} f_{ij}(\tau_{ij}) - pu_g \tag{4}$$

$$\text{subject to}: \sum_{j\in[J]} \lambda_{ij}y_{ijl}(t) \leq R_l, \forall t \in [T], \forall l \in [L], \tag{4a}$$

$$\sum_{t\in[T]}\sum_{l\in[L]} e_l^t \leq D - E + u_g, \tag{4b}$$

$$t\sum_{l\in[L]} y_{ijl}(t) \leq d_{ij} + \tau_{ij}, \forall t \in [T], \forall j \in [J] : a_{ij} \leq t, \tag{4c}$$

$$\sum_{l\in[L]} y_{ijl}(t) \leq 1, \forall j \in [J], \forall t \in [T], \tag{4d}$$

$$w_{ij}x_{ij} = \sum_{l\in[L]}\sum_{t\in[T]} y_{ijl}(t), \forall j \in [J], \tag{4e}$$

$$y_{ijl}(t), x_{ij} \in \{0, 1\},$$

$$\forall j \in [J], \forall l \in [L], \forall t \in [T], \tag{4f}$$

$$u_g \geq 0, \tau_{ij} \geq 0, \forall j \in [J]. \tag{4g}$$

In the above problem, e_l^t denotes the electricity consumption of cloudlet l at time slot t. An empirical study on cloudlet [27] formulates the energy consumption through a linear function: $e_l^t = \left(N_l P_{idle}^l + \left(P_{peak}^l - P_{idle}^l\right)\sum_{j\in J}\lambda_{ij}y_{ijl}(t)\right)\cdot \text{PUE}_l$, where N_l represents the number of running servers in cloudlet l. P_{idle} is the power consumption when the server is idle and P_{peak} is the sever power when the cloudlet is fully utilized. $\sum_{j\in J}\lambda_{ij}y_{ijl}(t)$ is the amount of workload. PUE_l, the power usage efficiency ratio, is determined by statistical records of the ratio between the datacenter facility power and computational consumption. Next, since the EDR requires the chosen cloudlet cluster to reduce E_i electricity, its expected power consumption is the original demand minus the EDR requirement, $D - E$.

Constraint (4a) guarantees that in each time slot, the cloudlet l has enough computing resource to execute tasks. Constraint (4b) ensures that the total power consumption does not exceed the sum of the EDR requirement and local generation. For any possible tasks to be scheduled, they should run between the arrival time and the deadline, which is described by (4c). Additionally, in (4d), we assume that each task runs on one cloudlet at most. Constraint (4e) connects two binary variables, x_{ij} and $y_{ijl}(t)$, to guarantee sufficient execution.

Challenges. We notice that the first ILP (2) is the classical knapsack problem. The challenge escalates as we need to ensure truthfulness and individual rationality in the auction design. For the second problem (4), if we let $u_g = 0$ and $\tau_{ij} = 0, \forall j \in J$, as well as ignore (4c), it becomes a knapsack problem. It is known to be NP-hard, let alone the difficulties concerning online scheduling. In Section 4, we propose a 2-approximation algorithm based on the primal-dual method to select winning clusters and compute payments for EDR event. In Section 5, we propose an online algorithm to schedule tasks while satisfying EDR requirement in the winners.

4 Methods for EDR auction mechanism

In Section 4.1, we propose an efficient algorithm to determine the winners and calculate payments in the first stage of EDR event. We analyze its performance in Section 4.2.

4.1 EDR auction design

In the winner determination process, we introduce a set of inequalities [28] and reformulate ILP (2) to obtain an approximate solution. Consider a set of cloudlet clusters S, i.e., $S \subset [I]$, we denote $\Delta(S) = E_{all} - \sum_{i\in S} E_i$ as the remaining electricity reduction goal when all bids in S are accepted. Furthermore, we define a variable, $E_i(S)$, to represent how a bid excluded from S can fill the gap between E_{all} and $\Delta(S)$, i.e., $E_i(S) = \min\{E_i, \Delta(S)\}$. We reformulate ILP (2) into:

$$\text{minimize} \sum_{i\in[I]} c_i z_i \tag{5}$$

subject to:

$$\sum_{i\in[I]\backslash S} E_i(S) z_i \geq \Delta(S), \forall S \subset [I] : \Delta(S) > 0. \tag{5a}$$

$$z_i \in \{0,1\}, \forall i \in [I]. \tag{5b}$$

Constraint (5a) states that when $S \subset [I]$ is chosen, we enumerate all possible items in the rest of the set, $[I] \backslash S$, to cover the EDR target. It can be seen that all feasible integer solutions in ILP (2) are feasible in (5) and vice versa. Therefore, the two ILPs have equivalent optimal solutions. Next, we introduce dual variable $m(S)$ to constraint (5a) and formulate the dual of (5) as:

$$\text{maximize} \sum_{\forall S\subset[I]:\Delta(S)>0} \Delta(S) m(S) \tag{6}$$

subject to:

$$\sum_{\forall S\subset[I]: i\subset[I]\backslash S, \Delta(S)>0} E_i(S) m(S) \leq c_i, \forall i \in [I], \tag{6a}$$

$$m(S) \geq 0, \forall S \subset [I] : \Delta(S) > 0. \tag{6b}$$

Based on the primal-dual method, when the i^*th constraint (6a) becomes tight, z_{i^*} in (5) is larger than 0. Due to the intrinsic $0-1$ nature, z_i will be automatically adjusted to 1. The basic idea is we initialize an empty set $\mathcal{R}$ to represent the accepted bid in accumulation and then gradually increase the value of dual variables, $m(S)$, to reach the energy reduction objective in the aggregated accepted bids. In each round, we add one cluster to the set $\mathcal{R}$ until no more clusters are left or the EDR target is successfully achieved.

AMEDR aims to tighten the constraints (6a) so that the corresponding cluster is selected and its indicated variable is set to 1. First, we initialize all primal and dual variables in line 1. Line 2 states that the algorithm terminates only when the EDR demand is met. We increase the dual variable until one of the dual constraints is tight, as is shown in line 3. After calculating the minimum required value of the dual variable $m(S)$, the corresponding cluster i^* is selected, added to $\mathcal{R}$ in lines 3–4. Next, we design a payment rule to ensure truthfulness in lines 5–9. By picking the second smallest value, $m(\widetilde{\mathcal{R}})$ from all $m(S)$, we find the critical bid and compute the payment based on it. Then, cost values are updated in line 10. Finally, the winner set $\widehat{S}$ and their payments are found. Although many other auction mechanisms have been designed for such problems, our method has nice properties which are listed as follows.

Algorithm 1 A 2-Approximation Auction Mechanism for EDR: AMEDR

```
Input: $\{c_i, E_i\}_{\{i \in [I]\}}, E_{all}$
Output: $\widehat{S}, P(\widehat{S})$
1:  Initialize $z_i = 0, \forall i \in [I]; m(S) = 0, \forall S \subset [I]; \mathcal{R} = \emptyset$.
2:  while $\Delta(\mathcal{R}) > 0$ do
3:      $i^* = \arg\min_{i \in [I]} \{\frac{c_i}{E_i(\mathcal{R})}\}$.
4:      $z_{i^*} = 1; \mathcal{R} = \mathcal{R} \cup \{i^*\}$.
5:      $m(\mathcal{R}) = \frac{c_{i^*}}{E_{i^*}(\mathcal{R})}$
6:      $[I] = [I] \setminus i^*$.
7:      $\tilde{i} = \arg\min_{i \in [I]} \{\frac{c_i}{E_i(\mathcal{R})}\}$.
8:      $m(\widetilde{\mathcal{R}}) = \frac{c_{\tilde{i}}}{E_{\tilde{i}}(\mathcal{R})}$.
9:      $P_{i^*} = c_{i^*} + (m(\widetilde{\mathcal{R}}) - m(\mathcal{R}))E_{i^*}(\mathcal{R})$.
10:     $c_i = c_i - E_i(\mathcal{R})m(\mathcal{R}), \forall i \in [I]$.
11: end while
12: $\widehat{S} = \mathcal{R}$.
13: $P(\widehat{S}) = \{P_{i^*}\}_{i^* \in \widehat{S}}$.
```

4.2 Theoretical analysis

4.2.1 Correctness and polynomial time

Lemma 1 *ILP (2) is equivalent to ILP (5).*

All missing proofs can be found in the Appendix.

Theorem 1 *AMEDR produces a feasible solution for ILP (2) (5) and dual LP (6).*

Theorem 2 *AMEDR runs in polynomial time.*

4.2.2 Approximation ratio

Definition 4 (approximation ratio) *: Let the social cost calculated by AMEDR be SC and the optimal objective value of ILP (2) be OPT. The approximation ratio is the upper bound ratio of SC to OPT.*

Theorem 3 *The approximation ratio of AMEDR is 2.*

Proof The primal problems (2) (5) have equivalent optimal value, OPT, with the dual problems (6). AMEDR terminates only when $\Delta(\mathcal{R}) \leq 0$. The last cluster picked by the algorithm is denoted as i_n, so we have $\Delta(\widehat{S} \setminus i_n) > 0$.

Since every i in $\widehat{S} \setminus i_n$ has a tight constraint in (6a), we reformulate the social cost among such items into:$\sum_{i \in \widehat{S}} \sum_{\forall S \subset [I]: i \subset [I] \setminus S, \Delta(S) > 0} E_i(S)m(S)$. However, due to the initialization, the non-negative variables $m(S)$ remains zero unless $S \subset \widehat{S} \setminus i_n$. The equality above is simplified as $\sum_{S \subset \widehat{S}} m(S) \sum_{i \in \widehat{S} \setminus S'} E_i(S')$. We further transform $\sum_{i \in \widehat{S} \setminus S} E_i(S)$ into

$\sum_{i\in\widehat{S}\backslash S} E_i(S) = \sum_{i\in\widehat{S}\backslash i_n} E_i - \sum_{i\in S} E_i + E_{i_n}(S)$, which is smaller than $E_{all} - \sum_{i\in S} E_i + E_{i_n}(S)$. Moreover, it is no larger than $2\Delta(S)$. Since $\Delta(\widehat{S}\backslash i_n) > 0$ holds, $SC \le 2\sum_{S'\subset\widehat{S}} m(S)\Delta(S) \le 2OPT$, ensuring that the approximation ratio is 2. □

4.2.3 Truthfulness and individual rationality

Theorem 4 *An auction mechanism is truthful if [29, 30]: (i) as the costs submitted by clusters decrease, and z_i is non-decreasing in its value; (ii) the winning bid payment is critical.*

Lemma 2 *Algorithm AMEDR is bid-monotonic: if cluster i^* submits an alternative cost $c_{\bar{i}}$ subject to $c_{\bar{i}} < c_{i^*}$ and $z_{i^*} = 1$, then $z_{\bar{i}} = 1$.*

Lemma 3 *The payment design in AMEDR is critical: the cost c_{i^*} submitted by winning bid i^* should be not larger than the payment P_{i^*}. If i^* bids c_{i^*} which is larger than P_{i^*}, i^* will fail in the auction.*

Theorem 5 *AMEDR is a truthful auction.*

Proof By combining Lemma 2, 3 and the definition in Theorem 4, we finish the proof. □

Theorem 6 *Algorithm AMEDR ensures individual rationality in each successful bid: the payments for bids are at least the cost of them.*

Proof Lemma 3 guarantees that any bid that violates $P_i < c_i$ cannot be chosen to win, and therefore, *individual rationality* holds during the auction. □

5 Methods for online task scheduling

In the second stage of EDR event, supposing cluster i is selected, a primal-dual algorithm is proposed to schedule tasks in Section 5.1. The theoretical analysis is presented in Section 5.2.

5.1 Online scheduling design

Reformulation. We consider how to schedule tasks in the chosen cloudlet cluster i. To deal with non-conventional scheduling constraints in (4c) and (4e), we reformulate (4) into an equivalent convex problem. Though the new formulation is packed with an exponential number of variables, it greatly simplifies the subsequent algorithm design. The new problem is formulated as follows:

$$\text{maximize} \sum_{j\in J}\sum_{h\in\zeta_{ij}} b'_{ijh}\chi_{ijh} - g(u) \tag{7}$$

subject to:

$$\sum_{j\in[J]}\sum_{h:t\in T(h),l\in L(h)} \lambda_{ij}\chi_{ijh} \le R_l, \forall l \in [L], \forall t \in [T], \tag{7a}$$

$$\sum_{l\in[L]}\sum_{j\in[J]}\sum_{h:l\in L(h)}\sum_{t:t\in T(h)} \beta_l\lambda_{ij}\chi_{ijh} \le u, \tag{7b}$$

$$\sum_{h \in \zeta_{ij}} \chi_{ijh} \leq 1, \forall j \in [J], \tag{7c}$$

$$\chi_{ijh} \in \{0,1\}, \forall j \in [J], \forall h \in \zeta_{ij}, \tag{7d}$$

$$u \geq 0. \tag{7e}$$

In the above program, ζ_{ij} is the set of all feasible schedules for task j. A feasible schedule is a vector $h = (x_{ij}, \{y_{ijl}(t)\}_{\forall l \in [L], \forall t \in [T]}, \tau_{ij})$ that satisfies constraints (4c) and (4e). Binary variable, χ_{ijh}, indicates whether task j is accepted and scheduled according to schedule h ($\chi_{ijh} = 1$) or not ($\chi_{ijh} = 0$). b'_{ijh} is the task value based on schedule h, i.e., $b'_{ijh} = b_{ij} - f_{ij}(\tau_{ij})$. $T(h)$ and $L(h)$ are the set of time slots and cloudlets indicating when and where task j is running based on schedule h. $g(u)$ is the local generation cost of cluster i. We let $D' = D - E - T(\sum_{l \in [L]} N_l P^l_{idle}) \cdot \mathrm{PUE}_l$ and $u = D' + u_g$. $g(u)$ can be defined as a piecewise function as follows:

$$g(u) = \begin{cases} 0, & u \leq D' \\ p(u - D'), & u > D'. \end{cases}$$

$g(u)$ indicates the energy consumption either below or above the EDR cap. We simplify the LHS of (4b), and let $\beta_l = (P^l_{peak} - P^l_{idle}) \cdot \mathrm{PUE}_l$. Constraints (7a) and (7b) are equivalent to (4a) and (4b).

Though we reformulate the problem into a packing structure, many challenges are still ahead of us. A *primal-dual* technique can be applied to solve the problem in polynomial time. By introducing dual variables $Z_l(t), C$ and ϕ_{ij}, as well as relaxing $\chi_{ijh} \in \{0,1\}$ to $\chi_{ijh} \geq 0$, the dual of the primal problem is:

$$\text{minimize} \sum_{l \in [L]} \sum_{t \in [T]} Z_l(t) R_l + \sum_{j \in [J]} \phi_{ij} + g^*(C) \tag{8}$$

subject to:

$$\phi_{ij} \geq b'_{ijh} - \sum_{l \in L(h)} \sum_{t \in T(h)} Z_l(t) \lambda_{ij} - \sum_{l \in L(h)} \sum_{t \in T(h)} C \beta_l \lambda_{ij}, \quad \forall j \in [J], \forall h \in \zeta_{ij}, \tag{8a}$$

$$Z_l(t), C, \phi_{ij} \geq 0, \forall j \in [J], \forall l \in L(h), \forall t \in T(h) \tag{8b}$$

where $g^*(C)$ is the Fenchel conjugate [31] of the function $g(u)$:

$$g^*(C) = \sup_{u \geq 0} \{Cu - g(u)\} = \begin{cases} +\infty, & u > D' \text{ and } C > p. \\ CD'. & \text{otherwise} \end{cases}$$

Allocation and scheduling. Based on the idea of *complementary slackness* [32], each χ_{ijh} has a corresponding constraint in (8a). Only when the constraint goes tight, can χ_{ijh} be updated to 1. In that case, we automatically assign 1 to x_{ij} and $\{y_{ijl}(t)\}_{l \in L(h), t \in T(h)}$ before we renew dual variables $\phi_{ij}, Z_l(t)$ and C. Since ϕ_{ij} in dual constraint is non-negative, we assign ϕ_{ij} as the maximum value between 0 and the RHS of (8a):

$$\phi_{ij} = \max\left\{0, \max_{h\in\zeta_{ij}}\left\{b'_{ijh} - \sum_{l\in L(h)}\sum_{t\in T(h)} Z_l(t)\lambda_{ij} - \sum_{l\in L(h)}\sum_{t\in T(h)} C\beta_l\lambda_{ij}\right\}\right\}. \quad (9)$$

When $\phi_{ij} > 0$, dual constraint (8a) holds tight so that the related primal variable $\chi_{ijh} > 0$. In this case, task j is accepted, and h is the corresponding schedule for j. Otherwise, if $\forall h \in \zeta_{ij}, \phi_{ij} = 0$, $b'_{ijh} - \sum_{l\in L(h)} \sum_{t\in T(h)} Z_l(t)\lambda_{ij} - \sum_{l\in L(h)} \sum_{t\in T(h)} C\beta_l\lambda_{ij} \leq 0$, this task is rejected.

The reason can be explained as follows: If we interpret $Z_l(t)$ as the per unit workload per unit time slot price for cloudlet l, and C as the per unit local generation cost, then the RHS of (8a) is the utility of task j. Therefore, when $\phi_{ij} > 0$, the utility of task j becomes positive so that the cluster is willing to process it. Note here $C = 0$ when the task does not exceed the EDR cap; otherwise, $C = p$. p is the local generation cost per unit. If there is a delay in the task completion, b'_{ijh} should contain the penalty expense; otherwise, $b'_{ijh} = b_{ij}$. When the value of the RHS of (9) is 0, the task is rejected. Equality (9) determines the cloudlets and slots to schedule tasks for the maximum utility, which is a key to the utility maximization.

Algorithm 2 Primal-Dual Based Online Allocation PD

```
Input: ${\beta_l, R_l}_{\{l\in L\}}, C, N, M, D'$.
1: Initialize $x_{ij} = 0, y_{ijl}(t) = 0, \tau_{ij} = 0, \phi_{ij} = 0, R_l(t) = 0, u = 0, \forall j \in [J], \forall l \in [L], \forall t \in [T]$, by default.
2: On the arrival of task $j$
3: Run CORE($\{R_l(t), Z_l(t)\}_{\{l\in[L], t\in[a_{ij},T]\}}, p, \Gamma_{ij}, u$).
4: if $x_{ij} = 1$ then
5:     Schedule the $j$th task according to $y_{ijl}(t)$.
6: else
7:     Reject the $j$th task.
8: end if
```

We next discuss the update of $Z_l(t)$. It is natural to think that as computing resource in a cloudlet decreases, the cluster may be reluctant to allocate more workload to this cloudlet. We develop a cost function for the cloudlet to reduce the possibility of accepting a task when it is almost fully occupied. The cost function is:

$$Z_l(t) = Z_l(R_l(t)) = \frac{N}{e\sigma}\left(\frac{e\sigma M}{N}\right)^{\frac{R_l(t)}{R_l}}.$$

Where $Z_l(t)$ starts at $\frac{N}{e\sigma}$ and grows exponentially with the increase of $R_l(t)$. N refers to the minimum value per unit workload per unit slot, and $\sigma = \frac{T}{\min_j\{w_{ij}\}}$. By the time l is fully utilized, $Z_l(t)$ is close to M, the maximum value per unit workload per unit slot. More specifically, $N = \min_{j\in[J]} \frac{b_{ij}}{w_{ij}\lambda_{ij}}, M = \max_{j\in[J]} \frac{b_{ij}}{w_{ij}\lambda_{ij}}$.

Algorithm 3 One-Round Task Scheduling: CORE

```
Input: $\{R_l(t), Z_l(t)\}_{\{l\in[L],t\in[a_{ij},T]\}}, p, \Gamma_{ij}, u$.
1:  Initialize $H = \emptyset, \widetilde{H} = \emptyset, h_t = \{t\}; x_{ij} = 0, y_{ijl}(t) = 0, \phi_{ij} = 0, \forall l \in [L], \forall t \in [T]$, by default.
2:  Add $(t,l), t \in [a_{ij}, T], l \in [L]$ to $H$ if $R_l(t) + \lambda_{ij} \le R_l, \forall t \in [T], \forall l \in [L]$.
3:  for all $(t,l) \in H$ do
4:      Calculate $q(t,l) = Z_l(t)\lambda_{ij}$.
5:      if $u > D'$ then
6:          $q(t,l) = q(t,l) + C\beta_l\lambda_{ij}$.
7:      end if
8:  end for
9:  Find $l_t = \arg\min_{l:(t,l)\in H} q(t,l), \forall t$. Add $(t,l_t), \forall t$ to $\widetilde{H}$.
10: Arrange time slots by sequence, and denote the $w_{ij}$th slot as $t_0$.
11: for all $t' \in \widetilde{H} : t_0 \le t' \le T$ do
12:     Select $w_{ij} - 1$ slots in $\widetilde{H}$ with minimum $q(t,l), t < t'$ and add them to $h_{t'}$.
13:     Calculate $Q(t') = \sum_{t\in h_{t'}} q(t, l_t)$.
14:     if $t' > d_{ij}$ then
15:         Calculate $b_{ij} = b_{ij} - f_j(t' - d_{ij})$.
16:     end if
17: end for
18: $\phi_{ij}(t') = b_{ij} - Q(t'), \forall t' \in \widetilde{H}$.
19: Find $t^* = \arg\max_{t'\in\widetilde{H}:t_0\le t'\le T} Q(t')$.
20: if $\phi_{ij} > 0$ then
21:     $R_{l_t}(t) = R_l(t) + \lambda_{ij}, \forall t \in h_{t^*}$.
22:     $Z_l(t) = \frac{N}{e\sigma}\left(\frac{e\sigma M}{N}\right)^{\frac{R_l(t)}{R_l}}, \forall l \in [L], \forall t \in [T]$.
23:     $x_{ij} = 1; y_{ijl}(t) = 1, \forall t \in h_{t^*}$.
24:     $u = u + \sum_{l:(t,l)\in H, t\in h_{t^*}} \beta_l\lambda_{ij}$.
25: end if
26: Output $\{x_{ij}, \{y_{ijl}(t), R_l(t)\}_{\{\forall l\in[L],\forall t\in[T]\}}, u\}$.
```

For task j, given $Z_l(t)$ and C, the key step is to find the best schedule that maximizes task j's utility. The scheduling algorithm works as follows: upon the arrival of task j, we firstly fix the schedule between $[a_{ij}, T]$. Since the original value of each task before the deadline is constant, we manage to calculate the resource consumption and energy expense in each plausible cloudlet and time slot. Then, in every situation, we fix the last time slot in order to find the minimal cost of the former $(w_{ij} - 1)$ slots. After adding the sum to the cost of the last time slot, we calculate the RHS of constraint (8a) and select the most economical one. Finally, we figure out the utility of task j according to (9) in each slot. We reject j if $\phi_{ij} = 0$; otherwise, j is accepted and we output the optimal schedule of task j.

Online scheduling algorithm. The online schedule algorithm PD for workload allocation in EDR is demonstrated in Algorithm 2. We input the needed variables for every cluster chosen by Algorithm 1. Initially, in line 1, all binary variables should be 0. Lines 2–8 call CORE in algorithm 3 and schedule each arriving task in the cluster. CORE computes x_{ij} and $y_{ijl}(t)$ for each task while also updating dual variables $Z_l(t)$, with estimated M, N values by former data. In CORE, upon the arrival of task j, we make initialization in line 1 and add all possible tuple (t, l) in line 2. Lines 3–8 compute the cost per time slot in the feasible set. Local generation cost is added if and only if processing task j exceeds the EDR cap. Then, we choose the cloudlets with minimum costs in each time slot to be the candidates for the schedule in line 9. In line 10, we mark the w_{ij}th slot. Lines 11–17 work as follows: by fixing the last slot for processing the task, the minimum cost of previous $w_{ij} - 1$ slots should be picked up. When the completion time t' violates the deadline, value for the task is diminished. Line 18 computes each possible utility and line 19 finds the max one. Lines 20–25 update binary and dual variables. If the previously selected utility is larger than 0, we accept the task and update cloudlet costs, amount of allocated resource, x_{ij} and $\{y_{ijl}(t)\}$; otherwise, no change is made.

5.2 Theoretical analysis of online task scheduling

5.2.1 Correctness and polynomial time

Theorem 7 *The Algorithm PD computes a feasible solution for problems in (4) (7) and (8).*

Theorem 8 *PD runs in polynomial time.*

5.2.2 Competitive ratio

The *competitive ratio* is defined as the upper bound ratio of the optimal objective value of (4) to the objective value achieved by PD. In reality, the ratio is always larger than 1.

We use OPT_1 and OPT_2 to denote the optimal objective values of (4) and (7). The equivalence between the two program indicates $OPT_1 = OPT_2$. Let $P^0 = 0$ and $D^0 = \sum_l \sum_t \frac{N}{e\sigma} R_l$ be the beginning primal and dual values. We assign different values to P^j and D^j in the accumulation process, until the algorithm finishes the tasks allocation by achieving P^J and D^J.

Lemma 4 *If one constant value α exists, such that (i) $P^j - P^{j-1} \geq \frac{1}{\alpha}(D^j - D^{j-1}), \forall j \in [J]$, ii) $P^0 = 0, D^0 \leq \frac{OPT_2}{e}$, the algorithm obtains $\frac{e}{e-1}\alpha$-competitive.*

Lemma 5 *D_0 in PD is at most $\frac{OPT_2}{e}$ under the condition that the optimal objective value of (4) is at least $\sum_l \frac{TN}{\sigma} R_l$.*

Definition 5 *The allocation-utility relationship for PD with a parameter α is $\lambda_{ij} Z_l^j(t) \geq \frac{1}{\alpha} R_l(Z_l^j(t) - Z_l^{j-1}(t)), \forall j \in [J], \forall l \in [L], \forall t \in [T]$.*

Lemma 6 *The allocation-utility relationship with α guarantees:*

$$\lambda_{ij} \sum_{l \in L(h)} \sum_{t \in T(h)} Z_l^j(t) + \phi_{ij} \geq \frac{1}{\alpha} \sum_{l \in L(h)} \sum_{t \in T(h)} R_l(Z_l^j(t) - \frac{1}{\alpha} Z_l^{j-1}(t))$$

$$+ \frac{1}{\alpha} \phi_{ij}, \forall j \in [J], \forall h \in \zeta_{ij}.$$

Lemma 7 *If α ensures the allocation-utility relationship for PD, then $P^j - P^{j-1} \geq \frac{1}{\alpha}(D^j - D^{j-1}), \forall j \in [J]$.*

We posit that the resource consumption of the workload is much less than the existing resource capacity, which is $\lambda_{ij} \ll R_l, \forall j, \forall l$. As a result, the differential $\mathrm{d}Z_l(t)$ equals $Z_l^j(t) - Z_l^{j-1}(t)$. The adaptation of allocation-utility relationship is interpreted in the version:

Definition 6 *The allocation-utility relationship in a differential version with* α *is* $\lambda_{ij} Z_l^j(t) \geq \frac{1}{\alpha} R_l dZ_l(t), \forall j \in [J], \forall l \in [L], \forall t \in [T]$.

Lemma 8 *We let* $\alpha_{ij} = \frac{1}{\lambda_{ij}}\left(\ln\left(\frac{\sigma M}{N}\right) + 1\right)$, *and it satisfies* $\lambda_{ij} Z_l^j(t) \geq \frac{1}{\alpha_{ij}}\left(R_l(Z_l^j(t) - Z_l^{j-1}(t))\right)$ *for task j.*

Proof According to our function,

$$Z_l(t) = Z_l(R_l(t)) = \frac{N}{e\sigma}\left(\frac{e\sigma M}{N}\right)^{\frac{R_l(t)}{R_l}},$$

$$\mathrm{d}Z_l(t) = Z_l((R_l(t)))' = \frac{N}{e\sigma}\left(\frac{e\sigma M}{N}\right)^{\frac{R_l(t)}{R_l}} \frac{1}{R_l} \ln\left(\frac{e\sigma M}{N}\right).$$

For any j, we can find an α_{ij} holds the inequality such that $\alpha_{ij} = \frac{R_l \mathrm{d}Z_l(t)}{\lambda_{ij} Z_l^j(t)} = \frac{1}{\lambda_{ij}}\left(\ln\left(\frac{\sigma M}{N}\right) + 1\right)$ □

Theorem 9 *With* $\alpha = \max_{j \in [J]}\{\frac{1}{\lambda_{ij}}\left(\ln\left(\frac{\sigma M}{N}\right) + 1\right)\}$, *the task scheduling algorithm PD is* $\frac{e}{e-1}\alpha$*-competitive.*

Proof Combining Lemma 4–8 and the definition of allocation-utility relationship, we can figure out the competitive ratio for all tasks in $[J]$, and hence, the proof is completed. □

6 Performance evaluation

6.1 Experiment setup

To simulate the whole edge system, we collect real-world data about EDR. According to a real EDR event happening in New York on August 28, 2018, which lasted for 6 h [33]. Besides, the real field tests show that the EDR energy reduction rate under 25% would be tolerable for a data center to sustain the normal operation [34]. As for cloudlet, it is a small-scale cloud data center with 1–40 servers [35]. In this case, the typical PUE value is around 2.1 [36]. As for the strike price for the EDR event, mostly it is around \$1100 to \$1800 per MWh [37]. The *EDR dispatch rate* (i.e., the percentage of the number of clusters to engage in EDR event) is around 58% [38]. The idle power for each server is 60 w, and the peak power is 180 w [39]. The diesel price for local generation is set to 0.32\$/kWh [40, 41].

We use the data stated above to construct our simulations. We assume that on average, a cloudlet should have a PUE of 2.2 and contain 20 servers. And the overall *energy-cutting rate* (i.e., the demand for energy reduction in the EDR event) is 25% with a 6-h duration. We randomly scale the number of servers for each cloudlet from 16 to 20 and the PUE value from 1.9 to 2.5. In the experiment setting, one cluster contains 15 distributed cloudlets and is capable of executing 40 tasks. We set the length of one time slot into 10 min and divide the whole EDR process into 36 time slots. We randomly generate each cluster's bidding price based on the cutting energy amount and the unit price. The original power for one cluster is estimated to be 600 kWh according to the number and power of servers and cloudlets. In the aspect of tasks, we randomly generate the workload from 0.4x to 1.0x toward normalized 20 servers. We use the Poisson distribution to randomly set each task's arriving time and randomly generate the deadline before the EDR ends. The value of a task is proportional to its workload and the number of time slots required. We set M to be the upper bound of unit price, and N to be the lower bound because we randomly generate the unit price of each task from 0.01\$ to 0.04\$.

To deal with the objective value of integer programming (4), we use the MINLP solver SCIP [42] to obtain the offline optimal solution. We find the average acceptance rate is higher than 95% without using any local power generation, indicating that our experiment setting is suitable to simulate the real-world scenario.

6.2 Results and discussion

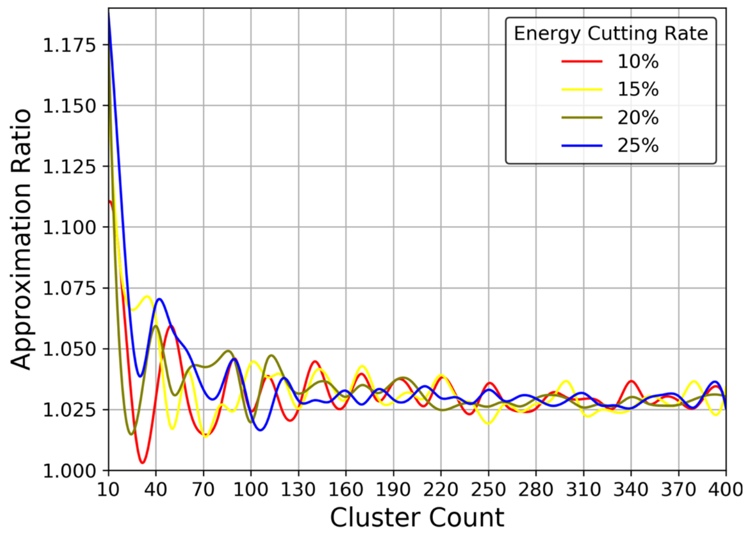
We consider several indexes to evaluate our algorithm: total social cost and approximation ratio on the first stage, and cluster utility, competitive ratio, and task acceptance rate on the second stage. Specifically, we compare four different approaches: (i) Greedy algorithm, which is based on [10], and always executes the maximum value task first in order to get optimal value; (ii) first-come, first-served (FCFS) algorithm, which always lets the early-arriving task schedules first and cannot be preempted by a later task [11]; (iii) Optimum programming, which returns the optimal offline solution; and (iv) mixed-integer programming (MIP), with PD on the second stage, which gets the optimal solution until current slot.

Performance of AMEDR. Figure 3a shows the trend of approximation ratio under different energy-cutting rates. Here, we fix the EDR dispatch rate to 60%, we can see that the ratio of AMEDR tends to converge to 1.025. This result in practice is much better than the theoretical bound. Figure 3b reflects the same result when we adjust the EDR dispatch rate. These simulations demonstrate that AMEDR has a close-to-optimal performance, especially with a large number of clusters.

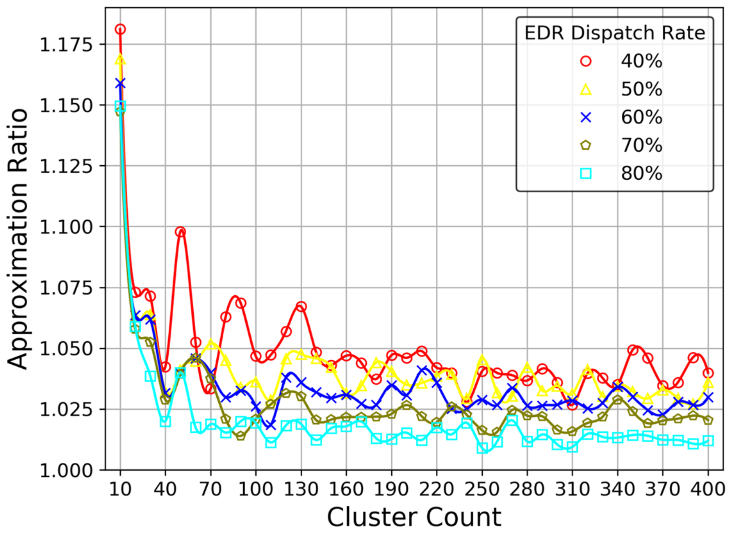
Performance of PD. Firstly, we have to consider the influence of setting different M and N, the maximum and the minimum value per unit workload per unit slot, on the performance of PD. In Fig. 4a, suppose we already know all the task information and try to vary this ratio to 0.5x, 1.0x, and 1.5x. The simulation result indicates that this ratio only slightly influences the optimal value. Besides, in the simulation, this figure also reflects that competitive ratio is likely to increase as the number of tasks rises.

Figure 4b is the comparison between three online algorithms. We implement two benchmark algorithms, the Greedy algorithm and the FCFS algorithm. Figure 4c modifies the EDR duration time. We can find that the performance of PD is better than the greedy algorithm and the FCFS algorithm. Both algorithms do not take the task's elastic deadline into consideration, so they will reject some of the tasks. PD performs better because it looks for flexible scheduling.

We change the energy-cutting rate to evaluate our algorithm in a relatively extreme situation. In Fig. 4c, we can see that the cluster's utility decreases because we do not have enough power to execute all the tasks, or we have to pay the local generation cost for power replenishment. However, PD still beats the other two algorithms as well, because PD includes local generation cost if we have to replenish electricity and compare this cost



(a) Comparison of approximation ratio between different energy cutting rate as cluster amount increases.

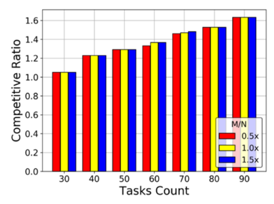


(b) Comparison of approximation ratio between different EDR dispatch rate as cluster amount increases.

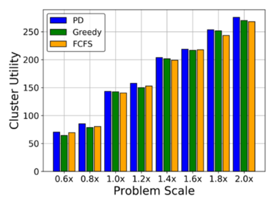
Fig. 3 Performance evaluation on AMEDR

with the task's utility. This utility measurement can avoid consuming power to execute low-value tasks.

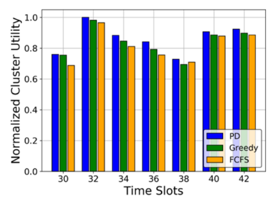
In Fig. 4e, we assign different weights to the penalty function and analyze sensitivity. We find the weight of the penalty function can influence the acceptance rate. This means PD rejects the task if the delay penalty is too heavy. As for the greedy algorithm or the FCFS algorithm, they do not consider this condition but finish the task before the



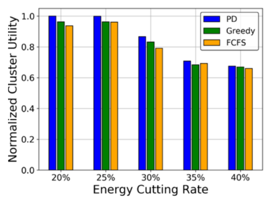
(a) Comparison of competitive ratio between different M/N scales with varying number of tasks J



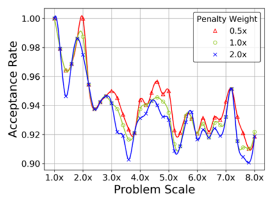
(b) Comparison of cluster utility between different online algorithms as problem scale increases.



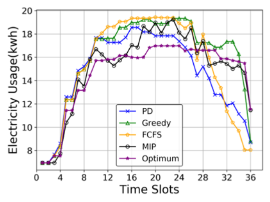
(c) Comparison of cluster utility between different online algorithms with varying time slots T.



(d) Comparison of cluster utility between different online algorithms as energy-cutting rate increases.



(e) Comparison of acceptance rate between different penalty weights as problem scale increases.



(f) Comparison of electricity usage between different algorithms in default problem scale.

Fig. 4 Performance evaluation on PD

deadline. However, this attribute is useful in an extreme condition with substantial worthless tasks to execute. We assume there are enough cloudlets to finish all the tasks. Hence, the differences are slight in the figure.

Figure 4f shows the electricity usage of different algorithms at each slot. We find that all algorithms perform well at low computation time, but PD can schedule tasks more efficiently at peak computation time. The local electricity usage of PD is 49.8%, 22.4% lower than the greedy algorithm, FCFS algorithm respectively, nearly close to the optimum. In this case, PD is more eco-friendly, compared with other algorithms.

7 Conclusion

In this work, we study how to enable edge emergency demand response via cloudlet clusters control. To address challenges in incentive mechanism design and task scheduling at participant cloudlet cluster, we propose a two stage control mechanism to facilitate edge EDR. In the first stage, a reverse auction, AMEDR, is proposed to select cost-efficient clusters and provide monetary remuneration to winners based on their energy reduction. We prove that AMEDR is computationally efficient, truthful, individual rational, and achieves 2-approximation in social cost. In the second stage, we design an online primal-dual algorithm, PD, for chosen cluster to schedule and allocate its workload while satisfy EDR energy reduction requirement. PD runs in polynomial time and achieves a provable competitive ratio. We conduct large-scale simulations to verify the efficiency and advantages of our method over existing methods.

Appendix

Appendix A: Proof of Lemma 1

Proof We first demonstrate that a feasible solution to (5) also satisfies the constraint of (2), via proof of contradiction. Let S^* denote the set of variables that equal 1 in a feasible solution to ILP (5), i.e., $z_i = 1$ *iff.* $i \in S^*$. Then, we transform (2a) into

$\sum_{i\in[S^*]} E_i \geq E_{all}$ and (5a) into $\sum_{i\in S^*\cap([I]\setminus S)} E_i(S) \geq \Delta(S), \forall S \subset [I]: \Delta(S) > 0$. Assume that this solution is not feasible to (2), $\sum_{i\in[S^*]} E_i < E_{all}$, which is equivalent to $\Delta(S^*) > 0$. Thus, we have $\sum_{i\in S^*\cap([I]\setminus S^*)} E_i(S) \geq \Delta(S^*)$. Since the left-hand side (LHS) of this inequality equals zero and the right-hand side (RHS) is greater than 0, we reach a contradiction. As a result, an arbitrary solution to (5) is also feasible to (2). Here, we omit the proof of the reverse that all the feasible solutions to ILP (2) are also feasible to (5), which can also be deducted by proof of contradiction. Since ILP (2) and (5) share the same set of solution as well as objective value, they are equivalent. □

Appendix B: Proof of Theorem 1

Proof (primal feasibility) Algorithm 1 stops at $\Delta(\mathcal{R}) \leq 0$, which suggests $E_{all} - \sum_{i\in\mathcal{R}} E_i \leq 0$. If we set all $z_i, i \in \mathcal{R}$ to 1, otherwise to 0, constraint (2a) is satisfied. Therefore, the feasibility in ILP (2) is proved. Combining Lemma 1, Algorithm 1 guarantees feasibility in both primal problems.

(Dual feasibility) Consider the situation that cluster i is chosen in a loop. Since $\mathcal{R}$ contains i, the update in the following loop will not change the variable in the previous loop and the constraints are not violated. As the number of tight constraints increases with $\mathcal{R}$ accumulating, no constraints in (6a) are violated. □

Appendix C: Proof of Therorem 2

Proof The `While` loop in line 2 iterates at most I times because only one winner $i \in [I]$ is added into the winner set $\widehat{S}$ at a time as long as the previous winners are not able to meet the whole EDR requirement. In each loop, calculations are performed at most $O(I)$ times in the minimum value selection and the assignments of $z_{i^*}, \mathcal{R}, [I]$ and c_{i^*}. After adding all the running time together, we obtain $O(I^2)$ in overall computation complexity. □

Appendix D: Proof of Lemma 2

Proof The chosen bid i^* has the minimum value for $\frac{c_i}{E_i(S_{i^*})}, i \in [I] \setminus S_{i^*}$. As $c_{\tilde{i}} < c_{i^*}$, two inequalities $\frac{c_{\tilde{i}}}{E_{\tilde{i}}(S_{i^*})} < \frac{c_{i^*}}{E_{i^*}(S_{i^*})}$ and $\frac{c_{\tilde{i}}}{E_{\tilde{i}}(S_{i^*})} < m(\widetilde{S_i})$ hold true. Therefore, $i^* = \arg\min_{i\in[I]}\{\frac{c_i}{E_i(\mathcal{R})}\}$. It can be seen that i^* is also selected in the auction. □

Appendix E: Proof of Lemma 3

Proof From lines 7–8 in AMEDR, we can deduce that $m(\widetilde{S_i})$ is the smallest value of $\frac{c_i}{E_i(S_{i^*})}, \forall i \in [I] \setminus (S_{i^*} \cup i^*)$. However, when we assume $\exists i^*, P_{i^*} < c_{i^*}$, we have $m(\widetilde{S_i}) < m(S_{i^*})$ according to line 9. In this case, cluster $\tilde{i}$ should be selected as the winning bid, and cluster i^* fails to earn the payment. □

Appendix F: Proof of Theorem 7

Proof To begin with, in Algorithm 3, line 2 provides feasible pairs of (t, l) in (4a). Constraint (4b) is guaranteed by line 3–8 in CORE, since we only utilize local generation cost when the consumption exceeds EDR cap. For each task j to be executed, we only add the time slots to the feasible set H if they are no earlier than a_{ij}, and based on line 11, we make it a rule that the $w_{ij} - 1$ task slots are scheduled between a_{ij} and $a_{ij} + \tau_{ij}$, which is described in (4c). (4d) mentions that in each time slot, the working cloudlet for task j is at most 1, a self-evident theory shown in line 9. Line 1 and 23 in CORE represent that x_{ij} and

$y_{ijl}(t)$ are binary, and the sum of $y_{ijl}(t)$ is 0 or w_{ij}, satisfying (4e) (4f) (4g). Furthermore, IP (7) provides a packing structure that satisfies (4c) (4e) in IP (4), as is shown in line 23. (7a), (7b), and (7e) are feasible for (4a), (4b), and (4g). (7c) and (7d) demonstrate at most one feasible set is accepted, and they are guaranteed by h_{t^*} in line 19. Thus, the algorithm can be proven feasible in IP (7). Last, by demonstrating ϕ_{ij} in the initialization and line 20, CORE ensures the non-negative quality of each variable for the dual problem (8). □

Appendix G: Proof of Theorem 8

Proof We first analyze Algorithm CORE. The search for feasible sets in task j, as shown in line 2, runs in $O(LT)$. The time complexity in line 3–8 is $O(LT)$. Line 9 and 10 work in $O(T)$ and $O(T^2)$, respectively. The loop from line 11 to 17 takes at most $O(w_{ij}T^2)$ time to run. Line 18–25 works to provide a feasible schedule, runs $O(LT)$ time. To sum up, the total running time is $O(LT + w_{ij}T^2)$.

Then PD simply runs all tasks via CORE, $O(LT + w_{ij}T^2)$. The result, $O(J(LT + \max_{j\in[J],i\in[I]}\{w_{ij}\}T^2))$, is also polynomial. □

Appendix H: Proof of Lemma 4

Proof By summing up the inequalities in each turn, we obtain $P^J = \sum_j(P^j - P^{j-1}) \geq \frac{1}{\alpha}\sum_j(D^j - D^{j-1}) = \frac{1}{\alpha}(D^J - D^0)$. On account of weak duality [32], $D^J \geq OPT_2$. Additionally, we assume $D^0 \leq \frac{OPT_2}{e}$. As a result of $P^J \geq \left(1-\frac{1}{e}\right)\frac{1}{\alpha}OPT_2 = \left(1-\frac{1}{e}\right)\frac{1}{\alpha}OPT_1$, the algorithm achieves an competitive ratio of $\frac{e}{e-1}\alpha$. □

Appendix I: Proof of Lemma 5

Proof We first make assumption that in a natural circumstance, when the resource and electricity are sufficient, tasks will be completed before deadline. Under such condition, if all cloudlets in the cluster process tasks in $\min w_{ij}, \forall j \in [J]$ time slots, i.e., $\frac{T}{\sigma}$, and each R_l is fully occupied by workloads with minimum value per workload per time slot, then the cluster would generate the minimum utility for the offline setting. Since $N = \min_{j\in[J]} \frac{b_{ij}}{w_{ij}\lambda_{ij}}$, the above circumstance will bring about $\sum_l \frac{TN}{\sigma}R_l$ utility for the cluster, definitely smaller than the objective value of (4).

Back to PD algorithm, the dual problem initializes its objective value D^0 at $\sum_l\sum_t \frac{N}{e\sigma}R_l = \sum_l \frac{TN}{e\sigma}R_l$. Obviously, $\frac{OPT_2}{e} \geq \sum_l \frac{TN}{e\sigma}R_l = D^0$. □

Appendix J: Proof of Lemma 6

Proof If $\chi_{ijh} = 0$, the inequality is obviously correct as the RHS should be $\frac{1}{\alpha}\phi_{ij}$, smaller than the value on the LHS. If $\chi_{ijh} = 1$, we sum all cloudlets and time slots of the allocation-utility relationship, and we have $\lambda_{ij}\sum_{l\in L(h)}\sum_{t\in T(h)} Z_l^j(t) \geq \frac{1}{\alpha}\left(\sum_{l\in L(h)}\sum_{t\in T(h)} R_l(Z_l^j(t) - Z_l^{j-1}(t))\right), \forall j \in [J]$. Next, we add ϕ_{ij} to each side of the inequality, and

$$\begin{aligned}
&\lambda_{ij}\sum_{l\in L(h)}\sum_{t\in T(h)} Z_l^j(t) + \phi_{ij}\\
\geq&\frac{1}{\alpha}\Big(\sum_{l\in L(h)}\sum_{t\in T(h)} R_l(Z_l^j(t) - Z_l^{j-1}(t))\Big) + \phi_{ij}\\
\geq&\frac{1}{\alpha}\Big(\sum_{l\in L(h)}\sum_{t\in T(h)} R_l(Z_l^j(t) - Z_l^{j-1}(t)) + \phi_{ij}\Big), \forall j \in [J]\,.
\end{aligned}$$

□

Appendix K: Proof of Lemma 7

Proof When the jth task is rejected, $P^j - P^{j-1} = D^j - D^{j-1} = 0$ certainly holds the inequality. Otherwise, $P^j - P^{j-1} = b'_{ijh} - (g(u^j) - g(u^{j-1}))$ represents the increased value in the scheduling, while $D^j - D^{j-1} = \sum_l \sum_t (Z_l^j(t) - Z_l^{j-1}(t))R_l + \phi_{ij} + (g^*(C^j) - g^*(C^{j-1}))$. Note that b'_{ijh} is the value for task j. Due to *complementary slackness*, the inequality in (8a) goes tight, we can replace b'_{ijh} with $\phi_{ij} + \lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} Z_l(t) + C^j\lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} \beta_l$. Then, $P^j - P^{j-1} = \phi_{ij} + \lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} Z_l(t) + C^j\lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} \beta_l - (g(u^j) - g(u^{j-1}))$

In order to simplify the proof, we discuss the problem in 3 subcases.

i) If $u^j \le D'$: in this case, since $C^j = 0$, we have

$$P^j - P^{j-1} = \lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} Z_l(t) + \phi_{ij},$$

$$D^j - D^{j-1} = \sum_{l\in[L]} \sum_{t\in[T]} R_l(Z_l^j(t) - Z_l^{j-1}(t)) + \phi_{ij}.$$

$\sum_l \sum_t (Z_l^j(t) - Z_l^{j-1}(t))$ can be further simplified to $\sum_{l\in L(h)} \sum_{t\in T(h)} (Z_l^j(t) - Z_l^{j-1}(t))$ as $Z_l^j(t) - Z_l^{j-1}(t) = 0$ if $l \notin L(h)$ or $t \notin T(h)$. According to Lemma **??**, PD ensures $P^j - P^{j-1} \ge \frac{1}{\alpha_1}(D^j - D^{j-1})$.

ii) If $u^{j-1} \ge D'$: $C^j\lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} \beta_l$ is equivalent with $(g(u^j) - g(u^{j-1}))$ because both represent an increment in the usage of local generator for processing task j. As a result, $P^j - P^{j-1}$ and $D^j - D^{j-1}$ are the same pattern with (i), and hence, the rest of the proof is similar to subcase (i).

(iii) if $u^{j-1} \le D' < u^j$: according to CORE, $C^{j-1} = 0, C^j = p, g(u^{j-1}) = 0, g(u^j) = p(u^{j-1} + \lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} \beta_l)$. Then,

$$P^j - P^{j-1} = \lambda_{ij} \sum_{l\in L(h)} \sum_{t\in T(h)} Z_l(t) + PD' + \phi_{ij},$$

$$D^j - D^{j-1} = \sum_{l\in[L]} \sum_{t\in[T]} R_l(Z_l^j(t) - Z_l^{j-1}(t)) + PD' + \phi_{ij}.$$

Since $PD' \ge 0$, it is easy to verify the lemma at this subcase. □

Abbreviations

EDR: Emergency demand response; IoT: Internet of Things; ILP: Integer linear programming; FPTAS: Fully polynomial time approximation scheme; VM: Virtual machine; AMEDR: An auction mechanism for emergency demand response; PD: A primal-dual based design for online allocation

Acknowledgements

The authors are grateful to the editor and anonymous referees for their valuable comments and suggestions. Only the authors are responsible for the views expressed and mistakes made.

Authors' contributions

Zhaoyan Song proposed the algorithms, provided the proofs of theorems and lemmas, and drafted the manuscript. Ruiting Zhou developed the ideas for the study and edited the manuscript. Shihan Zhao was responsible for the data simulations and the writing of the simulation parts. Shixin Qin also participated in the data analysis. John C.S. Lui and Zongpeng Li reviewed the manuscript. The authors read and approved the final manuscript.

Funding

This work was supported in part by the Technological Innovation Major Projects of Hubei Province (2017AAA125), the CUHK Grant (project # 3133067), and GRF grant 14201819. Part of this work by Ruiting Zhou was done while she was visiting CUHK.

Availability of data and materials

The data has been gathered from research papers and articles that are mentioned in the references.

Competing interests

The authors declare that they have no competing interests.

Received: 9 May 2020 Accepted: 14 August 2020
Published online: 10 September 2020

References

1. M. Satyanarayanan, The emergence of edge computing. IEEE Comput. **50**(1), 30–39 (2017)
2. Y. Zhang, K. Wang, Y. Zhou, Q. He, Enhanced adaptive cloudlet placement approach for mobile application on spark. Secur. Commun. Netw. **2018** (2018). https://doi.org/10.1155/2018/1937670
3. N. Ansari, X. Sun, Mobile edge computing empowers Internet of Things. IEICE Trans. Comm. **101**(3), 604–619 (2018)
4. A. Molina-Garcia, F. Bouffard, D. Kirschen, Decentralized demand-side contribution to primary frequency control. IEEE Trans. Power Syst. **26**(1), 411–419 (2010)
5. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
6. L. Zhang, S. Lei, C. Wu, Z. Li, in *Proc. of IEEE INFOCOM*, A truthful incentive mechanism for emergency demand response in colocation data centers, (2015), pp. 2632–2640. https://doi.org/10.1109/infocom.2015.7218654
7. J. Chen, D. Ye, S. Ji, Q. He, Y. Xiang, Z. Liu, in *Proc. of IEEE INFOCOM*, A truthful FTPAs mechanism for emergency demand response in colocation data centers, (2019), pp. 2557–2565. https://doi.org/10.1109/infocom.2019.8737468
8. X. Zhang, Z. Huang, C. Wu, Z. Li, F. Lau, in *Proc. ACM SIGMETRICS, vol. 43*, Online auctions in IAAs clouds: welfare and profit maximization with server costs, (2015), pp. 3–15. https://doi.org/10.1109/tnet.2016.2619743
9. R. Zhou, Z. Li, C. Wu, Z. Huang, An efficient cloud market mechanism for computing jobs with soft deadlines. IEEE-ACM Trans. Netw. **25**(2), 793–805 (2016)
10. N. Jain, I. Menache, J. S. Naor, J. Yaniv, Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters. ACM Trans. Parallel Comput. **2**(1), 3–1329 (2015)
11. Z. Dong, N. Liu, R. Rojas-Cessa, Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. J. Cloud Comput. **4**, 1–14 (2015)
12. D. Kim, J. Kim, Design of emergency demand response program using analytic hierarchy process. IEEE Trans. Smart Grid. **3**(2), 635–644 (2012)
13. H. Kwag, J. Kim, Reliability modeling of demand response considering uncertainty of customer behavior. Appl. Energy. **122**, 24–33 (2014)
14. R. Zhou, Z. Li, C. Wu, in *Proc. of IEEE INFOCOM*, An online procurement auction for power demand response in storage-assisted smart grids, (2015), pp. 2641–2649. https://doi.org/10.1109/infocom.2015.7218655
15. K. Ma, T. Yao, J. Yang, X. Guan, Residential power scheduling for demand response in smart grid. Int. J. Electr. Power Energy Syst. **78**, 320–325 (2016)
16. Q. Sun, S. Ren, C. C. Wu, Z. Li, in *Proc. of ACM e-Energy*, An online incentive mechanism for emergency demand response in geo-distributed colocation data centers, (2016), p. 3. https://doi.org/10.1145/2934328.2934331
17. S. Chen, L. Jiao, L. Wang, F. Liu, in *Proc. of IEEE INFOCOM*, An online market mechanism for edge emergency demand response via cloudlet control, (2019), pp. 2566–2574. https://doi.org/10.1109/infocom.2019.8737574
18. M. A. Islam, H. Mahmud, S. Ren, X. Wang, in *Proc. of IEEE HPCA*, Paying to save: reducing cost of colocation data center via rewards, (2015), pp. 235–245. https://doi.org/10.1109/hpca.2015.7056036
19. Q. Sun, C. Wu, Z. Li, S. Ren, Colocation demand response: joint online mechanisms for individual utility and social welfare maximization. IEEE J. Sel. Areas Commun. **34**(12), 3978–3992 (2016)
20. S. Ren, M. A. Islam, in *Proc. of USENIX ICAC*, Colocation demand response: why do i turn off my servers? (2014), pp. 201–208. https://www.usenix.org/conference/icac14/technical-sessions/presentation/ren
21. Z. Zhou, F. Liu, Z. Li, H. Jin, in *Proc. of IEEE INFOCOM*, When smart grid meets geo-distributed cloud: an auction approach to datacenter demand response, (2015), pp. 2650–2658. https://doi.org/10.1109/infocom.2015.7218656
22. R. Zhou, Z. Li, C. Wu, M. Chen, Demand response in smart grids: a randomized auction approach. IEEE J. Sel. Areas Commun. **33**(12), 2540–2553 (2015)
23. S. Anand, K. Garg, A. Kumar, in *Proc. of SODA*, Resource augmentation for weighted flow-time explained by dual fitting, (2012), pp. 1228–1241. https://doi.org/10.1137/1.9781611973099.97
24. N. R. Devanur, Z. Huang, Primal dual gives almost optimal energy-efficient online algorithms. TALG. **14**(1), 5 (2018)
25. S. Agrawal, N. R. Devanur, in *Proc. of SODA*, Fast algorithms for online stochastic convex programming, (2014), pp. 1405–1424. https://doi.org/10.1137/1.9781611973730.93
26. J. Wang, J. Pan, F. Esposito, in *Proc. of SmartIoT*, Elastic urban video surveillance system using edge computing, (2017). https://doi.org/10.1145/3132479.3132490
27. A. Qureshi, Power-demand routing in massive geo-distributed systems. PhD thesis, MIT (2010). https://dspace.mit.edu/handle/1721.1/62430
28. R. Carr, L. Fleischer, V. Leung, C. Phillips, in *Proc. of SODA*, Strengthening integrality gaps for capacitated network design and covering problems, (2000), pp. 106–115
29. A. Archer, E. Tardos, in *Proc. of IEEE Symposium on of Computer Science*, Truthful mechanisms for one-parameter agents, (2001), pp. 482–491. https://doi.org/10.1007/springerreference_57955
30. R. Myerson, Optimal auction design. Math. Oper. Res. **6**(1), 58–73 (1981)
31. R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. V. Vazirani, S. Yazdanbod, in *Proc. of ACM EC*, Convex program duality, fisher markets, and nash social welfare, (2017), pp. 459–460. https://doi.org/10.1145/3033274.3085109
32. S. Boyd, L. Vandenberghe, *Convex optimization*. (Cambridge, univ. press, Stanford University, 2004)
33. NYISO Summer 2018 Hot weather operations. [Online]. Available: http://www.nysrc.org. Accessed 18 July 2019
34. G. Ghatikar, V. Ganti, N. Matson, M. A. Piette, Demand response opportunities and enabling technologies for data centers: findings from field studies (2012). https://doi.org/10.2172/1174175
35. V. Bahl, Cloudlets for mobile computing (2014). https://www.microsoft.com/en-us/research/publication/cloudlets-for-mobile-computing-2/

36. M. Ganeshalingam, A. Shehabi, L. B. Desroches, Shining a light on small data centers in the U.S. (2017). https://escholarship.org/uc/item/8dh8j3kq
37. State of the Market Report for PJM. https://www.monitoringanalytics.com/reports/PJM_State_of_the_Market/2018/2018q3-som-pjm.pdf. Accessed 17 July 2019
38. 2018 Utility demand response market snapshot. https://www.peakload.org/2018-utility-dr-snapshot-report. Accessed 14 July 2019
39. D. Meisner, T. F. Wenisch, Peak power modeling for data center servers with switched-mode power supplies. ISLPED, 319–324 (2010). https://dl.acm.org/doi/abs/10.1145/1840845.1840911
40. Free online calculation of diesel generator power, energy and fuel consumption. https://power-calculation.com/generator-diesel-energy-calculator-genset.php. Accessed 20 July 2019
41. U.S. On-highway diesel fuel prices. https://www.eia.gov/petroleum/gasdiesel/. Accessed 15 July 2019
42. A. Gleixner, L. Eifler, T. Gally, et al., The SCIP optimization suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin (2018). http://nbn-resolving.de/urn:nbn:de:0297-zib-69361. Accessed 10 July 2019

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com