RESEARCH

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]

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; 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 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

*Correspondence: ruitingzhou@whu.edu.cn
[1]Wuhan University, Bayi Road, Wuchang District, 430070 Wuhan, China
Full list of author information is available at the end of the article

procurement [6, 7], or does not consider the electricity consumption in scheduling algorithm design [8, 9]. More discussions can be found in Sec. 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; 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 Sec. 2 and introduce the system model in Sec. 3. Algorithm designs for the first and second stage of edge EDR are presented in Sec. 4 and Sec. 5, respectively. Sec. 6 evaluates the performance of proposed algorithms and Sec. 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. [8] proposes 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, [23] and [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. But 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.

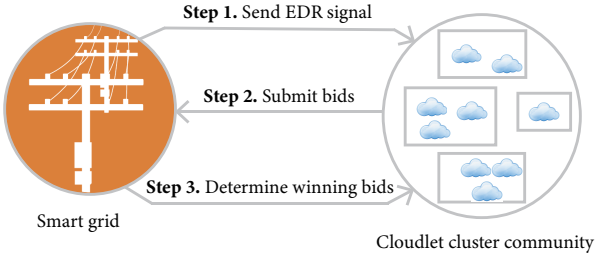


Figure 1: A demonstration of EDR via cloudlet clusters control.

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

Definition (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. The social cost minimization problem under truthful bidding ($c_i = v_i$) can be formulated into the following integer linear program (ILP):

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

$$\text{subject to:} \quad \sum_{i\in[I]} E_i z_i \geq E_{all}. \tag{2a}$$

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

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. 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_j and $y_{jl}(t)$, to indicate whether task j 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.

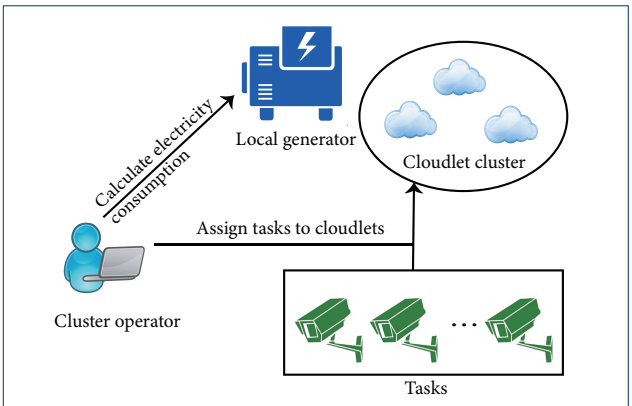


Figure 2: Task scheduling in the cluster.

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 maintenance cost. The objective is to maximize $P_i + \sum_{j\in[J]} b_j x_j - \sum_{j\in[J]} f_j(\tau_j) - pu_g - \theta_i$, 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:

$$\text{maximize} \quad \sum_{j\in[J]} b_j x_j - \sum_{j\in[J]} f_j(\tau_j) - pu_g \tag{4}$$

$$\text{subject to: } \sum_{j\in[J]} \lambda_j y_{jl}(t) \le R_l, \forall t \in [T], \forall l \in [L], \tag{4a}$$

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

$$t \sum_{l\in[L]} y_{jl}(t) \le d_j + \tau_j, \forall t \in [T], \forall j \in [J] : a_j \le t, \tag{4c}$$

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

$$w_j x_j = \sum_{l\in[L]} \sum_{t\in[T]} y_{jl}(t), \forall j \in [J], \tag{4e}$$

$$y_{jl}(t), x_j \in \{0,1\}, \quad \forall j \in [J], \forall l \in [L], \forall t \in [T], \tag{4f}$$

$$u_g \ge 0, \tau_j \ge 0, \forall j \in [J]. \tag{4g}$$

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: $e_l^t = (N_l P_{idle}^l + (P_{peak}^l - P_{idle}^l) \sum_{j\in J} \lambda_j y_{jl}(t)) \cdot \text{PUE}_l$, 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. $\sum_{j\in J} \lambda_j y_{jl}(t)$ 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_j and $y_{jl}(t)$, to guarantee sufficient execution.

Challenges. We notice that the first ILP 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, if we let $u_g = 0$ and $\tau_j = 0, \forall j \in J$, 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 Sec. 4, we propose a 2-approximation algorithm based on the primal-dual method to select winning clusters and compute payments for EDR event. In Sec. 5 we propose an online algorithm to schedule tasks while satisfying EDR requirement in the winners.

Table 1. Notation

I	# of cloud clusters	L_i	# of cloudlets in cluster i
T	# of time slots	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
$b_j(b_{ij})$	value of task j (in cluster i)
$a_j(a_{ij})$	the arrival time of task j (in cluster i)
$d_j(d_{ij})$	the deadline of task j (in cluster i)
$w_i(w_{ij})$	# of time slots required for task j (in cluster i)
$\lambda_j(\lambda_{ij})$	the workload of task j (in cluster i) in one slot
$\tau_j(\tau_{ij})$	# 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
u_g	total local generation
x_j	task j is accepted (1) or not (0)
$y_{jl}(t)$	task j is allocated in l at t (1) or not (0)

4 Methods for EDR Auction Mechanism

In Sec. 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 Sec. 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:

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

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

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:

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

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

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

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.

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: $\quad i^* = \arg\min_{i\in[I]}\{\frac{c_i}{E_i(\mathcal{R})}\}$.
4: $\quad z_{i^*} = 1; \mathcal{R} = \mathcal{R} \cup \{i^*\}$.
5: $\quad m(\mathcal{R}) = \frac{c_{i^*}}{E_{i^*}(\mathcal{R})}$
6: $\quad [I] = [I]\backslash i^*$.
7: $\quad \tilde{i} = \arg\min_{i\in[I]}\{\frac{c_i}{E_i(\mathcal{R})}\}$.
8: $\quad m(\widetilde{\mathcal{R}}) = \frac{c_{\tilde{i}}}{E_{\tilde{i}}(\mathcal{R})}$.
9: $\quad P_{i^*} = c_{i^*} + (m(\widetilde{\mathcal{R}}) - m(\mathcal{R}))E_{i^*}(\mathcal{R})$.
10: $\quad c_i = c_i - E_i(\mathcal{R})m(\mathcal{R}), \forall i \in [I]$.
11: **end while**
12: $\widehat{S} = \mathcal{R}$.
13: $P(\widehat{S}) = \{P_{i^*}\}_{i^*\in\widehat{S}}$.

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.

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 (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 $\Delta(\mathcal{R}) \le 0$. The last cluster picked by the algorithm is denoted as i_n, so we have $\Delta(\widehat{S}\backslash i_n) > 0$.

Since every i in $\widehat{S}\backslash i_n$ has a tight constraint in (6a), we reformulate the social cost among such items into:$\sum_{i\in\widehat{S}}\sum_{\forall S\subset[I]:i\subset[I]\backslash S,\Delta(S)>0} E_i(S)m(S)$. However, due to the initialization, the non-negative variables $m(S)$ remains zero unless $S \subset \widehat{S}\backslash i_n$. The equality above is simplified as $\sum_{S\subset\widehat{S}} m(S)\sum_{i\in\widehat{S}\backslash S'} E_i(S')$. We further transform $\sum_{i\in\widehat{S}\backslash S} E_i(S)$ 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's 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, 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 $c_{\tilde{i}}$ subject to $c_{\tilde{i}} < c_{i^*}$ and $z_{i^*} = 1$, then $z_{\tilde{i}} = 1$.*

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 Sec. 5.1. The theoretical analysis is presented in Sec. 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:

$$\text{maximize} \quad \sum_{j\in J}\sum_{h\in\zeta_j} b'_{jh}\chi_{jh} - g(u) \tag{7}$$

subject to:

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

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

$$\sum_{h\in\zeta_j}\chi_{jh} \le 1, \forall j\in[J], \tag{7c}$$

$$\chi_{jh}\in\{0,1\}, \forall j\in[J], \forall h\in\zeta_j, \tag{7d}$$

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

In the above program, ζ_j is the set of all feasible schedules for task j. A feasible schedule is a vector $h = (x_j, \{y_{jl}(t)\}_{\forall l\in[L],\forall t\in[T]}, \tau_j)$ that satisfies constraints (4c) and (4e). Binary variable, χ_{jh}, indicates whether task j is accepted and scheduled according to schedule h ($\chi_{jh}=1$) or not ($\chi_{jh}=0$). b'_{jh} is the task value based on schedule h, *i.e.,* $b'_{jh} = b_j - f_j(\tau_j)$. $T(h)$ 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 $D' = D - E - T(\sum_{l\in[L]} N_l P^l_{idle})\cdot \text{PUE}_l$ and $u = D' + u_g$. $g(u)$ can be defined as a piecewise function as follows:

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

$g(u)$ indicates the energy consumption either below or above the EDR cap. We simplify the LHS of (4b), and let $\beta_l = (P^l_{peak} - P^l_{idle})\cdot \text{PUE}_l$. 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_j, as well as relaxing $\chi_{jh}\in\{0,1\}$ to $\chi_{jh}\ge 0$, the dual of the primal problem is:

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

subject to:

$$\phi_j \ge b'_{jh} - \sum_{l\in L(h)}\sum_{t\in T(h)} Z_l(t)\lambda_j - \sum_{l\in L(h)}\sum_{t\in T(h)} C\beta_l\lambda_j, \quad \forall j\in[J], \forall h\in\zeta_j, \tag{8a}$$

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

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

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

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

$$\phi_j = \max\{0, \max_{h\in\zeta_j}\{b_{jh}{}' - \sum_{l\in L(h)}\sum_{t\in T(h)} Z_l(t)\lambda_j - \sum_{l\in L(h)}\sum_{t\in T(h)} C\beta_l\lambda_j\}\}. \tag{9}$$

When $\phi_j > 0$, dual constraint (8a) holds tight so that the related primal variable $\chi_{jh} > 0$. In this case, task j is accepted, and h is the corresponding schedule for j. Otherwise, if $\forall h\in\zeta_j, \phi_j = 0$, $b'_{jh} - \sum_{l\in L(h)}\sum_{t\in T(h)} Z_l(t)\lambda_j - \sum_{l\in L(h)}\sum_{t\in T(h)} C\beta_l\lambda_j \le 0$, 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 $\phi_j > 0$, 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'_{jh} should contain the penalty expense; otherwise $b'_{jh} = b_j$. 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 $x_j = 0, y_{jl}(t) = 0, \tau_j = 0, \phi_j = 0, R_l(t) = 0, u = 0, \forall j \in [J], \forall l \in [L], \forall t \in [T]$, by default.
2: **On the arrival of task** j
3: Run CORE$(\{R_l(t), Z_l(t)\}_{\{l\in[L],t\in[a_j,T]\}}, p, \Gamma_j, u)$.
4: **if** $x_j = 1$ **then**
5: Schedule the jth task according to $y_{jl}(t)$.
6: **else**
7: Reject the jth task.
8: **end if**

We next discuss the update of 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 $\sigma = \frac{T}{\min_j\{w_j\}}$. 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, $N = \min_{j\in[J]} \frac{b_j}{w_j\lambda_j}, M = \max_{j\in[J]} \frac{b_j}{w_j\lambda_j}$.

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 $[a_j, T]$. 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 $(w_j - 1)$ 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 $\phi_j = 0$; otherwise, j is accepted and we output the optimal schedule of task j.

Algorithm 3 One-Round Task Scheduling: CORE

Input: $\{R_l(t), Z_l(t)\}_{\{l\in[L],t\in[a_j,T]\}}, p, \Gamma_j, u$.

1: Initialize $H = \emptyset, \widetilde{H} = \emptyset, h_t = \{t\}; x_j = 0, y_{jl}(t) = 0, \phi_j = 0, \forall l \in [L], \forall t \in [T]$, by default.
2: Add $(t, l), t \in [a_j, T], l \in [L]$ to H if $R_l(t) + \lambda_j \le R_l, \forall t \in [T], \forall l \in [L]$.
3: **for all** $(t, l) \in H$ **do**
4: Calculate $q(t, l) = Z_l(t)\lambda_j$.
5: **if** $u > D'$ **then**
6: $q(t, l) = q(t, l) + C\beta_l\lambda_j$.
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_jth slot as t_0.
11: **for all** $t' \in \widetilde{H} : t_0 \le t' \le T$ **do**
12: Select $w_j - 1$ 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** $t' > d_j$ **then**
15: Calculate $b_j = b_j - f_j(t' - d_j)$.
16: **end if**
17: **end for**
18: $\phi_j(t') = b_j - Q(t'), \forall t' \in \widetilde{H}$.
19: Find $t^* = \arg\max_{t'\in\widetilde{H}:t_0\le t'\le T} Q(t')$.
20: **if** $\phi_j > 0$ **then**
21: $R_{l_t}(t) = R_l(t) + \lambda_j, \forall t \in h_{t^*}$.
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: $x_j = 1; y_{jl_t}(t) = 1, \forall t \in h_{t^*}$.
24: $u = u + \sum_{l:(t,l)\in H, t\in h_{t^*}} \beta_l\lambda_j$.
25: **end if**
26: **Output** $\{x_j, \{y_{jl}(t), R_l(t)\}_{\{\forall l\in[L],\forall t\in[T]\}}, u\}$.

Online Scheduling Algorithm. The online schedule algorithm PD for workload allocation in EDR is demonstrated in Algorithm 2. 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_j and $y_{jl}(t)$ 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_jth slot. Lines 11-17 work as follows: by fixing the last slot for processing the task, the minimum cost of previous $w_j - 1$ 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_j and $\{y_{jl}(t)\}$; 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 The *Allocation-Utility Relationship* for PD with a parameter α is $\lambda_j Z_l^j(t) \geq \frac{1}{\alpha} R_l(Z_l^j(t) - Z_l^{j-1}(t)), \forall j \in [J], \forall l \in [L], \forall t \in [T]$.

Lemma 6 *The Allocation-Utility Relationship with α guarantees:*

$$\lambda_j \sum_{l \in L(h)} \sum_{t \in T(h)} Z_l^j(t) + \phi_j \geq \frac{1}{\alpha} \sum_{l \in L(h)} \sum_{t \in T(h)} R_l(Z_l^j(t) - \frac{1}{\alpha} Z_l^{j-1}(t)) + \frac{1}{\alpha}\phi_j, \forall j \in [J], \forall h \in \zeta_j.$$

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 $\lambda_j \ll R_l, \forall j, \forall l$. 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 The *Allocation-Utility Relationship* in a differential version with α is $\lambda_j Z_l^j(t) \geq \frac{1}{\alpha} R_l \mathrm{dZ_l(t)}, \forall \mathrm{j} \in [\mathrm{J}], \forall \mathrm{l} \in [\mathrm{L}], \forall \mathrm{t} \in [\mathrm{T}]$.

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

Proof: According to our function,

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

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

For any j, we can find an α_j holds the inequality such that $\alpha_j = \frac{R_l \mathrm{dZ_l(t)}}{\lambda_j Z_l^j(t)} = \frac{1}{\lambda_j}\left(\ln\left(\frac{\sigma M}{N}\right) + 1\right)$ □

Theorem 9 *With $\alpha = \max_{j \in [J]}\{\frac{1}{\lambda_j}\left(\ln\left(\frac{\sigma M}{N}\right) + 1\right)\}$, 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 hours [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 $1800 per MWh [37]. The *EDR dispatch rate* (*i.e.* the percentage of the number of clusters to engage in EDR event) is around 58% [38]. The idle power for each server is 60w, and the peak power is 180w [39]. The diesel price for local generation is set to 0.32$/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-hour duration. We randomly scale the number of servers for each cloudlet from 16 to 20 and the PUE value form 1.9 to 2.5. In the experiment setting, one cluster contains 15 distributed cloudlets and is capable of

executing 40 tasks.the length of one time slot into 10 minutes 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 600kWh 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 tasks acceptance rate on the second stage.

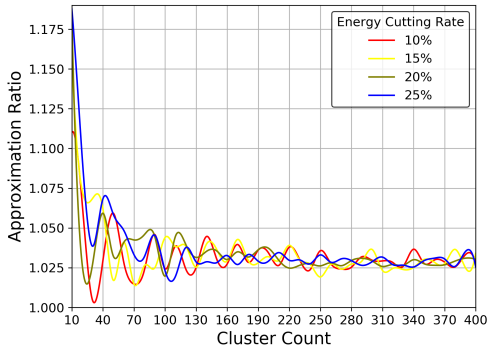
Performance of AMEDR. Fig. 3(a) shows the trend of approximation ratio under different energy-cutting rates. Here we fix the EDR dispatch rate to 60%, we can see that the ratio of AMEDR tends to converge to 1.025. This result in practice is much better than the theoretical bound. Fig. 3(b) reflects the 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. 4(a), 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.

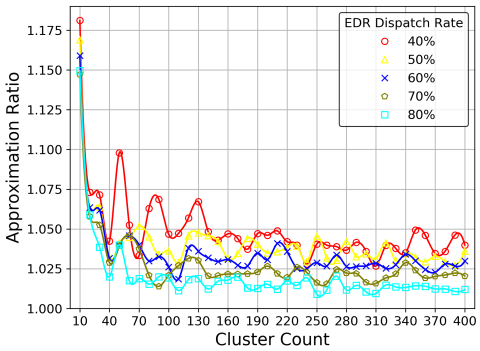
Fig. 4(b) is the comparison between three online algorithms. We implement two benchmark algorithms: i) a greedy algorithm based on [10], which always executes the maximum value task first in order to get optimal value; and ii) a first-come, first-served (FCFS) algorithm, which always lets the early-arriving task schedules first and cannot be preempted by a later task [11]. Fig. 4(c) 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. 4(c), 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 with the task's utility. This utility measurement can avoid consuming power to execute low-value tasks.

In Fig. 4(e), we assign different weights to the penalty function and analyze sensitivity. We find the weight of the penalty function can influence the accep-

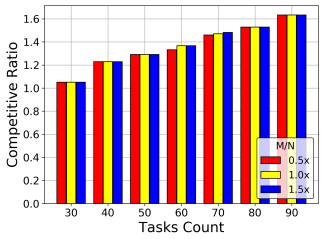


(a) Comparison of approximation ratio between different energy cutting rate as cluster amount increases.

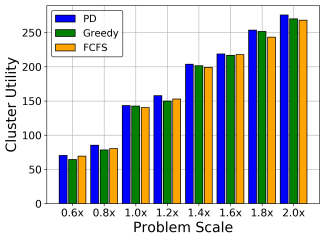


(b) Comparison of approximation ratio between different EDR dispatch rate as cluster amount increases.

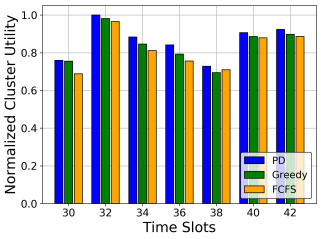
Figure 3: Performance evaluation on AMEDR



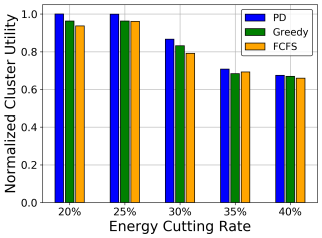
(a) Comparison of competitive ratio between different M/N scales with varying number of tasks J



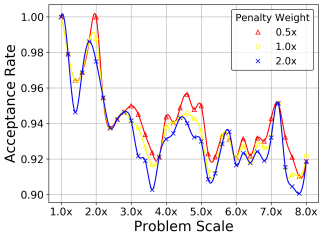
(b) Comparison of cluster utility between different online algorithms as problem scale increases.



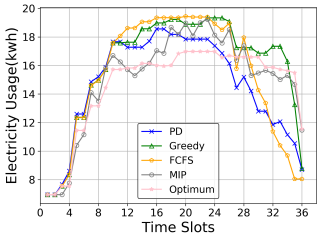
(c) Comparison of cluster utility between different online algorithms with varying time slots T.



(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.

Figure 4: Performance evaluation on PD

tance 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 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.

Fig. 4(f) 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.

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.

Author's 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.

Author details

[1]Wuhan University, Bayi Road, Wuchang District, 430070 Wuhan, China.
[2]Chinese University of Hong Kong, Shatin, NT, 999077 Hong Kong, China.

References

1. Satyanarayanan, M.: The emergence of edge computing. IEEE Computer **50**(1), 30–39 (2017)
2. Zhang, Y., Wang, K., Zhou, Y., He, Q.: Enhanced adaptive cloudlet placement approach for mobile application on spark. Security and Communication Networks **2018** (2018)
3. Ansari, N., Sun, X.: Mobile edge computing empowers internet of things. IEICE Trans. on Comm **101**(3), 604–619 (2018)
4. Molina-Garcia, A., Bouffard, F., Kirschen, D.: Decentralized demand-side contribution to primary frequency control. IEEE Trans. Power Syst. **26**(1), 411–419 (2010)
5. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
6. Zhang, L., Lei, S., Wu, C., Li, Z.: A truthful incentive mechanism for emergency demand response in colocation data centers. In: Proc. of IEEE INFOCOM, pp. 2632–2640 (2015)
7. Chen, J., Ye, D., Ji, S., He, Q., Xiang, Y., Liu, Z.: A truthful fptas mechanism for emergency demand response in colocation data centers. In: Proc. of IEEE INFOCOM, pp. 2557–2565 (2019)
8. Zhang, X., Huang, Z., Wu, C., Li, Z., Lau, F.: Online auctions in iaas clouds: Welfare and profit maximization with server costs. In: Proc. ACM SIGMETRICS, vol. 43, pp. 3–15 (2015)
9. Zhou, R., Li, Z., Wu, C., Huang, Z.: An efficient cloud market mechanism for computing jobs with soft deadlines. IEEE-ACM Trans. Netw. **25**(2), 793–805 (2016)
10. Jain, N., Menache, I., Naor, J.S., Yaniv, J.: Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters. ACM Trans. Parallel Comput. **2**(1), 3–1329 (2015)
11. Dong, Z., Liu, N., Rojas-Cessa, R.: Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. Journal of Cloud Computing **4**, 1–14 (2015)
12. Kim, D., Kim, J.: Design of emergency demand response program using analytic hierarchy process. IEEE Trans. Smart Grid **3**(2), 635–644 (2012)
13. Kwag, H., Kim, J.: Reliability modeling of demand response considering uncertainty of customer behavior. Appl. Energy **122**, 24–33 (2014)
14. Zhou, R., Li, Z., Wu, C.: An online procurement auction for power demand response in storage-assisted smart grids. In: Proc. of IEEE INFOCOM, pp. 2641–2649 (2015)
15. Ma, K., Yao, T., Yang, J., Guan, X.: Residential power scheduling for demand response in smart grid. Int. J. Electr. Power Energy Syst **78**, 320–325 (2016)
16. Sun, Q., Ren, S., C. Wu, C., Li, Z.: An online incentive mechanism for emergency demand response in geo-distributed colocation data centers. In: Proc. of ACM e-Energy, p. 3 (2016)
17. Chen, S., Jiao, L., Wang, L., Liu, F.: An online market mechanism for edge emergency demand response via cloudlet control. In: Proc. of IEEE INFOCOM, pp. 2566–2574 (2019)
18. Islam, M.A., Mahmud, H., Ren, S., Wang, X.: Paying to save: Reducing cost of colocation data center via rewards. In: Proc. of IEEE HPCA, pp. 235–245 (2015)
19. Sun, Q., Wu, C., Li, Z., Ren, S.: Colocation demand response: Joint online mechanisms for individual utility and social welfare maximization. IEEE J. Sel. Areas Commun. **34**(12), 3978–3992 (2016)
20. Ren, S., Islam, M.A.: Colocation demand response: Why do i turn off my servers? In: Proc. of USENIX ICAC, pp. 201–208 (2014)
21. Zhou, Z., Liu, F., Li, Z., Jin, H.: When smart grid meets geo-distributed cloud: An auction approach to datacenter demand response. In: Proc. of IEEE INFOCOM, pp. 2650–2658 (2015)
22. Zhou, R., Li, Z., Wu, C., Chen, M.: Demand response in smart grids: A randomized auction approach. IEEE J. Sel. Areas Commun. **33**(12), 2540–2553 (2015)
23. Anand, S., Garg, K., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proc. of SODA, pp. 1228–1241 (2012)
24. Devanur, N.R., Huang, Z.: Primal dual gives almost optimal energy-efficient online algorithms. TALG **14**(1), 5 (2018)
25. Agrawal, S., Devanur, N.R.: Fast algorithms for online stochastic convex programming. In: Proc. of SODA, pp. 1405–1424 (2014)
26. Wang, J., Pan, J., Esposito, F.: Elastic urban video surveillance system using edge computing. In: Proc. of SmartIoT (2017)
27. Qureshi, A.: Power-demand routing in massive geo-distributed systems. PhD thesis, MIT (2010)
28. Carr, R., Fleischer, L., Leung, V., Phillips, C.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proc. of SODA, pp. 106–115 (2000)
29. Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: Proc. of IEEE Symposium on of Computer Science, pp. 482–491 (2001)
30. Myerson, R.: Optimal auction design. Mathematics of operations research **6**(1), 58–73 (1981)
31. Cole, R., Devanur, N., Gkatzelis, V., Jain, K., Mai, T., Vazirani, V.V., Yazdanbod, S.: Convex program duality, fisher markets, and nash social welfare. In: Proc. of ACM EC, pp. 459–460 (2017)
32. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge, univ. press, ??? (2004)
33. NYISO Summer 2018 Hot Weather Operations. [Online]. Available: `http://www.nysrc.org`
34. G.Ghatikar, Ganti, V., Matson, N., Piette, M.A.: Demand response opportunities and enabling technologies for data centers: Findings from field studies (2012)
35. Bahl, V.: Cloudlets for mobile computing (2014)
36. Ganeshalingam, M., Shehabi, A., Desroches, L.B.: Shining a light on small data centers in the U.S. (2017)
37. State of the Market Report for PJM. `https://www.monitoringanalytics.com/reports/PJM_State_of_the_Market/2018/2018q3-som-pjm.pdf`
38. 2018 Utility Demand Response Market Snapshot. `https://www.peakload.org/2018-utility-dr-snapshot-report`
39. Meisner, D., Wenisch, T.F.: Peak power modeling for data center servers with switched-mode power supplies. ISLPED, 319–324 (2010)
40. Free online calculation of diesel generator power, energy and fuel consumption. `https://power-calculation.com/generator-diesel-energy-calculator-genset.php`
41. U.S. On-Highway Diesel Fuel Prices. `https://www.eia.gov/petroleum/gasdiesel/`
42. Gleixner, A., Eifler, L., Gally, T., et al.: The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin (July 2018). `http://nbn-resolving.de/urn:nbn:de:0297-zib-69361`