# Edge Emergency Demand Response Control via Scheduling in Cloudlet Cluster

Zhaoyan Song†, Ruiting Zhou† ‡, Shihan Zhao†, Shixin Qin†, John C.S. Lui‡, Zongpeng Li†
† Wuhan University, {*songzhaoyan, ruitingzhou, zhaosh, shixinqin, zongpeng*}*@whu.edu.cn*
‡ The Chinese University of Hong Kong, *cslui@cse.cuhk.edu.hk*

***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, one fundamental problem needs to be addressed: how to schedule and allocate workloads in a cloudlet cluster to satisfy EDR requirements. We propose an online task scheduling algorithm for the chosen cluster to dispatch workloads to guarantee target EDR power reduction. By exploiting the primal-dual optimization theory, we prove that our control scheme 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.**

## I. 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 placed at the edge of the network to provide low-latency and high bandwidth services for nearby mobile or IoT devices [1]. 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 an individual cloudlet uses a moderate amount of electricity, a cloudlet cluster utilizes 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 to stabilize the power grid by reducing and temporally shifting peak loads. To realize edge EDR for a chosen cloudlet cluster receiving task execution requests from mobile users online, one fundamental challenge 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 Sec. 2. In this work, we design a scheduling algorithm, tailored for a participant cloudlet cluster to dispatch workloads to guarantee target EDR power consumption. It has following goals: i) the algorithm is time efficient, making 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, *i.e.*, completed tasks' value minus the local generation 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. Our contributions are listed as follows:

**First**, we design an online 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, we introduce a set of new variables for each task to represent feasible schedules. Although the new formulation has an exponential size of variables, 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.

**Second**, we conduct large-scale simulations based on the real-world circumstances. We 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

1. Corresponding author: Ruiting Zhou.

2. 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.

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, the superiority of our method is demonstrated.

In the rest paper, we discuss related work in Sec. II and introduce the system model in Sec. III.Algorithm designs are presented in Sec. IV. Sec. V evaluates the performance of proposed algorithms and Sec. VI concludes the paper.

## II. Related Work

***Demand Response and Edge EDR.*** A series of recent studies exist on the reduction goal and social welfare maximization in demand response. For instance, in [12], a storage-assisted system is considered, with batteries and plug-in vehicles helping balance between supply and user demand. Chen *et al.* [13] 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 [14] and [15]. 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 scheduling takes place, targeting EDR in scheduling tasks online with more flexibility.

***Online Scheduling.*** Online scheduling is fundamental in cloud computing. [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, [16] and [17] 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.* [18] 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 in cloud computing, but also the flexibility to schedule tasks in heterogeneous clusters.

## III. System Model

### A. System Overview

We consider a community where a smart grid provisions electricity to $L$ heterogeneous cloudlets in a cluster. The cluster purchases $D$ units of electricity for $T$ time slots from the grid. While the grid is only in charge of power supply, the cloudlet cluster is responsible for its 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 $\{1, 2, \ldots, X\}$. A set of tasks $[J] = \{1, 2, \ldots, J\}$ runs in the cluster where they reside, receiving their workloads at end users via access points and transmitting them to the particular service providers. According to an agreement signed by the cluster operator and the smart grid, when an emergency event takes place, the cloudlet cluster needs to reduce $E$ units of electricity in the following $T$ time slots.

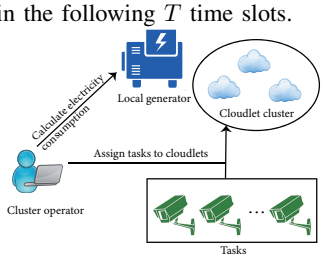



Fig. 1: Task scheduling in the cluster.

### B. Scheduling in Cloudlet Cluster

***Computing Task Information.*** Assume that each cloudlet $l \in [L]$ can process at most $R_l$ workloads. Let $[J]$ denote the task set. Each task $j \in [J]$ is expressed by a tuple: $\Gamma_j = \{b_j, a_j, d_j, w_j, \lambda_j, f_j(\tau_j)\}$, where $b_j$ is the value of task $j$ if it completes before its deadline. $a_j$ and $d_j$ are the arrival time and the deadline for task $j$, $w_j$ is the total number of time slots required to complete the task. The workload in one time slot is $\lambda_j$, so the total workload of task $j$ is $w_j\lambda_j$. $\tau_j$ refers to the level of deadline violation, whose penalty function is denoted by $f_j(\tau_j)$, with $f_{c_j}(\tau_j)$ non-decreasing and $f_{c_j}(0) = 0$:

$$f_j(\tau_j) = \begin{cases} f_{c_j}(\tau_j), & \tau_j \in [0, T - d_j] \\ +\infty, & \text{otherwise.} \end{cases} \tag{1}$$

***Decision Variable.*** As shown in Fig. 1, the cloudlet cluster schedules computing tasks to satisfy EDR energy consumption. 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 $t$.

***Problem Formulation.*** The cluster aims to maximize its own utility, *i.e.*, the sum of task value minus the sum of the delay penalty, and local generation cost. $p$ is the per-unit local generation cost and $u_g$ is the amount of local generation. The optimization problem is formulated as follows:

$$\text{maximize} \quad \sum_{j\in[J]} b_j x_j - \sum_{j\in[J]} f_j(\tau_j) - p u_g \tag{2}$$

$$\text{subject to:} \quad \sum_{j\in[J]} \lambda_j y_{jl}(t) \le R_l, \forall t \in [T], \forall l \in [L], \tag{2a}$$

$$\sum_{t\in[T]} \sum_{l\in[L]} e_l^t \le D - E + u_g, \tag{2b}$$

$$t \sum_{l\in[L]} y_{jl}(t) \le d_j + \tau_j, \forall t \in [T], \forall j \in [J] : a_j \le t, \tag{2c}$$

$$\sum_{l\in[L]} y_{jl}(t) \le 1, \forall j \in [J], \forall t \in [T], \tag{2d}$$

$$w_j x_j = \sum_{l\in[L]} \sum_{t\in[T]} y_{jl}(t), \forall j \in [J], \tag{2e}$$

$$y_{jl}(t), x_j \in \{0, 1\}, \quad \forall j \in [J], \forall l \in [L], \forall t \in [T], \tag{2f}$$

$$u_g \ge 0, \tau_j \ge 0, \forall j \in [J]. \tag{2g}$$

In the above problem, $e_l^t$ denotes the electricity consumption of cloudlet $l$ at time slot $t$. An empirical study on cloudlet [19] formulates the energy consumption through a linear function: $e_l^t = (N_l P_{idle}^l + (P_{peak}^l - P_{idle}^l) \sum_{j \in J} \lambda_j y_{jl}(t)) \cdot \mathrm{PUE}_l$, 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. $\mathrm{PUE}_l$, 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 cloudlet cluster to reduce $E$ electricity, its expected power consumption is the original demand minus the EDR requirement, $D - E$. Important notations are summarized in Table I.

Constraint (2a) guarantees that in each time slot, the cloudlet $l$ has enough computing resource to execute tasks. Constraint (2b) 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 (2c). Additionally, in (2d), we assume that each task runs on one cloudlet at most. Constraint (2e) connects two binary variables, $x_j$ and $y_{jl}(t)$, to guarantee sufficient execution.

***Challenges.*** We notice that if we let $u_g = 0$ and $\tau_j = 0, \forall j \in J$, as well as ignore (2c), it becomes a knapsack problem. It is known to be NP-hard, let alone the difficulties concerning online scheduling. Next, we propose an online algorithm to schedule tasks while satisfying EDR requirement.

TABLE I. Notation

| | |
|---|---|
| $L$ | # of cloudlets in the cluster |
| $T$ | # of time slots |
| $J$ | # of tasks in the cluster |
| $D$ | amount of electricity that the cloudlet purchased |
| $E$ | electricity reduction of the cluster |
| $b_j$ | value of task $j$ |
| $a_j$ | the arrival time of task $j$ |
| $d_j$ | the deadline of task $j$ |
| $w_j$ | # of time slots required for task $j$ |
| $\lambda_j$ | the workload of task $j$ in one slot |
| $\tau_j$ | # of slots that pass the deadline for task $j$ |
| $Z_l(t)$ | marginal price of unit workload in cloudlet $l$ at $t$ |
| $R_l(t)$ | the amount of allocated resources in cloudlet $l$ at $t$ |
| $p$ | 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) |

## IV. Online Task Scheduling

In this section, a primal-dual algorithm is proposed to schedule tasks in Sec. IV-A. The theoretical analysis is presented in Sec. IV-B.

### A. Online Scheduling Design

***Reformulation.*** We consider how to schedule tasks in the cloudlet cluster. To deal with non-conventional scheduling constraints in (2c) and (2e), we replace (2) with an equivalent convex problem. The new problem is formulated as follows:

$$\text{maximize} \quad \sum_{j \in J} \sum_{h \in \zeta_j} b'_{jh} \chi_{jh} - g(u) \tag{3}$$

subject to:

$$\sum_{j \in [J]} \sum_{h: t \in T(h), l \in L(h)} \lambda_j \chi_{jh} \le R_l, \forall l \in [L], \forall t \in [T], \tag{3a}$$

$$\sum_{l \in [L]} \sum_{j \in [J]} \sum_{h: l \in L(h)} \sum_{t: t \in T(h)} \beta_l \lambda_j \chi_{jh} \le u, \tag{3b}$$

$$\sum_{h \in \zeta_j} \chi_{jh} \le 1, \forall j \in [J], \tag{3c}$$

$$\chi_{jh} \in \{0,1\}, \forall j \in [J], \forall h \in \zeta_j, \tag{3d}$$

$$u \ge 0. \tag{3e}$$

In the above program, $\zeta_j$ 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 (2c) and (2e). Binary variable, $\chi_{jh}$, 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 $D' = D - E - T(\sum_{l \in [L]} N_l P_{idle}^l) \cdot \mathrm{PUE}_l$ and $u = D' + u_g$. $g(u)$ can be defined as a piecewise function as follows:

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

$g(u)$ indicates the energy consumption either below or above the EDR cap. We simplify the LHS of (2b), and let $\beta_l = (P_{peak}^l - P_{idle}^l) \cdot \mathrm{PUE}_l$. Constraints (3a) and (3b) are equivalent to (2a) and (2b).

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 $\phi_j$, as well as relaxing $\chi_{jh} \in \{0,1\}$ to $\chi_{jh} \ge 0$, the dual program is:

$$\text{minimize} \quad \sum_{l \in [L]} \sum_{t \in [T]} Z_l(t) R_l + \sum_{j \in [J]} \phi_j + g^*(C) \tag{4}$$

subject to:

$$\phi_j \ge b'_{jh} - \sum_{l \in L(h)} \sum_{t \in T(h)} Z_l(t) \lambda_j - \sum_{l \in L(h)} \sum_{t \in T(h)} C \beta_l \lambda_j, \quad \forall j \in [J], \forall h \in \zeta_j, \tag{4a}$$

$$Z_l(t), C, \phi_j \ge 0, \forall j \in [J], \forall l \in L(h), \forall t \in T(h) \tag{4b}$$

where $g^*(C)$ is the Fenchel conjugate [20] of $g(u)$:

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

***Allocation and Scheduling.*** Due to *complementary slackness* [21], each $\chi_{jh}$ has a corresponding constraint in (4a). Only when the constraint goes tight, can $\chi_{jh}$ 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 $\phi_j$ in dual constraint is non-negative, we assign $\phi_j$ as the maximum value between 0 and the RHS of (4a):

$$\phi_j = \max\left\{0, \max_{h\in\zeta_j}\{b'_{jh} - \sum_{l\in L(h)}\sum_{t\in T(h)} Z_l(t)\lambda_j - \sum_{l\in L(h)}\sum_{t\in T(h)} C\beta_l\lambda_j\}\right\}. \tag{5}$$

When $\phi_j > 0$, dual constraint (4a) 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 (4a) 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 (5) is 0, the task is rejected. Equality (5) determines the cloudlets and slots to schedule tasks for the maximum utility, which is a key to the utility maximization.

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:

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

$N$ and $M$ 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, $N = \min_{j\in[J]} \frac{b_j}{w_j\lambda_j}, M = \max_{j\in[J]} \frac{b_j}{w_j\lambda_j}$.

**Algorithm 1** Primal-Dual Based Online Allocation PD

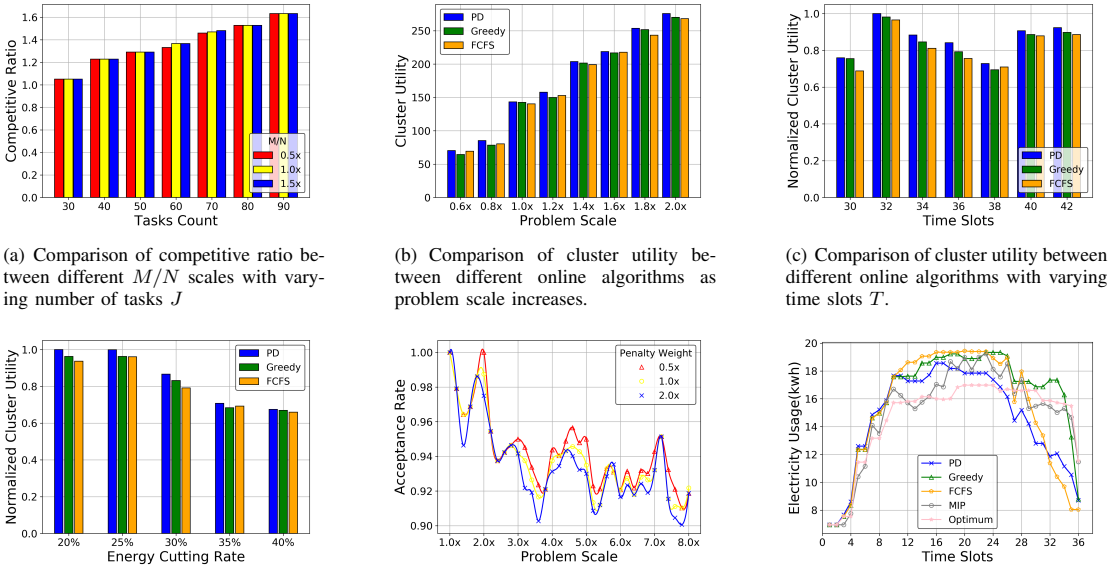
**Input**: $\{\beta_l, R_l\}_{\{l\in L\}}, C, N, M, D'$.
1: Initialize $x_j = 0, y_{jl}(t) = 0, \tau_j = 0, \phi_j = 0, R_l(t) = 0, u = 0, \forall j \in [J], \forall l \in [L], \forall t \in [T]$, by default.
2: **On the arrival of task** $j$
3: Run CORE$(\{R_l(t), Z_l(t)\}_{\{l\in[L],t\in[a_j,T]\}},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

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 (4a), and select the most economical one. Finally, we figure out the utility of task $j$ according to (5) in each slot. We reject $j$ if $\phi_j = 0$; otherwise, $j$ is accepted and we output the optimal schedule of task $j$.

**Algorithm 2** 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 $w_j$th 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 1. Initially, in line 1, all binary variables should be 0. Lines 2-8 call CORE in algorithm 2 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 the 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 $w_j$th 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.




(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. 2: Performance evaluation on PD

### B. Theoretical Analysis

*Theorem 1:* The Algorithm PD computes a feasible solution for problems in (2) (3) and (4).

**All missing proofs can be found in our technical report [22]**.

*Theorem 2:* PD runs in polynomial time.

The *Competitive Ratio* is defined as the upper bound ratio of the optimal objective value of (2) to the objective value achieved by PD. In reality, the ratio is always larger than 1.

*Theorem 3:* With $\alpha = \max_{j\in[J]}\{\frac{1}{\lambda_j}\left(\ln\left(\frac{\sigma M}{N}\right)+1\right)\}$, the competitve ratio of the task scheduling algorithm PD is $\frac{e}{e-1}\alpha$.

## V. Performance Evaluation

### A. Experiment Setup

To simulate the whole edge system, we utilize a real EDR event happening in New York on August 28, 2018, which lasted for 6 hours [23]. 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 [24]. As for the cloudlet, it is a small-scale cloud data center with 1-40 servers [25]. In this case, the typical PUE value is around 2.1 [26]. As for the strike price for the EDR event, mostly it is around \$1100 to \$1800 per MWh [27]. The *EDR dispatch rate* (*i.e.* the percentage of the number of clusters to engage in EDR event) is around 58% [28]. The idle power for each server is 60w, and the peak power is 180w [29]. The diesel price for local generation is set to 0.32\$/kWh[30].

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. We set 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 deadline. 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\$.

We use the MINLP solver SCIP to obtain the offline optimal solution for integer programming (2). 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.

### B. Data Analysis

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. In Fig. 2(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, this figure also reflects that the competitive ratio is likely to increase as the number of tasks rises.

Fig. 2(b) is the comparison between three online algorithms. We implement two benchmark algorithms: i) a greedy algorithm [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. 2(c) modifies the EDR duration time. We can find that the performance of PD is better than the other two algorithms. 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. 2(c), 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. 2(e), 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 deadline. However, this attribute is useful in an extreme condition with substantial worthless tasks to execute. Assume there are enough cloudlets to finish all the tasks, so the differences are slight in the figure.

Fig. 2(f) shows the electricity usage of different algorithms at each slot. Here we add the offline optimum and the online Mixed-integer Programming (MIP) which gets the optimal solution until current 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. PD is more eco-friendly, compared with other algorithms.

## VI. CONCLUSION

In this work, we study how to enable edge emergency demand response via cloudlet clusters control. To address challenges in task scheduling at a cloudlet cluster, we propose a control mechanism to facilitate edge EDR. We design an online primal-dual algorithm, PD, for the cluster to schedule and allocate its workload while satisfying EDR energy reduction requirement. PD runs in polynomial time and achieves a provable competitive ratio. Large-scale simulations verify the efficiency and advantages of our method over existing methods.

## REFERENCES


[1] M. Satyanarayanan, "The emergence of edge computing," *IEEE Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] Y. Zhang, K. Wang, Y. Zhou, and Q. He, "Enhanced adaptive cloudlet placement approach for mobile application on spark," *Security and Communication Networks*, vol. 2018, 2018.

[3] N. Ansari and X. Sun, "Mobile edge computing empowers internet of things," *IEICE Trans. on Comm*, vol. 101, no. 3, pp. 604–619, 2018.

[4] A. Molina-Garcia, F. Bouffard, and D. Kirschen, "Decentralized demand-side contribution to primary frequency control," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 411–419, 2010.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[6] L. Zhang, S. Lei, C. Wu, and Z. Li, "A truthful incentive mechanism for emergency demand response in colocation data centers," in *Proc. of IEEE INFOCOM*, 2015, pp. 2632–2640.

[7] J. Chen, D. Ye, S. Ji, Q. He, Y. Xiang, and Z. Liu, "A truthful fptas mechanism for emergency demand response in colocation data centers," in *Proc. of IEEE INFOCOM*, 2019, pp. 2557–2565.

[8] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau, "Online auctions in iaas clouds: Welfare and profit maximization with server costs," in *Proc. ACM SIGMETRICS*, vol. 43, 2015, pp. 3–15.

[9] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An efficient cloud market mechanism for computing jobs with soft deadlines," *IEEE-ACM Trans. Netw.*, vol. 25, no. 2, pp. 793–805, 2016.

[10] N. Jain, I. Menache, J. S. Naor, and J. Yaniv, "Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters," *ACM Trans. Parallel Comput.*, vol. 2, no. 1, pp. 3:1–3:29, Apr. 2015.

[11] Z. Dong, N. Liu, and R. Rojas-Cessa, "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers," *Journal of Cloud Computing*, vol. 4, pp. 1–14, 2015.

[12] R. Zhou, Z. Li, and C. Wu, "An online procurement auction for power demand response in storage-assisted smart grids," in *Proc. of IEEE INFOCOM*, 2015, pp. 2641–2649.

[13] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proc. of IEEE INFOCOM*, 2019, pp. 2566–2574.

[14] M. Islam, H. Mahmud, S. Ren, and X. Wang, "Paying to save: Reducing cost of colocation data center via rewards," in *Proc. of IEEE HPCA*, 2015, pp. 235–245.

[15] Q. Sun, C. Wu, Z. Li, and S. Ren, "Colocation demand response: Joint online mechanisms for individual utility and social welfare maximization," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3978–3992, 2016.

[16] S. Anand, K. Garg, and A. Kumar, "Resource augmentation for weighted flow-time explained by dual fitting," in *Proc. of SODA*, 2012, pp. 1228–1241.

[17] N. Devanur and Z. Huang, "Primal dual gives almost optimal energy-efficient online algorithms," *TALG*, vol. 14, no. 1, p. 5, 2018.

[18] S. Agrawal and N. R. Devanur, "Fast algorithms for online stochastic convex programming," in *Proc. of SODA*, 2014, pp. 1405–1424.

[19] A. Qureshi, "Power-demand routing in massive geo-distributed systems," Ph.D. dissertation, MIT, 2010.

[20] R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. V. Vazirani, and S. Yazdanbod, "Convex program duality, fisher markets, and nash social welfare," in *Proc. of ACM EC*, 2017, pp. 459–460.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, univ. press, 2004.

[22] Technical report, http://dwz.date/eG6.

[23] "NYISO Summer 2018 Hot Weather Operations," [Online], available: http://www.nysrc.org.

[24] G.Ghatikar, V. Ganti, N. Matson, and M. Piette, "Demand response opportunities and enabling technologies for data centers: Findings from field studies," 2012.

[25] V. Bahl, "Cloudlets for mobile computing," 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cloudlets-for-mobile-computing-2/

[26] M. Ganeshalingam, A. Shehabi, and L. Desroches, "Shining a light on small data centers in the U.S." 2017.

[27] "State of the Market Report for PJM." [Online]. Available: https://www.monitoringanalytics.com/reports/PJM_State_of_the_Market/2018/2018q3-som-pjm.pdf

[28] "2018 Utility Demand Response Market Snapshot." [Online]. Available: https://www.peakload.org/2018-utility-dr-snapshot-report

[29] D. Meisner and T. Wenisch, "Peak power modeling for data center servers with switched-mode power supplies," *ISLPED*, pp. 319–324, 2010.

[30] "U.S. On-Highway Diesel Fuel Prices." [Online]. Available: https://www.eia.gov/petroleum/gasdiesel/