# Cooperative Offloading of Tasks with Deadlines in Mobile Cloud Computing


## ABSTRACT

Mobile cloud computing (MCC) offers an agile solution that facilitates mobile applications to exploit cloud computing technology. Compared to offloading tasks to a remote cloud through wide area network (WAN), offloading to nearby cloudlets via WLAN links offers much smaller network delay. Resource capacity at cloudlets may become a bottleneck upon peak demands. A remote cloud then constitutes an appropriate back-up despite longer network delay. In this work, we design a cooperative task offloading scheme for tasks with hard deadlines, utilizing both local cloudlets and a remote cloud. By leveraging the *compact exponential optimization* technique, we design an online algorithm that simultaneously guarantees system efficiency, truthfulness, and social welfare maximization. A moderate parameterized competitive ratio is proven, and simulation studies reveal the algorithm's close-to-offline-optimum performance under practical settings.


## 1 INTRODUCTION

Mobile devices experienced explosive growth in the past few decades. According to the 2016-2021 Global Mobile Data Traffic Forecast [2], there will be 11.6 billion mobile devices by 2021, including M2M modules, exceeding the world's projected population at that time (7.8 billion). Widespread use of mobile devices has been accompanied by the rapid development of mobile applications. For computation-intensive and delay-sensitive applications, such as speech recognition, image processing, video analysis and augmented reality, resource limitations (*i.e.* battery life, storage and bandwidth) at mobile devices often become a concern. This motived recent studies that address the conflict between the increasing complexity of mobile applications and the limited local capabilities of mobile devices.

The recent paradigm of Mobile Cloud Computing (MCC) alleviates the challenge of resource constraints at portable devices, by offloading computation-intensive tasks to cloud servers. Example offloading solutions include COMET [13], MAUI [9] and CloneCloud [8]. Major cloud providers, such as Amazon EC2 [1], Microsoft Azure [5] and Google Cloud [3], provide computing services that can be remotely accessed by mobile users.

Offloading tasks to a remote cloud via wide area network (WAN) may incur long latencies. A natural solution is to offload deadline-constrained tasks to a nearby cloudlet for seamless interaction via a high-speed wireless local area network (WLAN). A *cloudlet* is a trusted, resource-limited cluster of idle devices (*e.g.* computers, servers), which are well-connected to the Internet and available to nearby mobile devices. Mobile users' demands for real-time interactive response can be met by low-latency, one-hop, high-bandwidth wireless access to cloudlets [10]. Fesehaye *et al.* [11] and Li *et al.* [15] show that one-hop cloudlets can avoid long data transfer delays and are often sufficient in practice. However, when the number of users in a certain area increases sharply, the particular cloudlet becomes overloaded. In marginal price based resource management schemes, where price increases as supply becomes tight, the marginal price becomes too high to afford and all requests will be rejected. This leads to lower social welfare, inferior quality of service (QoS), as well as lower quality of experience (QoE). The cost may even exceed the expense of offloading to a remote cloud with longer network delay. To balance system traffic load and improve system efficiency, we offload delay-insensitive tasks to remote cloud during peak hours, as long as their deadlines permit. The problem then becomes: where should each task be offloaded and how to schedule them. We propose a cooperative offloading scheme that aims to maximize social welfare.

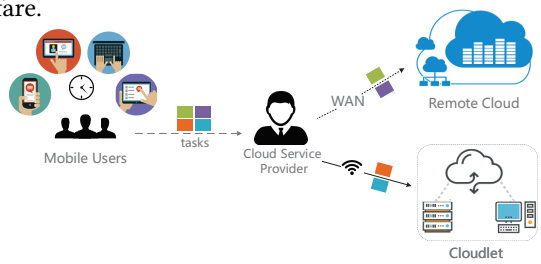


**Figure 1: A cooperative offloading scheme in MCC.**

In this work, we address the aforementioned challenges, and generalize existing offloading solutions by proposing an online cooperative offloading algorithm that handles tasks with execution deadlines, which is shown in Fig. 1. We focus on a scenario where a large number of mobile device users are served by both a cloudlet in their proximity, and a remote cloud. Our contributions can be summarized as follows:

- In order to solve the task offloading problem under limited resources and hard deadline constraints in MCC, we propose an online cooperative algorithm that balances the traffic load by utilizing a remote cloud for back-up resource supply.

- To incentivize the cloudlet and the remote cloud to participate in the cooperative offloading scheme, we design a truthful and efficient auction for task allocation and scheduling.
- We formulate this cooperative task offloading problem into an integer linear program (ILP), which cannot be directly solved by the classic primal-dual algorithm framework. By utilizing *compact exponential* method, we solve the problem in polynomial time. We conduct rigorous theoretical analysis to show that our algorithm achieves a small competitive ratio in typical scenarios.
- We further conduct simulation studies to verify the efficiency of our proposed online mechanism, and reveal its close-to-offline-optimal performance under realistic settings.

In the rest of this paper, we review related literature in Sec. 2. The cooperative system model is introduced in Sec. 3. The online cooperative auction is presented in Sec. 4. Simulations are presented in Sec. 5. Sec. 6 concludes the paper.

## 2 RELATED WORK

The concept of Cyber Foraging [19] was first proposed in 2001 to offload user equipments' tasks to idle devices for remote execution. With the emergence of MCC, many offloading solutions for mobile tasks were proposed. Bidoura *et al.* [14] first propose the task offloading scheme in MCC. Chen *et al.* [7] propose an offloading mechanism for multiple independent tasks to minimize system cost. MAUI [9] and CloneCloud [8] leverage virtualization technology to offload mobile applications (entirely or partially) to a remote cloud for execution. To overcome high access latencies and relieve traffic load on cellular networks, the cloudlet approach [10] is proposed, which deploys local cloud servers to provide computational resources for nearby user equipments. Although local cloudlets can help reduce network delay, their computation resources are limited, which may degrade system performance when traffic load is high. Roy *et al.* [18] propose an application-aware cloudlet selection strategy to make best use of resources. Gelenbe *et al.* [12] propose a load sharing strategy between cloudlet and remote cloud, aiming to strike a judicious balance between average response time and energy consumption. Different from the above literature, we are the first to utilize two different cloud platforms, *i.e.,* a remote cloud and a cloudlet, and combine resource, traffic and (hard) deadline constraints together to design a cooperative offloading scheme.

Large numbers of studies have been focused on auction mechanism design for cloud service provision and pricing. Wang *et al.* [21] study an offline resource allocation optimization problem in MCC. Shi *et al.* [20] first present an online combinatorial auction in virtual machine provisioning. To the authors' knowledge, the only primal-dual algorithm that addresses task scheduling problems with soft deadlines is proposed by Zhou *et al.* [22], who introduce a new *compact exponential* optimization framework , and utilize a dual oracle to efficiently update primal and dual variables. Our work has been partially inspired by that of Zhou's work, but the techniques cannot be directly adapted due to different model structure. The above schemes cannot be directly applied to our systems, since they do not consider hard deadlines and the trade-offs when choosing from two types of cloud service platforms.

## 3 SYSTEM MODEL

We consider a cloud service provider (CSP) who owns an MCC system, which comprises of: (i) a cloudlet with large numbers of mobile device users in its proximity, and (ii) a remote cloud for back-up resource supply. CSP acts as a auctioneer who receives bids and determines the resource allocation strategy, *i.e.,* decides whether to accept it, which cloud platform to choose, and how to schedule its task if the bid is accepted. We assume that the cloudlet and the remote cloud each hosts a pool of $K$ types of resource, *i.e.,* CPU, GPU, RAM, disk storage and network bandwidth. We use $[X]$ to denote the set {1,2,...,$X$}. A binary variable $s$ indicates the cloud platforms' type: the cloudlet (0) or the remote cloud (1). There are $C_0^k$ ($C_1^k$) units of type-$k$ resource in cloudlet (remote cloud). $I$ mobile users act as buyers arriving sequentially across the system life span 1,2,...,$T$. Each submits, upon its arrival at time $t_i$, a bid for multiple types of resources required for its task execution. We order simultaneous submitted bids randomly. We use $r_i^k$ to denote the amount of type-$k$ resource required by bid $i$ and $d_i$ to denote the hard deadline of bid $i$. Let $n_i$ denote the total number of time slots the $i$th user needs, which can be executed separately. We use $b_i$ to denote the total willingness-to-pay submitted in bid $i$, and $v_i$ to denote the true valuation of bidder $i$. Let $B_i$ denote the $i$th bid submitted at $t_i$ and $D_i$ denote the distance between user $i$'s mobile device and remote cloud, which is related to the network delay discussed in detail later. A bid can therefore be expressed as follows:

$$B_i = \{t_i, \{r_i^k\}_{k\in[K]}, b_i, d_i, n_i, D_i\} \tag{1}$$

Upon receiving each bid, CSP makes an immediate decision on whether to accept the bid, and how to assign its task if accepted. We set $x_{is}$ to 1 when its bid is accepted and served on platform $s$, otherwise, we reject the bid can set $x_{is}$ to 0.

If user $i$ wins its bid, CSP needs to compute a corresponding payment $p_i$. Let $p_s^k(t)$ be the unit price of type-$k$ resource in cloud platform $s$ at time $t$. Let $y_s^k(t)$ denote the

allocated amount of type-$k$ resource at time $t$ in cloud platform $s$. CSP updates the payment as the amount of allocated resources varies. We use a binary variable $h_s^i(t)$ to denote whether to allocate user $i$'s task on platform $s$ at slot $t$ or not. Let $u_i$ denote user $i$'s utility, which equals its bidding price minus its payment when it wins, and zero otherwise, *i.e.,* $u_i = (v_i - p_i)\sum_{s\in\{0,1\}} x_{is}$. If each user can always maximize its utility by bidding its true valuation, *i.e.*, for all $b_i \neq v_i, u_i(v_i) \geq u_i(b_i)$, we can say that this auction is truthful. The social welfare over $T$ is the sum of the cloud resource provider's profit $\sum_{i\in[I]} p_i$ and bidders' aggregate utility $\sum_{i\in[I]}\sum_{s\in\{0,1\}} v_i x_{is} - \sum_{i\in[I]} p_i$, and is simplified to $\sum_{i\in[I]}\sum_{s\in\{0,1\}} v_i x_{is}$.

**Table 1: List of Notations**

| | | | |
|---|---|---|---|
| $I$ | # of mobile users | $K$ | # of types of resources |
| $T$ | # of time slots | $n_i$ | # slots requested by user $i$ |
| $X$ | integer set {1,2,...X} | $t_i$ | user $i$'s arrival time |
| $p_i$ | user $i$'s payment | $u_i$ | user $i$'s utility |
| $d_i$ | user $i$'s deadline | $\tau_i$ | user $i$'s network delay |
| $s$ | cloudlet ($s$ = 0) and remote cloud ($s$ = 1) | | |
| $r_i^k$ | demand of type-$k$ resource by user $i$ | | |
| $b_i$ | bidding price of user $i$'s bid | | |
| $v_i$ | true valuation of user $i$'s bid | | |
| $x_{is}$ | serve bid $i$ on cloud platform $s$ (1) or not (0) | | |
| $C_s^k$ | capacity of type-$k$ resource on cloud platform $s$ | | |
| $p_s^k(t)$ | marginal price of type-$k$ resource on cloud platform $s$ | | |
| $y_s^k(t)$ | allocated type-$k$ resource at time $t$ on cloud platform $s$ | | |
| $h_s^i(t)$ | whether to allocate user $i$'s task in $t$ | | |
| $D_i$ | distance between user $i$ and remote cloud | | |

As for the network delay, we only consider propagation delay, processing delay and transmission delay here. Given that mobile users and the cloudlet are only one-hop apart, the corresponding propagation delay is negligible. The transmission delay can be significantly different (due to bandwidth) between mobile users and the cloud/cloudlet. The processing delay can also be much longer for processing at the cloudlet rather than in the cloud. In order to simplify calculation, we denote the network delay of user $i$'s work $\tau_i$ as: $\tau_i = f(D_i, BW_i, C_s^k)$. $f$ is a function defined over the distance of user $i$ and remote cloud. Since the distance is almost the same in a particular area, the delay $\tau_i$ is a constant number for bid $i$.

# 4 ONLINE COOPERATIVE ALGORITHM

In this section, we first formulate the cooperative task offloading problem into an ILP in Sec. 4.1, and utilize the *compact exponential* method to reformulate the original ILP. Then we design an online cooperative algorithm in Sec. 4.2, theoretical analysis is presented in Sec. 4.3.

## 4.1 Social Welfare Maximization Problem

Under the assumption of truthful bidding ($b_i = v_i$), the cooperative task offloading problem can be formulated as follows:

$$\text{maximize} \quad \sum_{i\in[I]}\sum_{s\in\{0,1\}} b_i x_{is} \tag{2}$$

subject to:

$$h_s^i(t)t \leq d_i x_{is}, \forall i \in [I], t \in [T] : t_i + \tau_i x_{is} s \leq t, s \in \{0,1\} \tag{2a}$$

$$n_i x_{is} \leq \sum_{t\in[T]: t_i+\tau_i x_{is} s \leq t} h_s^i(t), \forall i \in [I], s \in \{0,1\} \tag{2b}$$

$$\sum_{i\in[I], t_i+\tau_i \leq t} h_s^i(t) r_i^k \leq C_s^k, \forall k \in [K], t \in [T], s \in \{0,1\} \tag{2c}$$

$$\sum_{s\in\{0,1\}} x_{is} \leq 1, \forall i \in [I] \tag{2d}$$

$$x_{is}, h_s^i(t) \in \{0,1\}, \forall i \in [I], t \in [T], s \in \{0,1\} \tag{2e}$$

Constraint (2a) ensures that a task is processed between its arrival time and deadline, with network delay considered. Constraint (2b) guarantee that the allocated time slots are sufficient for a winning bid. Constraint (2c) limits the consumption of each type of resource by its capacity. Constraint (2d) indicates that each task can be offloaded to at most one cloud platform, *i.e.*, the nearby cloudlet or the remote cloud.

Even in the offline setting, ILP (2) is NP-hard as a knapsack problem. When we consider task deadline and online decision making, the challenge further escalates. The classic primal-dual framework, well-known for designing efficient solutions to NP-hard ILP problems, does not directly apply here, due to the unconventional constraints that model task deadlines.

To tackle these problems, we adopt the recent *compact exponential* optimization scheme [22]. The compact exponential framework reformulates a polynomial-size ILP with both conventional and non-conventional constraints into conventional constraints only, at the expense of introducing exponential number of variables. First, let $\xi_i$ denote the set of time schedules that satisfy constraints (2a), (2b) and (2d) for user $i$, *i.e.,* $l = \{h_s^i(t)\}_{\forall t\in[T], s\in\{0,1\}}$ Then, we reformulate the original ILP (2) into a simplified packing structure, at the expense of involving an exponential number of decision variables:

$$\text{maximize} \quad \sum_{i\in[I]}\sum_{l\in\xi_i} b_{il} x_{il} \tag{3}$$

subject to:

$$\sum_{i\in[I]}\sum_{l: t\in T(l), s\in S(l)} r_i^k x_{il} \leq C_s^k, \forall k \in [K], t \in [T], s \in \{0,1\} \tag{3a}$$

$$\sum_{l\in\xi_i} x_{il} \leq 1, \forall i \in [I] \tag{3b}$$

$$x_{il} \in \{0,1\}, \forall i \in [I], l \in \xi_i \tag{3c}$$

The value of $b_{il}$ is related to schedule $l$ and equals the corresponding $b_i$. $T(l)$ and $S(l)$ denote the set of time slots and the set of platforms when and where bid $i$ is executed in

schedule $l$, respectively. Constrains (3a) and (3b) are equivalent to (2c) and (2d). A feasible solution to (2) is also a feasible solution to (3), and vice versa. Therefore, the optimal objective values of both problems is the same. We next relax $x_{il} \in \{0,1\}$ to $x_{il} \geq 0$ and formulate the dual LP. By introducing dual variables $p_s^k(t)$ and $u_i$ to (3a) and (3b) respectively, the dual LP of the relaxed (3) is:

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in[I]} u_i + \sum_{k\in[K]}\sum_{t\in[T]}\sum_{s\in\{0,1\}} C_s^k p_s^k(t) && (4)\\ \text{subject to:} \quad & \\ & u_i \geq b_{il} - \sum_{k\in[K]}\sum_{l\in\xi_i} r_i^k p_s^k(t), \forall i \in [I], l \in \xi_i && (4a)\\ & p_s^k(t), u_i \geq 0, \forall i \in [I], t \in [T], k \in [K], s \in \{0,1\} && (4b) \end{aligned}$$

We next design an online algorithm which only updates a polynomial number of variables, but can simultaneously solve optimization problems in (2), (3), (4).

## 4.2 Online Auction Design

We next focus on the online auction design to maximize social welfare, assuming that all bids are truthful. Deciding whether to accept user $i$'s bid and how to schedule its task is a key point in such design. If CSP decides to offload task $i$ on cloud platform $s \in \{0,1\}$, then let $x_{is} = 1$ and increases the allocated resources $y_s^k(t)$ by $r_i^k$ on cloud platform $s$ for all type-$k$ resource ($k \in [K]$) on selected slots. Meanwhile, $\tau_i$ is calculated by function $f(D_i)$ if user $i$'s task is offloaded to the remote cloud. Otherwise, $x_{is}$ is zero.

**Task Allocation.** In order to solve (2), we resort to the help of (3) and its dual (4). For each primal variable $x_{il}$, there is a corresponding dual constraint (4a). According to complementary slackness, $x_{il}$ is updated based on its dual constraint, and it becomes 1 once the dual constraint becomes tight. Whenever a bid $i$ arrives, a dual variable $u_i \geq 0$ generated, which subject to constraint (4a), $u_i \geq b_{il} - \sum_{k\in[K]}\sum_{t\in[T]}\sum_{s\in\{0,1\}} r_i^k p_s^k(t)$. Therefore, let $u_i$ be the maximum of 0 and the right hand side (RHS) of constraints (4a):

$$u_i = \max\{0, \max_{l\in\xi_i}\{b_{il} - \sum_{k\in[K]}\sum_{t\in[T(l)]}\sum_{s\in[S(l)]} r_i^k p_s^k(t)\}\} \quad (5)$$

Accordingly, we adopt the following method to decide whether to accept user $i$'s bid and how to schedule its task based on $u_i$: If no schedule $l$ achieves a positive value on the RHS of (4a) ($u_{il} \leq 0, \forall l \in \xi_i$), user $i$'s bid is rejected. Otherwise, user $i$'s bid is served according to schedule $l^*$ which maximizes the RHS, *i.e.,* $x_{il^*} = 1$ and $x_{il} = 0, \forall l \neq l^*$.

The rationale is as follows: If we interpret $p_s^k(t)$ as the marginal price per unit of type-$k$ resource on cloud platform $s$ at time $t$, then the second term on RHS of (4a) is the total payment that user $i$ should pay. The RHS of (4a) is user $i$'s utility under schedule $l$. Therefore, (5) makes sure that we always schedule task $i$ according to schedule $l$ which maximizes its utility.

**Payment Design.** We next design a payment function that adjusts resource price based on realtime demand-supply conditions. Let marginal price $p_s^k(t)$ be a function of $y_s^k$. Moreover, payment should be independent of user $i$'s bid in order to guarantee truthfulness [16]. Let $L_i$ represent the minimum task value of user $i$ per unit resource per unit of time.

$$L_i = \frac{1}{2\beta}\min_{i\in[I]}\frac{b_i}{\sum_{k\in[K]} n_i r_i^k} \quad (6)$$

As shown above, the minimum task value $L_i$ is scaled down by a factor $\frac{1}{2\beta}$. Here $\frac{1}{\beta}$ is a lower bound of the ratio of allocated type-$k$ resource over its capacity accumulated in $T$ time slots. That is

$$\frac{1}{\beta} = \min_{k\in[K]}\frac{\sum_{t\in[1,T]}\sum_{s\in\{0,1\}} y_s^k(t)}{\sum_{s\in\{0,1\}} TC_s^k} \quad (7)$$

Let $U_i$ denote the maximum task value of user $i$ per unit resource per unit of time:

$$U_i = \max_{i\in[I]}\frac{b_i}{\sum_{k\in[K]} n_i r_i^k}. \quad (8)$$

The marginal price $p_s^k(t)$ increases as $y_s^k$ increases, for ensuring that a cloud platform will not waste all the resources to bids with low values. $p_s^k(t)$ starts at $L_i$ and exponentially increases until $y_s^k(t)$ is close to $C_s^k$. It reaches $U_i$ when the amount of allocated resources $y_s^k(t)$ reaches the total capacity $C_s^k$. Thus, the marginal price is defined as follows:

$$p_s^k(y_s^k(t)) = L_i(\frac{U_i}{L_i})^{\frac{y_s^k(t)}{C_s^k}} \quad (9)$$

**Auction Mechanism.** We next design a cooperative online algorithm $A_{online}$, as shown in Alg. 1, and the one-round scheduling algorithm $A_{core}$ in Alg. 2.

**Algorithm 1** A Cooperative Online Algorithm $A_{online}$

**Require:** $\{B_i\}$, $\{C_s^k\}$

1: Define marginal price function $p_s^k(y_s^k(t))$ accoring to (9);
2: Initialize $x_i = 0, y_s^k(t) = 0, h_s^i(t) = 0, \tau_i = 0, u_i = 0, p_s^k(t) = L_i$ $\forall i \in [I], k \in [K], t \in [T]$; Let $x_{il} = 0, \forall i \in [I], l \in [\xi_i]$, by default;
3: **Upon user $i$'s arrival**
4: $(x_{is}, \{h_s^i(t)\}, p_i, \{p_s^k(t)\}, \{y_s^k(t)\}) = A_{core}(B_i, \{C_s^k\}, \{p_s^k(t)\}, \{y_s^k(t)\})$
5: **if** $\exists s \in \{0,1\}, x_{is} = 1$ **then**
6: Accept user $i$'s bid and allocate resources to cloud platform $s$ according to $h_s^i(t)$; Charge $p_i$ from user $i$;
7: **else**
8: Reject bid $i$.
9: **end if**

Upon each user's arrival, CSP makes instant decision on whether to accept user $i$'s bid and how to schedule its task by algorithm $A_{core}$. The value of $x_{is_i}$ is also updated to 1 if bid $i$ is accepted ($u_i > 0$) and offloaded to cloud platform $s_i$. After

that, increasing utilization $y_s^k(t)$ for each type of resources on the selected cloud platform in the corresponding time slots according to schedule $l_i$. We next update dual marginal price variables $p_s^k(t)$ according to the pricing function we proposed.

**Algorithm 2** A Scheduling Algorithm $A_{core}$

**Require:** bidding language $\{B_i\}$, $\{C_s^k\}$, $\{p_s^k(t)\}$, $\{y_s^k(t)\}$
**Ensure:** $x_{is}$, $\{h_s^i(t)\}$, $p_i$, $\{p_s^k(t)\}$, $\{y_s^k(t)\}$
**for all** $s \in \{0,1\}$ **do**
  **if** $s = 1$ **then**
    $\tau_i = f(D_i)$;
  **else**
    $\tau_i = 0$;
  **end if**
  Add slot $t \in [t_i + \tau_i, d_i]$ to set $\chi$ if $y_s^k(t) + r_i^k \le C_s^k$, $\forall k \in [K], \forall s \in \{0,1\}$;
  Select the first $n_i$ slots $(t_1, t_2, ..., t_{n_i})$ from $\chi$ to be schedule $l_0$ ; Define $j = 1$;
  **while** $n_i + j \le |\chi|$ **do**
    $l_j = l_{j-1}$;
    Let $t_c$ be the $(n_i + j)$th slot in $\chi$;
    $p(t) = \sum_{k\in[K]} r_i^k p_s^k(t), \forall t \in \{t_1, t_2, ..., t_{n_i}, t_c\}$;
    $t_m = \arg\max_{t\in\{t_1,t_2,...,t_{n_i-1}\}} p(t)$;
    **if** $p(t_{n_i}) < p(t_m)$ **then**
      for schedule $l_j$, replace the slot $t_m$ with $t_{n_i}$ and save $t_c$ into $t_{n_i}$;
    **end if**
    $P_j = \sum_{t\in[T(l_j)]} p(t)$; $j = j + 1$;
  **end while**
  $s^* = argmin_j\{P_j\}$; $P_s^* = P_{s^*}$; $l_s^* = l_{s^*}$;
**end for**
$\tilde{s} = argmin_s\{P_s^*\}$; $\tilde{P} = P_{\tilde{s}}^*$, $\tilde{l} = l_{\tilde{s}}^*$;
**if** $b_i - \tilde{P} > 0$ **then**
  $x_{i\tilde{s}} = 1$; $h_{\tilde{s}}^i(t) = 1, \forall t \in T(\tilde{l})$; $x_{i\tilde{l}} = 1$;
  $u_i = b_i - \tilde{P}$; $p_i = \sum_{k\in[K]} \sum_{t\in T(\tilde{l})} r_i^k p_{\tilde{s}}^k(t)$;
  $y_{\tilde{s}}^k(t) = y_{\tilde{s}}^k(t) + r_i^k, \forall k \in [K], t \in [T(\tilde{l})]$;
  $p_{\tilde{s}}^k = p_{\tilde{s}}^k(y_{\tilde{s}}^k(t)), \forall k \in [K], t \in T(\tilde{l})$;
**end if**
**return** $x_{is}$, $\{h_s^i(t)\}$, $p_i$, $\{p_s^k(t)\}$, $\{y_s^k(t)\}$

When computing $u_i$, we face the problem that there are an exponential number of dual constraints related to it. We then design an efficient subroutine to output a polynomial size of schedules. Therefore, we only need to consider dual constraints related to selected schedules. We fix the finishing time of user $i$'s task to be $t_c$ ($t_c \in [t_i + \tau_i + n_i - 1, d_i]$) for each cloud platform $s \in \{0,1\}$. Then we select from all schedules with the same finishing time to find the best schedule $l_j$ which has the minimum price. The base case is schedule $l_0$ with completion time slots $t$ in $[t_i + \tau_i,\ (t_i + \tau_i + n_i) - 1]$, we next push the finishing time one slot forward each round. Let $p(t)$ denote the price for user $i$'s task running at $t$, *i.e.*, $p(t) = \sum_{k\in[K]} r_i^k p_s^k(t)$. During the process, we just compare the price of the old finishing time and $n_i - 1$ slots before it when replacing the finishing time. The process is repeated until $t_c$ reaches user $i$'s deadline $d_i$.

## 4.3 Theoretical Analysis

Lemma 1. **(Myerson 1981:)** *Let $Pr(B_i)$ be the probability of bidder i winning in an auction and $B_{-i}$ be the bidding prices except user i. A mechanism is truthful if and only if the following holds for a fixed $B_{-i}$ [17]:*

*1) $Pr(B_i)$ is monotonically non-decreasing in $B_i$;*

*2) bidder i is charged $B_i Pr(B_i) - \int_0^{B_i} Pr(B)dB$.*

Theorem 1. *Online auction $A_{online}$ is a truthful auction.*

*Proof:* (*Truthful in bidding price*) We can prove the truthfulness by showing that our auction mechanism holds for the two conditions in Lemma 1.

First, we assume that there are enough working time slots for each task, then if other bids are fixed, there must exist one and only one optimal schedule with the minimum payment denoted by $p_i^*$. Thus, $Pr(B_i)$ is as follow:

$$Pr(B_i) = \begin{cases} 0 & if\ B_i < p_i^* \\ 1 & if\ B_i \ge p_i^* \end{cases}$$

Therefore, $Pr(B_i)$ is monotonically no-decreasing in $B_i$ which supports condition 1, and the payment is as follow: If $B_i < p_i^*$, we have

$$B_i Pr(B_i) - \int_0^{B_i} Pr(B)dB = B_i * 0 - 0 = 0$$

Which is consistent with $A_{online}$ as the bid is rejected without payment.

Then if $B_i \ge p_i^*$, the payment is:

$$\begin{aligned} & B_i Pr(B_i) - \textstyle\int_0^{B_i} Pr(B)dB \\ &= B_i * 1 - \textstyle\int_0^{p_i^*} Pr(B)dB - \int_{p_i^*}^{B_i} Pr(B)dB \\ &= B_i - 0 - (B_i - p_i^*) \\ &= p_i^* \end{aligned}$$

Which is the same as the payment in $A_{online}$. If there are no available working time slots for bid $i$, the $Pr(B_i)$ is always 0 and the auction mechanism also holds for the above conditions. So it's obvious that our auction is truthful in bidding price.

(*Truthful in deadline*) On the one hand, if a bidder submits a delayed deadline, its task may not be completed on time. On the other hand, if a bidder submits a earlier deadline, its task may be allocated to the time slots with higher resource utilization rate, which leads to a higher payment and lower personal utility.

(*Truthful in resource occupation time*) Since the marginal price is non-decreasing, if a bidder requests more time slots

than its true demand, its payment will accordingly increase, thus leading to a lower utility. Besides, no bidder will drop part of its true occupation time slots due to the risk of failing to finish its task. □

Lemma 2. *The algorithm $A_{core}$ runs in $O(KT + ST^2)$ time.*

*Proof:* The definition of the network delay function in line 1 takes constant time. Inside each iteration of the *for* loop, lines 3-7 take constant time to compute corresponding network delay. Line 8 initializes a feasible slot set $\chi$ in $O(KT)$ steps. Line 9 defines a schedule $l_0$ in $n_i$ steps. The *while* loop (lines 10-19) aims to compute the best schedule $l_j$ when the completion time is fixed, and will iterate at most $T - n_i$ times. Inside the *while* loop body, lines 11-13 update $p_t$ in $O(n_i + 1)$ steps. The running time of finding the maximum price in line 14 is linear to $n_i$. It takes constant time to finish the comparison and replacement in lines 14-17. Accordingly, the whole *while* loop can be executed in $O((T - n_i)n_i)$ steps. Line 20 can be done in $O(T - n_i)$ steps to find the schedule with the minimum price. Lines 2-21 are the whole *for* loop, and the corresponding execution time is $O(S(T - n_i)n_i)$. The running time for the *if* body is $O(Kn_i)$. In summary, the total running time in algorithm $A_{core}$ is $O(Kn_i + Sn_i(T - n_i)) \le O(KT + ST^2)$. □

Theorem 2. *$A_{online}$ in Alg. 1 returns feasible solutions for convex problems (2), (3) and (4) in polynomial running time.*

*Proof:* The definition of the marginal price function in line 1 takes constant time and line 2 initializes all the primal and dual variables in linear time. By Lemma 2, $A_{core}$ in line 4 processes each user's bid in $O(KT + ST^2)$ time. The *if* statement in lines 5-9 can be executed in constant time. In conclusion, after handling the last bid, the total running time of $A_{online}$ is $O(I(KT + ST^2))$. □

In the next part, we analyze the competitive ratio of our cooperative auction, *i.e.*, the worst-case upper-bound ratio of social welfare achieved by the offline solution of (2) to the overal social welfare achieved by our algorithm at the end of $T$. We next introduce a primal-dual framework by proving Lemma 3, which states that if there exists a bound between the increment of primal objective value and the increment of dual objective value. It also means that competitive ratio is bounded. Let $P_0$ and $D_0$ be the initial values of primal and dual objectives. Let $P_i$ and $D_i$ denote the objective values of primal problem (3) and that of dual problem (4) returned by an algorithm after handling user $i$'s bid. Let $OPT_1$ and $OPT_2$ denote the optimal primal objective values of (2) and (3), and obviously $OPT_1 = OPT_2$. Note that $P_I$ and $D_I$ are the final primal and dual objective values at the end of $T$.

Lemma 3. *If there exists a constant $\alpha \ge 1$ such that $P_i - P_{i-1} \ge \frac{1}{\alpha}(D_i - D_{i-1})$ for every i, and $C \ge 1$ such that $D_0 \le \frac{1}{C}OPT$, then $A_{core}$ in $A_{online}$ is $\alpha \cdot \frac{C}{C-1}$-competitive.*

*Proof:* Summing up the inequality of each step $i$, we have,

$$P_I - P_0 = \sum_i (P_i - P_{i-1}) \ge \frac{1}{\alpha}\sum (D_i - D_{i-1}) = \frac{1}{\alpha}(D_I - D_0).$$

Here, we use the fact that $P_0 = 0$ and $D_0 = \sum_k \sum_t \sum_s C_s^k L_i$, thus $P_I = \frac{1}{\alpha}(D_I - D_0)$. By weak duality [6], $D_I \ge OPT$ and $(D_I - D_0) \ge (1 - \frac{1}{C}OPT)$, so we have $P_I \ge \frac{1}{\alpha}(1 - \frac{1}{C})OPT$. Therefore, the competitive ratio is $\frac{1}{\frac{1}{\alpha}(1-\frac{1}{C})} = \alpha \cdot \frac{C}{C-1}$. □

We next define a *Allocation Price Relationship* for $A_{online}$ and prove that if the *Allocation Price Relationship* holds for a given $\alpha$, which is sufficient to guarantee the inequality in Lemma 3. Let $p_{ks}^{i-1}(t)$ and $y_{ks}^i(t)$ respectively denote the price and the amount of allocated type-$k$ resource in server $s$ after processing user $i$'s bid.

Definition 1. *The* Allocation Price Relationship *for $A_{online}$ with $\alpha \ge 1$ is:*

$$\begin{aligned} &p_{ks}^{i-1}(t)(y_{ks}^i(t) - y_{ks}^{i-1}(t)) \ge \tfrac{1}{\alpha}C_s^k(p_{ks}^i(t) - p_{ks}^{i-1}(t)), \\ &\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall s \in S(l). \end{aligned} \tag{10}$$

Lemma 4. *If* Allocation Price Relationship *holds for a given $\alpha \ge 1$, then $A_{online}$ guarantees $P_i - P_{i-1} \ge \frac{1}{\alpha}(D_i - D_{i-1})$ for all $i \in [I]$.*

Proof: If bid $i$ is rejected, then $P_i - P_{i-1} = D_i - D_{i-1} = 0$. Then we assume that bid $i$ is accepted, and $l$ is the schedule of it. The increment of primal objective function after handling user $i$'s bid is:

$$\begin{aligned} &P_i - P_{i-1} = b_{il} \\ &= u_i + \textstyle\sum_{k\in[K]} \sum_{t\in T(l)} \sum_{s\in S(l)} r_i^k p_{ks}^{i-1}(t) \end{aligned}$$

The second equation holds because $A_{online}$ updates the value of dual variables such that constraint (**??**) becomes tight when user $i$'s bid is accepted by server $s$ with schedule $l$, then the left hand side of constraint (**??**) equals the right hand side of it. The increment of dual objective value is:

$$D_i - D_{i-1} = u_i + \sum_{t\in[T(l)]} \sum_{s\in S(l)} \sum_{k\in[K]} C_s^k(p_{ks}^i(t) - p_{ks}^{i-1}(t))$$

Since $y_{ks}^i(t) - y_{ks}^{i-1} = r_i^k$, by summing up *Allocation Price Relationship* over all $s \in S(l)$, $t \in T(l)$ and $k \in [k]$, we have:

$$P_i - P_{i-1} \ge u_i + \frac{1}{\alpha}(D_i - D_{i-1} - u_i).$$

Note that $u_i \ge 0$ and $\alpha \ge 1$, it is obvious that $P_i - P_{i-1} \ge \frac{1}{\alpha}(D_i - D_{i-1})$. □

Each inequality in *Allocation Price Relationship* involves variables only for type-$k$ resource in server $s$. In the following analysis, we try to identify corresponding $\alpha_{ks}$ for each pair of $s$ and $k$ that satisfies *Allocation Price Relationship*. Then $\alpha$ is just the maximum value among all $\alpha_{ks}$. We next define the differential version of *Allocation Price Relationship* based on the assumption that user $i$'s demand $r_i^k$ is much smaller than capacity $C_s^k$ in server $s$. As a result, $y_{ks}^i(t) - y_{ks}^{i-1} = \mathrm{d}y_s^k(t)$ and the *Differential Allocation Price Relationship* is as follow:

Definition 2. *The* Differential Allocation Price Relationship *for $A_{online}$ with $\alpha_{ks} \geq 1$ is:*

$$p_s^k(t)dy_s^k(t) \geq \frac{C_s^k}{\alpha_{ks}} dp_s^k(t),$$

$$\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall s \in S(l).$$

Lemma 5. *The marginal price function defined in (9) satisfies the* Differential Allocation Price Relationship *for $\alpha = ln(\frac{U_i}{L_i})$, with $L_i$ and $U_i$ defined in (6) and (8).*

*Proof:* According to (9), we have

$$p_s^k(y_s^k(t))' = \frac{L_i ln(\frac{U_i}{L_i})}{C_s^k} (\frac{U_i}{L_i})^{\frac{y_s^k(t)}{C_s^k}}$$

Since $\mathrm{d}p_s^k(t) = p_s^k(t)\mathrm{d}y_s^k(t)$, the *Differential Allocation Price Relationship* becomes:

$$L_i(\frac{U_i}{L_i})^{\frac{y_s^k(t)}{C_s^k}} dy_s^k(t) \geq \frac{C_s^k}{\alpha_{ks}} \frac{L_i ln(\frac{U_i}{L_i})}{C_s^k} (\frac{U_i}{L_i})^{\frac{y_s^k(t)}{C_s^k}} dy_s^k(t)$$

That is,

$$\alpha_{ks} \geq ln(\frac{U_i}{L_i})$$

Therefore this Lemma holds for $\alpha_{ks} = ln(\frac{U_i}{L_i})$ □

Theorem 3. *The online cooperated auction $A_{online}$ in Alg. 1 is $2\alpha$-competitve in social welfare with $\alpha = \max_{k\in[K]}\{\frac{U_i}{L_i}\}$. $L_i$ and $U_i$ are defined in (6) and (8).*

*Proof:* According to Lemma (5), the marginal function used in $A_{online}$ satisfy the *Differential Allocation Price Relationship* for $\alpha_{ks} \geq ln(\frac{U_i}{L_i})$, since $\mathrm{d}y_s^k(t) = y_{ks}^i(t) - y_{ks}^{i-1} = r_i^k$ is much smaller than the server capacity $C_s^k$, we have, $\mathrm{d}p_s^k(t) = p_s^k(y_s^k(t))'$. So the *Differential Allocation Price Relationship* inplies that the *Allocation Price Relationship* holds for $\alpha$. According to $A_{online}$, we have:

$$\begin{aligned} D_0 &= \textstyle\sum_{k\in[K]} \sum_{t\in[T]} \sum_{s\in\{0,1\}} C_s^k p_s^k(0) \\ &= \textstyle\sum_{k\in[K]} \sum_{t\in[T]} \sum_{s\in\{0,1\}} \frac{C_s^k}{2\beta} \min_{i\in[I]} \frac{b_i}{\sum_{k\in[K]} n_i r_i^k} \\ &= \textstyle\sum_{k\in[K]} \sum_{s\in\{0,1\}} \frac{TC_s^k}{2\beta} \min_{i\in[I]} \frac{b_i}{\sum_{k\in[K]} n_i r_i^k} \end{aligned} \tag{11}$$

since $\frac{1}{\beta} \leq \frac{\sum_{t\in[1,T]} \sum_{s\in\{0,1\}} y_s^k(t)}{\sum_{s\in\{0,1\}} TC_s^k}$, then equation (11) can be formaluated as:

$$D_0 \leq \frac{1}{2} \sum_{k\in[K]} \sum_{t\in[T]} \sum_{s\in\{0,1\}} y_s^k(t) \min_{i\in[I]} \frac{b_i}{\sum_{k\in[K]} n_i r_i^k} \tag{12}$$

As shown above, $\min_{i\in[I]} \frac{b_i}{\sum_{k\in[K]} n_i r_i^k}$ is the minimum per-time-slot per-resource-amount task value among all the bids, so we have:

$$OPT \geq \sum_{k\in[K]} \sum_{t\in[T]} \sum_{s\in\{0,1\}} y_s^k(t) \min_{i\in[I]} \frac{b_i}{\sum_{k\in[K]} n_i r_i^k}$$

Hence $D_0 \leq \frac{1}{2} OPT$.

Then by Lemma 3 ($C = 2$), Lemma 4 and 5, the theorem follows. □

## 5 SIMULATION STUDIES

**Simulation Settings**. We evaluate the performance of our online algorithm through trace-driven simulation studies based on Google Cluster Data [4], which contains tasks information such as resource demands, arrival times, deadlines, and execution time. We assume that each bid requests 2 types of resources. We assume that each task consumes [1, 12] time slot(s), each of 30 second long. Each bid arrives sequentially in 18 hours. User $i$'s deadline is generated randomly between its arrival time and the system end time. The bidding price of each task is equal to the unit marginal price multiplied by the corresponding requested amount of resource. We set the marginal price in the range $[L_i, U_i]$. The default values are $L_i = 1$ and $U_i = 30$ for $A_{online}$. For ease of calculation, the default capacity of type-$k$ resource is set to 100 in the cloudlet and 1000 in the remote cloud, to show the difference between their capacities. We set the default value of network delay as 20 ms.

**Performance of $A_{online}$**. Fig. 2 shows the social welfare achieved by $A_{online}$ under different numbers of users and $U_i/L_i$. We notice that social welfare becomes higher with the increment of $U_i/L_i$ and user population. Theorem **??** reveals that $U_i/L_i$ determines the competitive ratio as shown in Fig. 3. Here underestimation is slightly better than overestimation, since overestimation leads to a higher payment, and filters out tasks which should be accepted. As can be seen in Fig. 6, $A_{online}$ always achieves close-to-optimal performance regardless of the value of $n_i$. When $n_i \leq 18$, social welfare increases while each user requests more time slots, which is reasonable since social welfare is mostly contributed by the bidding price, and user $i$'s bidding price is higher when its task needs a long execution time. When $n_i > 20$, social welfare drops a little bit since the competition for resources in each time slot is fiercer with a larger $n_i$, so the percentage of winners would decrease, leading to a smaller overall social welfare.

We next simulate the impact of user $i$'s execution time $n_i$ on accept rates. As shown in Fig. 5, When the value of $n_i$ grows, both the cloudlet and remote cloud can accept fewer tasks, and the accept rate of cloudlet decreases sharply to zero due to the limited resource capacity. As shown in Fig. 4, with the increment of the upper bound of $r_i^k$ and the value of network delay, the social welfare increases. This is because, when $r_i^k$ increases, the bidding price becomes larger, then social welfare increases if the total accept rate does not change a lot. Fig. 7 shows that the overall percentage of winners of $A_{online}$ decreases when the number of users getting larger and the value of $U_i/L_i$ getting smaller. This is

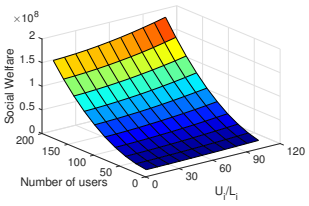


**Figure 2: Social Welfare of $A_{online}$ under different numbers of users and $U_i/L_i$.**

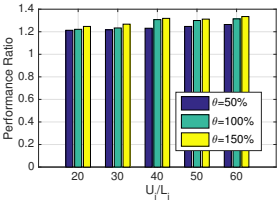


**Figure 3: Performance Ratio of $A_{online}$ under different $U_i/L_i$ and estimated $U_i$.**

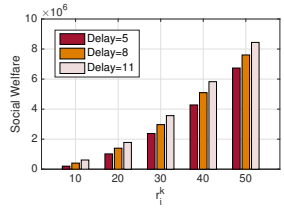


**Figure 4: Social Welfare of $A_{online}$ under different upper bounds of $r_i^k$ and values of $U_i/L_i$.**

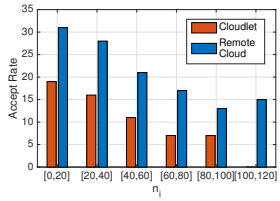


**Figure 5: Accept rate of cloudlet and remote cloud under different range of $n_i$.**

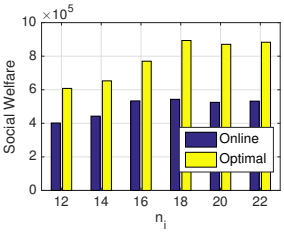


**Figure 6: social welfare under different $n_i$.**

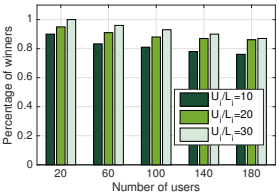


**Figure 7: Percentage of winners of $A_{online}$ with different numbers and $U_i/L_i$.**

because the number of winners is limited by the resource capacity and the current price of resource within $[L_i, U_i]$. When the value of $U_i/L_i$ becomes smaller, the difference between bidding prices gets smaller. Therefore, more bids with similar bidding price are rejected due to the increasing price during each round.

## 6 CONCLUSION

In this work, we propose a cooperative task offloading scheme with (hard) deadlines in MCC, to alleviate traffic load in nearby cloudlet during peak hours, by utilizing a remote cloud for back-up resource supply. We properly deal with the trade-offs between network delay and resource price. We utilize the *compact exponential* optimization method for efficient approximation algorithm design, and obtain a truthful auction that achieves near-to-optimal social welfare in polynomial time. Trace driven simulations validate our theoretical analysis and show good performance of our online algorithm.

## REFERENCES


[1] [n. d.]. *Amazon Elastic Compute Cloud Documentation.* https://amazonaws-china.com/documentation/ec2/.

[2] [n. d.]. *Global mobile data traffic forecast update, 2016-2021 white paper.* http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] [n. d.]. *Google Cloud Platform.* https://cloud.google.com/.

[4] [n. d.]. *Google Cluster Data.* https://github.com/google/cluster-data.

[5] [n. d.]. *Microsoft Azure.* https://azure.microsoft.com/zh-cn/.

[6] S. Boyd and L. Vandenberghe. 2004. *Convex optimization.* Cambridge University Press.

[7] M. H. Chen, B. Liang, and M. Dang. 2016. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In *Proc. of IEEE International Conference on Communications.*

[8] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. 2011. CloneCloud: Elastic execution between mobile device and cloud. In *Proc. of EuroSys.*

[9] E. Cuervo. 2010. Maui: Making smartphones last longer with code offload. In *Proc. of ACM MobiSys.*

[10] H. Dinh, C. Lee, D. Niyato, and P. Wang. 2013. A Survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing* 13, 18 (2013), 1587–1611.

[11] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang. 2012. Impact of cloudlets on interactive mobile cloud applications. In *Proc. of IEEE EDOC.*

[12] R. Genlenbe, R. lent, and M. Douratsos. 2012. Choosing a local or remote cloud. In *Proc. of Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on.*

[13] Mark S Gordon, Davoud Anoushe Jamshidi, Scott A Mahlke, Zhuoqing Morley Mao, and Xu Chen. 2012. COMET: Code Offload by Migrating Execution Transparently.. In *Proc. of OSDI*, Vol. 12. 93–106.

[14] B. A. Hridita, M. Irfan, and M. S. Islam. 2016. Task alocation for mobile cloud computing: State-of-the-art and open chanllenges. In *Proc. of ICIEV.*

[15] Y. Li and W. Wang. 2014. Can mobile cloudlets support mobile applications?. In *Proc. of IEEE INFOCOM.*

[16] R.B. Myerson. 1981. Optimal Auction Design. *INFORMS* 6, 1 (1981), 58–73.

[17] R. B. Myerson. 1981. Optimal Auction Design. *Mathematics of Operations Research* 6, 1 (1981), 58–73.

[18] D. G. Roy, D. De, A. Mukherjee, and R. Buyya. 2016. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. *Journal of Supercomputing* (2016), 1–19.

[19] M. Satyanaratanan. 2001. Pervasive computing: Vision and challenges. *IEEE Pers. Commun* 8, 4 (2001), 10–17.

[20] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. 2014. An online auction framework for dynamic resource provisioning in cloud computing. In *Proc. of ACM SIGMETRCS.*

[21] K. Wang, K. Yang, and C. Magurawalage. 2015. Joint energy minimization and resource allocation in C-RAM with mobile cloud. *Comupting Science* PP, 99 (2015), 1–1.

[22] Ruiting Zhou, Zongpeng Li, Chuan Wu, and Zhiyi Huang. 2017. An efficient cloud market mechanism for computing jobs with soft deadlines. *IEEE/ACM Transactions on networking* 25, 2 (2017), 793–805.