Cost-Efficient Workload Scheduling in Cloud Assisted Mobile Edge Computing

Xiao Ma*, Shan Zhang†, Wenzhuo Li*, Puheng Zhang*, Chuang Lin*, Xuemin (Sherman) Shen†
*Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
†Department of Electrical and Computer Engineering, University of Waterloo, Waterloo N2L 3G1, Canada

***Abstract*—Mobile edge computing is envisioned as a promising computing paradigm with the advantage of low latency. However, compared with conventional mobile cloud computing, mobile edge computing is constrained in computing capacity, especially under the scenario of dense population. In this paper, we propose a Cloud Assisted Mobile Edge computing (CAME) framework, in which cloud resources are leased to enhance the system computing capacity. To balance the tradeoff between system delay and cost, mobile workload scheduling and cloud outsourcing are further devised. Specifically, the system delay is analyzed by modeling the CAME system as a queuing network. In addition, an optimization problem is formulated to minimize the system delay and cost. The problem is proved to be convex, which can be solved by using the Karush-Kuhn-Tucker (KKT) conditions. Instead of directly solving the KKT conditions, which incurs exponential complexity, an algorithm with linear complexity is proposed by exploiting the linear property of constraints. Extensive simulations are conducted to evaluate the proposed algorithm. Compared with the fair ratio algorithm and the greedy algorithm, the proposed algorithm can reduce the system delay by up to 33% and 46%, respectively, at the same outsourcing cost. Furthermore, the simulation results demonstrate that the proposed algorithm can effectively deal with the challenge of heterogeneous mobile users and balance the tradeoff between computation delay and transmission overhead.**

I. Introduction

With the development of mobile devices, new types of computation-intensive, delay-sensitive mobile applications are drawing increasing attention, such as augmented reality [1] and recognition assistance [2]. Mobile edge computing is envisioned as a promising paradigm which helps to relieve the conflict between these applications and the resource-constrained mobile devices [3]-[6]. By endowing wireless access network with powerful computing capabilities, mobile edge computing enables mobile users access abundant computing resources without suffering from the uncontrolled Internet delay and jitters [3]. However, compared with public clouds (e.g., Amazon AWS and Microsoft Azure), the computing capacity of mobile edge is limited, especially when serving large number of mobile users.

Hybrid cloud is an effective solution to address this problem [8], [9]. In a hybrid cloud framework, overwhelming workloads in the internal infrastructure are outsourced to a public cloud in times of overload. Based on the idea of hybrid cloud, we propose the Cloud Assisted Mobile Edge computing (CAME) framework. As shown in Fig. 1, cloud resources, encapsulated as instances [13], are tenanted by the mobile edge operator to enhance the computing capacity. Mobile computation tasks can be offloaded to the mobile edge via wireless access network. Excessive workloads in the mobile edge are further outsourced to the cloud through the Internet.

The CAME system exhibits heterogeneous computation resources. The computation resources consist of three types, i.e., resources on mobile devices, mobile edge and the cloud. In addition, mobile devices have different computation capacities. To improve quality of service (QoS) of the system, an effective workload scheduling algorithm is required to fully utilize the computation resources, which coordinates computation offloading decisions of multiple mobile users and balances the workloads of mobile edge and the cloud.

System delay is a key QoS metric in mobile edge computing. In the CAME system, the system delay consists of both computation delay and communication delay. Offloading computation workloads from mobile users is beneficial to access abundant computation resources, which reduces computation delay. However, additional communication traffic is meanwhile yielded in the wireless access network (e.g., photos are transmitted in face recognition applications [10]), increasing communication delay. An effective workload scheduling algorithm should jointly consider the computation workload scheduling and relevant communication burden, and balance the tradeoff between computation delay and communication overhead.

Minimizing the system cost (including internal cost and outsourcing cost) is an essential objective of mobile edge operators. In this work, the workload scheduling problem is analyzed with the objective of minimizing the system delay and cost. The workload scheduling problem is challenging in three-fold. First, mobile users are heterogeneous in computing capabilities and mobile requests. The computation resources of mobile edge and the cloud further augment the heterogeneity. Second, computation and communication resources are jointly shared by the mobile users. A workload scheduling algorithm should consider both the computation workload scheduling and the relevant communication overhead. Finally, renting more cloud resources can effectively reduce computation delay, while introducing higher outsourcing cost. The workload scheduling algorithm should also consider the tradeoff between system delay and cost.

978-1-5386-2704-4/17/$31.00© 2017 IEEE

To address these challenges, the workload scheduling decision is made by the management component in the mobile edge, which consists of offloading decision maker and outsourcing decision maker. As shown in Fig. 1, the offloading decision maker implements the computation offloading decisions for mobile users. The workloads offloaded from each mobile user are determined by taking into account the heterogeneous user capacities and the tradeoff between computation delay and communication overhead. The outsourcing decision maker optimizes the usage of cloud resources and balances the workloads between mobile edge and the cloud, by considering the tradeoff between system delay and cost.

The CAME system is modeled as a queuing network to analyze the system delay. An optimization problem is formulated with the objective of minimizing the system delay and cost. The optimization problem is proved to be a convex problem with $(2N+6)$ inequation constraints (where N denotes the amount of mobile users). Solving this problem based on the Karush-Kuhn-Tucker (KKT) conditions [11] yields a complexity of $O(4^N)$, which is impractical in real systems. By exploiting the linear property of the constraints, we propose an algorithm that achieves optimality with linear complexity.

The contributions of this paper are summarized as follows.

- The CAME framework is proposed to enhance the system computing capacity. In addition, the workload scheduling problem is analyzed, in which computation and communication workloads are jointly scheduled. Considering the heterogeneity of computation resources and mobile tasks, computation offloading decisions of mobile users are coordinated, and outsourcing decision is made which optimizes the usage of cloud resources and balances the load between mobile edge and the cloud.
- The system delay is analyzed based on queueing network modeling. An optimization problem is formulated with the objective of minimizing the system delay and cost. Complexity of the optimization problem is analyzed, and by exploiting the linear property of constraints, an algorithm with linear complexity is proposed.
- Extensive simulations are conducted to evaluate the proposed algorithm. The results demonstrate that at the same system cost, the proposed algorithm can significantly reduce the system delay compared with the fair ratio algorithm and the greedy algorithm. Furthermore, the proposed algorithm achieves desired performance in dealing with heterogeneity of mobile users and balancing the tradeoff between computation resources and transmission overhead.

The paper is organized as follows. The system model is presented in Section II. Problem formulation and algorithm design are illustrated in Section III and Section IV, respectively. In Section V, extensive simulations are conducted and in Section VI, related work is discussed. Finally, the work is concluded in Section VII.

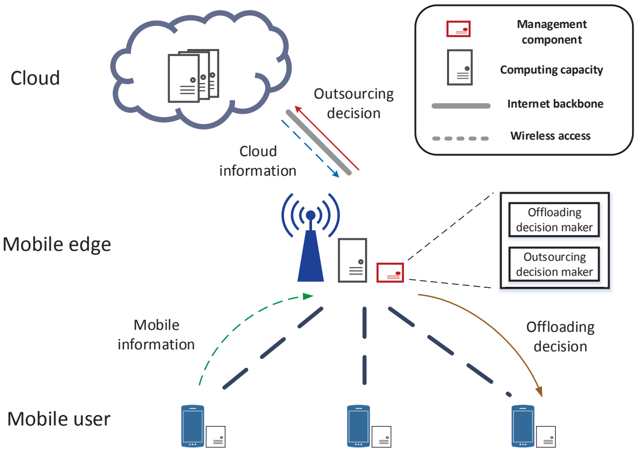

Fig. 1. The Cloud Assisted Mobile Edge computing (CAME) framework.

II. System Model

A. Overview of the CAME System

Fig. 1 shows the architecture of the CAME system. The mobile edge, which is connected to the cloud through the Internet, has computation resources (edge computing capacity) and communication resources (the wireless access network). Mobile users have local computing capabilities and are able to offload excessive mobile requests to mobile edge through the wireless access network.

When mobile requests arrive at the system, the management component in the mobile edge collects mobile information (i.e., mobile requests and computing capacities of mobile devices) from each mobile user and the cloud information (i.e., prices and computing capacities of cloud instances). Computation offloading decisions are implemented by the offloading decision maker to determine the workloads offloaded from each mobile user. Heterogeneity of mobile users is taken into account and the tradeoff between computation delay and transmission overhead is balanced. When overwhelming workloads are offloaded to the mobile edge, cloud resources are leased to augment the computing capacity. The outsourcing decision is made to determine the optimal usage of cloud resources and balance the workloads between the mobile edge and the cloud.

B. Analytical Model

In this section, the analytical model of the CAME system is presented. There is a base station s with edge computation resources and a set of $\Upsilon = \{1, 2, ..., N\}$ mobile users, each of which has computation-intensive, delay-sensitive applications to be executed. Cloud resources are leased by the mobile edge operator to enhance the system computing capacity. To illustrate the function of each part in the CAME system, a queuing network model is proposed, as shown in Fig. 2.

1) Mobile User Model: The arrival of computation requests at user i is modeled as a Poisson Process with the parameter of λ_i $(i \in \Upsilon)$. Denote by μ_i the serving rate of user i $(i \in \Upsilon)$.

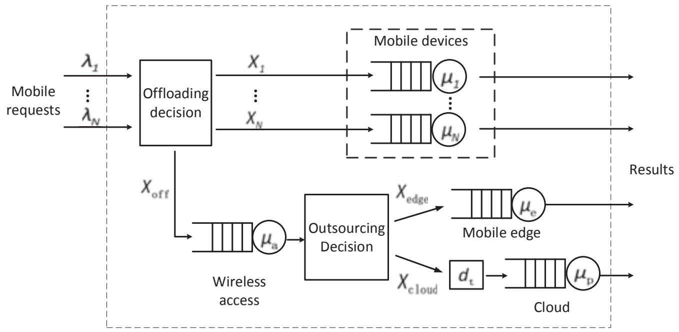

Fig. 2. Analytical model of the CAME system.

Offloading computation requests is beneficial to access abundant computation resources, which, however, incurs additional transmission overhead in the access network. To deal with this tradeoff and fully exploit the computing capacities of heterogeneous mobile users, computation offloading decisions are made to determine the workloads offloaded from each mobile user. Let x_i be average rate of computation requests executed on mobile device i,

$$0 \leq x_i \leq \lambda_i, \tag{1}$$

then the average computation delay on user i can be given by

$$D_i = \frac{1}{\mu_i - x_i}. \tag{2}$$

To provision high quality of service, it should be ensured that

$$x_i < \mu_i. \tag{3}$$

2) Wireless Access: Denote by x_{off} the overall rate of computation requests offloaded from the mobile users, then

$$x_{\text{off}} = \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} x_i. \tag{4}$$

When offloading mobile computation requests through the wireless access network, additional communication traffic (e.g. parameters and program codes) is generated. There are c units of transmission requests in the wireless access network when one unit of computing request is transmitted. Let λ_{a} be the arrival rate of transmission requests in the access network, then

$$\lambda_{\text{a}} = c x_{\text{off}}. \tag{5}$$

Denote by μ_{a} the serving rate of the access network. According to the analysis in [12], the average transmission delay in the access network can be given by

$$D_{\text{a}} = \frac{1}{\mu_{\text{a}} - c x_{\text{off}}}, \tag{6}$$

where

$$0 \leq x_{\text{off}} < \frac{\mu_{\text{a}}}{c}. \tag{7}$$

Equation (6) indicates that the transmission delay significantly increases with the amount of offloaded computation requests, degrading the QoS of the system.

3) The Mobile Edge and the Cloud: In the CAME system, cloud resources are leased to enhance the system computing capacity. The outsourcing decision is made to optimize the usage of cloud resources and determine the workloads outsourced to the cloud. Denote by x_{edge} the arrival rate of computation requests executed at the mobile edge and μ_{e} as the computing rate of mobile edge hosts, then the computation delay in the mobile edge is

$$D_{\text{edge}} = \frac{1}{\mu_{\text{e}} - x_{\text{edge}}}, \tag{8}$$

where

$$0 \leq x_{\text{edge}} < \mu_{\text{e}}. \tag{9}$$

Let x_{cloud} be the computation requests outsourced to the cloud,

$$x_{\text{cloud}} = x_{\text{off}} - x_{\text{edge}}. \tag{10}$$

In the cloud, computation resources are provided as encapsulated instances, e.g., AWS instances [13]. Denote k as number of the tenanted cloud instances,

$$0 \leq k \leq M. \tag{11}$$

Then, the serving rate of the cloud is

$$\mu_{\text{p}} = k \mu_{\text{ins}}, \tag{12}$$

where μ_{ins} denotes the serving rate of a single instance. Thus the computation delay in the cloud is given by

$$D_{\text{cloud}} = \frac{1}{k\mu_{\text{ins}} - x_{\text{cloud}}}, \tag{13}$$

where

$$0 \leq x_{\text{cloud}} < k\mu_{\text{ins}}. \tag{14}$$

C. System Delay and Cost

The system delay can be considered as a weighted sum of delay in each part of the system, with the weight represented by the ratio of computation requests. Thus the system delay can be computed as follows

$$\begin{aligned} D =& \sum_{i=1}^{N} \frac{x_i}{\lambda} D_i + \frac{x_{\text{off}}}{\lambda} D_{\text{a}} + \frac{x_{\text{edge}}}{\lambda} D_{\text{edge}} + \frac{x_{\text{cloud}}}{\lambda} (D_{\text{cloud}} + d_{\text{t}}) \\ =& \frac{1}{\lambda} (\sum_{i=1}^{N} \frac{x_i}{\mu_i - x_i} + \frac{x_{\text{off}}}{\mu_{\text{a}} - c x_{\text{off}}} + \frac{x_{\text{edge}}}{\mu_{\text{e}} - x_{\text{edge}}} + \\ & \frac{x_{\text{cloud}}}{k\mu_{\text{ins}} - x_{\text{cloud}}} + x_{\text{cloud}} d_{\text{t}}), \end{aligned} \tag{15}$$

where

$$\lambda = \sum_{i=1}^{N} \lambda_i, \tag{16}$$

and d_{t} is the transmission delay between mobile edge and the cloud.

The system cost consists of internal cost and outsourcing cost. Internal cost represents the cost on infrastructure of mobile edge. As constant computing capacity is provisioned in mobile edge, the internal cost is a constant, denoted as

c_m. On-demand instances (such as AWS on-demand instances) are selected as the representative of cloud instances, which are charged hourly without upfront fees or long-term commitments [13]. Denote by c_p the price of a single on-demand instance, then the outsourcing cost is

$$C_\mathrm{p} = kc_\mathrm{p}. \tag{17}$$

Thus the system cost is given by

$$C = kc_\mathrm{p} + c_\mathrm{m}. \tag{18}$$

III. Problem Formulation and Analysis

In the CAME system, an effective workload scheduling mechanism is required to provision optimal QoS with minimized cost. In this section, the workload scheduling problem is formulated as an optimization problem, with the analysis and solutions presented in the following.

As mentioned in Sec. I, workload scheduling in the CAME system is implemented by the management component in the mobile edge. Computation offloading decisions are made to determine the workloads executed on mobile devices ($x_i, i \in \Upsilon$) and those offloaded (x_off). The outsourcing decision maker determines the optimal usage of cloud instances (k) and balance the workloads between mobile edge (x_edge) and the cloud (x_cloud). Thus, in the workload scheduling problem, the decision vector is denoted as

$$\hat{a} \triangleq < x_1, x_2, ..., x_N, x_\mathrm{off}, x_\mathrm{edge}, x_\mathrm{cloud}, k > . \tag{19}$$

System delay is a critical metric of QoS. In addition, minimizing the system cost is an essential objective. Thus, the workload scheduling problem can be formulated as

$$\begin{aligned} &\min \quad f(\hat{a}) = C(\hat{a}) + VD(\hat{a}) \\ &s.t. (1)(3)(4)(7)(9)(10)(11)(14). \end{aligned} \tag{20}$$

Here, V is a weight constant. $C(\hat{a})$ and $D(\hat{a})$ are as shown in (15) and (18), respectively.

As the number of cloud instances k is a discrete variable in $\{0, 1, 2, ..., M\}$, problem (20) is solved by the following two steps:

Step1. For each given $k \in \{0, 1, 2, ..., M\}$, solve the optimization problem

$$\begin{aligned} &\min f(\hat{x}) = C(\hat{x}) + VD(\hat{x}) \\ &s.t. (1)(3)(4)(7)(9)(10)(14), \end{aligned} \tag{21}$$

where

$$\hat{x} \triangleq < x_1, x_2, ..., x_N, x_\mathrm{off}, x_\mathrm{edge}, x_\mathrm{cloud} > . \tag{22}$$

Denote $\hat{x}^*(k)$ as the optical solution of (21).

Step2. Determine the optimal $k^* \in \{1, 2, ..., M\}$, where

$$k^* = \underset{k \in \{0,1,2,...M\}}{\arg} \min\{f(\hat{x}^*(k), k)\}. \tag{23}$$

Theorem 1. *Problem (21) is a convex optimization problem.*

Proof: Please refer to Appendix A.

As problem (21) is a convex optimization problem, the solutions are analyzed by introducing the Lagrange Function [11]

$$L(\hat{x}, \hat{\eta}, \hat{\sigma}) = f(\hat{x}) + \sum_{i=1}^{2N+6} \eta_i g_i(\hat{x}) + \sum_{i=1}^{2} \sigma_i h_i(\hat{x}). \tag{24}$$

Here, η_i $(i \in \{1, 2, ..., 2N+6\})$ and σ_i $(i \in \{1, 2\})$ are the Lagrange multipliers. $g_i(\hat{x})$ represents the standard form of inequation constraints in problem (21) as follows:

$$\begin{array}{ll} g_i(\hat{x}) = x_i - u_i & i = 1, 2, ..., N \\ g_{i+N}(\hat{x}) = -x_i & i = 1, 2, ..., N \\ g_{2N+1}(\hat{x}) = x_\mathrm{off} - \frac{\mu_\mathrm{a}}{c} & \\ g_{2N+2}(\hat{x}) = -x_\mathrm{off} & \\ g_{2N+3}(\hat{x}) = x_\mathrm{edge} - \mu_\mathrm{e} & \\ g_{2N+4}(\hat{x}) = -x_\mathrm{edge} & \\ g_{2N+5}(\hat{x}) = x_\mathrm{cloud} - k\mu_\mathrm{ins} & \\ g_{2N+6}(\hat{x}) = -x_\mathrm{cloud}, & \end{array} \tag{25}$$

where u_i denotes the upper bound of x_i $(i = 1, 2, ..., N)$. $h_i(\hat{x})$ $(i \in \{1, 2\})$ represent the standard form of equality constraints, defined as

$$\begin{array}{l} h_1(\hat{x}) = \sum\limits_{i=1}^{N} \lambda_i - \sum\limits_{i=1}^{N} x_i - x_\mathrm{off} \\ h_2(\hat{x}) = x_\mathrm{edge} + x_\mathrm{cloud} - x_\mathrm{off}. \end{array} \tag{26}$$

For a convex optimization problem, points $< \hat{x}, \hat{\eta}, \hat{\sigma} >$ that satisfy the KKT conditions suffice to be the optimal solutions [11]. Therefore, problem (21) can be solved by searching for points that satisfy the KKT conditions, i.e.,

$$\begin{array}{ll} \nabla f(\hat{x}) + \sum\limits_{i=1}^{2N+6} \eta_i \nabla g_i(\hat{x}) + \sum\limits_{i=1}^{2} \sigma_i \nabla h_i(\hat{x}) = 0 & \\ \eta_i g_i(\hat{x}) = 0, & i = 1, 2, ..., 2N+6 \\ \eta_i \geq 0, & i = 1, 2, ..., 2N+6 \\ g_i(\hat{x}) \leq 0, & i = 1, 2, ..., 2N+6 \\ h_i(\hat{x}) = 0, & i = 1, 2. \end{array} \tag{27}$$

Searching for points that satisfy the equation constraints in (27) can induce 2^{2N+6} results, due to the $(2N+6)$ inequation constraints in problem (21). Therefore, solving problem (21) based on the KKT conditions yields an exponential complexity $O(4^N)$, which is impractical in real situations.

IV. Algorithm Design

In this section, to simplify the search for the optimal solutions of problem (21), an algorithm with linear complexity is designed by exploiting the linear property of the constraints. The algorithm is presented with complexity analysis in the following.

A. Algorithm Overview

According to the analysis in the above section, the main factor that incurs exponential complexity is the inequation constraints in problem (21). To simplify the search for optimal solutions, the inequation constraints are removed first. Then $\hat{x}$

can be represented as a function of Lagrange multipliers $\hat{\sigma}$ according to the gradient equation

$$\nabla f(\hat{x}) + \sum_{i=1}^{2} \sigma_i \nabla h_i(\hat{x}) = 0. \tag{28}$$

Since x_i is linearly constrained in $[0, u_i]$ (i.e., the inequation constraints in problem (21)), when searching for the optimal value of $\hat{\sigma}$, x_i is constrained by a piecewise function of $\hat{\sigma}$ in $[0, u_i]$. Then, the optimal solutions of problem (21) can be obtained by the bisection search method.

B. Proposed Algorithm with Linear Complexity

According to (28), $\hat{x}$ can be represented by a function of $\hat{\sigma}$ as

$$\begin{array}{l} l_i(\sigma_1) = \mu_i - \sqrt{\frac{V\mu_i}{\sigma_1 \sum\limits_{j=1}^{N} \lambda_j}} \quad i = 1, 2, ..., N \\ l_{\text{off}}(\sigma_1, \sigma_2) = \frac{1}{c}(\mu_{\text{a}} - \sqrt{\frac{V\mu_{\text{a}}}{(\sigma_1+\sigma_2) \sum\limits_{j=1}^{N} \lambda_j}}) \\ l_{\text{edge}}(\sigma_2) = \mu_{\text{e}} - \sqrt{-\frac{V\mu_{\text{e}}}{\sigma_2 \sum\limits_{j=1}^{N} \lambda_j}} \\ l_{\text{cloud}}(\sigma_2) = k\mu_{\text{ins}} - \sqrt{-\frac{Vk\mu_{\text{ins}}}{\sigma_2 \sum\limits_{j=1}^{N} \lambda_j + Vd_{\text{t}}}}. \end{array} \tag{29}$$

Since x_i is constrained in $[0, u_i]$, where

$$\begin{array}{l} u_i = \min\{\lambda_i, \mu_i - \varsigma\} \; i = 1, 2, ...N \\ u_{\text{off}} = \frac{\mu_{\text{a}} - \varsigma}{c} \\ u_{\text{edge}} = \mu_{\text{e}} - \varsigma \\ u_{\text{cloud}} = k\mu_{\text{ins}} - \varsigma \end{array} \tag{30}$$

x_i can be given by

$$x_i = \begin{cases} 0 & \text{if } l_i(\sigma_1, \sigma_2) < 0 \\ u_i & \text{if } l_i(\sigma_1, \sigma_2) > u_i \\ l_i(\sigma_1, \sigma_2) & \text{otherwise.} \end{cases} \tag{31}$$

Here, $i \in \{1, 2, ..N, \text{off}, \text{edge}, \text{cloud}\}$ and ς is a positive constant.

By substituting (31) into equation (4), σ_2 can be derived as a function of σ_1 as

$$\sigma_2 = \frac{V\mu_{\text{a}}}{\sum\limits_{i=1}^{N} \lambda_i [\mu_{\text{a}} + c(\sum\limits_{i=1}^{N} x_i(\sigma_1) - \sum\limits_{i=1}^{N} \lambda_i)]^2} - \sigma_1. \tag{32}$$

Thus the equation constraint $h_2(\hat{x})$ can be given by

$$h_2(\sigma_1) = x_{\text{edge}}(\sigma_1) + x_{\text{cloud}}(\sigma_1) - x_{\text{off}}(\sigma_1). \tag{33}$$

Theorem 2. *$h_2(\sigma_1)$ is monotonically increasing with σ_1.*

Proof: Please refer to Appendix B.

Due to the monotonicity of $h_2(\sigma_1)$, the σ_1 that satisfies $h_2(\sigma_1) = 0$ can be obtained with the bisection search method. Thus the optimal solution $\hat{x}^*$ can be determined according to (31) and (32). The algorithm is summarized in Algorithm 1.

Algorithm 1 Workload Scheduling with Linear Complexity

```
Represent $\hat{x} = \{x_i, ..., x_N, x_{\text{off}}, x_{\text{edge}}, x_{\text{cloud}}\}$ as a function of $\hat{\sigma}$ based on $f(\hat{x}) + \sum_{i=1}^{2} \sigma_i \nabla h_i(\hat{X}) = 0$.
Denote $x_i = l_i(\sigma_1, \sigma_2)$, $i \in \{1, 2, ..., N, \text{off}, \text{edge}, \text{cloud}\}$ as shown in (29).
for each $i \in \{1, 2, ..., N\}$ do
    if $\{x_i(\sigma_1) < 0\}$ then
        $x_i = 0$.
    else $\{x_i(\sigma_1) > \min\{\lambda_i, \mu_i - \varsigma\}\}$
        $x_i = \min\{\lambda_i, \mu_i - \varsigma\}$.
    end if
end for
if $x_{\text{off}}(\sigma_1, \sigma_2) < 0$ then
    $x_{\text{off}} = 0$.
else $\{x_{\text{off}} > \frac{\mu_{\text{a}} - \varsigma}{c}\}$
    $x_{\text{off}} = \frac{\mu_{\text{a}} - \varsigma}{c}$.
end if
if $x_{\text{edge}}(\sigma_2) < 0$ then
    $x_{\text{edge}} = 0$.
else $\{x_{\text{edge}} > \mu_{\text{e}} - \varsigma\}$
    $x_{\text{edge}} = \mu_{\text{e}} - \varsigma$.
end if
if $x_{\text{cloud}}(\sigma_2) < 0$ then
    $x_{\text{cloud}} = 0$.
else $\{x_{\text{cloud}} > k\mu_{\text{ins}} - \varsigma\}$
    $x_{\text{cloud}} = k\mu_{\text{ins}} - \varsigma$.
end if
Represent $\sigma_2$ with a function of $\sigma_1$ as in (32).
Compute $\sigma_1$ that satisfies $h_2(\sigma_1) = 0$ with the bisection search method.
Determine the value of $\hat{x}(\sigma_1)$ according to (31) and (32).
```

C. Optimality and Complexity Analysis

Theorem 3. *The results derived from Algorithm 1 are optimal solutions to problem (21).*

Proof Sketch: Problem (21) can be considered as an optimization problem with the objective $\min f(\hat{x}) = C(\hat{x}) + VD(\hat{x})$ contrained in a close space. The close space is bounded by the inequations constraints $g_i(\hat{x}) \leq 0$ $(i = 1, 2, ..., 2N + 6)$. The KKT conditions in (27) gives solutions to the optimization problem which simultaneously take into account the extreme points ($\nabla f(\hat{x}) + \sum_{i=1}^{2} \sigma_i \nabla h_i(\hat{x}) = 0$) and the points on the boundaries ($g_i(\hat{x}) = 0$ $(i = 1, 2, ..., 2N+6)$). Directly searching for the optimal solutions based on the KKT conditions is proved to be exponentially complex. To simplify the search, the extreme points are obtained first by solving the gradient constraint (28). Since problem (21) is a convex optimization problem, if the extreme points are within the close space bounded by the inequations $g_i(\hat{x}) \leq 0$ $(i = 1, 2, ..., 2N + 6)$, then the extreme points are the optimal solutions to problem (21). If one of the inequation constraints is not satisfied by the solutions of (28), which means that the extreme points are not within the close space, then the optimal solutions must be on the boundary. This can be explained by a simple example: if $x_1 \geq 0$ is not satisfied by the solutions of (28), then on the extreme point, there is $x_1 < 0$. Since problem (21) is a convex optimization problem over $\hat{x}$, the objective function is monotonically increasing with x_1 when $x_1 \geq 0$. The objective function is thus minimized when $x_1 = 0$. Therefore, the results derived from Algorithm 1 are equivalent to the solutions of the KKT conditions in (27) and Algorithm 1 provides optimal solutions to problem (21).

Remark 1. *According to the description in Algorithm 1, for each given Lagrange multiplier σ_1, the value of $\hat{x}$ is determined. Traversal over $\hat{x}$ is conducted to ensure that x_i is linearly constrained in $[0, u_i]$, $i \in \{1, .., N, \mathrm{off}, \mathrm{edge}, \mathrm{cloud}\}$, which, as a result, generates a complexity of $(2N+6)$. Since the optimal σ_1 is searched with the bisection search method, the times of searching equal $\left\lceil \log_2 \frac{r-l}{\varphi} \right\rceil$, where l and r denote the left and right bounds of σ_1, respectively, and φ denotes the searching precision. Therefore, the computing complexity of Algorithm 1 is $2(N+3)\left\lceil \log_2 \frac{r-l}{\varphi} \right\rceil$, i.e., $O(N)$.*

According to the above analysis, the optimal solution $\hat{x}^*$ which minimizes $C(\hat{x}) + VD(\hat{x})$ when given the number of cloud instances k is obtained based on Algorithm 1 (i.e., problem (21)). To solve the original optimization problem (20), the optimal usage of cloud instances k remains to be determined. This is solved in Algorithm 2.

Algorithm 2 Determining Optimal Usage of Cloud Resources

```
for each given $k \in \{0, 1, 2, ..., M\}$ do
    Compute the optimal $\hat{x}^*(k)$, which minimizes $C(\hat{x}) + VD(\hat{x})$, based on Algorithm 1.
end for
Find the optimal $k^*$ that satisfies $k^* = \arg\min_{k \in [0,M]} \{C(\hat{x}^*(k)) + VD(\hat{x}^*(k))\}$.
```

Notice that the system cost $C(k)$ increases linearly with the number of cloud instances k and the system delay $D(k)$ decreases with k. There are two probabilities about the trend of $C(k) + VD(k)$ with k. First, the computing capacity of the mobile edge is the bottleneck of improving the system delay. In this case, the mobile users are incapable to execute computing tasks on mobile devices and the access network provides sufficient communication resources. Then, increasing cloud instances can effectively reduce system delay. With the increasing usage of cloud resources, the computing capacity is no longer the bottleneck. Then the system delay decreases slowly with k. In this case, $C(k) + VD(k)$ decreases first and then increases with k. Second, the computing capacity of the mobile edge is not the bottleneck. Then increasing k will not reduce the system delay significantly. In this case, $C(k)+VD(k)$ purely increases with k and is minimized when $k = 0$.

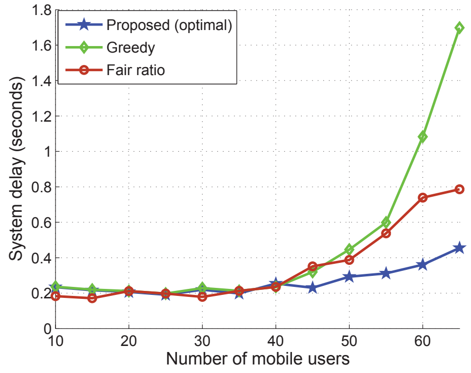

Fig. 3. System delay of different algorithms.

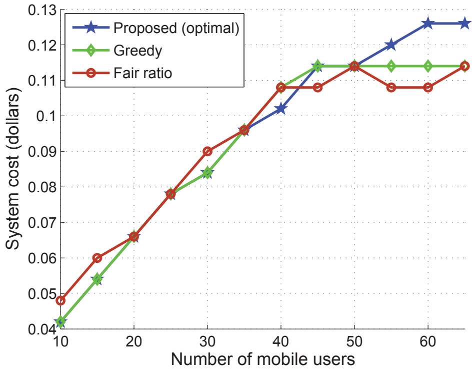

Fig. 4. System cost of different algorithms.

V. Simulation and Analysis

In this section, extensive simulations are conducted to evaluate the proposed algorithm. A scenario is considered in which a small cell base station has a coverage of 50 m [14]. The base station has mobile edge hosts with a constant computing capacity of $\mu_1 = 7.2$ GHz and a wireless access network with the transmission rate of 120 Mbps. Cloud resources are leased to augment the system computing capacity. Amazon T2.nano is the representative of cloud instances [13], each of which has a computing capacity of 2.4 GHz and is priced at 0.006 dollar/h. There are N mobile users, the computing capacities of which are heterogeneous.

The performance of the proposed algorithm is compared with two baseline algorithms: the greedy algorithm and the fair ratio algorithm. In the greedy algorithm, mobile users with the largest utilization (represented by $\frac{\lambda_i}{\mu_i}$) are endowed the highest priority to offload the mobile requests. In the fair ratio algorithm, all mobile users are treated equally and offload a fair ratio of the mobile requests. The complexity of the fair ratio algorithm is $O(1)$. Since all mobile users are sorted in the greedy algorithm, the complexity is $O(N \log N)$.

A. *Evaluation of the Proposed Algorithm*

In this section, the performance of the proposed algorithm is evaluated by comparing with the greedy algorithm and the fair ratio algorithm. A set of $N \in [10, 65]$ mobile users are considered, which have heterogeneous mobile requests. The arrival rates of the mobile computation requests are uniformly distributed in $[0.3, 1.3]$ Giga CPU cycles per second. The computing capacities of mobile devices are heterogeneous among $[0.4, 1.3]$ GHz. Augmented reality is selected as the representative of delay-sensitive and computation-intensive mobile applications. As demonstrated in [1], an augmented reality application can be considered as continuous objective recognition tasks. Take face recognition as an example, of which the computation requests are 1000 Mega CPU cycles [24]. When offloading the face recognition task through the wireless access network, the transmitted data (i.e., a photo) is 3000 KB. Thus the ratio of communication over computing requests c is 3. In practical applications, the transmitted data can also be face features extracted from a photo. Then the size of transmitted data depends on the feature extraction techniques in different face recognition applications. The influence of c is illustrated in the latter part.

The three algorithms are evaluted in terms of the system delay and cost, and the results are shown in Fig. 3 and Fig. 4. Combining the results in the two figures, it can be concluded that with the increasing number of mobile users, the proposed algorithm can significantly reduce the system delay over the other two algorithms while maintaining the system cost at an acceptable level. In details, when the number of mobile users $N \leq 40$, there is no apparent difference in system delay and cost among the three algorithms. When N is small, the access network can provide relatively sufficient communication resources to mobile users, and offloading mobile tasks is beneficial to reduce the system delay. As a result, most of the computing tasks are offloaded and the three algorithms generate similar results. As N increases (larger than 40), the access network cannot accommodate the increasing communication requests, and more mobile requests are scheduled to be executed on mobile devices. In the fair ratio algorithm, all mobile users are treated equally, without taking into account the heterogeneous mobile computing capacities, which results in underutilization of mobile resources. In the greedy algorithm, mobile tasks can not be partially offloaded. As a result, the system computation resources are underutilized. In the proposed algorithm, the mobile users with higher computing capacities are more probable to be allocated with more mobile requests, which helps to achieve load balance among the heterogenous mobile users. When $N = 50$, the proposed algorithm can reduce the system delay by 33% and 46% over the other two algorithms, respectively, while incurring the same system cost.

B. Tradeoff between System Delay and Cost

System delay and cost are two critical considerations of mobile edge operators. In this section, the tradeoff between system delay and cost is evaluated by changing the weight constant V. The results are shown in Fig. 5 and Fig. 6.

Fig. 5 illustrates the system cost of the algorithms with different weight constant V. In the proposed algorithm, the system cost increases with V. When V increases, decreasing the system delay become the main goal of optimization. More cloud instances are therefore tenanted, which incurs increasing system cost. By comparing the results of the three algorithms when $V = 0.05$ (reducing the system cost is the main objective), it can be concluded that the proposed algorithm outperforms the other two algorithms in reducing system cost.

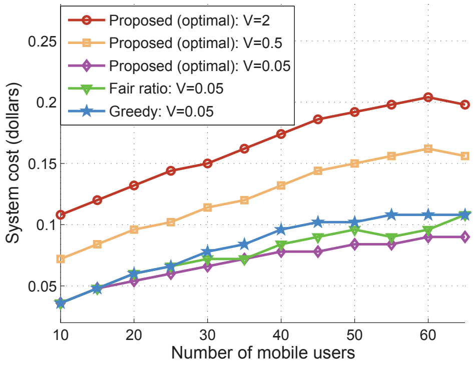

Fig. 5. System cost over weight constant.

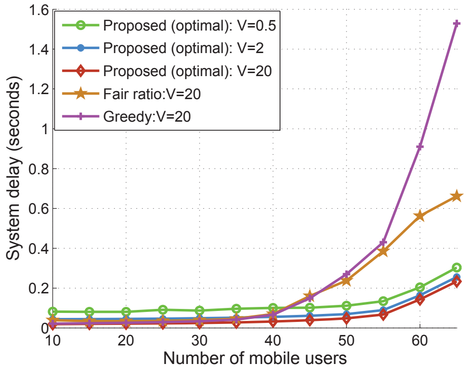

Fig. 6. System delay over weight constant.

Fig. 6 illustrates the system delay of the algorithms with different weight constant V. In the proposed algorithm, the system decreases with V. When V increases, reducing the system delay becomes the main purpose of optimization. Comparing the three algorithms when $V = 20$, it can be concluded that the proposed algorithm outperforms the other two algorithms in reducing the system delay.

C. Heterogeneity of Mobile Users

When making computation offloading decisions for mobile users, how to deal with the heterogeneity of mobile users is a critical challenge. In the fair ratio algorithm, all mobile users are treated equally without consideration of the heterogeneity in mobile computing capabilities. In this section, the proposed algorithm is evaluated with respect to the heterogeneity of mobile users. There are $N = 60$ mobile users, all of which have the same arrival rate of 1000 Mega CPU cycles. The serving rates of mobile devices follow the normal distribution

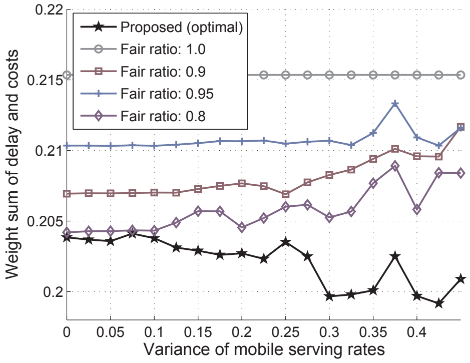

Fig. 7. System performance with heterogeneous mobile users.

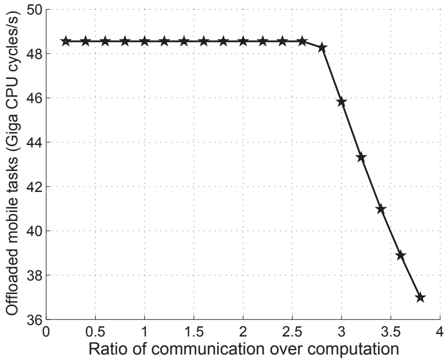

Fig. 8. Offloaded workloads with c.

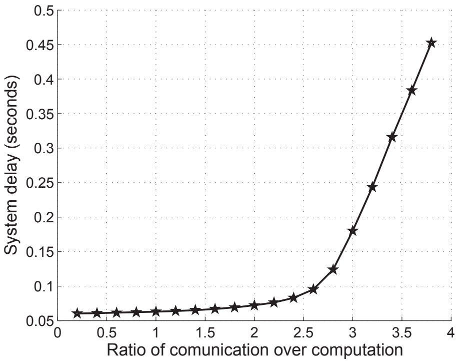

Fig. 9. System delay with c.

with the average of 1.2 GHz, and the variance σ represents the heterogeneity of mobile users. Fig. 7 shows the results of the proposed algorithm and the fair ratio algorithm with different offloading ratios.

As shown in Fig. 7, the weighted sum of delay and cost increases with σ in the fair ratio algorithm. This is because according to the fair ratio algorithm, all mobile users are treated equally. Mobile users with larger mobile computing capabilities can not fully utilize the computing capacities of mobile devices. The proposed algorithm in this work allocates workloads to mobile users according to the computing capacities of mobile devices. Thus the system delay and cost can be significantly decreased compared with the fair ratio algorithm, especially when the heterogeneity is large. Therefore, the proposed algorithm is ideal to deal with the heterogeneous mobile users.

D. Tradeoff between Communication and Computation Delay

The performance of the wireless access network is a critical factor that affects the system delay due to the constrained communication resources. Excessive communication requirements in the access network can generate huge transmission delay, which results in degradation of the system delay. How to implement the computation offloading decisions for mobile users, which deal with the tradeoff between computation delay and communication overhead, is an important challenge. In this section, the proposed algorithm is evaluated with respect to this tradeoff.

As demonstrated in Part. A, there are different values of c in different applications. Here, c denotes the ratio of communication requirements over computation requirements. The offloaded computation workloads from mobile users are influenced by c due to the tradeoff between computation delay and communication overhead, and therefore the system delay is influenced. In this simulation, the computation requirements remain unchanged, and the communication requirements change with varying c. The overall offloaded computation workloads and the system delay are selected as the metrics to evaluate the tradeoff. The results are shown in Fig. 8 and Fig. 9.

As shown in Fig. 8, when $c \leq 2.7$, the offloaded workloads remain unchanged as c increases. This is because when c is small, the wireless access network can provide sufficient communication resources for the offloaded computing tasks, and the computation delay is the key component of the system delay. Thus the system delay does not change apparently with c, as shown in Fig. 9. When the communication ratio is large ($c > 2.7$), the offloaded workloads decrease sharply with c. In this situation, excessive communication requirements arrive at the access network. The transmission overhead becomes the key component of the system delay. More computation requirements are allocated to the mobile devices to reduce the transmission overhead, which, however, incurs the increasing computation delay. Therefore, the system delay increases with c when c is large.

VI. Related Work

Mobile edge computing (MEC) is a promising computing paradigm with reduced latency. Many efforts have been devoted to exploiting the potential of MEC. A hierarchical edge cloud architecture has been designed in [7] to handle the peak workloads from mobile users. Cloud-fog interoperation has been achieved in [15] based on the proposed SDN enabled framework, to improve quality of service and optimize usage of network resource. An auction based resource allocation mechanism has been designed in [16] to share cloudlets in

MEC. In this work, we propose the cloud assisted mobile edge computing (CAME) framework to enhance the computing capacity of mobile edge.

Workload scheduling in the CAME framework consists of computation offloading of mobile tasks and load balancing between mobile edge and cloud. Many works have devoted efforts to computation offloading decision problems from the perspective of a single mobile user [17]-[22]. Huang *et al.* in [17] have proposed a dynamic computation offloading mechanism based on Lyapunov Optimization Theorem. Wen *et al.* in [20] have sought to achieve energy efficiency by optimizing the clock frequency and scheduling data transmission. Xian *et al.* in [22] have proposed an energy efficient computation offloading algorithm by optimally configuring the timeout. Wolski *et al.* in [21] have designed a framework to make computation offloading decisions by predicting bandwidth data.

Lots of research interests have been devoted to computation offloading problems of multiple mobile users [23]-[28]. In [23], computation and communication resources have been jointly allocated among multiple mobile users with delay constraints to minimize energy consumption. Chen *et al.* have solved the computation offloading problem of multiple users based on Game Theory in [24], [27]. In [24], they have formulated the computation offloading problem of multiple users via a single wireless access point as a computation offloading game. Furthermore, in [27], they have explored the computation offloading problem of multiple users in the multi-channel wireless environment. Xiao *et al.* in [28] have focused on the computation offloading problem of multiple mobile users via multiple wireless WiFi access points. Yang *et al.* in [26] have proposed a framework which enables data stream applications to be partitioned and executed in mobile cloud computing and supports multiple mobile users to share computation instances.

All the above studies have focused on the computation offloading problems from the perspective of mobile users. We seek to achieve a coordination of computation offloading decisions of mobile users to fully utilize the system resources. Based on the frameworks that support task partitioning, e.g., [26], we propose a workload scheduling algorithm which jointly schedules computation and communication workloads to minimize system delay.

Hybrid cloud is a promising solution that augments computing capacity of a private cloud and enables elastic services. Load balance between internal infrastructure and the public cloud is essential to reduce outsourcing cost while ensuring system performance. In [29], Zhang *et al.* have designed a static intelligent workload factoring service to enforce high utilization of cloud resources and reduced outsourcing cost, which neglects the dynamics of workloads. An online algorithm has been presented in [30] to schedule Mapreduce workloads in the hybrid cloud to achieve a minimum outsourcing cost within delay constraints. Niu *et al.* in [31] have dealt with the burst and fluctuating web services which are delay-sensitive by a hybrid cloud solution. They have designed a cost-efficient online algorithm for workload scheduling by leveraging Lyapunov optimization theorem to achieve arbitrarily close to the optimal respond time. AWS Auto Scaling has been adopted to determine the amount of cloud instances, which is coarse-grained. In this work, a fine-grained workload scheduling algorithm is designed to optimize the usage of cloud resources, and a load balance between mobile edge and the cloud is achieved.

VII. Conclusion

In this paper, we have proposed the CAME framework to provision elastic services in close proximity to mobile users. Workload scheduling problem in the CAME framework has been devised to minimize the system delay and cost by applying queueing network and convex optimization theories. Specifically, a practical workload scheduling algorithm has been proposed by exploiting the linear property of inequation constraints in the problem, which can obtain the optimal solutions with linear complexity. Simulation results have demonstrated that the proposed algorithm outperforms the fair ratio algorithm and the greedy algorithm in reducing the system delay while maintaining the system cost at an acceptable level. Extensive simulations have been further conducted to evaluate the proposed algorithm in dealing with the heterogeneous mobile users, as well as the tradeoff between computation delay and transmission overhead.

For the future work, the internal cost of the mobile edge will be taken into account in the optimal offloading design. Optimal computing capacity of the mobile edge will be further explored, and the resource provisioning problem will be analyzed.

VIII. Acknowledgements

This work is funded by the National Natural Science Foundation of China (No. 61472199).

Appendix A
Proof of Theorem 1

To prove problem (21) is a convex optimization problem, it should be proved that the objective function $f(\hat{x})$ in (21) is a convex function over $\hat{x}$.

Substitute $D(\hat{x})$ in (15) and $C(\hat{x})$ in (18) into problem (21), then $f(\hat{x})$ can be given by

$$\begin{aligned} f(\hat{x}) =& \frac{V}{\lambda}\Big(\sum_{i=1}^{N}\frac{x_i}{\mu_i - x_i} + \frac{x_{\text{off}}}{\mu_{\text{a}} - c x_{\text{off}}} + \frac{x_{\text{edge}}}{\mu_{\text{e}} - x_{\text{edge}}} + \\ & \frac{x_{\text{cloud}}}{k\mu_{\text{ins}} - x_{\text{cloud}}} + x_{\text{cloud}} d_{\text{t}}\Big) + c_{\text{p}}k + c_{\text{m}}. \end{aligned} \tag{34}$$

Based on (34), the Hessian matrix of $f(\hat{x})$ can be represented as

$$H(f) = [h_{ij}]_{(N+3)\times(N+3)}, \tag{35}$$

where $h_{ij} = 0$ when $i \neq j$, and

$$h_{ii} = \frac{2V\mu_i}{\lambda(\mu_i - x_i)^3} \quad i = 1, 2, ..., N, \tag{36}$$

$$h_{ii} = \frac{2Vc\mu_{\rm a}}{\lambda(\mu_{\rm a} - cx_{\rm off})^3} \quad i = N+1, \tag{37}$$

$$h_{ii} = \frac{2V\mu_{\rm e}}{\lambda(\mu_{\rm e} - x_{\rm edge})^3} \quad i = N+2, \tag{38}$$

$$h_{ii} = \frac{2Vk\mu_{\rm ins}}{\lambda(k\mu_{\rm ins} - x_{\rm cloud})^3} \quad i = N+3. \tag{39}$$

According to the constraints in problem (21), for any $i \in \{1, 2, ..., N+3\}$, $h_{ii} > 0$. Therefore, $H(f)$ is a positive semi-definite matrix and $f(\hat{x})$ is a convex function over $\hat{x}$ [11]. Furthermore, the inequation constraints in problem (21) are convex functions over $\hat{x}$ and the equation constraints are affine functions over $\hat{x}$. According to the definition of convex optimization problem in [11], the convexity of problem (21) can be proved.

Appendix B
Proof of Theorem 2

According to $l_i(\sigma_1)$ in (29), $l_i(\sigma_1)$ is monotonically increasing with σ_1. Since $x_i(\sigma_1)$ $(i = 1, 2, ..., N)$ are piecewise functions over σ_1 as presented in (31), $x_i(\sigma_1)$ $(i = 1, 2, ..., N)$ are monotonically increasing with σ_1. In addition, $x_{\rm off}$ decreases with σ_1 based on 4. As presented in (32),

$$\sigma_2 = \frac{V\mu_{\rm a}}{\sum\limits_{i=1}^{N} \lambda_i(\mu_{\rm a} - cx_{\rm off})^2} - \sigma_1,$$

since $x_{\rm off}$ decreases with σ_1 and $\mu_{\rm a} > cx_{\rm off}$, σ_2 decreases with σ_1.

According to (29) and (31), $x_{\rm edge}$ and $x_{\rm cloud}$ decreases with σ_2. As a result, $x_{\rm edge}$ and $x_{\rm cloud}$ increases with σ_1.

Therefore, $h_2(\sigma_1)$ is monotonically increasing with σ_1.

References

[1] T. Chen, L.Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *ACM SenSys'15*, Seoul, South Korea, Nov. 2015, pp. 155-168.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *ACM MobiSys'14*, Bretton Woods, New Hampshire, USA, June 2014, pp. 68-81.

[3] M. Satyanarayanan, P.Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14-23, Oct. 2009.

[4] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan, "The case for mobile edge-clouds," in *IEEE UIC/ATC'13*, Vietri sul Mare, Italy, Dec. 2013, pp. 209-215.

[5] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *ACM MobiSys'13*, Taipei, Taiwan, June 2013, pp. 153-166.

[6] B. Liang, "Mobile edge computing," in *Key Technologies for 5G Wireless Systems*, V. W. S. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, Eds., Cambridge University Press, 2017.

[7] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM'16*, San Francisco, CA, USA, July 2016, pp. 1-9.

[8] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet comput.*, vol. 13, no. 5, Sep. 2009.

[9] R. V. Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *IEEE CLOUD'10*, Miami, Florida, USA, July 2010, pp. 228-235.

[10] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *IEEE ISCC'12*, Cappadocia, Turkey, July 2012, pp. 59-66.

[11] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[12] J. Wu, S. Zhou, and Z. Niu, "Traffic-aware base station sleeping control and power matching for energy-delay tradeoffs in green cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 8, pp. 4196-4209, July 2013.

[13] "Amazon EC2," Online available: https://aws.amazon.com/cn/ec2/.

[14] T. Q. S. Quek, G. Roche, İ. Güvenç, and M. Kountouris, *Small cell networks: Deployment, PHY techniques, and resource management*. Cambridge University Press, 2013.

[15] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. Shen, "Catalyzing cloud-fog interoperation in 5G wireless networks: An sdn approach," *arXiv:1612.05291*, 2016.

[16] A. L. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Trans. Emerg. Topic Comput.*, vol. PP, no. 99, Oct. 2015.

[17] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991-1995, Apr. 2012.

[18] K. Kumar, J. Liu, Y. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129-140, 2013.

[19] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 1, pp. 19-26, Jan. 1998.

[20] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *IEEE INFOCOM'12*, Orlando, Florida, USA, May 2012, pp. 2716-2720.

[21] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *IEEE IPDPS'08*, Miami, Florida, USA, Apr. 2008, pp. 1-8.

[22] C. Xian, Y. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *IEEE ICPADS'07*, Hsinchu, Taiwan, Dec. 2007, pp. 1-8.

[23] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *IEEE SPAWC'13*, Darmstadt, Germany, June 2013, pp. 26-30.

[24] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974-983, Apr. 2015.

[25] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *IEEE CLOUD'13*, Santa Clara Marriott, CA, USA, June 2013, pp. 75-82.

[26] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23-32, Mar. 2013.

[27] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, Oct. 2016.

[28] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *ACM MSWiM'15*, Cancun, Mexico, Nov. 2015, pp. 271-278.

[29] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent workload factoring for a hybrid cloud computing model," in *IEEE World Conference on Services-I*, Los Angeles, CA, USA, July 2009, pp. 701-708.

[30] X. Qiu, W. L. Yeow, C. Wu, and F. C. Lau, "Cost-minimizing preemptive scheduling of mapreduce workloads on hybrid clouds," in *IEEE/ACM IWQoS'13*, Montreal, Canada, June 2013, pp. 1-6.

[31] Y. Niu, B. Luo, F. Liu, J. Liu, and B. Li, "When hybrid cloud meets flash crowd: Towards cost-effective service provisioning," in *IEEE INFOCOM'15*, Hong Kong, China, Apr. 2015, pp. 1044-1052.