This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

Multi-user Multi-task Computation Offloading in Green Mobile Edge Cloud Computing

Weiwei Chen, *Member IEEE,* Dong Wang and Keqin Li, *Fellow IEEE,*

Abstract—Mobile Edge Cloud Computing (MECC) has becoming an attractive solution for augmenting the computing and storage capacity of Mobile Devices (MDs) by exploiting the available resources at the network edge. In this work, we consider computation offloading at the mobile edge cloud that is composed of a set of Wireless Devices (WDs), and each WD has an energy harvesting equipment to collect renewable energy from the environment. Moreover, multiple MDs intend to offload their tasks to the mobile edge cloud simultaneously. We first formulate the multi-user multi-task computation offloading problem for green MECC, and use Lyaponuv Optimization Approach to determine the energy harvesting policy: how much energy to be harvested at each WD; and the task offloading schedule: the set of computation offloading requests to be admitted into the mobile edge cloud, the set of WDs assigned to each admitted offloading request, and how much workload to be processed at the assigned WDs. We then prove that the task offloading scheduling problem is NP hard, and introduce centralized and distributed Greedy Maximal Scheduling algorithms to resolve the problem efficiently. Performance bounds of the proposed schemes are also discussed. Extensive evaluations are conducted to test the performance of the proposed algorithms.

Index Terms—Mobile Edge Computing, Computation Offloading, Energy Harvesting, Multi-user Mult-task Scheduling

1 Introduction

Nowadays, Mobile Devices (MDs) such as smartphones, tablet computers, wearable devices and etc., are playing more and more important role in our daily life. Moreover, mobile applications running on MDs, such as immersive gaming, human-computer interaction, often demand stringent delay and processing requirements. Consequently, despite the tremendous improvement of MDs' processing capacity, executing applications solely on a MD is still incapable of providing satisfactory Quality of Experience (QoE) to users. Besides, MDs are energy constrained, which further increases the tension between resource limited MDs and computation intensive mobile applications.

Fortunately, with the advanced networking and multi-task technology, offloading part of the workload of an application (or simply put it as a task) from MDs to a remote cloud or a nearby mobile edge cloud [1] (also known as cloudlet) provides a promising alternative to reduce the task execution time as well as prolong the lifetime of MDs. A mobile edge cloud consists of either one edge server or a set of devices that serve mobile users cooperatively. Though the mobile edge cloud is less powerful compared with a remote cloud, as it is located at the edge of the network the transmission latency between a MD and the mobile edge cloud is much lower than that of the remote cloud. Besides, a mobile edge cloud can explore the idle computing and storage resources at the network edge efficiently. In this work, we mainly focus on task offloading in a mobile edge cloud composed of multiple devices. Moreover, we focus on discuss offloading computation intensive and delay sensitive tasks to a mobile edge cloud, in which the amount of energy consumed for communication (i.e., transmitting a task and its result between a mobile device and the mobile edge cloud) is way much lower than the energy required for computing the task in a limited time duration.

Extensive research attention has been drawn to resource allocation for task offloading in both academia and industry [2]. In single user task offloading scenario, the CPU frequency at the local MDs or devices in the mobile edge cloud, and the amount of workload to be offloaded to each device in the mobile edge cloud are investigated in [3], [4], [5], [6], [7], [8], [9], [10]. In regards to multi-user multi-task computation offloading, centralized algorithms are proposed in [11], [12], [13], [14], [15], [16], [17], [18] to balance workload from different mobile edge clouds or virtual machines in the same cloud server. However, to further enhance the agility of the mobile edge cloud, especially when the system consists of a set of distributed Wireless Devices (WDs), online distributed algorithms is more preferred.

Furthermore, though task offloading can save energy for mobile devices significantly, the mobile edge cloud is still challenged with high energy consumption and carbon footprint. As a result, many mobile edge clouds, as has been discussed in [2] and [19] use renewable energy to realize green computing. Likewise, when mobile edge cloud is composed of a set of WDs, renewable energy opens up the possibility of convenient and perpetual operation at wireless devices even without connecting them to the power grid. Renewable energy can be generated in various sources, such as solar radiation, wind and in-door light. Though renewable energy is environmental friendly, the amount of harvestable energy is time varying. Consequently, the

- *Weiwei Chen is with the College of Computer Science and Electronic Engineering, Hu Nan University, China. E-mail: avachen@hnu.edu.cn*
- *Dong Wang is with the College of Computer Science and Electronic Engineering, Hu Nan University, China.*
- *Keqin Li is with the Department of Computer Science, the State University of New York.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

energy harvesting policy and the task offloading policy together will determine the efficiency of a green mobile edge cloud, i.e., the amount of workload to be completed, the amount of energy saved by the MDs, and etc.

Despite the efforts made in the previous works, in regards to a green mobile edge cloud that composed of a set of wireless devices, since the workload from a mobile device can be computed by multiple wireless devices, how to assign workload from MDs to WDs with guaranteed system performance still remains to be an open issue. Moreover, to facilitate easy system implementation, how to make sure all the MDs and WDs adjust to both energy and task demand variations distributively and efficiently is even challenging. In this work, we strive to tackle these issues. In detail, the design goal is to exploit the computing capacity in green mobile edge cloud efficiently, e.g. finish as many offloading requests as possible, and make the best use of the harvestable energy at each WD in the mobile edge cloud. To this end, the energy harvesting policy and the offloading scheduling scheme are carefully planned so that the amount of energy harvested at the mobile edge cloud and the amount of energy consumed for task offloading are well balanced. In addition, to meet different implementation requirements, both energy harvesting and scheduling decisions can be determined either in a centralized or in a distributed way. The main contributions of this work are listed in the following.

1) We propose a multi-user multi-task computation offloading framework for a green mobile edge cloud computing system. In the proposed approach, the dynamics of energy arrivals at the mobile edge cloud as well as task arrivals at different MDs are jointly considered;
2) We introduce the concept of energy links (not real wireless links) and show that offloading a task from a MD to mobile edge cloud is equivalent to routing the harvested energy from mobile edge cloud to a MD. With energy links, the multi-user multi-task offloading problem is then cast into a Maximal Independent Set (MIS) problem; and
3) Since the problem is NP-hard, centralized and distributed greedy maximal scheduling algorithms and their performance bound are studied. Both theoretical analysis and simulation results indicate the proposed centralized and distributed greedy scheduling algorithms achieve similar performance. Moreover, the proposed scheduling algorithms provide 18.8% to 31.9% higher average system utility than a random scheduling scheme.

The rest of the paper is organized as follows. Prior works are reviewed in Sec. 2. The system model is discussed in Sec. 3. Sec. 4 formulates the multi-user multi-task computation offloading problem. Since the problem is NP-hard, a centralized and a distributed greedy maximal scheduling algorithms are presented. Sec. 5 analyzes the implementation complexity as well as the performance bound of the proposed scheduling algorithms. Simulation results are investigated in Sec. 6. In Sec. 7, a conclusion is drawn.

2 Related Works

In this section, we will give a brief overview about some related works in regards to resource allocation for task offloading in mobile edge cloud computing. Task offloading in mobile edge cloud computing can be supported by either power grid or renewable energy. When the mobile edge cloud has reliable energy supply from the power grid, integer programming is proposed in [5] to determine which part of the task is offloaded to the mobile edge cloud and which part should be executed at the MD for single task offloading scenario. The prior work [20] applies game theory to coordinate task offloading from various users with various task requests(task length, task response delay requirement, etc) to the mobile edge cloud. When the mobile edge cloud is consisted of a set of devices that will compute tasks simultaneously for a mobile device [21], and [7] propose to offload a task from a MD to multiple MDs through a single hop or multiple hops, so as to minimize the overall task response time. To reduce the transmission delay caused by multi-hop task dissemination, the prior work [6] offloads a task to multiple MDs that are directly connected to the task initiator. Semi-definite Cone Programming is discussed in [10] to facilitate task offloading from a MD to a mobile edge cloud that contains multiple WDs.

For multi-task offloading problem, the authors in [4], [14] introduce heuristic algorithms to assign sub-tasks to different cores or Virtual Machines (VMs) in a cloud server. Chen. et al. [16] propose a game theoretical approach to let multiple MDs to decide whether they should offload their tasks to the mobile edge cloud or not. As the computing capacity of the server in the mobile edge cloud is assumed to be extremely high, the bottleneck for offloading tasks from multiple users is the bandwidth the network can provide. Guo et al. [11] approximate the joint power allocation and sub-task assignment problem with a convex optimization approach. By assuming almost infinite computing capability at the mobile edge cloud server, it mainly concerns which subset of sub-tasks should be offloaded to the mobile edge cloud server, and their offloading sequence. However, these works cannot be applied to the case when the computing capacity of the servers in the mobile edge cloud are limited.

In the context of multiple users and multiple mobile edge clouds/servers/VMs in a cloud, centralized resource allocation strategies [15], [22] are developed to balance the workload of each mobile edge cloud/VM in a cloud. Prior work [12] assumes a set of MDs form a mobile edge cloud and help to execute tasks from each other so as to minimize the average energy consumption of all MDs. To encourage the cooperation among different MDs, incentive schemes are introduced to prevent one MD from overriding the energy, bandwidth or computing resources from other MDs. The ad hoc mobile edge cloud setting is also discussed in [13]. In Chen's work, graph theory is used to pair different D2D mobile devices so as to minimize the overall system energy consumption. Multi-users offloading tasks to multi-servers in the mobile edge cloud computing is discussed in [23]. Even though, in [12], [13], [23], one task is assumed to be offloaded to only one MD or server in the mobile edge cloud. This assumption, however, is not applicable to the case when a task is relatively large and has to be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

processed by multiple devices in the mobile edge cloud. Opportunistic offloading strategy is discussed in [3], a MD can dynamically determine whether it offloads its task to its nearby MDs via a 3G/4G, WIFI or D2D link based on the MD's location. Despite the efforts made in the previous works, in regards to a self-organized mobile edge cloud with a set of MDs, an efficient distributed resource allocation algorithm is more preferred to a centralized scheme so as to facilitate more agile task offloading in mobile edge cloud computing.

On the other hand, energy harvesting technology has been studied recently to reduce the carbon footprint. Moreover, if wireless devices are powered by renewable energy harvested from the environment, they can be operated perpetually even without energy supply from the power grid. When energy harvesting is blended with mobile edge cloud computing, it is expected to use the network edge computing resources more efficiently. You et al. [8] assume wireless energy transfer technology is used for a MD such as a wireless sensor or a wearable device. A binary offloading decision is made every time when a task arrives to the mobile device based on whether offloading the task to the mobile edge cloud will save its energy or not. Joint transmit energy beam-former at AP, CPU processing frequency as well as task offloading strategies for multi-users is presented in [24]. Moreover, in this work the mobile users are charged by its associated AP via Wireless Power Transfer (WPT). In [9], the authors assume a MD is able to harvest energy from the environment. Since the harvestable energy at the MD is highly dynamic, it then uses Lyapunov optimization approach to design the task offloading strategy so as to maximize the time average utility of the MD. The prior work [8] and [9] only consider resource allocation optimization for one MD. As to the multi-user case, to handle the uncertainty in the amount of harvestable energy, the authors in [19] propose a reinforcement learning based resource allocation algorithm to dynamically offload workload to different servers in the mobile edge cloud so that the long-term service delay and system operation cost can be minimized.

The approaches mentioned above are more suitable for a centralized system that can monitor all the nodes in the network and make system-wide decisions jointly. Beside, they only focus on task offloading from users to a single mobile edge cloud server. When a green mobile edge cloud is composed of a set of wireless devices, how to implement resource allocation strategies for multi-user multi-task offloading efficiently, especially when multiple devices can compute workload from a single user, still remains to be an open issue. In this work, we will design efficient scheduling algorithms to address this problem.

3 System Model

The mobile edge cloud consists of a set of WDs (denoted as W) and a set of MDs (denoted as M) where $|W|$ and $|M|$ is the cardinality of W and M respectively. The WDs, such as Wi-Fi routers, femto-cell base stations, printers etc., are equipped with energy harvesting devices so as to harvest energy from the environment and provide computation offloading services to the MDs. MDs, via buying the offloading services, can offload their tasks to the mobile edge cloud to extend their own computing ability and reduce their energy consumption. Moreover, a MD can only offload its task to WDs that are within direct communication range of itself. A WD, though can accept tasks from different MDs at different time, is only capable of computing workload from one MD at a time. In addition, different wireless links use orthogonal channels (such as in LTE [25], [26] or IEEE 802.11ax system [27]) for data transmission so that one wireless link will not experience interference from other wireless links. Fig. 1 illustrates a typical computation offloading scenario in which a set of wireless APs constitute a mobile edge cloud. In this figure, MD1, MD2 and MD3 can establish a direct communication link to AP1, AP2, AP3 and AP4. The workload from MD1, MD2 and MD3 are denoted as WK1, WK2 and WK3 respectively. Based on MDs' location as well as the amount of workload to be offloaded at each MD, we can schedule task offloading in the following way. AP1 and AP2 serve MD1, AP3 serves MD2, and AP4 serves MD3.

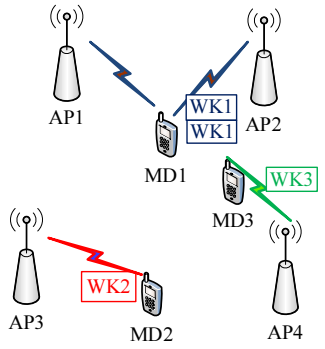

Fig. 1. An illustrative scenario for computation offloading in mobile edge cloud computing. In this figure, the mobile edge cloud is composed of a set of wireless APs.

Furthermore, time is divided into frames, where a frame contains a control sub-frame and a computation offloading sub-frame. In the control sub-frame, control information is exchanged among WDs and MDs to determine the offloading schedule. In the computation offloading sub-frame, workload is first transmitted to the WDs, and WDs return the result to the MDs after they finishing processing the workload. Here workload refers to the amount of sub-task from a MD.

A task to be offloaded to the mobile edge cloud from MD i can be represented as a tuple $A_i(t) = \{\rho_i(t), \beta_i(t), r_i(t), u_i(t)\}$, and $\rho_i(t), \beta_i(t), r_i(t), u_i(t)$ denotes the input data size, the ratio of the output data size to the input data size, the ratio of the number of CPU cycles to the input data size, and the revenue the mobile edge cloud can get if the task has been successfully completed by the mobile edge cloud. Moreover, $A_i(t) = \{0, 0, 0, 0\}$ corresponds to the case when no task offloading is requested by MD i at frame t. The tasks to be offloaded are fine grained and can be divided into arbitrary size. As a certain amount of communication overhead is always needed to facilitate task offloading between a MD and a WD, to guarantee the task offloading efficiency it is required that the amount of workload offloaded to a WD is no less than $\gamma \cdot \rho_i(t)$ if a task $A_i(t)$ is to be offloaded to the mobile edge cloud. Furthermore, a task has to be finished in a frame as long as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

it is admitted into the mobile edge cloud.

One WD can only compute workload from one MD at a time. However, one WD can serve different MDs at different time. On the other hand, multiple WDs can compute tasks for one MD cooperatively. Denote $s_{ij}(t)$ as an indicator. $s_{ij}(t) = 1$ if WD j serves MD i at frame t and $s_{ij}(t) = 0$ otherwise. Let M_j be the set of MDs connected directly to WD j. Hence

$$\sum_{i \in M_j} s_{ij}(t) \leq 1, \qquad \forall j \in W. \tag{1}$$

Let $\bar{s}_i(t)$ be a binary number, where $\bar{s}_i(t) = 1$ indicates that a task initiated by MD i at frame t is accepted by a set of WDs in the mobile edge cloud, and $\bar{s}_i(t) = 0$ implies that the task from MD i is not accepted. Denote W_i as the set of WDs connected directly to MD i, and $x_{ij}(t)$ as the amount of workload scheduled to WD j from MD i. To meet the task offloading feasibility constraint, it requires

$$\sum_{j \in W_i} x_{ij}(t) s_{ij}(t) = \rho_i(t) \bar{s}_i(t), \qquad \forall i \in M; \tag{2a}$$

$$x_{ij}(t) \geq \gamma s_{ij}(t) \rho_i(t), \qquad \forall i \in M, j \in W_i. \tag{2b}$$

Let the CPU frequency of WD j be f_j, and c_{ij} be the data rate between MD i and WD j. Due to the delay constraint, the total amount of time for offloading workload x_{ij}, i.e., the time used for transmitting the workload from MD i to WD j, computing the workload at WD j and returning the result to MD i, should be less or equal to the duration of the computation offloading sub-frame (denoted as $\pi_{t,o}$). Hence

$$\begin{aligned} & x_{ij}(t) \left(\frac{\beta_i(t)}{c_{ij}} + \frac{1}{c_{ij}} + \frac{r_i(t)}{f_j} \right) s_{ij}(t) \\ & = x_{ij}(t) w_{ij}(t) s_{ij}(t) \leq \pi_{t,o} \qquad \forall j \in W_i. \end{aligned} \tag{3}$$

Here $w_{ij}(t) = \frac{1}{c_{ij}} \beta_i(t) + \frac{1}{c_{ij}} + \frac{r_i(t)}{f_j}$. $w_{ij}(t)$ is the total amount of time WD j needs to finish 1 bit of workload initiated by MD i at frame t.

Similarly, WD j will consume energy for receiving and computing the workload, as well as returning its result to the task initiator. Let the transmitting and receiving power at WD j be pt_j and pr_j respectively. Denote the amount of energy consumed at WD j for MD i at frame t as $e_{ij}(t)$. $e_{ij}(t)$ can be expressed as

$$\begin{aligned} e_{ij}(t) = & \left(\frac{pt_j \beta_i(t)}{c_{ij}} + \frac{pr_j}{c_{ij}} + \kappa f_j^{\sigma} r_i(t) \right) x_{ij}(t) s_{ij}(t) \\ = & x_{ij}(t) z_{ij}(t) s_{ij}(t) \qquad \forall i \in M, j \in M_i. \end{aligned} \tag{4}$$

where κ is the effective switched capacitance, σ is a value close to 2 (we set it to 2 in this work), and $z_{ij}(t) = \frac{pt_j \beta_i(t)}{c_{ij}} + \frac{pr_j}{c_{ij}} + \kappa f_j^{\sigma} r_i(t)$. The overall amount of energy consumed by WD j, denoted as $e_j(t)$, can be expressed as

$$e_j(t) = \sum_{i \in M_j} e_{ij}(t), \qquad \forall j \in W. \tag{5}$$

In addition, the amount of harvestable energy in the environment arrives to a WD randomly. Let $\bar{h}_j(t)$ and $h_j(t)$ be the maximum amount of harvestable energy arrives to WD j and the actual amount of energy WD j harvests at frame t.

TABLE 1
Notations

M	The set of MDs (Mobile Devices).
N	The set of WDs (Wireless Devices).
M_j	The set of MDs connected with WD j.
W_i	The set of WDs connected with MD i.
$\rho_i(t)$	The input data size of the task from MD i at frame t.
$\beta_i(t)$	The ratio of the output to the input data size of the task from MD i at frame t.
$\pi_{t,o}$	The duration of a computation offloading sub-frame.
$r_i(t)$	The ratio of the CPU cycles to the input data size of the task from MD i at framt t.
$u_i(t)$	The revenue of successfully offloading the task from MD i.
$w_{ij}(t)$	The amount of time WD j needs to finish one bit of workload from MD i.
$z_{ij}(t)$	The total amount of energy WD j needs to process one bit of workload from MD i.
$\bar{s}_i(t)$	$\bar{s}_i(t) = 1$ indicates a task from MD i is accepted by a set of WDs in the system, otherwise $\bar{s}_i(t) = 0$.
s_{ij}	$s_{ij} = 1$ if WD j serves MD i at frame t, otherwise $s_{ij}(t) = 0$.
$x_{ij}(t)$	The amount of workload assigned to WD j from MD i at frame t.
$e_j(t)$	The amount of energy consumed by WD j at frame t .
$q_j(t)$	The energy queue for WD j at frame t .
$h_j(t)$	The amount of energy harvested by WD j at frame t.
$\bar{h}_j(t)$	The maximum amount of harvestable energy at WD j in frame t.
θ_j	Energy drift at WD j.
v_0	A weight, used to balance the energy queue length and the system utility (revenue).
μ_l	The task initiator of energy link l.
$R_l(t)$	The set of WDs assigned to MD μ_l in energy link l at frame t.
$b_l(t)$	The weight of energy link l at frame t.
$EL(t)$	The set of all the energy links at frame t.
$I_s(t)$	The maximum independent energy link set at frame t.
$LS(t)$	The set of all independent energy link sets at frame t.
d_{max}	The maximum conflict degree of all possible $EL(t)$
η	The number of iterations in the proposed DGMS based algorithm.
$U_{v_0}^{CGMS}$	The overall system utility of the CGMS-based scheme.
$U_{v_0}^{DGMS}$	The overall system utility of the DGMS-based scheme.

Obviously, $\bar{h}_j(t) \leq h_j(t)$. WD j maintains an energy queue for computation offloading. The energy harvested from the previous frame can be used in the current frame. Denote the energy queue for WD j at frame t as $q_j(t)$. $q_j(t)$ evolves as

$$q_j(t+1) = q_j(t) + h_j(t) - e_j(t), \qquad \forall j \in W. \tag{6}$$

Note that, as the energy consumption of MD j is no more than the available energy in its energy queue, it requires that

$$e_j(t) \leq q_j(t), \qquad \forall j \in W. \tag{7}$$

The notations used in this article are listed in Table-I.

4 The Proposed Centralized and Distributed Multi-user Multi-task Offloading Schemes

4.1 Formulation

In this work, we intend to maximize the overall revenue of the WDs. Hence, the multi-user multi-task offloading

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

problem can be formulated as:

$$\max \lim_{t_s\to\infty} \frac{1}{t_s}\sum_{t=1}^{t_s}\sum_{i\in M} u_i(t)\bar{s}_i(t) \tag{8a}$$

$$\text{s.t.}(1)-(7) \tag{8b}$$

Notice that, our approach can be easily extended to achieve other design goals, e.g., to minimize the energy consumption of all the MDs, etc.

According to (8), if we can exhaustively list all possible task arrivals of all the MDs and the energy arrival of all the WDs, we can design a specific multi-user multi-task offloading scheme for each combination of task and energy arrivals, the optimal system revenue (the optimal value of (8)) will be achieved. However, as the task and energy arrivals are time variant, the total number of task and energy arrivals for all of the users in the system grows exponentially with the network size. For instance, if there are y different levels of harvestable energy at each WD, ξ different types of tasks, m MDs and n WDs, the total number of combinations is $y^n(\xi+1)^m$.

To this end, we use the Lyapunov optimization approach proposed in [28] to resolve (8) without exhaustively listing all the combinations. In detail, let θ_j be a constant, and it is named as the energy drift of energy queue $q_j(t)$. Here

$$\begin{aligned}
&\theta_j = e_j^{max} + \phi(v_0);\\
&e_j^{max} = \max_{\{1\le t\le t_s, i\in W_j\}}\left\{\frac{z_{ij}\pi_{t,o}}{w_{ij}}\right\};\\
&h_j^{max} = \max_{\{1\le t\le t_s\}} \bar{h}_j(t);\\
&h^{max} = \max_{\{j\in W\}} \bar{h}_j^{max};\\
&z_{min} = \min_{\{i\in M, j\in W_i, 1\le t\le t_s\}} z_{ij}(t);\\
&z_{max} = \max_{\{i\in M, j\in W_i, 1\le t\le t_s\}} z_{ij}(t);\\
&\rho_{min} = \min_{\{1\le t\le t_s, i\in M\}} \rho_i(t); \text{and}\\
&\phi(v_0) = \frac{\rho_{min}(1-\gamma)h^{max}z_{max} + v_0 u_k}{\gamma\rho_{min}z_{min}}.
\end{aligned}$$

Notice that, e_j^{max} represents the maximum amount of energy wireless device j consumed for computing the workload from any of its neighboring mobile devices W_j. Hence it takes the maximum value of $\frac{z_{ij}\pi_{t,o}}{w_{ij}}$ among all mobile device $i \in W_j$. Moreover, the energy drift θ_j, a function of v_0, forces the energy queue to stay around θ_j. The higher the v_0, the more energy buffered at the energy queue, the more available energy to be used. Let $Q'(t) = \left\{q'_j(t) = q_j(t) - \theta_j \mid j \in W\right\}$ be the set of virtual queues for all the WDs in W. Define $L(t)$ and $\Delta(t)$ as

$$L(t) = \sum_{j\in W}(q'_j(t))^2; \tag{9a}$$

$$\Delta(t) = E\left\{L(t+1) - L(t) \mid Q'(t)\right\}; \tag{9b}$$

$$\delta = \frac{1}{2}\sum_{j\in W}(h_j^{max})^2 + \frac{1}{2}\sum_{j\in W}(e_j^{max})^2. \tag{9c}$$

We have,

$$\begin{aligned}
&\Delta(t) - v_0 E\left\{\sum_{i\in M} u_i(t)\bar{s}_i(t) \mid Q'(t)\right\}\\
&\le -\sum_{i\in M} v_0 E\left\{u_i(t)\bar{s}_i(t) \mid Q'(t)\right\}\\
&+ E\left\{\sum_{j\in W}\left(q_j(t)-\theta_j\right)\left(h_j(t) - \sum_{i\in M_j} e_{ij}(t)\right) \mid Q'(t)\right\} + \delta.
\end{aligned} \tag{10}$$

To maintain system stability, Lyaponuv optimization approach aims at finding an energy harvesting strategy $\{h_j(t)\}$ and a task offloading strategy $\{\bar{s}_i(t), s_{ij}(t), x_{ij}(t)\}$ so as to minimize the right-hand side of (10). Therefore, for the given virtual energy queues $Q'(t)$, $\{\bar{s}_i(t), s_{ij}(t), x_{ij}(t), h_j(t)\}$ can be determined by

$$\begin{aligned}
&\arg \inf_{(1)-(5),(7)} \left\{-\sum_{j\in M}(q_j(t)-\theta_j(t))\sum_{i\in M_j} x_{ij}(t)z_{ij}(t)\right.\\
&\left.+\sum_{j\in W}(q_j(t)-\theta_j)h_j(t) - \sum_{i\in M} v_0 u_i(t)\bar{s}_i(t)\right\}.
\end{aligned} \tag{11}$$

Equation (11) is then decoupled with the following two separate sub-problems.

1. ***Energy harvesting strategy***

$$h_j(t) = \begin{cases} h'_j(t), & q_j(t) < \theta_j(t);\\ 0, & \text{otherwise.}\end{cases} \tag{12}$$

2. ***Multi-task multi-user task offloading strategy***

$$\begin{aligned}
\{\bar{s}_i(t), s_{ij}(t), x_{ij}(t)\} = \arg \max_{(1)-(5),(7)} \sum_{i\in M}\Bigg(&v_0 u_i(t)\bar{s}_i(t)\\
&+\sum_{j\in W_i}(q_j(t)-\theta_j)z_{ij}(t)x_{ij}(t)\Bigg).
\end{aligned} \tag{13}$$

Before discussing how to solve (13), we fist introduce Lemma 1 which guarantees a WD can always provide enough energy for computing and transmitting the workload from a MD once the WD is recruited for task offloading.

Lemma 1. *$\forall i \in W_j$, if $e_j(t) < e_j^{max}, x_{ij}(t) = 0$, and WD j will not participate in task computing in frame t .*

Proof. The proof is provided in Appendix A. □

Recall that e_j^{max} is the maximum amount of energy WD j consumes for computing workload from a MD.

Lemma 1 implies that, for any mobile device i that is the neighborhood of wireless device j ($i \in W_j$), if its current available energy $e_j(t)$ is smaller than e_j^{max}, mobile device i cannot get any service from wireless device j ($x_{ij}(t) = 0$). With Lemma 1, constraint (7) can be safely removed from (13). Moreover, as will be shown in Appendix B, by removing (7), we can easily derive a lower bound of the scheduling schemes proposed in the following sub-sections.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

Lemma 2. *The feasibility of the given $\{\bar{s}_i(t) \mid i \in M\}$, and $\{s_{ij}(t) \mid i \in M, j \in W_i\}$ can be verified with the following set of equations.*

$$\sum_{j \in W_i} \frac{\pi_{t,o}}{w_{ij}(t)} s_{ij}(t) \geq \rho_i(t)\bar{s}_i(t); \tag{14a}$$

$$\sum_{i \in M_j} s_{ij}(t) \leq 1, \qquad \forall j \in W; \tag{14b}$$

$$\left(\sum_{j \in W_i} s_{ij}\right) \cdot \gamma \leq 1; \tag{14c}$$

$$q_j(t) \geq e_j^{max} s_{ij}(t), \qquad \forall j \in W. \tag{14d}$$

Moreover, if $\{\bar{s}_i(t) \mid i \in M\}$, and $\{s_{ij}(t) \mid i \in M, j \in W_i\}$ is feasible, the associated optimal workload assignment $x_{ij}(t)$ can be derived from (15).

$$x_{ij}^*(t) = \begin{cases} \frac{\pi_{t,o}}{w_{ij}(t)}, & i \in T'_{i,1,k^*}; \\ y_{n^*,k^*}(t) - \sum\limits_{j \in T'_{i,1,k^*}(t)} \frac{\pi_{t,o}}{w_{ij}(t)} & i = \tau_{k^*+1}; \\ \rho_i(t)\gamma, & i \in T_{i,k^*+2,n^*+1}; \\ 0, & \textit{otherwise.} \end{cases} \tag{15}$$

where $\tau_j(t)$ is the WD whose $(q_j(t) - \theta_j)z_{ij}(t)$ is the jth largest, and

$$n^* = \arg\left\{ n \mid \sum_{j \in T'_{i,1,n}} \frac{1}{w_{ij}(t)} < \frac{\rho_i(t)}{\pi_{t,o}} \leq \sum_{j \in T'_{i,1,n+1}} \frac{1}{w_{ij}(t)} \right\};$$

$$y_{n^*,k}(t) = \rho_i(t)\Big(1 - (n^* - k^*)\gamma\Big);$$

$$k^* = \arg\left\{ k \middle| \sum_{j \in T'_{i,1,k}} \frac{\pi_{t,o}}{w_{ij}(t)} < y_{n^*,k}(t), \sum_{j \in T'_{i,1,k+1}} \frac{\pi_{t,o}}{w_{ij}(t)} \geq y_{n^*,k-1}(t) \right\};$$

$$T'_{i,k,n} = \{\tau_k(t), \cdots, \tau_n(t)\}, \ \ 1 \leq k \leq n \leq |W_i(t)|.$$

Proof. Equations (14a)-(14b) guarantee that there is enough WDs to process the workload from MD i, and each WD only process workload from one MD. Moreover,(14c) implies a WD will either compute no less than $\gamma\rho_i(t)$ amount of workload for MD i, or not compute any workload from it. Equation (14d) makes sure the MDs that will participate into task offloading have more than e_j^{max} amount of energy. Hence if a schedule satisfies (14a)-(14d), it is a feasible schedule. As to (15), it is a straight derivation of (13) when $\{\bar{s}_i(t) \mid i \in M\}$ and $\{s_{ij}(t) \mid i \in M, j \in W_i\}$ are given. □

Lemma 2 indicates that MDs with higher virtual queue length $(q_j(t) - \theta_j(t))$ have higher chance to be assigned with more workload.

We further associate a weight $b\Big(\bar{s}_i(t), \{s_{ij}(t)\}\Big)$ to the given $\bar{s}_i(t)$, and $\{s_{ij}(t)\}$. $b\left(\bar{s}_i(t), \{s_{ij}(t)\}\right)$ is expressed as

$$b\Big(\bar{s}_i(t), \{s_{ij}(t)\}\Big) = v_0 u_i(t) + \sum_{j \in W_i} \Big(q_j(t) - \theta_j\Big) z_{ij}(t) x_{ij}^*(t). \tag{16}$$

where $x_{ij}^*(t)$ is the solution derived from (15).

Next, we introduce the concept of the energy link which is then used to search $\{\bar{s}_i(t)\}$ and $\{s_{ij}(t)\}$ efficiently.

An ***Energy link*** is represented as $\{\mu_l(t), R_l(t)\}$, where μ_l means energy link l is used to offload the task initiated by MD μ_l, and R_l is the set of WDs compute the task cooperatively. Since multiple energy links may or may not have the same task initiator and the same set of wireless devices compute the job for the task initiator, only $\mu_l(t)$ or $R_l(t)$ is not able to identify an energy link. Hence the combination $(\mu_l(t), R_l(t)$ is introduced to represent an energy link. To meet the task feasibility constraints (2-3), R_l is selected as

$$R_l(t) = \Big\{ W'_i \mid W'_i \subseteq W_i, i = \mu_l(t), \sum_{j \in W'_i} \frac{z_{ij}(t)}{w_{ij}(t)} \geq \frac{\rho_i(t)}{\pi_{t,o}}, |W'_i| \cdot \gamma \leq 1 \Big\}. \tag{17}$$

Note that, an energy link is used to offload a task from a MD to a set of WDs, and it is not a real wireless link. For each energy link l, we also associate a weight (denoted as $b_l(t)$) to it and $b_l(t)$ is set according to (16).

Moreover, energy link l_1 and l_2 cannot be activated simultaneously, or l_1 and l_2 "interfere" with each other when:

1) Two energy links share the same task initiator: $\mu_{l_1} = \mu_{l_2}$;
2) There exists at least one WD that belongs to both $R_{l_1}(t)$ and $R_{l_2}(t)$, that is $R_{l_1}(t) \cap R_{l_2}(t) \neq \Phi$.

Let the set of energy links be EL. We can then establish an interference graph IG. IG can be expressed as

$$IG = \{g_{l_1,l_2} \mid l_1, l_2 \in EL\}. \tag{18}$$

Here $g_{l_1,l_2} = 1$ if l_1 interferes with l_2. With IG, a maximum independent energy link set, denoted as I_s, should satisfy the following constraints:

1) $\forall l_1, l_2 \in I_s, g_{l_1,l_2} = 0$;
2) $\forall l_3 \notin I_s, \exists l_1 \in I_s$, such that $g_{l_1,l_3} = 1$.

We also define $N_l(t)$ as the set of energy links interfere with energy link l ($N_l(t) = \{l_1 \mid g_{l,l_1} = 1\}$). The maximum degree of $N_l(t)$, denoted as ϖ_{max}, can then be expressed as

$$\varpi_{max} = \max_{l \in EL} |N_l(t)|. \tag{19}$$

Here $|N_l(t)|$ is the cardinality of $N_l(t)$. Note that, with the concept of energy link, task offloading can be viewed as ***routing energy from the mobile edge cloud (WDs) to a MD through an energy link***. Hence to maximize the overall revenue of the mobile edge cloud is equivalent to route the harvested energy from the mobile edge cloud to MDs as efficiently as possible. Designate LS as the set of all possible I_s, and $\Big\{\bar{s}_i^*(t), s_{i,j}^*(t)\Big\}$ as the optimal solution of (13). $\Big\{\bar{s}_i^*(t), s_{i,j}^*(t)\Big\}$ is determined by the following equation

$$\{\bar{s}_i(t), s_{ij}(t)\} = \arg \max_{I_s \in LS} \sum_{l \in I_s} b_l(t). \tag{20}$$

With (20), we can easily derive the following theorem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

Theorem 1. *The multi-user multi-task offloading problem (13) is NP-hard.*

Proof. (13) can be reduced to (20), and (20) is a maximum independent set problem, which is NP-hard as has been discussed in [29]. Consequently, (13) is also NP-hard. □

When the network size is small, the optimal schedule $\{\bar{s}_i(t), s_{ij}(t)\}$ can be derived via exhaustively searching all possible maximum independent set (I_s). However, the associated computation complexity grows exponentially with the network size. Hence on-line heuristic algorithms are needed to find a good I_s efficiently.

Algorithm 1 The Multi-user Multi-task Offloading Algorithm

Data: $\{\bar{h}_j(t)\}$
Initialization: $\forall j \in W, q_j(0) \rightarrow 0$;
foreach *frame* t **do**
  foreach $j \in W$ **do**
    Harvest energy according to (12) and broadcast its available energy to its associated MDs;
  end
  Use CGMS or DGMS algorithm to search $\{\bar{s}_i(t)\}$,$\{s_{ij}(t)\}$ and $\{x_{ij}(t)\}$;
  Schedule the admitted task offloading requests;
  foreach $j \in W$ **do**
    Update $q_j(t)$ according to (6).
  end
end

Algorithm 2 The Centralized Greedy Maximal Scheduling Algorithm

Data: EL,$\{A_i(t)|i \in M\}$
Result: $\bar{s}_i(t)$,$\{s_{ij}(t)\}$,$\{x_{ij}(t)\}$
Initialization : $EL'EL$ A centralized controller collects the virtual queue information $q'_j(t)$ for all $j \in W$;
foreach $j \in W$ **do**
  if $q_j(t) < e_j^{max}$ **then**
    $EL' \rightarrow EL' \setminus \{l \mid j \in R_l(t)\}$
  end
end
$f_m \rightarrow true$;
while f_m **do**
  $l^* \rightarrow \arg\max_{l \in EL'} b_l(t)$;
  if $b_{l^*}(t) > 0$ **then**
    $\bar{s}_{\mu_{l^*}}(t) \rightarrow 1, s_{\mu_{l^*},j}(t) \rightarrow 1$;
    foreach $j \in R_{l^*}(t)$ **do**
      Set $x_{\mu_{l^*},j}(t)$ according to (15), $EL' \rightarrow EL' \setminus \{l \mid g_{l,l^*} = 1\}$;
      Send a task offloading confirmation and the corresponding workload assignment to MD μ_{l^*} and WDs in $R_{l^*}(t)$;
    end
  else
    $f_m \rightarrow false$
  end
end

4.2 The proposed centralized multi-user multi-task offloading scheme

In this sub-section, we will discuss how to determine the multi-user multi-task offloading scheme with a centralized approach. We use a Greedy Maximal Scheduling (GMS) to derive a schedule for (20), the details of the GMS-based multi-user multi-task offloading scheduling algorithm is shown in Algorithm 1. Algorithm 1 uses either the centralized GMS algorithm (abbreviated as CGMS) or the Distributed GMS algorithm (abbreviated as DGMS) to search a feasible schedule.

In the CGMS algorithm (shown in Algorithm 2), a centralized controller accepts one task offloading request and assigns its workload to WDs greedily so as to maximize the system utility in the current iteration. This procedure will be conducted iteratively until no more request can be admitted in the current frame.

4.3 The proposed distributed multi-user multi-task offloading scheme

As has been discussed before, a centralized greedy algorithm requires a centralized controller to admit and schedule workload from different MDs jointly. When the network size grows, or when there is no wired connections between different WDs, to coordinate workload scheduling in a centralized way will incur high system implementation complexity. Hence a distributed scheduling algorithm is highly preferred. In this section, we will propose a distributed multi-user multi-task offloading algorithm. To this end, we extend the algorithm discussed in [30] to search a feasible schedule for (20). The details of the DGMS (Distributed GMS) based multi-user multi-task offloading algorithm is shown in Algorithm 3.

In the algorithm, we first define

$$\eta = min\{\varpi_{max}, nl_{max}\},$$

where nl_{max} is the maximum number of energy links that can be activated simultaneously ($nl_{max} = max_{I_s \in LS}|I_s|$), and ϖ_{max} is the maximum degree of the interference graph IG (refer to (19)). Moreover, a MD i that needs to offload its task to other WDs, will maintain a WD set $W'_i(t)$, and choose WDs in $W'_i(t)$ to construct its energy link. The algorithm iterates η rounds. In each round, a MD i whose task offloading request has not been responded will choose an energy link l with the highest weight ($b_l(t)$) among the remaining available energy links, and broadcast a task offloading request to all the WDs in $R_l(t)$. The task offloading request message contains the general task information as well as the amount of workload needs to be offloaded to all the WDs in $R_l(t)$. When a WD receives one or more task offloading requests, it will choose the one with the highest weight, and send a task offloading accept signal to that MD. If a MD receives the task offloading accept signal from all the WDs in $R_l(t)$, it implies that energy link l has the highest weight among all the energy links interfere with itself. The MD will then acknowledge all the WDs in $R_l(t)$ that its task will be offloaded to them. Upon receiving the task offloading confirmation signal from MD i, WD j is ready to serve MD i. Hence it will inform other MDs connecting to it

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

with an occupation message. If MD i receives an occupation message from WD j, it will delete j from $W_i'(t)$.

Notice that, by removing j from $W_i'(t)$, a set of energy links interfering with the confirmed energy link will also be excluded from future use. Hence, through the four-way handshake, only the energy links whose weight is larger than its remaining interfering energy links will be selected. Hence we can argue that in each iteration the system utility of the DGMS-based scheduling scheme is no less than that of the CGMS-based scheme.

Algorithm 3 The Distributed Greedy Maximal Scheduling Algorithm

Data: M_j, W_i,$\{A_i(t)|i \in M\}$, $TW \to W$
Result: $\bar{s}_i(t)$,$\{s_{ij}(t)\}$,$\{x_{ij}(t)\}$
Initialization : $W_i'(t) \to W_i \setminus \left\{j \mid q_j(t) < e_j^{max}\right\}$;
for $sub_i = 1$ ***to*** η **do**
  foreach *MD* $i \in \{i_1 \mid i_1 \in M, \bar{s}_{i_1}(t) = 0, \rho_{i_1}(t) > 0\}$ **do**
    Sort WDs in $W_i'(t)$ in descending order, denote the new sequence as $T'_{i,1,|W_i'(t)|}$ select the largest $n_{i^*}(t)+1$ WDs such that $\rho_i(t) \leq \sum_{j \in T'_{i,1,n_{i^*}+1}} \frac{\pi_{t,o}}{w_{ij}(t)}$ and $\gamma \cdot (n^*+1) \leq 1$;
    Construct an energy link l with $R_l(t) = T'_{i,1,n_{i^*}+1}(t)$, and compute its weight $b_l(t)$ according to (16);
    Send a task request to MDs in $R_l(t)$;
  end
  foreach *WD* $j \in M \setminus TW$ **do**
    $m_j \to \arg\max_{\mu_l \in M_j} b_l(t)$ and respond a task accept signal to MD m_j;
  end
  foreach *MD* $i \in \{i_1 \mid i_1 \in M, \bar{s}_{i_1}(t) = 0, x_i(t) > 0\}$ **do**
    if *MD* i *receives task accept signals from all MDs in* $R_l(t)$ **then**
      Acknowledge all MDs in $R_l(t)$ with a task offloading confirmation signal;
      $\bar{s}_i(t) \to 1$,
      foreach *WD* $j \in R_l(t)$ **do**
        $s_{ij}(t) \to 1$, $x_{ij}(t)$ is set according to (15);
      end
    end
  end
  foreach *WD* $j \in \{M \setminus TW\}$ **do**
    if *WD* j *receives a task offloading confirmation signal* **then**
      $TW \to TW \cup j$ and send an occupation message to $i \in M_j \setminus m_j$.
    end
  end
  foreach *MD* $i \in \{i_1 \mid i_1 \in M, \bar{s}_{i_1}(t) = 0, x_i(t) > 0\}$ **do**
    if *MD i receives an occupation message from WD* j **then**
      $W_i'(t) \to W_i'(t) \setminus j$
    end
  end
end

Moreover, since the maximum number of simultaneously activated energy links is nl_{max}, it is expected that after nl_{max} rounds no more energy link can be activated. This is because at least one energy link will be accepted in one iteration if there exists such an energy link. Likewise, if an energy link is going to be admitted by the mobile edge cloud, in the worst case, it will be admitted after all its interfering energy links have been tried (and the corresponding task offloading requests are not accepted). This will take at most ϖ_{max} round. As a consequence, the maximum number of iterations is confined to η.

In addition, a distributed scheduling algorithm requires that all MDs report their true energy link weights to the WDS. If a selfish MD, who wants to offload its own task with higher priority, report its energy link weight higher than the true value. This MD will deprive other MDs computing resources, and resulting in decreased overall system utility.

The issue, however, can be easily addressed by creating an agent for each MD in one of its connected WDs or introducing brokers as middleware to coordinate task offloading between MDs and WDs. The agents or brokers will then maintain the accurate weight information of the energy links.

5 System Performance Analysis

5.1 Implementation complexity analysis

According to (12), when $q_j(t) > \theta_j$, $h_j(t) = 0$. Hence

$$q_j(t) \leq \theta_j + h_j^{max} = e_j^{max} + \phi(v_0) + h_j^{max}. \quad (21)$$

The battery size of WD j, denoted as $\zeta_j(v_0)$ is then set to

$$\zeta_j(v_0) = e_j^{max} + \phi(v_0) + h_j^{max}, \ \forall j \in W. \quad (22)$$

Notice that, the higher the v_0, the higher the battery size, and the higher the energy drift at WD j.

For the CGMS-based scheduling scheme, a centralized controller needs to collect the energy queue information of all the WDs as well as the task offloading requests from MDs. After selecting the set of energy links to be scheduled, the controller will pass the schedule to all the MDs and WDs, and the MDs will then offload the tasks to WDs. The complexity of the algorithm is $O(nl_{max})$ (recall that, $nl_{max} = \max_{I_s \in LS} |I_s|$).

As to the DGMS-based scheduling scheme, in the first part of the control sub-frame, WDs will pass its energy queue length information to its associated MDs. The algorithm then runs η iterations. In each iteration, 4 control sub-slots are required to finish the 4-way handshake task offloading confirmation. In total, $1+4\eta$ control sub-slots are needed and the complexity is $O(\eta)$. To be more specific, for a network with $|N| = 20$, and the control sub-slot equals to 50μs, since $\eta \leq |N|$, the number of control sub-slot is upper bounded by $1 + 4|N|$. If the frame length is 50ms, the control overhead is upper bounded by 8.01%. When the network size increases, we can further partition the network into several non-overlapping regions and perform DGMS for each region separately. How to partition the network into multiple regions is out of the scope of this work, and will be discussed in our future work.

5.2 Utility analysis

We first define the following terms:

1) U^{OPT}: the optimal system utility of the multi-user multi-task offloading problem;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

2) U_{v_0}: the optimal system utility of the Lyaponuv drift approach with exhaustive search (the sechedule is derived from the optimal value of (20)); and
3) $U_{v_0}^{CGMS}/U_{v_0}^{DGMS}$: the system utility of the CGMS/DGMS based scheduling scheme.

The performance gaps between the optimal multi-user multi-task offloading scheduling algorithm and the CGMS or DGMS based scheduling scheme are revealed in the following theorem.

Theorm 2. *For a given* v_0,

$$U_{v_0}^{CGMS} \geq \frac{U^{OPT}}{d_{max}} - \frac{\delta}{d_{max}v_0} \tag{23}$$

and

$$U_{v_0}^{DGMS} \geq \frac{U^{OPT}}{d_{max}} - \frac{\delta}{d_{max}v_0}, \tag{24}$$

where

$$d_{max} = \max_{\{l \in EL, I_s \in LS\}} |I_s \cap N_l|.$$

Proof. Please refer to Appendix B. □

The battery size is $O(v_0)$ and the utility of both CGMS and DGMS based scheme is $O(\frac{1}{v_0})$. It implies a lower energy storage trades for a lower achievable utility, and vice versa. Thus, we can balance the system implementation complexity and the system performance by choosing an appropriate control parameter v_0.

Theorem 2 also implies that compared with a CGMS-based scheduling scheme, the DGMS-based scheduling scheme will not degrade the system performance. However, allowing WDs to schedule tasks from MDs locally (a centralized scheduler is no longer needed), several rounds of information exchange are required to update the resource usage status among MDs and WDs. Hence, DGMS-based scheduling scheme uses more control overhead to trade for lower implementation complexity. Moreover, as has been discussed in the previous subsection, the scheduling overhead of the DGMS-based scheme is still relatively low. This implies that the proposed DGMS-based scheduling scheme provides an efficient alternative when a centralized controller is not available.

6 Simulations

6.1 Simulation Settings

We evaluate the system performance of the proposed CGMS and DGMS based multi-user multi-task offloading scheme in this section. We randomly generate (5 MDs,10 WDs), (10 MDs, 20 WDs), (15 MDs, 30 WDs), (20 MDs, 40 WDs) 4 topologies. The topologies to be tested are shown in Fig. 2.

The transmission range between a MD and a WD is 30 meters, and a pair of (MD, WD) can communicate as long as they are within each other's transmission range. Once the topology has been drawn, the transmission rate of each link is determined as

$$c_{ij} = B\log_2\left(1 + \frac{pt_i L_{i,j}}{n_0}\right), \tag{25}$$

where B, the bandwidth of the network is set to 20MHz, and the noise power n_0 is 90dBm. $L_{i,j} = dis_{i,j}^{\varphi}$ is the path loss between MD i and WD j (with distance equals to $dis_{i,j}$ and φ is set to 3.5).

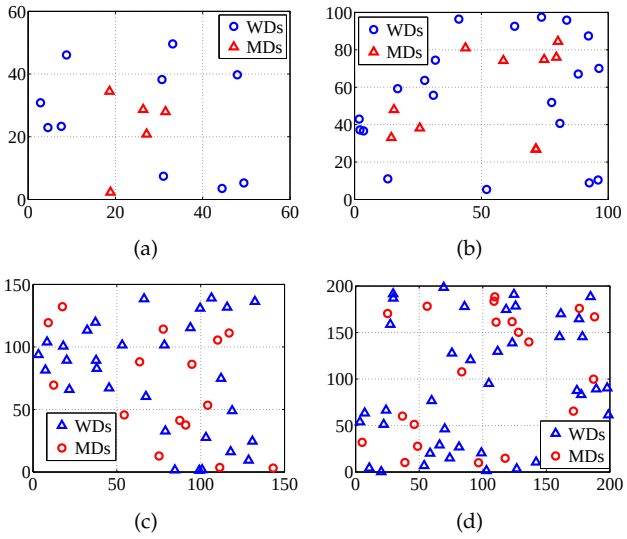

Fig. 2. The topologies of the four tested networks

TABLE 2
System Parameters

Transmitting/receiving power (pt_j/pr_j)	0.1W/0.07W
Noise (n_0)	90dBm
Path loss parameter (φ)	3.5
Frame duration	50ms
Control sub-slot duration	50μs
γ	0.1
$A_1 = \{\rho_1, \beta_1, r_1, u_1\}$	$\{29.5\text{KB}, 0.50, 750.0, 1.0\}$
$A_2 = \{\rho_2, \beta_2, r_2, u_2\}$	$\{27.6\text{KB}, 0.63, 734.5, 0.99\}$
$A_3 = \{\rho_3, \beta_3, r_1, u_3\}$	$\{32.8\text{KB}, 0.69, 418.7, 0.97\}$
$A_4 = \{\rho_4, \beta_4, r_4, u_4\}$	$\{29.7\text{KB}, 0.27, 861.6, 1.0\}$

The energy and task arrival rate as well as the CPU processing frequency are randomly generated with mean equals to 0.9W, 3 tasks/s and 2.2GHz respectively. We then randomly generate 4 tuples to represent 4 different task types (refers to A_1, A_2, A_3, A_4 in Table 2) to be used in the simulation. In each frame, a MD will first determine whether it has a task to be offloaded or not based on its task arrival rate. If a task arrives, it randomly selected a task type and send a task offloading request to the mobile edge cloud. The task statistics are in accordance to [11], since most of the energy are consumed for computing the task but not transmitting it, the revenue of a task is set to $u_i(t) = c_0 * \log(1 + \rho_i(t)d_i(t)$, and $c_0 = 19.1$ (so that the utility of A_1 can be normalized to 1). The system parameters are listed in Table 2.

6.2 Simulation Results

In this sub-section, we first study the convergence speed of the proposed CGMS and DGMS based scheduling scheme. We then evaluate the system performances in regards to different v_0, energy arrival rates, task arrival rates, and CPU processing frequency. To be more specific, as it is not convenient to evaluate the impacts of the average task arrival rate on the overall system performance by simply

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

changing the task arrival rate of one user (recall that we have multiple mobile devices in the system), we introduce task arrival rate α_T. In detail, α_T, is a scaler and is used to scale the task arrival rates of all the mobile devices. Similarly, α_E and α_F are introduced as the scaling factor for the average energy arrival rates and the CPU processing frequencies of all the wireless devices.

Moreover, beside the proposed CGMS and DGMS based multi-user multi-task offloading algorithm, we also study the performance of the optimal scheduling scheme and a Random Scheduling (RS) scheme. The optimal scheduling scheme is realized by exhaustively searching all possible scheduling policies and choose the best one in every frame. However, since it is too time consuming to derive the optimal scheduling scheme for the (10 MDs, 20 WDs), (15 MDs, 30 WDs), (20 MDs, 40 WDs) topologies, we only evaluate the performance of the optimal scheduling scheme for the (5 MDs, 10 WDs) topology. In the RS scheme, a MD will broadcast a task offloading request to all the WDs associated with it. A WD will randomly select one task offloading request and respond to it. If a MD receives enough responds from its associated WDs, it will issue a task offloading confirmation signal and the corresponding WDs will serve the MD in the task offloading sub-frame. Since the procedure is similar to that of the DGMS-based scheme, we omit the details of the RS scheme.

6.2.1 Convergence speed of the proposed Greedy Maximal Scheduling based algorithms

Fig. 3 shows the convergence time of the bottleneck energy queue for the tested 4 topologies in the proposed DGMS-based scheme. The bottleneck energy queue refers to the one with the highest average energy queue length. In the four tested topologies, the energy arrival rate α_E is set to 1 (refers to the case when every WD has enough energy for task offloading) and the task arrival rate α_A is set to 10 (refer to the case when every MD always needs to offload its task) so as to maintain the energy queue length variations at a high level. Moreover, the battery sizes are set to 9.11J,12.61J,12.31J and 14.61J for the 4 networks respectively so that the maximum system utility can be achieved. From Fig.3, we can observe that the 4 tested topologies will converge within 15s. Notice that, the convergence speed can be further accelerated if we reduce v_0, i.e., the battery size of each WD.

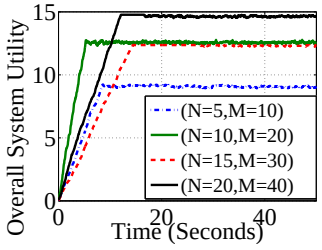

Fig. 3. Convergence time of the bottleneck queue for the 4 tested topologies

6.2.2 Battery size versus the overall system utility

The impact of the battery storage on the overall system performance is revealed in Fig. 4. The maximum required battery size is defined as $\zeta_{max}(v_0) = \max_{j \in W} \zeta_j(v_0)$. The four curves in 4(a) represent the performance of 4 different scheduling schemes. Note that, the red curve with cross marks (denoted as "Exhaustive") shows the performance of the optimal scheduling scheme. In the figure, the task arrival scaling parameter is set to a large value (i.e., $\alpha_A = 10$), the energy arrival scaling parameter α_E is 1, and the CPU frequency scaling parameter α_F is 1.

From Fig. 4, we can observe that the higher the battery size, the higher the overall system utility. When ζ_{max}=9.11J,12.61J,12.3J1J,14.61J, the overall system utility converges in the four tested topologies respectively. Furthermore, as the convergence performance of the CGMS-based scheduling scheme is almost the same as that of the DGMS-based scheduling scheme, we omit the convergence performance of the CGMS-based scheduling scheme here for brevity.

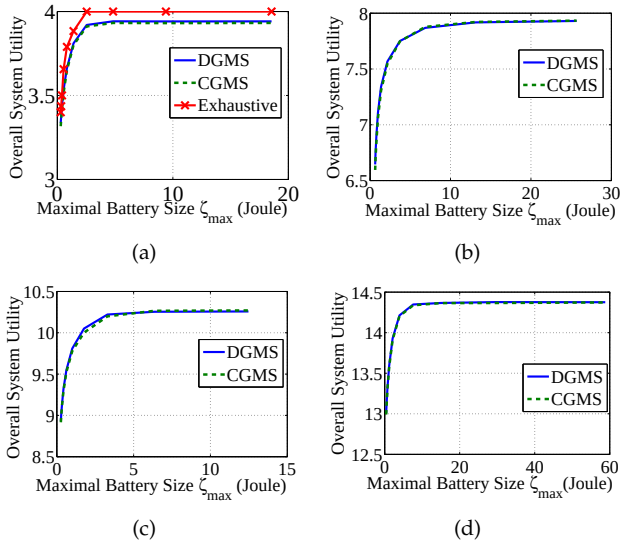

Fig. 4. Battery size versus the overall system utility of the: (a). (5MDs, 10WDs) network; (b). (10MDs, 20WDs) network; (c). (15MDs, 30WDs) network; (d). (20MDs, 40WDs) network

6.2.3 Energy arrival rate versus the overall system utility

In Fig. 5, the task arrival scaling parameter α_A is set to 10 and the CPU frequency scaling parameter α_F is 1. Moreover, ζ_{max} is set to 9.1J,12.61J,12.31J and 14.61J for the four tested topologies respectively. As shown in Fig. 5, the higher the energy arrival rate (higher α_E), the higher the computing ability of each WDs, and the higher the overall system utility. However, when the energy arrival rate reaches to a certain point, the overall system utility converges. This is because initially as the energy arrival rate increases, the probability that WDs will be used for task offloading (the probability that $q_j(t) \geq \theta_j$) increases, hence the overall system utility increases. When the energy arrival scaling parameter exceeds a certain value, we can expect that all the wireless devices tend to participate into task computation offloading as often as possible (overly energy supply for all the wireless devices).

We also observe that the proposed CGMS and DGMS based scheduling schemes achieve almost the same performance. Moreover, at low energy regime (with very small

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

α_E), the performance differences among four schemes are not obvious. This is due to the reason that with low energy supply, the number of devices can be used for task offloading is limited, in this case, the four schemes can achieve similar performance. As the energy supply increases, the performance gaps between the optimal scheme and CGMS/DGMS based scheduling schemes or the RS scheme become larger and larger. It implies that the optimal scheduling scheme can balance the energy usage and task offloading in the most effective way. When the energy supply is high enough, the performance gaps between the optimal scheme and the CGMS/DGMS based scheduling schemes shrink. However, the performance gap between the RS and other schemes remain significant. In conclusion, the optimal scheme shows at most 20.7% higher overall system utility than that of the CGMS/DGMS based scheduling scheme. As to the RS scheme, in the four tested topologies, its overall system utility is at most 28.2%, 28.1%, 19.4%, 31.9% lower than that of the proposed CGMS/DGMS based scheduling scheme.

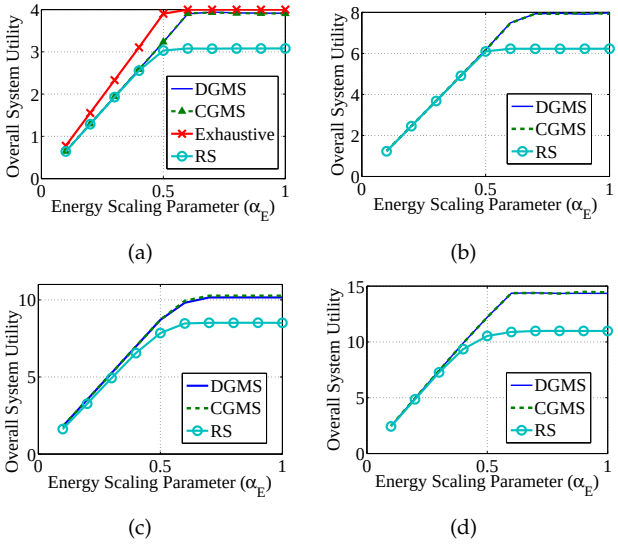

Fig. 5. Energy Arrival Scaling Parameter versus the overall system utility of the: (a). (5MDs, 10WDs) network; (b). (10MDs, 20WDs) network; (c). (15MDs, 30WDs) network; (d). (20MDs, 40WDs) network

6.2.4 Task arrival rate versus the overall system utility

Fig. 6 shows the impact of the task arrival rate on the overall system utility. Here the energy arrival scaling parameter α_E and the CPU frequency scaling parameter α_F are set to 1, and v_0 is set to 9.1J,12.6J,12.3J and 14.6J for the four tested topologies respectively. From Fig. 6(a), it is obvious that the higher the task arrival rate, the more tasks to be served, and the higher the overall system utility. In terms of the performance gaps among different schemes, we can observe similar trends as explained in the previous subsection. Furthermore, the optimal scheme can achieve at most 9.4% higher overall system utility than that of the proposed CGMS scheme from (5MDs, 10WDs) network. Likewise, the proposed DGMS scheme achieves 31.6%, 27.4%, 18.8% and 31.3% higher overall system utility than that of the RS scheme.

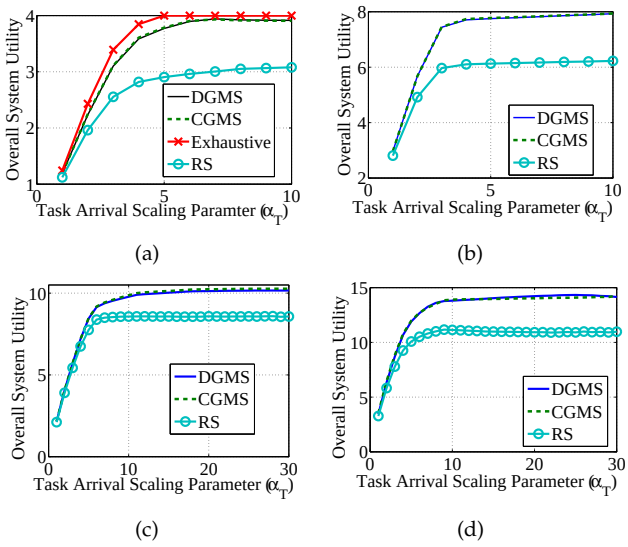

Fig. 6. Task Arrival Scaling Parameter versus the overall system utility of the: (a). (5MDs, 10WDs) network; (b). (10MDs, 20WDs) network; (c). (15MDs, 30WDs) network; (d). (20MDs, 40WDs) network

6.2.5 CPU Frequency versus the overall system utility

Fig. 7 shows the impact of different CPU processing frequencies on the overall system utility. Here the energy arrival scaling parameter α_E and the task arrival scaling parameter α_A are set to 1 and 10 respectively, and v_0 is set to 9.1J,12.6J,12.3J and 14.6J for the four tested topologies respectively. From Fig. 7(a), we can expect that the optimal scheme can achieve at most 9.9% higher overall system utility than that of the proposed CGMS/DGMS based scheduling scheme from (5MDs, 10WDs) network. Likewise, the proposed CGMD/DGMS based scheduling scheme achieve 24.1%, 27.3%, 16.4% and 32.8% higher overall system utility than that of the RS scheme.

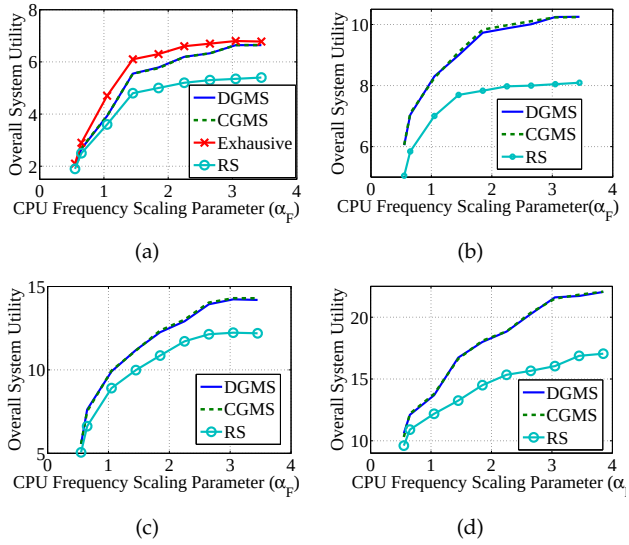

Fig. 7. CPU Frequency Scaling Parameter versus the overall system utility of the: (a). (5MDs, 10WDs) network; (b). (10MDs, 20WDs) network; (c). (15MDs, 30WDs) network; (d). (20MDs, 40WDs) network

7 Conclusions

In this work, we discuss multi-user multi-task offloading scheduling schemes in a renewable mobile edge cloud system. As the mobile edge cloud is composed of a set of WDs with relatively low processing ability (the processing ability is not as high as that of a server), scheduling schemes needs to map the workload from a MD to multiple WDs. Moreover, scheduling schemes also needs to handle uncertain energy supply at each WD properly so as to make the best of the mobile edge cloud system. In detail, the scheduling algorithms determine the energy harvesting strategy, a set of offloading requests to be admitted, and a sub-set of WDs to compute the workload for the admitted offloading request so as to maximize the overall system utility. To exploit the computing capacity at the green mobile edge cloud thoroughly, the scheduling algorithms match the offloading energy consumption at the mobile edge cloud to its harvestable energy. Since the scheduling problem is NP hard, centralized and distributed greedy maximal scheduling algorithms are proposed. As shown in Sec. 5, the proposed DGMS and CGMS algorithms can achieve at least $\frac{1}{d_{max}}$ of the overall system utility to that of the optimal scheduling scheme. Moreover, from the simulations, we can observe that compared with a random scheduling scheme, the proposed DGMS and CGMS algorithm can provide 16.4% to 32.8% higher overall system utility.

References

[1] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, PP(99):1–1, 2017.

[2] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. Mobile edge computing: Survey and research outlook. *arXiv preprint arXiv:1701.01090*, 2017.

[3] Min Chen, Yixue Hao, Yong Li, and Chin Feng Lai. On the computation offloading at ad hoc cloudlet: architecture and service modes. *Communications Magazine IEEE*, 53(6):18–24, 2015.

[4] Xue Lin, Yanzhi Wang, Qing Xie, and Massoud Pedram. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. *IEEE Transactions on Services Computing*, 8(2):175–186, 2015.

[5] S. Eman Mahmoodi, R. N. Uma, and K. P. Subbalakshmi. Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2016.

[6] Weiwei Chen, Chin Tau Lea, and Kenli Li. Dynamic resource allocation in ad-hoc mobile cloud computing. In *IEEE Wireless Communications and NETWORKING Conference*, pages 1–6, 2017.

[7] Sheng Zhang, Jie Wu, and Sanglu Lu. Distributed workload dissemination for makespan minimization in disruption tolerant networks. *IEEE Transactions on Mobile Computing*, 15(7):1661–1673, 2016.

[8] Changsheng You, Kaibin Huang, and Hyukjin Chae. Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE Journal on Selected Areas in Communications*, 34(5):1757–1771, 2016.

[9] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590 – 3605, 2016.

[10] Thinh Quang Dinh, Jianhua Tang, Quang Duy La, and Tony Q. S. Quek. Adaptive computation scaling and task offloading in mobile edge computing. In *IEEE Wireless Communications and NETWORKING Conference*, 2017.

[11] Songtao Guo, Bin Xiao, Yuanyuan Yang, and Yang Yang. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In *IEEE INFOCOM 2016 - the IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[12] Lingjun Pu, Xu Chen, Jingdong Xu, and Xiaoming Fu. D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration. *IEEE Journal on Selected Areas in Communications*, PP(99):1–1, 2016.

[13] Xu Chen, Lingjun Pu, Lin Gao, Weigang Wu, and Di Wu. Exploiting massive d2d collaboration for energy-efficient mobile edge computing. *IEEE Wireless Communications*, 24(4):64–71, 2017.

[14] Lei Yang, Jiannong Cao, Hui Cheng, and Yusheng Ji. Multi-user computation partitioning for latency sensitive mobile cloud applications. *IEEE Transactions on Computers*, 64(8):2253–2266, 2015.

[15] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. Cloudlet load balancing in wireless metropolitan area networks. In *IEEE INFOCOM 2016 - the IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[16] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5):2795–2808, 2016.

[17] Yuyi Mao, Jun Zhang, S. H. Song, and Khaled B. Letaief. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, PP(99):1–1, 2017.

[18] Jian Song, Yong Cui, Minming Li, Jiezhong Qiu, and Rajkumar Buyya. Energy-traffic tradeoff cooperative offloading for mobile cloud computing. In *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pages 284–289. IEEE, 2014.

[19] Jie Xu and Shaolei Ren. Online learning for offloading and autoscaling in renewable-powered mobile edge computing. In *GLOBECOM 2016 - 2016 IEEE Global Communications Conference*, pages 1–6, 2016.

[20] Slacdjana Jovsilo and Gyorgy Dan. A game theoretic analysis of selfish mobile computation offloading. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

[21] Yujin Li and Wenye Wang. Can mobile cloudlets support mobile applications? In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1060–1068, 2014.

[22] Jonathan Chase and Dusit Niyato. Joint optimization of resource provisioning in cloud computing. *IEEE Transactions on Services Computing*, 10(3):396–409, 2017.

[23] Tuyen X Tran and Dario Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *arXiv preprint arXiv:1705.00704*, 2017.

[24] Feng Wang, Jie Xu, Xin Wang, and Shuguang Cui. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *arXiv preprint arXiv:1702.00606*, 2017.

[25] David Astély, Erik Dahlman, Anders Furuskär, Ylva Jading, Magnus Lindström, and Stefan Parkvall. Lte: the evolution of mobile broadband. *IEEE Communications magazine*, 47(4), 2009.

[26] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 9–16. IEEE, 2015.

[27] Der-Jiunn Deng, Kwang-Cheng Chen, and Rung-Shiang Cheng. Ieee 802.11 ax: Next generation wireless local area networks. In *Heterogeneous networking for quality, reliability, security and robustness (QShine), 2014 10th international conference on*, pages 77–82. IEEE, 2014.

[28] M Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):211, 2010.

[29] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2):313–322, 2003.

[30] Changhee Joo, Xiaojun Lin, Jiho Ryu, and Ness B. Shroff. Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels. *IEEE/ACM Transactions on Networking*, 24(3):1476–1488, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2826544, IEEE Transactions on Services Computing

Weiwei Chen received the B.S. degree in electronic engineering from the Beijing University of Posts and Telecommunications University, Beijing, China, in 2007, and the Ph.D degree from the Hong Kong University of Science and Technology, Hong Kong in 2013. She is now with the College of Computer Science and Electronic Engineering, Hunan University, China. Her research interests include resource allocation in mobile edge computing, wireless networking for IoT system, future Internet architecture.

Dong Wang received the B.S. degree and PhD degree in computer science from Hunan University in 1986 and 2006. From December 2004 to December 2005, he was a visiting scholar in University of Technology Sydney, Australia. Since 1986, he has been working with Hunan University, China. Currently, he is professor of Hunan University. His main research interests include network test and performance evaluation, wireless communications and mobile computing, etc.

Keqin Li is a SUNY Distinguished Professor of computer science in the State University of New York. He is also a Distinguished Professor of Chinese National Recruitment Program of Global Experts (1000 Plan) at Hunan University, China. He was an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Infor- mation Science and Technology, Tsinghua University, Beijing, China, during 2011-2014. His current research interests include parallel computing and high performance computing, distributed computing, energy efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 520 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, and IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.