Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling

Thinh Quang Dinh, *Student Member, IEEE*, Jianhua Tang, *Member, IEEE*, Quang Duy La, *Member, IEEE*, and Tony Q. S. Quek, *Senior Member, IEEE*

***Abstract*— In this paper, we propose an optimization framework of offloading from a single mobile device (MD) to multiple edge devices. We aim to minimize both total tasks' execution latency and the MD's energy consumption by jointly optimizing the task allocation decision and the MD's central process unit (CPU) frequency. This paper considers two cases for the MD, i.e., fixed CPU frequency and elastic CPU frequency. Since these problems are NP-hard, we propose a linear relaxation-based approach and a semidefinite relaxation (SDR)-based approach for the fixed CPU frequency case, and an exhaustive search-based approach and an SDR-based approach for the elastic CPU frequency case. Our simulation results show that the SDR-based algorithms achieve near optimal performance. Performance improvement can be obtained with the proposed scheme in terms of energy consumption and tasks' execution latency when multiple edge devices and elastic CPU frequency are considered. Finally, we show that the MD's flexible CPU range can have an impact on the task allocation.**

***Index Terms*— Mobile edge computing, fog computing, semidefinite relaxation, computation offloading, dynamic voltage and frequency scaling.**

I. Introduction

RECENTLY, mobile devices have become an indispensable part of modern life. On a single mobile phone, there are numerous applications from calling, web browsing to speech recognition and navigation [1]. Hence, energy-efficient data processing is obviously vital for battery-empowered MDs. Mobile cloud computing (MCC) has been considered to be a potential solution [2]. Since the cloud servers have much higher computation and storage resources than MDs, migrating computation-intensive tasks to cloud servers can significantly reduce the burden of computing, storage, and computation energy on the MDs. In order to execute computation on cloud servers, the MDs and the servers are required to operate offloading frameworks, such as MAUI [3] and ThinkAir [4]. With these frameworks, the MDs can partition their application into executable tasks, and these tasks can be executed on cloud servers with different hardware architectures. However, cloud servers are usually logically and spatially far from MDs, which leads to huge communications latency. Small cell networks could be the key to deal with these limitations [5]. Due to the spatial proximity from small cell access points (APs) to users, if small cell APs are capable of processing tasks of MDs, this architecture could reduce the application time response.

The idea of bringing both communication and computational capacities close to users was firstly introduced as cloudlet in 2009 [6], where users can connect to nearby servers through a wireless LAN. Several years later, the term mobile edge computing (MEC) was used to describe the service executions at a mobile base station by IBM and Nokia Siemens Network [7]. Since then, MEC has attracted a lot of attention from academic areas [8], [9]. In particular, there are several works considering resource allocation for MEC. Some works focused on single user scenario [10], [11], while other works focused on multiuser scenario [12]–[14]. Munoz *et al.* [10] minimized the MD's energy consumption by jointly optimizing the transmission time and the amount of data offloaded to a femto AP. You *et al.* [11] proposed a framework where a MD can harvest energy from a base station or offload to it. You *et al.* [12] studied a centralized offloading framework for a multiuser MEC system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) aiming to minimize the MD's energy consumption. In a centralized manner, Sardellitti *et al.* [13] jointly optimized the users' transmit precoding matrices and the cloud's computational resource to minimize the users' energy consumption. Chen *et al.* [14], designed an efficient distributed computation offloading algorithm that can achieve a Nash equilibrium for multiuser scenario. However, in all the above works, a MD can only associate with either a single edge device or a remote server. Since APs will be densely deployed in future networks [5], we can exploit this diversity by considering a scenario where a MD can offload to multiple nearby APs with computational capacities, instead of only a single AP.

Beside task offloading, dynamic voltage frequency scaling (DVFS), a technique to adjust the frequency of a microprocessor, is another solution to reduce MDs' energy

Manuscript received September 2, 2016; revised January 4, 2017, March 16, 2017, and April 24, 2017; accepted April 24, 2017. Date of publication April 28, 2017; date of current version August 14, 2017. This work was supported in part by the MOE ARF Tier 2 under Grant MOE2015-T2-2-104, the Startup Funds of Chongqing University of Posts and Telecommunications under Grant A2016-114, the National Natural Science Foundation of China (NSFC) under Grant 61601071, and the Open Foundation of State Key Lab of Integrated Services Networks of Xidian University under Grant ISN17-01. The associate editor coordinating the review of this paper and approving it for publication was Y. Li. *(Corresponding author: Thinh Quang Dinh.)*

T. Q. Dinh and Q. D. La are with Singapore University of Technology and Design, Singapore 487372 (e-mail: quangthinh_dinh@mymail.sutd.edu.sg; quang_la@sutd.edu.sg).

J. Tang was with the Singapore University of Technology and Design, Singapore 487372. He is now with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, 400065 China (e-mail: jtang4@e.ntu.edu.sg).

T. Q. S. Quek is with the Singapore University of Technology and Design, Singapore 487372, and also with the Department of Electronics Engineering, Kyung Hee University, Yongin-si 17104, South Korea (e-mail: tonyquek@sutd.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2017.2699660

consumption [15]. Earlier works either did not take into account this aspect as they only considered a fixed CPU frequency at the MD [10], [16], [17], or assumed a simplified task model [18]–[20] with either no or infinite granularity in offloadable data. For example, Zhang *et al.* [18] jointly optimized the MD's CPU frequency and transmission rate with offloading decision which is either local processing or total offloading to a cloud server. Wang *et al.* [19], [20] considered computation partition which allowed offloadable data to be partially offloaded to a nearby femto-cloud. However, in [19] and [20], they assumed that the amount of offloadable data can be partitioned as small as possible. In practice, offloadable data contains some files or tasks which cannot be divided into smaller parts.

Last but not least, computational offloading affects both the MD's energy consumption and its tasks' execution latency. However, most of the previous work either focused only on energy consumption [21]–[23] or execution latency [24]–[26]. Li *et al.* [21] proposed a partitioning scheme that statistically divides program into server tasks and client tasks in order to minimize energy consumption. Xian *et al.* in [23] proposed an adaptive offloading scheme to improve the energy saving of mobile devices based on a timeout. Ra *et al.* [24] proposed a dynamic offloading strategy using a greedy algorithm to minimize the completion time of applications. Gu *et al.* [25] minimized tasks' execution latency of a MD with memory constraint. Yang *et al.* [26] designed an offline heuristic partitioning of multiple users' computation to minimize the average completion time for all users.

In this paper, we study the effect when a single MD is able to allocate tasks to multiple APs with computational capacities, and to scale its CPU frequency. We propose a framework where the MD minimizes both its energy consumption and its tasks' execution latency by jointly optimize the task allocation decision and its CPU frequency. By exploiting this diversity in term of allocation decision and CPU frequency, the MD's energy consumption and task latency performance can be improved. In addition, our task model considers practical aspects which are absent from previous works investigating this problem. In terms of granularity in offloadable tasks, we assume that they can be partitioned into independent batches of varying sizes. We will consider two scenarios for the MD, i.e., fixed CPU frequency and elastic CPU frequency. Since these problems are NP-hard, we propose approximation approaches to find solutions in our problems. In summary, the main contributions of this paper are as follows:

- In terms of the system model and problem formulation, we investigate the benefits of computational offloading, jointly in energy consumption and latency reduction, when the MD can offload its computational tasks to multiple APs. Our work offers a new approach to computational offloading with coupled DVFS with task offloading by dealing with irregular granularity in offloadable data. Two scenarios, i.e., fixed and elastic MD's CPU frequency, will be investigated.
- In terms of the mathematical framework, we propose algorithms based on semidefinite relaxation (SDR) to efficiently solve the problems resulting from the new task model and multiple-AP context. In the fixed CPU scenario, linear programming relaxation (LR) based algorithm is proposed as an alternative to SDR. In the elastic CPU scenario, we firstly consider an exhaustive search (ES) based algorithm as a benchmark and then apply the SDR-based algorithm to find the near optimal solution.
- Numerical results show that SDR-based approaches can achieve close to optimal performance. Moreover, we observe performance gain in the MD's energy consumption and the tasks' execution latency when multiple edge devices and frequency scaling are considered. Finally, we show that the MD's CPU range can affect the task allocation decision.

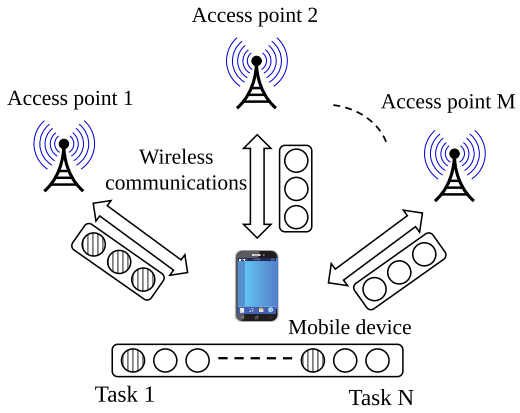


Fig. 1. Offloading framework with multiple APs.

The rest of the paper is organized as follows. The system model is introduced in Section II. We then propose the tasks offloading framework without and with CPU frequency scaling in Sections III and IV, respectively. Section V presents analysis on the solutions of the proposed offloading problems in special cases. We then present the numerical results in Section VI and final conclusion in Section VII.

II. System Model and Problem Formulation

We consider a MD with N independent tasks. Each of these tasks can be potentially offloaded to a CPU of any of the M wireless APs or locally processed by the MD' CPU as shown in Fig. 1. Our AP can, in fact, be modeled after a Fog-Radio Access Network (F-RAN) node in [27] where an equipment not only possesses communication capacities, but also provides application services. We denote the set of tasks as $\mathcal{N} = \{1, \ldots, N\}$, and the set of CPUs as $\mathcal{M} = \{0, \ldots, M\}$, where CPU 0 denotes the MD's CPU.[1] In the following, we consider APs occupying orthogonal wireless channels, i.e., signal from different APs cannot interfere with each other. The MD needs to decide whether tasks will be processed locally or remotely.[2]

[1]Since we treat the MD's CPU as CPU 0, in what follows, when we mention CPUs, we also include the MD's CPU.

[2]We only consider one user; however, for multiple users, they can be decoupled into independent single-user problems as in our model which is true in many real systems assigning non-overlapping resource blocks to users. This is commonly assumed in similar works in computational offloading [10], [20].

We use a tuple $\{\alpha_i, \beta_i, w_i\}$ to represent task i, for $i \in \mathcal{N}$ in which α_i is the input data size (in bits), β_i is the output data size (in bits), and w_i is the number of CPU cycles that is required to process the task, respectively. In the following, we assume that the amount of data to be processed, which is of irregular sizes, is known before any execution.[3] Moreover, N independent tasks are partitioned into $M+1$ disjoint sets, where each set is considered as a batch, i.e., a "big" task. Each AP provides the MD with fixed service rate (CPU frequency) r_k (cycles/sec).[4] The uplink and downlink rates between the user and APs are quantified as

$$C_k^{\mathrm{UL}} = B_k^{\mathrm{UL}} \log_2\left(1 + \frac{P^{\mathrm{Tx}} h_{ik}^{\mathrm{UL}}}{\varpi_0}\right),$$
$$C_k^{\mathrm{DL}} = B_k^{\mathrm{DL}} \log_2\left(1 + \frac{P_{\mathrm{AP}} h_{ik}^{\mathrm{DL}}}{\varpi_0}\right) \quad (1)$$

respectively, where B_k^{UL} and B_k^{DL} are the uplink and downlink channel bandwidths an AP k allocates for the MD, respectively; P^{Tx} and P_{AP} are the transmission powers of the MD and APs, respectively; h_{ik}^{UL} and h_{ik}^{DL} are the uplink and downlink channel gains between AP k and the MD, respectively; and ϖ_0 is the white noise power level. When the coherence time of channels is comparable to offloading duration (e.g., when the application deadline is short for some tasks), h_{ik}^{UL} and h_{ik}^{DL} can be considered constant. In this work, APs are assumed to have full channel state information (CSI). The MD can obtain these values from the APs and therefore, C_k^{UL} and C_k^{DL} are known to the MD. On the other hand, if the offloading duration is much longer than coherence time, the values of h_{ik}^{UL} and h_{ik}^{DL} vary which leads C_k^{UL} and C_k^{DL} to fluctuate within the duration. Thus, we assume that the effective rates C_k^{UL} and C_k^{DL} over the whole duration can be approximated by their average values. For local processing, there is no uplink and downlink transmission latency, i.e., $C_0^{\mathrm{UL}} = +\infty$ and $C_0^{\mathrm{DL}} = +\infty$. We leverage an indicator x_{ik}, $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{M}$, to represent the task allocation, i.e.,

$$x_{ik} = \begin{cases} 1, & \text{if task } i \text{ is assigned to CPU } k, \\ 0, & \text{otherwise.} \end{cases}$$

We denote $\mathbf{X} = \{x_{ik}\} \in \{0,1\}^{N \times (M+1)}$ the *task allocation matrix*, and $\mathbf{x} = [\mathbf{x}_0^T, \mathbf{x}_1^T, \ldots, \mathbf{x}_k^T, \ldots, \mathbf{x}_M^T]^T$, where $\mathbf{x}_k = \left[x_{1k}, x_{2k}, \ldots, x_{Nk}\right]^T$, the column vector corresponding to $\mathbf{X}$.

Define d_{ik}^{UL}, d_{ik}^{DL} and d_{ik}^{Comp} as the upload, download, and computational latency, respectively, when task i is offloaded to CPU k, where $d_{ik}^{\mathrm{UL}} = \frac{\alpha_i}{C_k^{\mathrm{UL}}}$, $d_{ik}^{\mathrm{DL}} = \frac{\beta_i}{C_k^{\mathrm{DL}}}$ and $d_{ik}^{\mathrm{Comp}} = \frac{w_i}{r_k}$. Hence, the total upload latency T_k^{UL}, download latency T_k^{DL}, and computation latency T_k^{Comp} of the batch offloaded to CPU k are defined as $T_k^{\mathrm{UL}} = \frac{\sum_{i \in \mathcal{N}} x_{ik} \alpha_i}{C_k^{\mathrm{UL}}}$, $T_k^{\mathrm{DL}} = \frac{\sum_{i \in \mathcal{N}} x_{ik} \beta_i}{C_k^{\mathrm{DL}}}$, and $T_k^{\mathrm{Comp}} = \frac{\sum_{i \in \mathcal{N}} x_{ik} w_i}{r_k}$, respectively. Each AP $k \in \mathcal{M}$ will receive, process its batch, and return the result to the MD in sequence. In fact, an AP can start to process either all tasks after receiving all of them or some tasks after it receives the first few while still receiving more tasks. The second case will make the analysis more intractable due to the arrival order of tasks at the AP tremendously enlarging the decision domain. Hence, for simplicity, we assume non-overlapping steps at APs, which means an AP begins processing tasks after receiving all. The execution latency of the batch offloaded to CPU k, $\forall k \in \mathcal{M}$ is given by

$$T_k = T_k^{\mathrm{UL}} + T_k^{\mathrm{Comp}} + T_k^{\mathrm{DL}} = \sum_{i \in \mathcal{N}} x_{ik} \left(\frac{\alpha_i}{C_k^{\mathrm{UL}}} + \frac{w_i}{r_k} + \frac{\beta_i}{C_k^{\mathrm{DL}}}\right)$$
$$= \sum_{i \in \mathcal{N}} x_{ik} D_{ik},$$

where $D_{ik} = \frac{\alpha_i}{C_k^{\mathrm{UL}}} + \frac{w_i}{r_k} + \frac{\beta_i}{C_k^{\mathrm{DL}}}$. The energy consumption of the MD is mainly contributed by computational processing and wireless transmission.

1) Computational Energy Consumption: The computational energy consumption of the MD is defined as

$$E^{\mathrm{Comp}} = \rho r_0^{\zeta} \sum_{i \in \mathcal{N}} x_{i0} D_{i0},$$

where ρr_0^{ζ} is the computational power of the MD. As in [15], ρ is a constant that depends on the average switched capacitance and the average activity factor.[5] The value ζ ($\zeta \geq 2$) is a constant (usually close to 3) [15]. Here, we set $\zeta = 3$.

2) Wireless Transmission Energy Consumption: The wireless transmission energy consumption of the MD is defined as

$$E^{\mathrm{TR}} = P^{\mathrm{Tx}} \sum_{k \in \mathcal{M} \setminus \{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\mathrm{UL}} + P^{\mathrm{Rx}} \sum_{k \in \mathcal{M} \setminus \{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\mathrm{DL}},$$

where P^{Tx} and P^{Rx} are the transmitting and receiving power, respectively, which are regarded as constants.[6]

The two main objectives of the MD are to minimize 1) the tasks' execution latency, and 2) its energy consumption. These metrics are introduced as follows.

1) Execution Latency: Since APs process tasks in parallel, the execution latency is given by

$$t(\mathbf{X}, r_0) = \max_{k \in \mathcal{M}} T_k.$$

[3]This assumption is valid with data-partitioned-oriented applications, such as face detection, virus scanning, and G-zip compression [10].

[4]It is assumed that by guaranteeing a fixed service rate for the MD to process tasks, the AP can therefore process tasks from other MDs simultaneously. In practice, although the APs' computational capacities may vary during processing time, our assumption still holds if (a) the APs' computational capacities are very large, which allows APs to maintain the service rates they guarantee for each MD; and (b) the presence of a federation of fog and cloud allows them to dynamically scale up their computational capacities to guarantee service rates [28].

[5]The power coefficient ρ may depend also on the CPU frequency. However, it is beyond the scope of this paper.

[6]In more general models, the MD can adaptively control the transmission power according to the channel gains and latency requirements. In this case, the problem could be approximated by an equivalent homogeneous quadratic constrained quadratic programming. Subsequently, we can apply SDR to find an approximate solution. However, new methods should be devised in order to recover to a feasible solution, which could be the topic of in a future study.

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

Notation	Definition
i	index of a task
k	index of a AP's CPU or MD's CPU
α_i	size of the input data of task i
β_i	size of the out data of task i
w_i	required number of CPU cycles to process task i
λ_t	scalar weight of tasks' execution latency
λ_e	scalar weight of the MD's energy consumption
$C_k^{\mathrm{UL}}, C_k^{\mathrm{DL}}$	uplink and downlink data rate between the MD and AP k
D_{ik}	execution latency when task i is offloaded to CPU k
r_k	service rate of the AP's CPU k
r_0	CPU frequency of the MD
ρ	model dependent constant for the CPU frequency of the computational processing energy consumption
P^{Comp}	computational power of the MD
$P^{\mathrm{Tx}}, P^{\mathrm{Rx}}$	transmitting and receiving power of the MD
E^{Comp}	computational energy consumption of the MD
E^{TR}	transmission energy consumption of the MD
x_{ik}	decision variable assigning task i to CPU k
$\mathbf{X}$	task allocation matrix
$t(\mathbf{X}, r_0)$	tasks' execution latency
$e(\mathbf{X}, r_0)$	total energy consumption of the MD
$r_{\min}, r_{\max}$	minimum and maximum CPU frequency of the MD

2) Total Energy Consumption: The total energy consumption of MD is given by

$$e(\mathbf{X}, r_0) = E^{\mathrm{Comp}} + E^{\mathrm{TR}}.$$

However, the two objectives above are coupled by $\{\mathbf{X}, r_0\}$, and, hence, cannot be minimized independently and simultaneously. In the following, we investigate the trade-off between the two objectives and define the joint objective function (or total cost) as

$$\psi(\mathbf{X}, r_0) \triangleq \lambda_t t(\mathbf{X}, r_0) + \lambda_e e(\mathbf{X}, r_0), \tag{2}$$

where $\lambda_t, \lambda_e \in [0, 1]$ are scalar weights. Equation (2) can be considered as a weighted sum approach of a general multi-objective optimization problem. As mentioned in Proposition 3.9 of [29], given positive weights, minimizing $\mathscr{E}1$ will reach an efficient solution of the multi-objective optimization problem. If any of the weights is zero, it will reach a weakly efficient solution. The weighted-sum approach is extensively used because it is simple to understand and easy to implement. Also, the weight itself reflects the relative importance (preference) between energy and latency. For example, when the MD battery is low and the user only cares about the energy consumption, the MD can set $\lambda_t = 0$ and $\lambda_e = 1$. Therefore, our optimization problem is formulated as

$$\begin{aligned}
\mathscr{E}1: \ & \min_{\mathbf{X}, r_0} \ \psi(\mathbf{X}, r_0), \\
& \text{s.t} \sum_{k \in \mathcal{M}} x_{ik} = 1, \quad \forall i \in \mathcal{N}, && \text{(3a)} \\
& \quad r_0 \in [r_{\min}, \ r_{\max}], && \text{(3b)} \\
& \quad x_{ik} \in \{0, 1\}, && \text{(3c)}
\end{aligned}$$

We denote the feasible region of problem $\mathscr{E}1$ as $\mathcal{X}$. The constraints (3a) guarantee that a task can only be assigned to one AP.[7] Note that from now on, for ease of exposition, we use prefix $\mathscr{F}$ to label the fixed MD's CPU frequency problem and $\mathscr{E}$ to label the elastic counterpart. Moreover, all the previously introduced notations and their definitions have been summarized in Table I for easy reference.

Proposition 1: $\mathscr{E}1$ is a NP-hard problem.

Proof: Consider a special case when the frequencies of all APs are the same, which means r_0 is fixed, $\lambda_t = 1$ and $\lambda_e = 0$, and there is no transmission latency. Hence, $D_{i0} = D_{i1} = \cdots = D_{iM} = D_i, \forall i \in \mathcal{N}$. Problem $\mathscr{E}1$ is equivalent to

$$\begin{aligned}
& \min_{\mathbf{X}} \ \max_{k \in \mathcal{M}} \left(\sum_{i \in \mathcal{N}} x_{ik} D_i \right), \\
& \text{s.t} \sum_{k \in \mathcal{M}} x_{ik} = 1, \quad \forall i \in \mathcal{N}, \\
& \quad x_{ik} \in \{0, 1\}.
\end{aligned}$$

The special case is so called the makespan minimization problem for identical parallel machines [30], where makespan is equivalent to the completion time of the last task. From [31], the special case is NP-hard and hence, so is $\mathscr{E}1$. □

III. Optimization of Task Offloading With a Fixed CPU Frequency

In this section, we propose an optimization framework to determine an optimal task allocation decision that minimizes the weighted sum of latency and energy consumption at the MD when the CPU frequency of MD is fixed. Due to fixed CPU frequency, D_{i0} and $P^{\mathrm{Comp}} = \rho r_0^{\zeta}$ become constant. As such, the objective function in (2) is simplified as follows:

1) The makespan is given by:

$$t(\mathbf{X}) = \max_{k \in \mathcal{M}} \left(\sum_{i \in \mathcal{N}} x_{ik} D_{ik} \right).$$

2) The total energy is simplified as:

$$\begin{aligned}
e(\mathbf{X}) = \ & P^{\mathrm{Comp}} \sum_{i \in \mathcal{N}} x_{i0} D_{i0} + P^{\mathrm{Tx}} \sum_{k \in \mathcal{M} \setminus \{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\mathrm{UL}} \\
& + P^{\mathrm{Rx}} \sum_{k \in \mathcal{M} \setminus \{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\mathrm{DL}}.
\end{aligned}$$

As such, the optimization framework for fixed CPU frequency can be rewritten as

$$\begin{aligned}
\mathscr{F}1: \ & \min_{\mathbf{X}} \ \lambda_t t(\mathbf{X}) + \lambda_e e(\mathbf{X}), \\
& \text{s.t} \sum_{k=0}^{M} x_{ik} = 1, \quad \forall i \in \mathcal{N}, && \text{(4a)} \\
& \quad x_{ik} \in \{0, 1\}. && \text{(4b)}
\end{aligned}$$

Remark 1: Using Proposition 1, $\mathscr{F}1$ is also NP-hard.

To determine the optimal solution of $\mathscr{F}1$, we first introduce a new variable t with additional constraint

[7] A more general task model where a task is splittable and can be partially offloaded to multiple APs or partially locally processed could be considered. As such, the binary constraints can be replaced by $x_{i,k} \in [0, 1]$. In the fixed CPU frequency scenario, the problem can be solved by linear programming. In the elastic counterpart, new methods should be devised.

$t \geq \max_{k\in\mathcal{M}} \left(\sum_{i\in\mathcal{N}} x_{ik} D_{ik}\right)$ to transform into $\mathscr{F}2$ as follows:

$$\begin{aligned}\mathscr{F}2: \min_{\mathbf{X},t}\ & \lambda_t t + \lambda_e P^{\text{Comp}} \sum_{i\in\mathcal{N}} x_{i0} D_{i0} \\ & + \lambda_e P^{\text{Tx}} \sum_{k\in\mathcal{M}\setminus\{0\}} \sum_{i\in\mathcal{N}} x_{ik} d_{ik}^{\text{UL}} \\ & + \lambda_e P^{\text{Rx}} \sum_{k\in\mathcal{M}\setminus\{0\}} \sum_{i\in\mathcal{N}} x_{ik} d_{ik}^{\text{DL}} \\ \text{s.t } & \sum_{i\in\mathcal{N}} x_{ik} D_{ik} \leq t, \quad \forall k \in \mathcal{M}, \qquad (5a) \\ & \sum_{k\in\mathcal{M}} x_{ik} = 1, \quad \forall i \in \mathcal{N}, \qquad (5b) \\ & x_{ik} \in \{0,1\}, \qquad (5c)\end{aligned}$$

where $\mathscr{F}2$ is a mixed-integer linear programming (MILP) problem. Although $\mathscr{F}2$'s optimal solution can be found via Branch-and-Bound (BnB) algorithm [32], this algorithm incurs huge computational complexity, especially when the dimension of $\mathcal{X}$ is large. Thus, we propose two efficient methods to solve $\mathscr{F}2$, namely, the LR and SDR approaches.

A. Linear Programing Relaxation Approach

In this subsection, we propose a LR-based algorithm to solve $\mathscr{F}2$. The key idea is to relax the binary variables x_{ik} to real numbers $x_{ik} \in [0,1]$. The resulting relaxed problem $\mathscr{F}2$-1 is then solved by interior points method which has the complexity in polynomial time $O\left(\nu^{3.5}K^2\right)$, where ν is the number of variables and K is the number of the bits in the input [33], [34]. Denote $\mathbf{y} = [\mathbf{x}^T, t]^T$, $\mathbf{D}_k = [D_{1k}, \ldots, D_{Nk}]^T$, $\forall k \in \mathcal{M}$, and $\mathbf{d}_k^{\text{UL}} = \left[d_{1k}^{\text{UL}}, \ldots, d_{Nk}^{\text{UL}}\right]^T, \forall k \in \mathcal{M}\setminus\{0\}$. The LR of $\mathscr{F}2$ can be vectorized as follows:

$$\begin{aligned}\mathscr{F}2\text{-}1: \min_{\mathbf{y}}\ & \mathbf{b}_0^T \mathbf{y}, \\ \text{s.t } & \mathbf{A}_1 \mathbf{y} \leq \mathbf{0}_{(M+1)\times 1}, \qquad (6a) \\ & \mathbf{A}_2 \mathbf{y} = \mathbf{1}_{N\times 1}, \qquad (6b) \\ & y_j \in [0,1], \quad \forall j = 1, \ldots, NM+N, \qquad (6c)\end{aligned}$$

where y_j is the jth element of vector $\mathbf{y}$, and

$$\begin{aligned}\mathbf{b}_0 &= \left[\lambda_e P^{\text{Comp}} \mathbf{D}_0^T, \lambda_e \mathbf{b}_0'^T, \lambda_t\right]^T, \\ \mathbf{b}_0' &= P^{\text{Tx}} \left[\mathbf{d}_1^{\text{UL}^T}, \mathbf{d}_2^{\text{UL}^T}, \ldots, \mathbf{d}_M^{\text{UL}^T}\right]^T \\ &\quad + P^{\text{Rx}} \left[\mathbf{d}_1^{\text{DL}^T}, \mathbf{d}_2^{\text{DL}^T}, \ldots, \mathbf{d}_M^{\text{DL}^T}\right]^T, \\ \mathbf{A}_1 &= \begin{bmatrix} \mathbf{D}_0^T & \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \cdots & \mathbf{0}_{1\times N} & -1 \\ \mathbf{0}_{1\times N} & \mathbf{D}_1^T & \mathbf{0}_{1\times N} & \cdots & \mathbf{0}_{1\times N} & -1 \\ \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \mathbf{D}_2^T & \cdots & \mathbf{0}_{1\times N} & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \cdots & \mathbf{D}_M^T & -1 \end{bmatrix}, \\ \mathbf{A}_2 &= \left[\mathbf{I}_{N,0}, \mathbf{I}_{N,1}, \mathbf{I}_{N,2}, \ldots, \mathbf{I}_{N,k}, \ldots, \mathbf{I}_{N,M}, \mathbf{0}_{N\times 1}\right],\end{aligned}$$

where $\mathbf{I}_{N,k}$ is a N-dimensional identity matrix, $\forall k \in \mathcal{M}$. Here, the constraints (6a), (6b), and (6c) are equivalent to the constraints (5a), (5b), and (5c), respectively.

Let $\hat{\mathbf{y}} = [\hat{\mathbf{x}}, \hat{t}\ ]$ denote the optimal solution of $\mathscr{F}2$-1. We first transform $\hat{\mathbf{x}}$ to the equivalent fractional matrix $\hat{\mathbf{X}}$, whose elements are in $[0,1]$ by a "reshape" operation. The term "reshape" means to change the size of a vector or a matrix while its number of elements is unchanged. If all components of $\hat{\mathbf{X}}$ are binary, it is also the optimal solution to $\mathscr{F}1$. Otherwise, to recover binary characteristic of $\hat{\mathbf{X}}$, for each row of $\hat{\mathbf{X}}$, we set the highest element to 1 and set the rest to 0. By doing this procedure, we obtain the solution $\mathbf{X}^{\text{LR}}$. We summarize the LR-based algorithm in Algorithm 1. We notice that, in step 1 of Algorithm 1, $\mathscr{F}2$-1 is solved by a linear programming solver. In step 3 and 4, $\hat{x}_{ik}$ and x_{ik}^{LR} represent the element in row i column k of matrices $\hat{\mathbf{X}}$ and $\mathbf{X}^{\text{LR}}$, respectively. The complexity of this algorithm is $O\left(\nu^{3.5}K^2\right)$, where $\nu = N(M+1)+1$.

Algorithm 1 Fixed CPU Frequency Task Offloading-LR-Based Algorithm

Input: $M, N, L, \alpha_i, \beta_i, w_i, r_k, C_k^{\text{UL}}, C_k^{\text{DL}}, \forall i \in \mathcal{N},\ \forall k \in \mathcal{M}$;
1: Solve $\mathscr{F}2$-1 to achieve $\hat{\mathbf{X}}$;
2: **if** $\hat{\mathbf{X}}$ is a binary matrix, **then**
3: $\quad \mathbf{X}^{\text{LR}} = \hat{\mathbf{X}}$;
4: **else**
5: $\quad$ **for** $i = 1$ to N **do**
6: $\quad\quad k^* \leftarrow \arg\max_{k\in\mathcal{M}} \hat{x}_{ik}$;
7: $\quad\quad x_{ik}^{\text{LR}} \leftarrow 0, \forall\ k \in \mathcal{M}\setminus\{k^*\}$ and $x_{ik^*}^{\text{LR}} \leftarrow 1$;
8: $\quad$ **end for**
9: **end if**
Output: $\mathbf{X}^{\text{LR}}$

B. Semidefinite Relaxation Approach

In this subsection, we propose an alternative algorithm to solve $\mathscr{F}2$. Following [35], $\mathscr{F}2$ is transformed to an equivalent homogeneous quadratic constrained quadratic programming (QCQP). By dropping the rank-one constraint, a homogeneous QCQP problem becomes a Semidefinite programming (SDP) which is a convex problem and can be solved using the interior point method with a worst case complexity of $O(\max\{m,n\}^4 n^{1/2}\log(1/\epsilon))$, where n is the dimension of the symmetric matrix $\mathbf{Z}$, m is the number of constraints, and $\epsilon > 0$ is the solution accuracy. As a result, the binary conditions are first replaced by quadratic constraints

$$x_{ik}(1 - x_{ik}) = 0 \quad \forall i,k \iff x_{ik} \in \{0,1\}. \qquad (7)$$

Then, the problem $\mathscr{F}2$ is transformed into

$$\begin{aligned}\mathscr{F}3: \min_{\mathbf{y}}\ & \mathbf{b}_0^T \mathbf{y}, \\ \text{s.t } & \mathbf{A}_1 \mathbf{y} \leq \mathbf{0}_{(M+1)\times 1}, \qquad (8a) \\ & \mathbf{A}_2 \mathbf{y} = \mathbf{1}_{N\times 1}, \qquad (8b) \\ & \mathbf{y}^T \text{diag}(\mathbf{u}_p)\mathbf{y} - \mathbf{u}_p^T \mathbf{y} = 0, \quad p = 1, \ldots, NM+N, \qquad (8c)\end{aligned}$$

where $\mathbf{u}_p$ is a $(NM+N+1)\times 1$ unit vector with the pth entry being 1; and $\text{diag}(\mathbf{u}_p)$ is a diagonal matrix whose diagonal entries starting in the upper left corner are the elements of $\mathbf{u}_p$. Here, the constraints (8a), (8b), and (8c) are

equivalent to the constraints (5a), (5b), and (5c), respectively. The constraint (8c) is equivalent to (7).

Defining $\mathbf{Z} = \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}^T & 1 \end{bmatrix}$ and $Q = MN + N + 1$, $\mathscr{F}3$ is then homogenized to

$$\begin{aligned} \mathscr{F}3\text{-}1: \ & \min_{\mathbf{Z}} \ \mathrm{Tr}(\mathbf{B}_0\mathbf{Z}), \\ & \text{s.t } \mathrm{Tr}(\mathbf{H}_h\mathbf{Z}) \le 0, \quad h = 1, \ldots, M+1, && \text{(9a)} \\ & \mathrm{Tr}(\mathbf{J}_j\mathbf{Z}) = 1, \quad j = 1, \ldots, N, && \text{(9b)} \\ & \mathrm{Tr}(\mathbf{G}_p\mathbf{Z}) = 0, \quad p = 1, \ldots, NM+N, && \text{(9c)} \\ & \mathbf{Z} \succeq 0, \quad \mathrm{rank}(\mathbf{Z}) = 1, \quad z_{Q+1,Q+1} = 1, && \text{(9d)} \end{aligned}$$

where

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{0}_{Q\times Q} & \frac{1}{2}\mathbf{b}_0 \\ \frac{1}{2}\mathbf{b}_0^T & 0 \end{bmatrix}, \quad \mathbf{H}_h = \begin{bmatrix} \mathbf{0}_{Q\times Q} & \frac{1}{2}\mathbf{A}_{1,h}^T \\ \frac{1}{2}\mathbf{A}_{1,h} & 0 \end{bmatrix},$$

$$\mathbf{J}_j = \begin{bmatrix} \mathbf{0}_{Q\times Q} & \frac{1}{2}\mathbf{A}_{2,j}^T \\ \frac{1}{2}\mathbf{A}_{2,j} & 0 \end{bmatrix}, \quad \mathbf{G}_p = \begin{bmatrix} \mathrm{diag}(\mathbf{u}_p) & -\frac{1}{2}\mathbf{u}_p \\ -\frac{1}{2}\mathbf{u}_p & 0 \end{bmatrix}.$$

Moreover, $z_{Q+1,Q+1} = 1$ is the element at row $Q+1$ and column $Q+1$ of matrix $\mathbf{Z}$; $\mathbf{A}_{1,h}$ and $\mathbf{A}_{2,j}$ are the hth row vector of matrix $\mathbf{A}_1$ and the jth row vector of matrix $\mathbf{A}_2$, respectively. Here, the constraints (9a), (9b), and (9c) are equivalent to the constraints (8a), (8b), and (8c), respectively.

In problem $\mathscr{F}3$-1, only the rank constraint rank($\mathbf{Z}$) = 1 is non-convex, whereas the objective function and all other constraints are convex. Denote $\mathbf{Z}^*$ the optimal solution of $\mathscr{F}3$-1 without the rank one constraint which is found by a convex programming solver. If $\mathbf{Z}^*$ is of rank one, we can construct the optimal solution to problem $\mathscr{F}1$ as follows. $\mathbf{Z}^* = \begin{bmatrix} \mathbf{y}^* \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}^{*T} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}^* \\ t^* \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{*T} & t^* & 1 \end{bmatrix}$. We extract the upper-left $(MN+N)\times(MN+N)$ sub-matrix of $\mathbf{Z}^*$, called $\mathbf{Z}'$, where $\mathbf{Z}'$ is positive semidefinite [36]. We have $\mathbf{Z}' = \mathbf{x}^*\mathbf{x}^{*T}$. Because $x_{ik} \in \{0, 1\}$, $x_{ik}x_{ik} = x_{ik}$. Hence, $\mathbf{x}^*$ is the diagonal of $\mathbf{Z}'$. Subsequently, we can construct the optimal task allocation matrix $\mathbf{X}^*$ from $\mathbf{x}^*$.

If $\mathbf{Z}^*$ is not of rank one, we propose an algorithm based on Gaussian randomization to obtain an approximate solution of $\mathscr{F}1$. We highlight the main ideas of the randomization algorithm as follows:

1) Randomly generate L feasible solutions of $\mathscr{F}1$ based on a multivariate Gaussian distribution having zero mean and $\mathbf{Z}'$ as the covariance matrix;
2) Select the solution that minimizes the objective value for $\mathscr{F}1$.

Let L denote the sample size of the randomization, and let the superscript (l) denote the index of a random sample. The proposed algorithm is summarized in Algorithm 2. Algorithm 2 can be considered a variant of randomization approaches mentioned in [37] and [38].

Once extracting the upper-left $(MN + N) \times (MN + N)$ sub-matrix $\mathbf{Z}'$, we can generate L random $(MN + N) \times 1$ vectors, the lth vector denoted by $\xi^{(l)}$, which is based on $\xi^{(l)} \sim \mathcal{N}\left(\mathbf{0}_{(MN+N)\times 1}, \mathbf{Z}'\right)$, for $l = 1, \ldots, L$. We note that the dimension of $\xi^{(l)}$ is equal to the dimension of $\mathbf{x}$. Here, we need to recover the binary characteristic of $\xi^{(l)}$ to make them feasible to $\mathscr{F}1$. If we use the element-wise sign function as mentioned in [35] and [37], the recovered vectors may not satisfy the constraints (3a). Thus, we map each vector $\xi^{(l)} \in \mathbb{R}^{MN+N}$ into a vector $\hat{\mathbf{x}}^{(l)} \in [0, 1]^{MN+N}$, by a sigmoid function $sig(x) \triangleq \frac{1}{1+\exp(-\mu x)}$, where $\mu \gg 1$. After that, using the idea of Algorithm 1 to recover the binary characteristic while satisfying the constraints (3a), we reshape each vector $\hat{\mathbf{x}}^{(l)} \in [0, 1]^{MN+N}$ to its corresponding fractional task allocation matrix $\hat{\mathbf{X}}^{(l)} \in [0, 1]^{N\times(M+1)}$. For each row of $\hat{\mathbf{X}}^{(l)}$, we set the highest element to 1, and set the rest to 0. By performing this procedure, for each matrix $\hat{\mathbf{X}}^{(l)}$, a matrix $\bar{\mathbf{X}}^{(l)} \in \{0, 1\}^{N\times(M+1)}$ is obtained which satisfies the constraints (3a). Finally, by searching the minimum over all L matrices $\bar{\mathbf{X}}^{(l)}$, we can obtain the solution $\left\{\mathbf{X}^{\mathrm{SDR}}, r_0^{\mathrm{SDR}}\right\}$ to $\mathscr{F}1$.

We summarize the SDR-based Algorithm for fixed CPU frequency in Algorithm 2. The symbols $\hat{x}_{ik}^{(l)}$ and $\bar{x}_{ik}^{(l)}$ represent the element in row i and column k of $\hat{\mathbf{X}}$ and $\bar{\mathbf{X}}^{(l)}$, respectively. We notice that there is a trade-off between the number of random samples L and the algorithm performance. As in [35], near optimality could be achieved at a small value of L in comparison with the size of the decision space. The complexity of this algorithm is $O(m^4 n^{1/2}\log(1/\epsilon) + LNM)$, where $n = N(M+1)$ and $m = N(M+1) + M + N + 2$.

Algorithm 2 Fixed CPU Frequency Task Offloading-SDR-Based Algorithm

Input: $M, N, L, \alpha_i, \beta_i, w_i, r_k, C_k^{\mathrm{UL}}, C_k^{\mathrm{DL}}, \forall i \in \mathcal{N}, \ \forall k \in \mathcal{M}$;
Solve $\mathscr{F}3$-1 without the rank-1 constraint to achieve $\mathbf{Z}^*$;
Extract the upper left $(MN + N) \times (MN + N)$ sub-matrix $\mathbf{Z}'$ from $\mathbf{Z}^*$;
if rank($\mathbf{Z}^*$)= 1 **then**
  $\mathbf{x}^*$ is the diagonal of $\mathbf{Z}'$;
  Construct $\mathbf{X}^*$ from $\mathbf{x}^*$;
else
  for $l = 1$ to L **do**
    Generate $\xi^{(l)} \sim \mathcal{N}\left(\mathbf{0}_{(MN+N)\times 1}, \mathbf{Z}'\right)$;
    Map $\xi^{(l)}$ to $\hat{\mathbf{x}}^{(l)}$ element-wise by a sigmoid function;
    Reshape $\hat{\mathbf{x}}^{(l)} \in \mathbb{R}^{MN+N}$ to $\hat{\mathbf{X}}^{(l)} \in \mathbb{R}^{N\times(M+1)}$;
    for $i = 1$ to N **do**
      $k^* \leftarrow \arg\max_{k\in\mathcal{M}} \hat{x}_{ik}^{(l)}$;
      $\bar{x}_{ik}^{(l)} \leftarrow 0, \forall\, k \in \mathcal{M}\backslash\{k^*\}$ and $\bar{x}_{ik^*}^{(l)} \leftarrow 1$;
    end for
    Achieve $\bar{\mathbf{X}}^{(l)}$;
  end for
  Determine $\mathbf{X}^{\mathrm{SDR}}$ by finding out the minimum $\psi\left(\bar{\mathbf{X}}^{(l)}\right)$ over L matrices $\bar{\mathbf{X}}^{(l)}$;
end if
Output: $\mathbf{X}^{\mathrm{SDR}}$

IV. Joint Optimization of Task Offloading and Frequency Scaling

Unlike the previous section, for the general case when the MD's CPU frequency is elastic, there is no standard method as to find the optimal solution. Therefore, we first discuss an exhaustive search approach, which obtains the global optimal solution. Then, we also propose a low-complexity SDR approach for practical implementation.

A. Exhaustive Search Approach

In this subsection, we present an exhaustive search method to find the optimal solution in a finite number of iterations. As this method is costly, we consider it only for comparison with other lower complexity methods. We denote $\bar{r}_0$ as the optimal solution for problem $\mathscr{E}1$ with given $\bar{\mathbf{X}}$. We have the following proposition.

Algorithm 3 Elastic CPU Frequency Task Offloading-Exhaustive Search Algorithm

Input: $M, N, \alpha_i, \beta_i, w_i, r_k, C_k^{\mathrm{UL}}, C_k^{\mathrm{DL}}, \forall i \in \mathcal{N},\ \forall k \in \mathcal{M}, r_{\min}, r_{\max}$;
1: For each feasible $\bar{\mathbf{X}}$, find $\bar{r}_0$ by applying Proposition 2;
2: Determine $\{\mathbf{X}^*, r_0^*\}$ by finding out the minimum $\psi(\bar{\mathbf{X}}, \bar{r}_0)$ over all feasible $\bar{\mathbf{X}}$;
Output: $\{\mathbf{X}^*, r_0^*\}$;

Proposition 2: Given a certain matrix $\bar{\mathbf{X}}$ for problem $\mathscr{E}1$, then the optimal $\bar{r}_0$ is given by

1) *When $x_{i0} = 0, \forall i \in \mathcal{N}$, $\bar{r}_0$ is arbitrary within the range $[r_{\min},\ r_{\max}]$.*[8]
2) *When $\exists\ x_{i0} \neq 0$,*
   a) *If $r_U < r_{\min}$, $\bar{r}_0 = r_{\min}$.*
   b) *If $r_U \in [r_{\min},\ r_{\max}]$,*

$$\bar{r}_0 = \begin{cases} \underset{r_0 \in \{r_U, r_{\min}\}}{\arg\min}\ \psi\left(\bar{\mathbf{X}}, r_0\right), & \text{when } r_\lambda < r_{\min}, \\ \underset{r_0 \in \{r_U, r_\lambda\}}{\arg\min}\ \psi\left(\bar{\mathbf{X}}, r_0\right), & \text{when } r_\lambda \in [r_{\min}, r_U], \\ r_U, & \text{when } r_\lambda > r_U. \end{cases} \tag{10}$$

   c) *If $r_U > r_{\max}$,*

$$\bar{r}_0 = \begin{cases} r_{\min}, & \text{when } r_\lambda < r_{\min}, \\ r_\lambda, & \text{when } r_\lambda \in [r_{\min},\ r_{\max}], \\ r_{\max}, & \text{when } r_\lambda > r_{\max}, \end{cases} \tag{11}$$

where $r_U = \frac{\sum_{i\in\mathcal{N}} x_{i0} w_i}{\max_{k\in\mathcal{M}\setminus\{0\}} \left(\sum_{i\in\mathcal{N}} x_{ik} D_{ik}\right)}$ and $r_\lambda = \left(\frac{\lambda_t}{2\lambda_e \rho}\right)^{\frac{1}{3}}$.

Proof: See Appendix. □

Based on Proposition 2, we perform an ES to find the optimal solution for problem $\mathscr{E}1$. Firstly, we find all feasible task allocation matrices $\bar{\mathbf{X}}$ with up to $(M+1)^N$ possible cases. For each feasible $\bar{\mathbf{X}}$, we find the optimal $\bar{r}_0$ according to Proposition 2. Then, by searching all possible $\bar{\mathbf{X}}$, we can obtain the optimal $\left\{\mathbf{X}^*, r_0^*\right\}$. We summarize the ES algorithm in Algorithm 3.

B. Semidefinite Relaxation Approach

In the following, a SDR-based algorithm is proposed to efficiently achieve a near optimal solution. We introduce two new variables $\upsilon = r_0^2$ and t such that $t \geq \max_{k\in\mathcal{M}} T_k$. Also, the binary constraints are replaced by the quadratic constraints like (7). Then, the problem $\mathscr{E}1$ is transformed into

$$\begin{aligned} \mathscr{E}2: \min_{\mathbf{X}, r_0, \upsilon, t}\ & \lambda_t t + \lambda_e \rho \upsilon \sum_{i\in\mathcal{N}} x_{i0} w_i \\ & + \lambda_e P^{\mathrm{Tx}} \sum_{k\in\mathcal{M}\setminus\{0\}} \sum_{i\in\mathcal{N}} x_{ik} d_{ik}^{\mathrm{UL}} \\ & + \lambda_e P^{\mathrm{Rx}} \sum_{k\in\mathcal{M}\setminus\{0\}} \sum_{i\in\mathcal{N}} x_{ik} d_{ik}^{\mathrm{DL}}, \\ \text{s.t } & \sum_{i\in\mathcal{N}} x_{ik} D_{ik} \leq t, \quad \forall k \in \mathcal{M}\setminus\{0\}, && (12a) \\ & \sum_{i\in\mathcal{N}} x_{i0} w_i \leq t r_0, && (12b) \\ & \sum_{k\in\mathcal{M}} x_{ik} = 1, \quad \forall i \in \mathcal{N}, && (12c) \\ & r_0^2 - \upsilon = 0, && (12d) \\ & x_{ik}(1 - x_{ik}) = 0, \quad \forall i \in \mathcal{N}, k \in \mathcal{M}, && (12e) \\ & r_0 \in [r_{\min},\ r_{\max}]. && (12f) \end{aligned}$$

Define $\mathbf{y} = [\mathbf{x}^T, r_0, \upsilon, t]^T$, $\mathbf{w} = [w_1, \ldots, w_N]^T$, $\mathbf{d}_k^{\mathrm{UL}} = [d_{1k}^{\mathrm{UL}}, \ldots, d_{Nk}^{\mathrm{UL}}]^T$ and $\mathbf{D}_k = [D_{1k}, \ldots, D_{Nk}]^T$, $\forall k \in \mathcal{M}\setminus\{0\}$, and $\mathbf{u}_p$ is the $(NM + N + 3) \times 1$ unit vector with the pth component being one. Then, the problem $\mathscr{E}2$ is vectorized as follows

$$\begin{aligned} \min_{\mathbf{y}}\ & \mathbf{y}^T \mathbf{A}_0 \mathbf{y} + \mathbf{b}_0^T \mathbf{y}, \\ \text{s.t } & \mathbf{A}_1 \mathbf{y} \leq \mathbf{0}_{(M+1)\times 1}, && (13a) \\ & \mathbf{y}^T \mathbf{A}_2 \mathbf{y} + \mathbf{b}_2^T \mathbf{y} \leq 0, && (13b) \\ & \mathbf{A}_3 \mathbf{y} = \mathbf{1}_{N\times 1}, && (13c) \\ & \mathbf{y}^T \mathbf{A}_4 \mathbf{y} + \mathbf{b}_4^T \mathbf{y} = 0, && (13d) \\ & \mathbf{y}^T \mathrm{diag}(\mathbf{u}_p)\mathbf{y} - \mathbf{u}_p^T \mathbf{y} = 0, \quad p = 1, \ldots, NM + N, && (13e) \\ & r_{\min} \leq \mathbf{b}_5^T \mathbf{y} \leq r_{\max}, && (13f) \end{aligned}$$

where

$$\mathbf{A}_0 = \begin{bmatrix} \mathbf{0}_{Q_1\times Q_1} & \frac{1}{2}\mathbf{a}_0 & \mathbf{0}_{Q_1\times 1} \\ \frac{1}{2}\mathbf{a}_0^T & 0 & 0 \\ \mathbf{0}_{1\times Q_1} & 0 & 0 \end{bmatrix},$$

$$\mathbf{a}_0 = [\mathbf{w}^T, \mathbf{0}_{1\times(MN+1)}]^T, \quad Q_1 = MN + N + 1,$$

$$\mathbf{b}_0 = \left[\mathbf{0}_{1\times N}, \lambda_e \mathbf{b}_0'^T, 0, 0, \lambda_t\right]^T,$$

$$\mathbf{b}_0' = P^{\mathrm{Tx}}\left[\mathbf{d}_1^{\mathrm{UL}^T}, \mathbf{d}_2^{\mathrm{UL}^T}, \ldots, \mathbf{d}_M^{\mathrm{UL}^T}\right]^T + P^{\mathrm{Rx}}\left[\mathbf{d}_1^{\mathrm{DL}^T}, \mathbf{d}_2^{\mathrm{DL}^T}, \ldots, \mathbf{d}_M^{\mathrm{DL}^T}\right]^T,$$

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{0}_{1\times N} & \mathbf{D}_1^T & \mathbf{0}_{1\times N} & \cdots & \mathbf{0}_{1\times N} & 0 & 0 & -1 \\ \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \mathbf{D}_2^T & \cdots & \mathbf{0}_{1\times N} & 0 & 0 & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & \cdots & \mathbf{D}_M^T & 0 & 0 & -1 \end{bmatrix},$$

[8] Note that in simulations, when $x_{i0} = 0, \forall i \in \mathcal{N}$, as $\bar{r}_0$ is arbitrary, we set it to $r_{\min}$.

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{0}_{Q_2\times Q_2} & \frac{1}{2}\mathbf{a}_2 \\ \frac{1}{2}\mathbf{a}_2^T & 0 \end{bmatrix}, \quad Q_2 = MN + N + 2,$$

$$\mathbf{a}_2 = [\mathbf{0}_{1\times(MN+N)}, -1, 0]^T, \mathbf{b}_2 = [\mathbf{w}^T, \mathbf{0}_{1\times(MN+3)}]^T,$$

$$\mathbf{A}_3 = \left[\mathbf{I}_{N,0}, \mathbf{I}_{N,1}, \mathbf{I}_{N,2}, \ldots, \mathbf{I}_{N,k}, \ldots, \mathbf{I}_{N,M}, \mathbf{0}_{N\times 3}\right],$$

where $\mathbf{I}_{N,k}$ is a N-dimensional identity matrix, $\forall k \in \mathcal{M}$,

$$\mathbf{A}_4 = \begin{bmatrix} \mathbf{0}_{Q_3\times Q_3} & \mathbf{0}_{Q_3\times 1} & \mathbf{0}_{Q_3\times 2} \\ \mathbf{0}_{1\times Q_3} & 1 & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{2\times Q_3} & \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 2} \end{bmatrix}, \quad Q_3 = MN + N,$$

$$\mathbf{b}_4 = [\mathbf{0}_{1\times Q_1}, -1, 0]^T, \quad \mathbf{b}_5 = [\mathbf{0}_{1\times Q_3}, -1, \mathbf{0}_{1\times 2}]^T.$$

Here, the constraints (12a)-(12f) are equivalent to the constraints (13a)-(13f), respectively.

Define $\mathbf{Z} = \begin{bmatrix}\mathbf{y}\\ 1\end{bmatrix}\begin{bmatrix}\mathbf{y}^T & 1\end{bmatrix}$ and $Q_4 = MN + N + 3$. Then, the problem $\mathscr{E}2$ is homogenized to:

$$\mathscr{E}2\text{-}1: \min_{\mathbf{Z}} \ \mathrm{Tr}(\mathbf{B}_0\mathbf{Z}),$$

$$\text{s.t } \mathrm{Tr}\ (\mathbf{H}_h\mathbf{Z}) \le 0, \quad h = 1, \ldots, M, \tag{14a}$$

$$\mathrm{Tr}(\mathbf{B}_2\mathbf{Z}) \le 0, \tag{14b}$$

$$\mathrm{Tr}(\mathbf{J}_j\mathbf{Z}) = 1, \quad j = 1, \ldots, N, \tag{14c}$$

$$\mathrm{Tr}(\mathbf{B}_4\mathbf{Z}) = 0, \tag{14d}$$

$$\mathrm{Tr}(\mathbf{G}_p\mathbf{Z}) = 0, \quad p = 1, \ldots, NM + N, \tag{14e}$$

$$r_{\min} \le \ \mathrm{Tr}(\mathbf{B}_5\mathbf{Z}) \le r_{\max}, \tag{14f}$$

$$\mathbf{Z} \succeq 0, \quad \mathrm{rank}(\mathbf{Z}) = 1, \ z_{Q_4+1,Q_4+1} = 1, \tag{14g}$$

where

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{A}_0 & \frac{1}{2}\mathbf{b}_0 \\ \frac{1}{2}\mathbf{b}_0^T & 0 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} \mathbf{A}_2 & \frac{1}{2}\mathbf{b}_2 \\ \frac{1}{2}\mathbf{b}_2^T & 0 \end{bmatrix},$$

$$\mathbf{B}_4 = \begin{bmatrix} \mathbf{A}_4 & \frac{1}{2}\mathbf{b}_4 \\ \frac{1}{2}\mathbf{b}_4^T & 0 \end{bmatrix},$$

$$\mathbf{B}_5 = \begin{bmatrix} \mathbf{0}_{Q_4\times Q_4} & \frac{1}{2}\mathbf{b}_5 \\ \frac{1}{2}\mathbf{b}_5^T & 0 \end{bmatrix}, \quad \mathbf{H}_h = \begin{bmatrix} \mathbf{0}_{Q_4\times Q_4} & \frac{1}{2}\mathbf{A}_{1,h}^T \\ \frac{1}{2}\mathbf{A}_{1,h} & 0 \end{bmatrix},$$

$$\mathbf{J}_j = \begin{bmatrix} \mathbf{0}_{Q_4\times Q_4} & \frac{1}{2}\mathbf{A}_{3,j}^T \\ \frac{1}{2}\mathbf{A}_{3,j} & 0 \end{bmatrix}, \ \mathbf{G}_p = \begin{bmatrix} \mathrm{diag}(\mathbf{u}_p) & -\frac{1}{2}\mathbf{u}_p \\ -\frac{1}{2}\mathbf{u}_p & 0 \end{bmatrix}.$$

We note that z_{Q_4+1,Q_4+1} is the element at row Q_4+1 and column Q_4+1 of matrix $\mathbf{Z}$; $\mathbf{A}_{1,h}$ and $\mathbf{A}_{3,j}$ are the hth row vector of matrix $\mathbf{A}_1$ and the jth row vector of matrix $\mathbf{A}_3$, respectively. Here, the constraints (13a)-(13f) are equivalent to the constraints (14a)-(14f), respectively.

Similar to $\mathscr{F}3$-1, by dropping the rank constraint, the problem $\mathscr{E}2$-1 can be solved efficiently. Denote $\mathbf{Z}^*$ as the optimal solution of $\mathscr{E}2$-1 without this rank constraint. If $\mathbf{Z}^*$ is of rank one, we can construct the optimal solution to problem $\mathscr{E}1$ as follows. $\mathbf{Z}^* = \begin{bmatrix}\mathbf{y}^*\\ 1\end{bmatrix}\begin{bmatrix}\mathbf{y}^{*T} & 1\end{bmatrix}$, where $\mathbf{y} = [\mathbf{x}^T \ r_0 \ v \ t]^T$. We extract the upper-left $(MN+N)\times(MN+N)$ sub-matrix of $\mathbf{Z}^*$, called $\mathbf{Z}'$. We have $\mathbf{Z}' = \mathbf{x}^*\mathbf{x}^{*T}$. Because $x_{ik} \in \{0,1\}$, $x_{ik}x_{ik} = x_{ik}$. Hence, $\mathbf{x}^*$ is the diagonal of $\mathbf{Z}'$. Subsequently, we can construct the optimal task allocation matrix $\mathbf{X}^*$ from $\mathbf{x}^*$. The optimal frequency $r_0^* = \sqrt{z^*_{Q_1,Q_1}}$, where $z^*_{Q_1,Q_1}$ is the element at row Q_1 and column Q_1 of matrix $\mathbf{Z}^*$.

Like Algorithm 2, we use Gaussian randomization procedure to obtain an approximate solution of $\mathscr{E}1$ if $\mathbf{Z}^*$ is not of rank one. Utilizing Algorithm 2, we can obtain L matrices $\bar{\mathbf{X}}^{(l)} \in \{0,1\}^{N\times(M+1)}$. For each matrix $\bar{\mathbf{X}}^{(l)}$, we find the optimal $\bar{r}_0^{(l)}$ according to Proposition 2. Then, by searching the minimum over all L matrices $\bar{\mathbf{X}}^{(l)}$, we can obtain the solution $\left\{\mathbf{X}^{\mathrm{SDR}}, r_0^{\mathrm{SDR}}\right\}$. We summarize the SDR-based algorithm in Algorithm 4.

Algorithm 4 Elastic CPU Frequency Task Offloading-SDR-based Algorithm

Input: $M, N, L, \alpha_i, \beta_i, w_i, r_k, C_k^{\mathrm{UL}}, C_k^{\mathrm{DL}}, \forall i \in \mathcal{N}, \ \forall k \in \mathcal{M}, r_{\min}, r_{\max}$;
1: Solve $\mathscr{E}2$-1 without the rank-1 constraint to achieve $\mathbf{Z}^*$;
2: Extract the upper left $(MN+N)\times(MN+N)$ sub-matrix $\mathbf{Z}'$ from $\mathbf{Z}^*$;
3: **if** rank($\mathbf{Z}^*$)= 1 **then**
4: $\mathbf{x}^*$ is the diagonal of $\mathbf{Z}'$;
5: Construct $\mathbf{X}^*$ from $\mathbf{x}^*$;
6: $r_0^* \leftarrow \sqrt{z^*_{Q_1,Q_1}}$;
7: **else**
8: Utilize Algorithm 2 to obtain L matrices $\bar{\mathbf{X}}^{(l)}$;
9: For each matrix $\bar{\mathbf{X}}^{(l)}$, find $\bar{r}_0^{(l)}$ by applying Proposition 2;
10: Determine $\left\{\mathbf{X}^{\mathrm{SDR}}, r_0^{\mathrm{SDR}}\right\}$ by finding out the minimum $\psi\left(\bar{\mathbf{X}}^{(l)}, \bar{r}_0^{(l)}\right)$ over all L matrices $\bar{\mathbf{X}}^{(l)}$;
11: **end if**
Output: $\left\{\mathbf{X}^{\mathrm{SDR}}, r_0^{\mathrm{SDR}}\right\}$;

V. Analysis on the MD's CPU Frequency Range

In this part, we investigate the impact of the MD's CPU frequency range on the task allocation decision. Denote $r^{\text{off\_a}} = \sqrt{\rho^{-1}\max_{i\in\mathcal{N}}\min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}w_i} + P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}w_i}\right)}$ and $r^{\text{off\_b}} = \sqrt{\rho^{-1}\min_{i\in\mathcal{N}}\min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}w_i} + P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}w_i}\right)}$.

Proposition 3: When the MD's objective includes only energy consumption, i.e. $\lambda_e = 1$ and $\lambda_t = 0$, the MD offloads all tasks (i.e., $x_{i0} = 0, \forall i \in \mathcal{N}$) if the MD's optimal CPU frequency is $r_0^* > r^{off\_a}$. On the other hand, if $r_0^* < r^{off\_b}$, the MD locally processes all tasks (i.e., $x_{i0} \neq 0, \forall i \in \mathcal{N}$).

Proof: When $\lambda_e = 1$ and $\lambda_t = 0$, our objective becomes

$$\min_{\mathbf{X},r_0} \ \rho r_0^2 \sum_{i\in\mathcal{N}} x_{i0}w_i + P^{\mathrm{Tx}} \sum_{k\in\mathcal{M}\setminus\{0\}}\sum_{i\in\mathcal{N}} x_{ik}d_{ik}^{\mathrm{UL}} + P^{\mathrm{Rx}} \sum_{k\in\mathcal{M}\setminus\{0\}}\sum_{i\in\mathcal{N}} x_{ik}d_{ik}^{\mathrm{DL}}. \tag{15}$$

The MD offloads all tasks when

$$\begin{aligned}
&\rho r_0^{*2} w_i \\
&> \min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}}+P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}}\right), \quad \forall i\in\mathcal{N} \\
&\Leftrightarrow r_0^{*2} > \frac{\min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}}+P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}}\right)}{\rho w_i}, \quad \forall i\in\mathcal{N} \\
&\Leftrightarrow r_0^{*2} > \rho^{-1}\min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}w_i}+P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}w_i}\right), \quad \forall i\in\mathcal{N} \\
&\Leftrightarrow r_0^{*2} > \rho^{-1}\max_{i\in\mathcal{N}}\min_{k\in\mathcal{M}}\left(P^{\mathrm{Tx}}\frac{\alpha_i}{C_k^{\mathrm{UL}}w_i}+P^{\mathrm{Rx}}\frac{\beta_i}{C_k^{\mathrm{DL}}w_i}\right).
\end{aligned}$$

Similarly, the MD locally processes all tasks when $r_0^* < r^{\text{off\_b}}$. □

According to [10] which considers only one AP, the conditions for the MD to either offload all or locally process all tasks are found under no MD's CPU frequency constraints. Thus, Proposition 3 could be considered as an extended result from [10] when multiple APs and frequency scaling are regarded.[9]

Remark 2: When the MD's objective includes only energy consumption, from (15), the objective value monotonically increases with respect to r_0. Thus, the MD's optimal CPU frequency will be $r_0^* = r_{\min}$. Hence, in this scenario, $r_{\min}$, the minimum MD's CPU frequency, controls the final task allocation decision.

Proposition 4: Let $R = [r_{\min}, r_{\max}]$ be the current frequency range of the MD, R' be a new frequency range in which $R \subset R'$, and ψ^* and $\psi^{*\prime}$ are the optimal cost functions when R and R' are considered, respectively. Then, $\psi^{*\prime} \leq \psi^*$.

Proof: Denote $\mathcal{X}'$ the new feasible region corresponding with R'. As $R \subset R'$, $\mathcal{X} \subset \mathcal{X}'$. Assuming that there is an optimal solution $\{\mathbf{X}^*, r_0^*\}$ in $\mathcal{X}$, it can be derived that

$$\{\mathbf{X}^*, r_0^*\} = \arg\min_{\{\mathbf{X}, r_0\}\in\mathcal{X}} \psi(\mathbf{X}, r_0) \in \mathcal{X}'.$$

Then, it follows that $\psi^{*\prime} = \min_{\{\mathbf{X}, r_0\}\in\mathcal{X}'} \psi(\mathbf{X}, r_0) \leq \psi(\mathbf{X}^*, r_0^*) = \psi^*$. □

VI. Numerical Results

In this section, we evaluate the performance of our proposed algorithms based on the parameters referred in [39] and [40]. Specifically, we consider the N810 device with a CPU frequency of 400×10^6 cycles/sec and computational power 0.8W, so ρ is computed as $\frac{0.8}{(400\times10^6)^3} = 1.25\times10^{-26}$ (J/cyc). In the elastic CPU frequency case, the MD's CPU frequency range is from $r_{\min} = 200\times10^6$ to $r_{\max} = 800\times10^6$ cycles/sec. Moreover, P^{Tx} and P^{Rx} are set to be 1.258W and 1.181W, respectively. To show the relationship between the number of computational cycles and the input bits, we choose Gzip as the application where $w_i = \kappa\alpha_i$. According to [39, Table 3], $\kappa = 330$ cycles/byte. The output data size β_i is equal to 20% of the input data size α_i, which can vary during the simulation. The AP configurations as well as the number of APs are design parameters which are modified depending on the purposes of simulations. The uplink gain of each channel between an AP and the MD h_{ik}^{UL} is assumed to be equal to its downlink channel gain h_{ik}^{DL}. We control the distribution of h_{ik}^{UL} and h_{ik}^{DL} in such a way that the realizations of C_k^{UL} and C_k^{DL} are in some desirable ranges (at 95% confidence level) for ease of investigation. The data rate of each link is controlled in the range $[10, 20]$ Mbps for each channel realization, unless stated otherwise. We set the parameter μ in the sigmoid function as 10, and the weights are related by $\lambda_t = 1 - \lambda_e$. All simulation results are obtained by averaging over 500 channel realizations. Our simulations are run on Matlab using a PC with Intel Xeon CPU E5-1620 @ 3.6 Ghz processor. We summarize our default simulation parameters in Table II, if not specified.

TABLE II
Default Parameter Setup

Parameter	Value
P^{Tx}	1.258W
P^{Rx}	1.181W
β_i	$0.2\alpha_i$
ρ	1.25×10^{-26}
w_i	$330\alpha_i$
$[r_{\min}, r_{\max}]$	$[200\times10^6, 800\times10^6]$ cycles/sec
No. channel realizations	500

A. Algorithm Comparison

Figs. 2 and 3 compare the performance of different algorithms versus the number of tasks N with respect to different data rate regions. The two figures correspond to the algorithms' performance when the MD's CPU frequency is fixed or elastic, respectively. We set the number of tasks $N = 10$, the size of input data $\alpha_i = 0.5$MB, $\forall i \in \mathcal{N}$, the number of Gaussian samples $L = 100$, and $\lambda_e = 0.5$. The number of APs is two, whose service rates are 2×10^9 and 2.2×10^9 cycles/sec. In these figures, the data rates are controlled in three ranges, from 500kbps to 1Mbps, 2Mbps to 10Mbps, and 20Mbps to 50Mbps representing low, average and high data ranges, respectively. In Fig. 2, which is related to the problem $\mathscr{F}$, the proposed SDR-based algorithm is compared with the following baseline methods, namely, (a) Local Processing: all of the tasks will be processed at the MD, (b) Random Assignment: each task is randomly assigned, (c) All to Cloud: all tasks are offloaded to a non-computational AP which forwards the tasks to a cloud server via a fiber link, and (d) Branch and Bound: it provides the optimal solution in exponential running time. To provide more details about All to Cloud, the fiber links between the non-computational AP and the cloud servers have the rate of 1Gbps. The value of cloud server' service rate, denoted by r_{Cloud}, is chosen at 4×10^9 cycles/sec (case 1) and 10×10^9 cycles/sec (case 2) which are approximately 2 times and 5 times faster than an AP's. The data rate of the link between the MD has the same distribution with the

[9]When the MD's main objective is tasks' latency, i.e. $\lambda_e = 0$ and $\lambda_t = 1$, although it is non-trivial to prove, the tasks are partially offloaded when the maximum MD's CPU frequency is not so small in comparison to the service rate of APs.

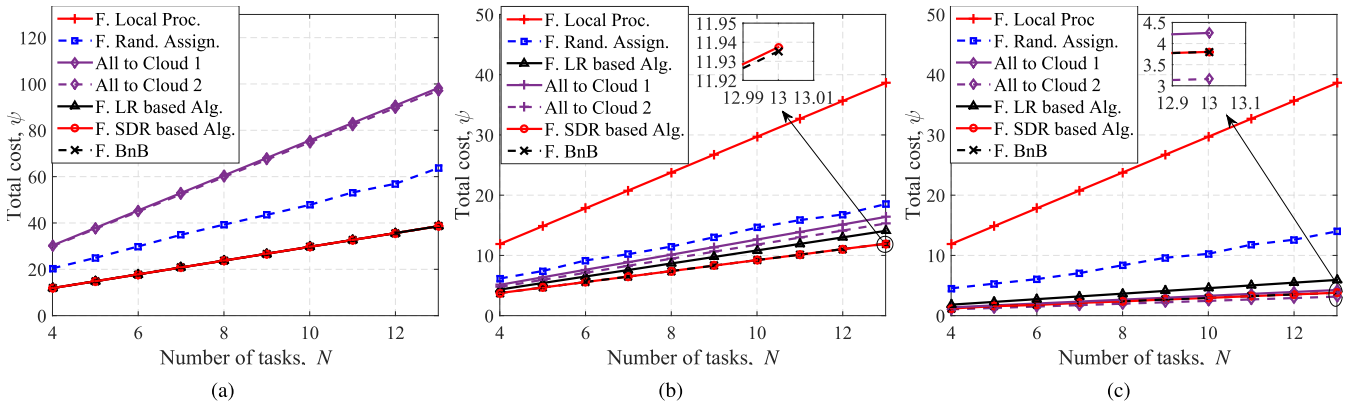


Fig. 2. Algorithm comparison for fixed MD's CPU frequency when high data rates are randomly generated (a) from 500kbps to 1Mbps (b) from 2Mbps to 10Mbps (c) from 20Mbps to 50Mbps.

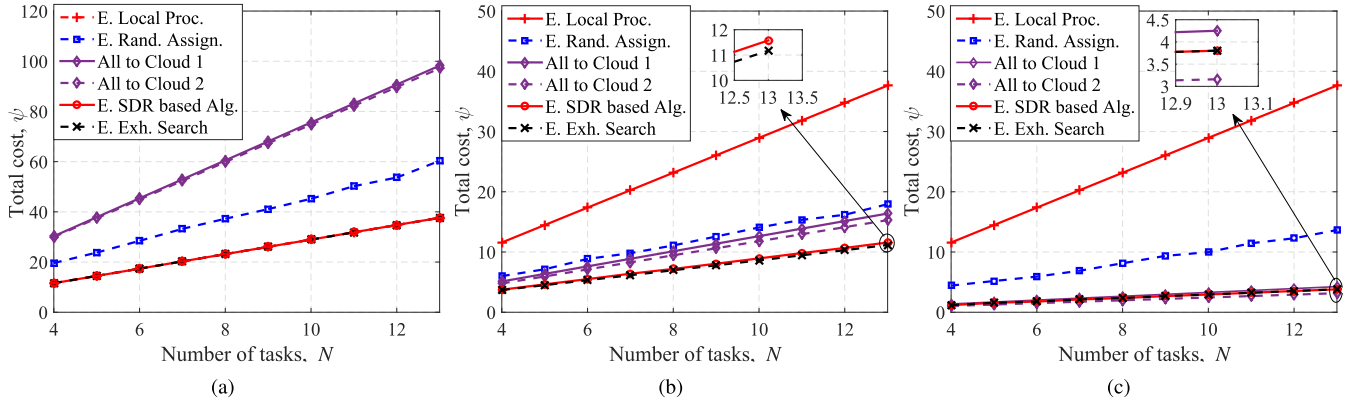


Fig. 3. Algorithm comparison for elastic MD's CPU frequency when high data rates are randomly generated (a) from 500kbps to 1Mbps (b) from 2Mbps to 10Mbps (c) from 20Mbps to 50Mbps.

links to computational APs. In this strategy, the uplink and downlink latencies are computed as $T^{\text{UL}}_{\text{Cloud}} = \frac{\sum_{i\in\mathcal{N}} \alpha_i}{C^{\text{UL}}_{\text{NC}}}$ and $T^{\text{DL}}_{\text{Cloud}} = \frac{\sum_{i\in\mathcal{N}} \beta_i}{C^{\text{DL}}_{\text{NC}}}$, respectively, where the subscript "NC" is for "non-computational". The processing latency is computed as $T^{\text{Comp}}_{\text{Cloud}} = \frac{\sum_{i\in\mathcal{N}} w_i}{r_{\text{Cloud}}}$. Since the cloud server is far from the MD, the propagation delay has to be taken into account. Hence, the total latency is computed as $T = T^{\text{UL}}_{\text{Cloud}} + T^{\text{DL}}_{\text{Cloud}} + T^{\text{Comp}}_{\text{Cloud}} + 2\times T_{\text{prop}}$, where T_{prop} is the propagation delay. We assume that our mobile device is in Singapore and the desired cloud server in Los Angeles. By pinging from our computer to UCLA's website, we get the mean round trip time (RTT) $\approx$ 0.2 s. Thus, we set the propagation delay at 0.1s. Similarly, in Fig. 3, which is related to the problem $\mathscr{E}$, the SDR-based algorithm is compared with (a) Local Processing with Frequency Scaling: all tasks are locally processed and the frequency is optimal to the task allocation matrix, (b) Random Assignment with Frequency Scaling: each task is randomly assigned and the optimal CPU frequency of MD is computed with each random task allocation matrix, and (c) All to Cloud and (d) Exhaustive Search. From both figures, the SDR-based algorithms achieve near optimal performance with a small number of Gaussian samples. When APs' data rates are average and high as in Figs. 2b, 2c, 3b, and 3c, Random Assignment is better than Local Processing, which means offloading is beneficial. For the proposed LR-based algorithm, its cost function under average and high data rates is higher than the SDR-based algorithm's for fixed CPU frequency. However, the LR-based algorithm is still better than Random Assignment. Meanwhile, when APs' data rates are low, the optimal offloading decision is Local Processing. This result is also achieved by the LR-based algorithm and SDR-based algorithm. In Fig. 2, although the total cost of LR-based algorithm is higher than or equal to the total cost of SDR as aforementioned, the algorithm's total cost is acceptable with a lower time complexity (of $O\left(\nu^{3.5}K^2\right)$) compared to SDR-based algorithm (of $O(m^4n^{1/2}\log(1/\epsilon) + LNM)$). In Figs. 2b and 2c, when $N = 10$, the ratios of the total cost between LR-based algorithm and SRD-based algorithm $\frac{\psi^{\text{LR}}}{\psi^{\text{SDR}}}$ are $\frac{10.83}{9.225} = 1.174$ and $\frac{4.563}{2.946} = 1.548$ for average and high data rates, respectively; while the ratios between Local Processing and Random Assignment with SDR-based algorithm are $\frac{29.7}{9.225} = 3.219$ and $\frac{14.6}{9.225} = 1.582$, respectively for average rates; and $\frac{29.7}{2.946} = 10.081$ and $\frac{10.6}{2.946} = 3.598$ for high data rates, respectively. As such, the LR-based algorithm can be a viable alternative to the SDR-based algorithm. Regarding the All to Cloud's performance, when the data rate is high, the cost of All to Cloud are approximately equal to the optimal cost of the proposed methods for MEC. However, as the data rates decrease, the total cost of All to Cloud strategy increases because of higher transmission delay, making it more desirable to utilize the proposed schemes. The performance of All to Cloud is even worse than Local

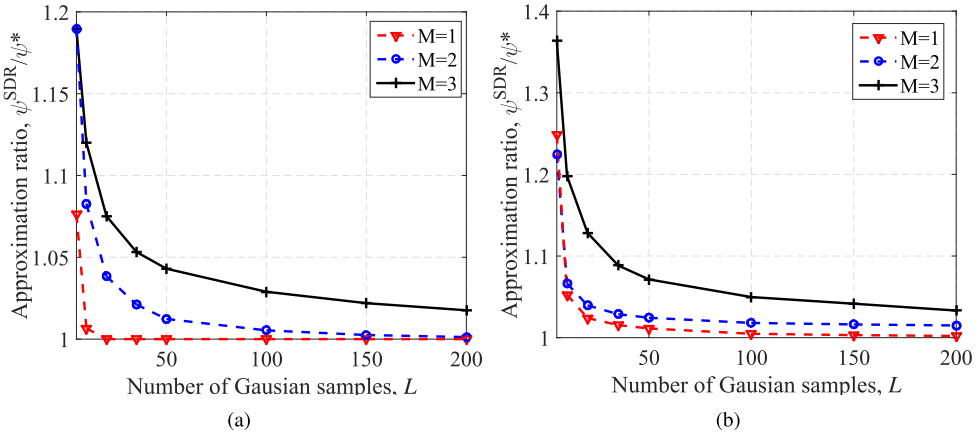


Fig. 4. The approximation ratios of the SDR-based algorithms versus the number of Gaussian samples L (a) for the problem $\mathscr{F}$, and (b) for the problem $\mathscr{E}$.

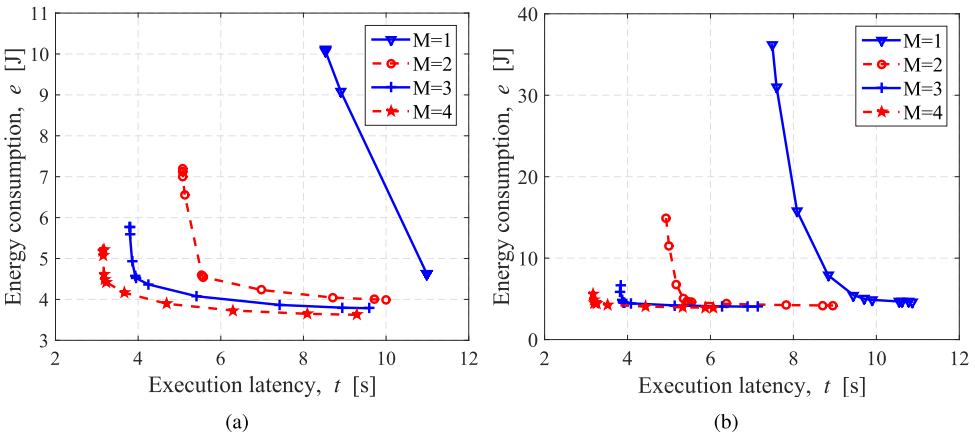


Fig. 5. The trade-offs between MD's energy consumption and total tasks' execution latency w.r.t. different number of APs M (a) for the problem $\mathscr{F}$, and (b) for the problem $\mathscr{E}$.

Processing when the data rates are low. Comparing with Fig. 2, the elastic CPU framework in Fig. 3 shows better or at least equal performance than the fixed one. For example, in Figs. 2b and 3b, when $N = 10$, the total costs of Local Processing of fixed and elastic CPU cases are 29.7 and 28.9, respectively. Also, the total cost of SDR-based algorithms for the fixed and elastic ones are 9.225 and 8.937, respectively. Detailed comparisons between fixed CPU frequency and elastic CPU frequency will be provided in Section VI-C. Figs. 4a and 4b show the ratio of the final cost of SDR algorithms to the optimal cost from BnB or exhaustive search. We set the number of tasks $N = 10$, the value of $\alpha_i \in \{0.1, 0.2, \ldots, 1\}$ MB, and $\lambda_e = 0.5$. The computational service rate of each AP is $r_k = 2 \times 10^9$ (cycles/sec). From these figures, we observe that the higher the value of L is, the closer to optimum the performance is. However, the approximation ratios begin to slow down considerably about after $L = 100$, which means that beyond this point we have to use a much higher L in order to achieve a marginal performance gain. For example, in Fig. 4a, when $M = 3$, to decrease the approximation ratio from about 1.04 to about 1.03, L needs to increase from 50 to 100; meanwhile, to decrease the approximation ratio from about 1.03 to about 1.02, L needs to increase from 100 to 200. Due to such trade-offs, it is reasonable to set $L = 100$ for the rest simulations. Second, we observe that for the same L, the performance of algorithms could decline as the number of APs M increases. As shown by Figs. 4a and 4b, the SDR algorithms' approximation ratios decline faster for problem $\mathscr{F}$ than for $\mathscr{E}$, which may be due to the fact that $\mathscr{F}$ has less variables and constraints than $\mathscr{E}$.

B. Multiple APs

In Figs. 5a and 5b, we investigate the trade-off between the MD's energy consumption and tasks' execution latency. We set the number of tasks $N = 10$, the value of $\alpha_i \in \{0.1, 0.2, \ldots, 1\}$ MB. The computational service rate of each AP is 2×10^9 (cycles/sec). The value of λ_e is chosen such that $\frac{\lambda_e}{\lambda_t} = 10^q$, where $q \in [-2, 2]$. The value of q increases from -2 to 2 with the step size 0.4. For each λ_e, we find out the near optimal energy consumption and tasks' execution latency by using the SDR-based algorithms. These points correspond to the several markers on each of the energy-latency curve on Fig. 4, with $q = -2$ to the furthest left and $q = 2$ to the right. From these figures, there are significant performance gains in terms of energy and latency reduction for both fixed CPU frequency case and elastic CPU frequency case when M increases, as the energy-latency curve is shifted down and to the left. The results confirm that it is beneficial for the MD to allocate tasks to more than one AP. However, the performance

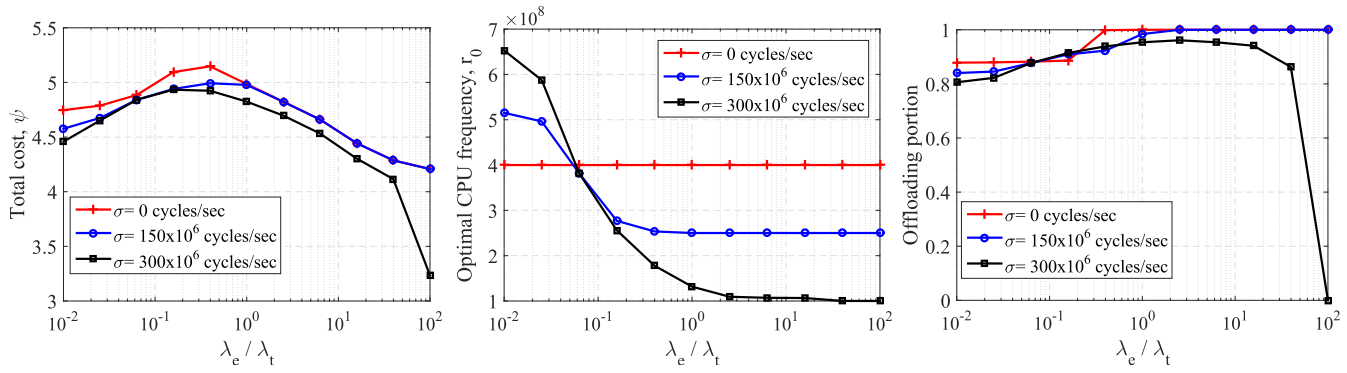


Fig. 6. Impacts of the ratio of λ_e to λ_t on (a) the total cost, (b) the optimal frequency, and (c) the offloading portion.

gain seems to be marginally diminishing as the number of APs M gets larger.

C. Comparison Between Fixed and Elastic CPU Frequency Cases

In Figs. 6a, 6b, and 6c, we investigate the effect of different MD's CPU frequency ranges. The parameter settings are similar to those of Section VI-B; except that the number of APs is two. We now consider the CPU frequency range of $400 \times 10^6 \pm \sigma$ cycles/sec, where $\sigma \in \{0, 150 \times 10^6, 300 \times 10^6\}$ cycles/sec. The case $\sigma = 0$ corresponds to the fixed MD's CPU frequency case, and σ can be viewed as the elasticity of the MD's CPU frequency. In Fig. 6a, when the MD's CPU frequency elasticity increases, the MD's total cost decreases for all ratios of $\frac{\lambda_e}{\lambda_t}$. This agrees with Proposition 3, which states that widening the frequency range can reduce the optimal objective value. Furthermore, we notice a similar trend for all three cases. For instance, with $\sigma = 0$, the total cost goes up when the ratio of $\frac{\lambda_e}{\lambda_t}$ increases from 10^{-2} to $10^{-0.8}$. It then falls down, once reaching its peak at $\frac{\lambda_e}{\lambda_t} \approx 0.3$. When $\frac{\lambda_e}{\lambda_t}$ is small, the MD likely only minimizes tasks' latency. Meanwhile, when $\frac{\lambda_e}{\lambda_t}$ is large, the MD likely only minimizes energy consumption. As $\frac{\lambda_e}{\lambda_t} > 1$, the total costs of $\sigma = 0$ and $\sigma = 150 \times 10^6$ cycles/sec coincide as seen in Fig. 6a. The reason for that, as observed in Fig. 6c, is because the optimal decisions of both scenarios are to offload all tasks to APs where $\frac{\lambda_e}{\lambda_t} > 1$. Fig. 6b shows the optimal CPU frequency value versus the ratio of $\frac{\lambda_e}{\lambda_t}$. Starting from very high value, the MD's CPU frequency decreases until it reaches $r_{\min}$ as the ratio $\frac{\lambda_e}{\lambda_t}$ increases. For example, when $\sigma = 150 \times 10^6$ cycles/sec, $r_{\min} = 250 \times 10^6$, the optimal frequency starts from more than 500×10^6 cycles/sec, then decreases to 250×10^6 as shown in Fig. 6b. In Fig. 6c, we investigate the offloading portion versus the ratio of $\frac{\lambda_e}{\lambda_t}$. Because each task has a different number of CPU cycles to be processed, the offloading portion is defined as the ratio of the number of CPU cycles which are offloaded, to the total number of CPU cycles the MD needs to process all tasks. In Fig. 6c, when the ratio of $\frac{\lambda_e}{\lambda_t}$ is small, the MD partially offloads its CPU cycles. When $\frac{\lambda_e}{\lambda_t}$ increases, more cycles are offloaded. However, when $\frac{\lambda_e}{\lambda_t}$ reaches 100, all cycles are local processed when $\sigma = 300 \times 10^6$ cycles/sec, while the MD offloads all cycles in the two other cases. The results can be explained by Proposition 3. Since $\frac{\lambda_e}{\lambda_t} \gg 1$, it is approximately analogous to the case of MD prioritizing energy consumption. Due to the parameter settings, $r^{\text{off\_a}} = r^{\text{off\_b}} \approx 150 \times 10^6$ cycles/sec. Hence, $r_{\min} > r^{\text{off\_a}}$ when $\sigma = 0$ and 150×10^6 cycles/sec, and $r_{\min} < r^{\text{off\_b}}$ when $\sigma = 300 \times 10^6$ cycles/sec.

VII. Conclusion

In this paper, we proposed a computational offloading framework where a single MD can offload tasks to multiple APs. We aimed to minimize the total cost which includes both the MD's energy consumption and total tasks' execution latency by coupling task allocation decisions and frequency scaling. We also considered both fixed CPU frequency and elastic CPU frequency at the MD. In both cases, due to the overall optimization problems being NP-hard, we proposed SDR-based algorithms which can be computed in polynomial time to efficiently find the solutions. Our simulation results showed that the proposed SDR approach can achieve near optimal performance and outperform some referenced schemes such as Local Processing or Random Assignment. Since there are more degrees of freedom, in term of allocation decision and CPU frequency, we showed that the MD can obtain reduction in its energy consumption and its tasks' execution latency. Finally, we investigated how the CPU frequency range, as well as the APs' data and service rates can influence the task allocation decision. In this work, we currently assume APs process all tasks offloaded by the MD, which may not hold if there are multiple MDs offloading to the same AP, causing some tasks to be dropped or experience extra latency. Therefore, this scenario will be a good avenue for future research.

Appendix
Proof of Proposition 2

Given a task allocation matrix $\bar{\mathbf{X}}$, all the quantities $\max_{k \in \mathcal{M}\setminus\{0\}} \left(\sum_{i \in \mathcal{N}} x_{ik} D_{ik}\right)$, $\sum_{i \in \mathcal{N}} x_{i0} w_i$, $P^{\text{Tx}} \sum_{k \in \mathcal{M}\setminus\{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\text{UL}}$, and $P^{\text{Rx}} \sum_{k \in \mathcal{M}\setminus\{0\}} \sum_{i \in \mathcal{N}} x_{ik} d_{ik}^{\text{DL}}$ are constants which are denoted by $\mathscr{C}_A$, $\mathscr{C}_B$, $\mathscr{C}_C$, and $\mathscr{C}_D$, respectively; and the objective function depends only on r_0. Denote U the point of intersection of the horizontal line $\phi(r_0) = \mathscr{C}_A$ and the curve $\varphi(r_0) = \frac{\mathscr{C}_B}{r_0}$, so $r_U = \frac{\mathscr{C}_B}{\mathscr{C}_A}$.

When $x_{i0} = 0, \forall i \in \mathcal{N}$, all tasks are offloaded. In this case, the MD's CPU frequency r_0 does not affect the total cost, so $\bar{r}_0$ is arbitrary within the range $[r_{\min}, r_{\max}]$.

When $\exists\ x_{i0} \neq 0$,

1) If $r_U < r_{\min}$, $\frac{\mathscr{C}_B}{r_0} < \mathscr{C}_A, \forall r_0 \in [r_{\min}, r_{\max}]$. The objective function of $\mathscr{E}1$ becomes

$$\psi\left(\bar{\mathbf{X}}, r_0\right) = \lambda_e\left(\rho\mathscr{C}_B r_0^2 + \mathscr{C}_C + \mathscr{C}_D\right) + \lambda_t\mathscr{C}_A.$$

The first order derivative is derived as follows

$$\frac{\partial\psi}{\partial r_0} = 2\lambda_e\rho\mathscr{C}_B r_0.$$

Since $\frac{\partial\psi}{\partial r_0} > 0, \forall r_0 \in [r_{\min}, r_{\max}]$, $\psi\left(\bar{\mathbf{X}}, r_0\right)$ monotonically increases within the range $[r_{\min},\ r_{\max}]$. Hence, $\bar{r}_0 = r_{\min}$.

2) If $r_U > r_{\max}$, $\frac{\mathscr{C}_B}{r_0} > \mathscr{C}_A, \forall r_0 \in [r_{\min}, r_{\max}]$. The objective function becomes

$$\psi\left(\bar{\mathbf{X}}, r_0\right) = \lambda_e\left(\rho\mathscr{C}_B r_0^2 + \mathscr{C}_C + \mathscr{C}_D\right) + \lambda_t\frac{\mathscr{C}_B}{r_0}.$$

The first order derivative is derived as follows

$$\frac{\partial\psi}{\partial r_0} = 2\lambda_e\rho\mathscr{C}_B r_0 - \frac{\lambda_t\mathscr{C}_B}{r_0^2}.$$

Thus, $\frac{\partial\psi}{\partial r_0} = 0$ if and only if $r_0 = \left(\frac{\lambda_t}{2\lambda_e\rho}\right)^{\frac{1}{3}}$. For notational convenience, we denote $r_\lambda = \left(\frac{\lambda_t}{2\lambda_e\rho}\right)^{\frac{1}{3}}$.

   a) When $r_\lambda < r_{\min}$, $\frac{\partial\psi}{\partial r_0} > 0, \forall r_0 \in [r_{\min}, r_{\max}]$, we have $\bar{r}_0 = r_{\min}$.
   b) When $r_\lambda \in [r_{\min}, r_{\max}]$, we have $\bar{r}_0 = r_\lambda$.
   c) When $r_\lambda > r_{\max}$, $\frac{\partial\psi}{\partial r_0} < 0, \forall r_0 \in [r_{\min}, r_{\max}]$, we have $\bar{r}_0 = r_{\max}$.

   In short, if $r_U > r_{\max}$,

$$\bar{r}_0 = \begin{cases} r_{\min}, & \text{when } r_\lambda < r_{\min}, \\ r_\lambda, & \text{when } r_\lambda \in [r_{\min},\ r_{\max}], \\ r_{\max}, & \text{when } r_\lambda > r_{\max}, \end{cases}$$

3) If $r_U \in [r_{\min}, r_{\max}]$, we divide the domain into two sub-domains: $[r_{\min}, r_U]$ and $[r_U, r_{\max}]$.

   a) Assuming that $\bar{r}_0$ is in the sub-domain $[r_U, r_{\max}]$, it follows that $\frac{\mathscr{C}_B}{r_0} < \mathscr{C}_A$. Like when $r_U < r_{\min}$, $\psi\left(\bar{\mathbf{X}}, r_0\right)$ is monotonically increasing within the sub-domain $[r_U, r_{\max}]$. Thus, $\bar{r}_0 = r_U$.
   b) Assuming that $\bar{r}_0$ is in the sub-domain $[r_{\min}, r_U]$, it follows that
      - When $r_\lambda < r_{\min}$, $\frac{\partial\psi}{\partial r_0} > 0, \forall r_0 \in [r_{\min}, r_U]$, and we have $\bar{r}_0 = r_{\min}$.
      - When $r_\lambda \in [r_{\min}, r_U]$, we have $\bar{r}_0 = r_\lambda$.
      - When $r_\lambda > r_t$, $\frac{\partial\psi}{\partial r_0} < 0, \forall r_0 \in [r_{\min}, r_U]$, and we have $\bar{r}_0 = r_U$.

   Hence, if $r_U \in [r_{\min}, r_{\max}]$,

$$\bar{r}_0 = \begin{cases} \underset{r_0 \in \{r_U, r_{\min}\}}{\arg\min}\ \psi\left(\bar{\mathbf{X}}, r_0\right), & \text{when } r_\lambda < r_{\min}, \\ \underset{r_0 \in \{r_U, r_\lambda\}}{\arg\min}\ \psi\left(\bar{\mathbf{X}}, r_0\right), & \text{when } r_\lambda \in [r_{\min}, r_U], \\ r_U, & \text{when } r_\lambda > r_U. \end{cases}$$

□

REFERENCES

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

[2] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2014.

[3] E. Cuervo *et al.*, "Maui: Making smartphones last longer with code offload," in *Proc. ACM MobiSys*, San Francisco, CA, USA, Jun. 2010, pp. 49–62.

[4] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[5] T. Q. Quek, G. D. L. Roche, I. Güvenç, and M. Kountouris, *Small Cell Networks Deployment Phy Techniques and Resource Management*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[6] 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.

[7] IBM. (2013). *IBM and Nokia Siemens Networks Announce World's First Mobile Edge Computing Platform*. [Online]. Available: http://www-03.ibm.com/press/us/ en/pressrelease/40490.wss

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. (2017). "Mobile edge computing: Survey and research outlook." [Online]. Available: https://arxiv.org/abs/1701.01090

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, to be published, doi: 10.1109/COMST.2017.2682318.

[10] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[11] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[12] C. You, K. Huang, H. Chae, and B.-H. Kim. (2016). "Energy-efficient resource allocation for mobile-edge computation offloading." [Online]. Available: https://arxiv.org/abs/1605.08518.

[13] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[14] 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.

[15] M. E. T. Gerards, J. L. Hurink, and J. Kuper, "On the interplay between global DVFS and scheduling tasks with precedence constraints," *IEEE Trans. Comput.*, vol. 64, no. 6, pp. 1742–1754, Jun. 2015.

[16] M.-H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE SPAWC*, Stockholm, Sweden, Jun./Jul. 2015, pp. 186–190.

[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, Jun. 2012.

[18] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[19] Y. Wang *et al.*, "Energy-Optimal partial computation offloading using dynamic voltage scaling," in *Proc. IEEE ICCW*, Jun. 2015, pp. 2695–2700.

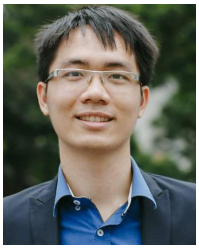
[20] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[21] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proc. Int. Conf. Compil., Archit., Synth. Embedded Syst.*, Atlanta, GA, USA, Nov. 2001, pp. 238–246.

[22] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[23] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *Proc. Int. Conf. Parallel Distrib. Syst.*, Hsinchu, Taiwan, Dec. 2007, pp. 1–8.

[24] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. ACM MobiSys*, Bethesda, MD, USA, Jun. 2011, pp. 43–56.
[25] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *IEEE Pervasive Comput.*, vol. 3, no. 3, pp. 66–73, Jul. 2004.
[26] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.
[27] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Netw.*, vol. 31, no. 1, pp. 52–58, Jan. 2017.
[28] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. ACM Workshop Mobile Big Data*, Hangzhou, China, 2015, pp. 37–42.
[29] M. Ehrgott, *Multicriteria Optimization*. New York, NY, USA: Springer, 2006.
[30] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.
[31] M. R. Garey and D. S. Johnson, "'Strong' NP-completeness results: Motivation, examples, and implications," *J. ACM*, vol. 25, no. 3, pp. 499–508, 1978.
[32] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6. Belmont, MA, USA: Athena Scientific, 1997.
[33] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, Dec. 1984.
[34] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, PA, USA: SIAM, 1994.
[35] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.
[36] F. Zhang, *Matrix Theory: Basic Results and Techniques*. Berlin, Germany: Springer-Verlag, 1999.
[37] D. Bertsimas and Y. Ye, "Semidefinite relaxations, multivariate normal distributions, and order statistics," in *Handbook of Combinatorial Optimization*. Boston, MA, USA: Kluwer, 1998.
[38] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, "Approximation bounds for quadratic optimization with homogeneous quadratic constraints," *SIAM J. Optim.*, vol. 18, no. 1, pp. 1–28, Feb. 2007.
[39] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, Boston, MA, USA, Jun. 2010, pp. 4–11.
[40] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. 1st Int. Conf. Energy-Efficient Comput. Netw.*, Passau, Germany, 2010, pp. 75–84.



Thinh Quang Dinh (S'17) received the B.Eng. degree (Hons.) in electrical and electronic engineering (specializing in telecommunications), Ho Chi Minh City University of Technology, Vietnam, in 2013. He is currently pursuing the Ph.D. degree with the Singapore University of Technology and Design (SUTD) under the SUTD President's Graduate Fellowship. His main research interests are the mathematical application of optimization, machine learning and game theory to communication, and networking and resource allocation problems in mobile edge computing.

Jianhua Tang (S'11–M'15) received the B.E. degree in communication engineering from Northeastern University, China, in 2010, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2015. He was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design from 2015 to 2016. He is currently with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, China. He is also a Research Assistant Professor with the Department of Electrical and Computer Engineering, Seoul National University, South Korea. His current research interests include cloud computing, contentcentric network, and cloud radio access network.

Quang Duy La (S'09–M'14) received the B.Eng. degree (Hons.) and the Ph.D. degree, under the Nanyang Presidents Research Scholarship, in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2008 and 2013, respectively. He is currently a Post-Doctoral Research Fellow with Temasek Laboratories, Singapore University of Technology and Design. He has participated in research projects on radio resource allocation, smart grid, body area networks, and network security. He is a Co-Author of the monograph *Potential Game Theory: Applications in Radio Resource Allocation* (Springer, 2016). His research interests lie in game-theoretic and distributed algorithms for next generation communications systems and big data analytics for the IoT.

Tony Q. S. Quek (S'98–M'08–SM'12) received the B.E. and M.E. degrees in electrical and electronics engineering from the Tokyo Institute of Technology and the Ph.D. degree in electrical engineering and computer science, MIT. He is currently a tenured Associate Professor with the Singapore University of Technology and Design (SUTD). He also serves as the Associate Head of ISTD Pillar and the Deputy Director of SUTD-ZJU IDEA. His main research interests are the application of mathematical, optimization, and statistical theories to communication, networking, signal processing, and resource allocation problems, and his specific current research topics include heterogeneous networks, wireless security, Internet-of-Things, and big data processing.

He is a Co-Author of the book *Small Cell Networks: Deployment, PHY Techniques, and Resource Allocation* (Cambridge University Press, 2013) and *Cloud Radio Access Networks: Principles, Technologies, and Applications* (Cambridge University Press, 2017). He has been actively involved in organizing and chairing sessions, and has served as a member of the Technical Program Committee and symposium chairs in a number of international conferences. He is currently an elected member of the IEEE Signal Processing Society SPCOM Technical Committee. He is serving as the Workshop Chair of the IEEE Globecom in 2017, the Tutorial Chair of the IEEE ICCC in 2017, and the Special Session Chair of the IEEE SPAWC in 2017. He was an Executive Editorial Committee Member of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS, and the IEEE WIRELESS COMMUNICATIONS LETTERS.

Dr. Quek was honored with the 2008 Philip Yeo Prize for Outstanding Achievement in Research, the IEEE Globecom 2010 Best Paper Award, the 2012 IEEE William R. Bennett Prize, the IEEE SPAWC 2013 Best Student Paper Award, the IEEE WCSP 2014 Best Paper Award, the 2015 SUTD Outstanding Education Awards–Excellence in Research, the 2016 Thomson Reuters Highly Cited Researcher, and the 2016 IEEE Signal Processing Society Young Author Best Paper Award.