Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling

Yanting Wang, Min Sheng, *Senior Member, IEEE*, Xijun Wang, *Member, IEEE*, Liang Wang, and Jiandong Li, *Senior Member, IEEE*

***Abstract*— The incorporation of dynamic voltage scaling technology into computation offloading offers more flexibilities for mobile edge computing. In this paper, we investigate partial computation offloading by jointly optimizing the computational speed of smart mobile device (SMD), transmit power of SMD, and offloading ratio with two system design objectives: energy consumption of SMD minimization (ECM) and latency of application execution minimization (LM). Considering the case that the SMD is served by a single cloud server, we formulate both the ECM problem and the LM problem as nonconvex problems. To tackle the ECM problem, we recast it as a convex one with the variable substitution technique and obtain its optimal solution. To address the nonconvex and nonsmooth LM problem, we propose a locally optimal algorithm with the univariate search technique. Furthermore, we extend the scenario to a multiple cloud servers system, where the SMD could offload its computation to a set of cloud servers. In this scenario, we obtain the optimal computation distribution among cloud servers in closed form for the ECM and LM problems. Finally, extensive simulations demonstrate that our proposed algorithms can significantly reduce the energy consumption and shorten the latency with respect to the existing offloading schemes.**

***Index Terms*— Partial computation offloading, dynamic voltage scaling, mobile-edge computing, collaboration between communication and computation resources.**

I. Introduction

With smart mobile devices (SMDs) gaining enormous popularity, users expect to run desktop-level applications such as speech recognition (e.g., Siri), online game, and reality augmentation on SMDs anywhere. However, it is very challenging for the SMD to support these resource-hungry applications due to its limited resources of energy, computation, and storage. To address this limitation, computation offloading provides a promising technique to broaden the capability of SMDs, which migrates part or all of the data processing of mobile applications from resource-limited mobile devices to powerful computing platforms in network edge [1]–[3].

Manuscript received October 26, 2015; revised April 12, 2016 and July 1, 2016; accepted August 5, 2016. Date of publication August 11, 2016; date of current version October 14, 2016. This work is supported by the National Natural Science Foundation of China (61231008, 61301176, and 91338114), 863 project (No.2014AA01A701), and 111 Project (B08038). The associate editor coordinating the review of this paper and approving it for publication was V. Wong *(Corresponding author: Min Sheng)*.

Y. Wang, M. Sheng, X. Wang, and J. Li are with the State Key Laboratory of Integrated Service Networks, Institute of Information Science, Xidian University, Xi'an, 710071, China (e-mail: yantingwangxd@gmail.com; msheng@mail.xidian.edu.cn; xjwang22@gmail.com; jdli@mail.xidian.edu.cn).

L. Wang is with the School of Computer Science, Shannxi Normal University, Xi'an, 710119, China (e-mail:wangliang@snnu.edu.cn).

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.2016.2599530

Several architectures for mobile computation offloading are proposed, e.g., MAUI, CloneCloud, ThinkAir, CONCERT, and femto-cloud proposed in TROPIC [4]–[8]. Particularly, the project TROPIC incorporates computation offloading into heterogeneous networks, where several femto access points (FAPs) equipped with some amount of computation and storage capabilities collaborate to form a femto-cloud. The femto-cloud is an example of mobile-edge computing which aims to provide information technology and cloud-computing capacities within the radio access network to offer a service environment characterized by proximity, low latency, and high rate access [9]. On the other hand, computation offloading schemes are studied in [9]–[17]. Since offloading introduces additional communication overhead, a key technical challenge is how to balance the tradeoff between computation cost and communication cost to support applications with enhanced user experience, such as lower latency and energy consumption.

To improve network performance, references [9]–[12] focus on full offloading strategies to minimize latency or energy consumption. Compared with full offloading, partial offloading is more suitable for the application with more stringent latency requirement, since it takes advantage of parallelism between the SMD and cloud. Additionally, it is more reasonable to offload partial rather than entirety of application since bandwidth is limited in wireless networks [2]. Therefore, many works are devoted to partial offloading [13]–[17]. Particularly, the authors in [13] minimize the energy consumption of SMD by jointly optimizing the uplink time, downlink time, and processed data size at the SMD and cloud. A dynamic offloading algorithm based on Lyapunov optimization is proposed in [14] to achieve energy saving. In [15], the transmit power and constellation size are jointly optimized to minimize the energy expenditure of SMD under the latency constraint. The authors in [16] jointly study the partitioning of computations and the scheduling of offloaded computations on the cloud resources, with the goal of achieving minimum average completion time for all the users. A framework for partitioning and executing data stream applications is proposed in [17] to achieve maximum speed. However, the aforementioned researches only consider a fixed computational speed of SMD, which is neither energy-optimal nor latency-optimal at the SMD's side.

Dynamic voltage scaling (DVS) is a technique that varies the supply voltage and clock frequency based on the

0090-6778 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

computation load to provide desired performance [18]. Using DVS technology, the SMD could adaptively adjust its computational speed to reduce energy consumption or shorten the computing time. Therefore, the incorporation of DVS technology into computation offloading offers more flexibilities for strategy design. Among the few works focused on this topic, the authors in [19] consider the optimization of computational speed of SMD and the transmission rate under Gilbert-Elliott channel to make offloading decision between local execution and total offloading. Although [19] provides some hints about the impact of computational speed on offloading decision, it does not consider computation partition. Moreover, it ignores the cloud processing time and energy consumption of receiving at the SMD, which limits the scope of application. Note that DVS technology further complicates partial computation offloading decisions by affecting not only the transmit power of SMD but also the offloading ratio. Consequently, the existing offloading designs cannot be directly utilized for partial computation offloading when SMD has the capability of DVS. This motivates the study of this paper.

In this paper, we focus on jointly optimizing communication and computation resources for partial computation offloading using DVS technology. Specifically, we optimize the computational speed of SMD, transmit power of SMD, and offloading ratio to achieve two system goals: energy consumption of SMD minimization (ECM) and latency of application execution minimization (LM). In the scenario where the SMD is severed by a single cloud, we formulate these two problems as nonconvex problems and design two algorithms to solve them. Further, we investigate the ECM problem and LM problem in a multiple cloud servers scenario, where the SMD could offload computation to a set of cloud servers. To our knowledge, we are the first to design the energy-optimal and latency-optimal partial computation offloading strategies when SMD has the capability of DVS. Part of the results have been presented in our previous work, which investigates the ECM problem in a single-cloud scenario and obtains its locally optimal solution by the alternating minimization method [20]. The main contributions of this paper are as follows:

- We propose an energy-optimal partial computation offloading (EPCO) algorithm to recast the nonconvex ECM problem into a convex one based on the variable substitution technique and obtain the optimal computational speed of SMD and optimal transmit power of SMD in closed-form.
- Through analyzing the optimal solutions, we derive the necessary and sufficient condition under which local execution is optimal. Besides, we analyze the optimality of total offloading and reach a conclusion that total offloading cannot be optimal when DVS is utilized.
- To handle the nonconvex and nonsmooth LM problem, we propose a locally optimal algorithm based on the univariate search technique [21].
- We further study the ECM problem and LM problem in a multiple cloud servers scenario, where the optimal computation distribution among cloud servers and the optimal user association are obtained.

TABLE I

PARTIAL OFFLOADING PARAMETERS

Symbol	Description
$W_{\mathrm{U}}/W_{\mathrm{D}}$	uplink/downlink channel bandwidth
d	distance from the SMD to its serving FAP
h_1/h_2	uplink/downlink channel fading coefficients
N_0	white Guassian noise power
P_t	transmit power of SMD
P_0	static power consumption of SMD
k_t	efficient factor of power amplifier of SMD
P_r	receive power consumption of SMD
P_{F}	transmit power of the serving FAP
$P_{t_{\max}}$	maximum transmit power of SMD
f_l	computational speed of SMD
f_c	computational speed of cloud
$f_{l_{\max}}$	maximum computational speed of SMD
k	a coefficient depending on chip architecture (using for modeling computation energy consumption)
I	amount of computation input data bits
$L_{\max}$	application-dependent latency requirement
$E_{\max}$	maximum energy supplied by SMD
C	number of cycles needed for the application
β_1	a coefficient accounting for the overhead in uplink transmission
β_2	a coefficient jointly accounting for the overhead in downlink transmission and the ratio of output to input bits offloaded to the cloud
λ	ratio of locally executed amount of bits to the total input data bits
$R_{\mathrm{U}}/R_{\mathrm{D}}$	uplink/downlink rate
$t_{\mathrm{U}}/t_{\mathrm{D}}$	uplink/downlink transmit delay
τ_c	execution time in the cloud
t_l/E_l	time/energy consumption for local execution part
t_c/E_c	time/energy consumption for offloading part
L/E	total time/energy consumption of SMD
N	number of FAPs in femto-cloud
L_c	cloud latency in multi-FAP scenario
t_{cm}	time for offloading part in multi-FAP scenario
f_n	computational speed of FAP n
w_n	allocated computation bits to FAP n
$\delta_{Tx,bh}(n)$	one way communication latency from the associated FAP to FAP n
$\delta^r_{Tx,bh}(n)$	one way communication latency from FAP n to the associated FAP

The rest of this paper is organized as follows: Section II presents the system model and problem formulation. In Section III, we propose an algorithm to solve the ECM problem and do some extension. The LM problem is studied in Section IV. Finally, extensive simulation results and conclusions are provided in Sections V and VI, respectively.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first introduce the system architecture design and application model. Then, the energy consumption and latency model are presented. Finally, we formulate two

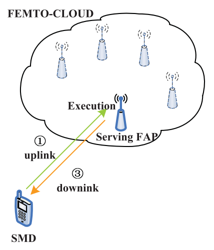


Fig. 1. An illustration of computation offloading.

optimization problems. The parameters used in the following are listed in Table I.

A. System Architecture Design

Since cellular network can manage communication but not computation, a new computation control entity is needed in the computation offloading architecture, which is called small cell cloud manager (SCM) in TROPIC [8]. The SCM consists of three modules related to computation offloading, i.e., offloading module, operator module, and optimization module. Through collaboration among these modules, the SCM manages computing-related activities in small cell cloud, i.e., femto-cloud in this paper. To put it simply, the SCM obtains the information of both the small cell and the SMD, e.g., the instantaneous channel information, cloud computational resource availability, parameters of SMD, and QoS requirements of SMD through operator module. Then, the SCM executes the optimization mechanisms in optimization module. Finally, the offloading decision is delivered to the SMD and the femto-cloud through offloading module.[1]

Based on this mobile computation offloading architecture, we consider a set of cloud-enhanced FAPs collaborate to form a femto-cloud and provide the SMD with proximity access to the cloud computing services [8], which is shown in Fig.1. When the SMD has a computation-intensive application to process, it sends resource request to the SCM. The SCM judiciously determine whether to offload the application or not and which portion is needed to be offloaded to the cloud. Once the SCM decides to offload, three phases need to be performed sequentially. Firstly, the SMD sends data to the cloud through uplink channel. Secondly, the cloud executes the offloaded data. Finally, the results are sent back to the SMD through downlink channel.[2] With the goal of saving SMD's energy consumption, the decision of offloading depends on several aspects, such as computational speed of SMD, transmit power of SMD, channel condition, etc. In order to investigate the effect of these factors on offloading decisions, we consider the case that the nearest serving FAP is the only server allowed to execute the offloaded part [13]. Therefore, we do not distinguish FAP, cloud server, and cloud hereinafter.

[1]Actually, the decision can be made in the SMD, which is an alternative in TROPIC. Moreover, there is little change in offloading procedure except that the SMD should be informed of some information, e.g., the computational speed of cloud and the channel conditions.

[2]Note that if the SMD has the capability of DVS, the operator module in the SCM should collect only one more parameter, i.e., the maximum computational speed of SMD, and the optimization module in the SCM should execute a more complex optimization mechanism. Hence, we can see it only increases the information interaction while has little impact on the existing mobile computation offloading architecture if the SMD has the capability of DVS.

B. Application Model

There are a significant number of applications that can support the cloud service. Focusing on different types of applications, the resource management needs to be tackled in different ways. In general, applications can be classified into the following three major groups [13]:

- Data partitioned oriented applications. The amount of data to be processed is known beforehand for this kind of applications. Besides, we could take advantage of parallelism by processing a portion of total data at the local side and the remaining part at the cloud side concurrently [9], [12], [13], [19], [22]. The typical examples are the virus scan application and the file/figure compression application. Note that a load balancer divides all the files or figures into several independent subsets that can be processed in parallel.
- Code partitioned oriented applications. This type of applications can be divided into several components. Some of the components can be parallel, while some need to be sequential since the output data of some components is the input data of some others. The relationship among these components can be expressed by the component dependency graph (CDG) or call graph (CG) [15], [23], [24]. For different CDGs or CGs, different mathematical formulations should be made.
- Continuous execution applications. It is not known beforehand for how long the application runs. Cloud mobile gaming and other interactive applications belong to this group. One Quality of Service (QoS) metric of this type of applications is the average delay, which includes communication delay, queuing delay, and computing delay [25]–[28].

In this paper, we focus on data partitioned oriented applications. Particularly, the application is abstracted into a profile with two parameters, i.e., $(I, L_{\max})$ [19], where I and $L_{\max}$ denote the amount of computation input data bits and the application-dependent latency requirement, respectively. Following [13] and [20], we model the number of cycles C needed for the application as the number of computation input data multiplied by a factor, i.e., $C = \alpha I$, where α $(\alpha > 0)$ depends on the nature of application, e.g., computational complexity of application. Besides, we define $\lambda\,(0 \leq \lambda \leq 1)$ as the ratio of locally executed amount of bits to the total input data bits. In order to simplify the analysis, we assume full granularity in data partition [13]. In other words, the application could be partitioned into subsets of any size, despite that only several partitions are possible in practice. Accordingly, the optimal solution in this paper could be served as a performance upper-bound of realistic offloading strategies. In other words, the optimal solution obtained in our work should be further quantized [29]. The quantization methods will be provided in Section III-B and Section IV-B.

C. Energy Consumption and Latency Model

1) Local Computing Cost: We model the power consumption of CPU as $P = kf^3$ as in [19], where f and k are the CPU's computational speed and a coefficient depending on chip architecture, respectively. As f is equal of cycles per second, the energy consumption per cycle is thus kf^2. For the SMD, its computation energy consumption can be minimized by optimally configuring computational speed via DVS technology. When the amount of data bits processed at the SMD is λI, the execution time t_l is

$$t_l = \frac{\alpha \lambda I}{f_l}, \tag{1}$$

where f_l is computational speed of SMD. The energy consumption E_l is given by

$$E_l = \alpha \lambda I k f_l^2. \tag{2}$$

2) Offloading Cost: In this work, frequency division duplex (FDD) is considered as the duplex mode. The uplink and downlink channels are assumed to be the frequency-flat block-fading Rayleigh channels, with block length no less than the maximum latency requirement of the application. The path loss between the SMD and its serving FAP is modeled as $d^{-\upsilon}$, where d and υ denote the distance from the SMD to its serving FAP and the path loss exponent, respectively. Moreover, the uplink and downlink channel fading coefficients are denoted by h_1 and h_2, respectively, both of which are modeled as circularly symmetric complex Gaussian random variables. Considering the white Guassian noise power is N_0, the uplink rate R_{U} and downlink rate R_{D} are given by

$$R_{\mathrm{U}} = W_{\mathrm{U}} \log_2\left(1 + \frac{P_t d^{-\upsilon} |h_1|^2}{\mathrm{N}_0}\right), \tag{3}$$

$$R_{\mathrm{D}} = W_{\mathrm{D}} \log_2\left(1 + \frac{P_{\mathrm{F}} d^{-\upsilon} |h_2|^2}{\mathrm{N}_0}\right), \tag{4}$$

where W_{U}, W_{D}, P_t, and P_{F} denote the uplink channel bandwidth, downlink channel bandwidth, transmit power of SMD, and transmit power of its serving FAP, respectively.

With $(1-\lambda) I$ bits offloaded to the cloud, the amount of uplink data is $\beta_1 (1-\lambda) I$, where $\beta_1 (\beta_1 > 0)$ denotes the overhead in uplink transmission, such as channel encoding, data encryption. Besides, the downlink data size is $\beta_2 (1-\lambda) I$, where $\beta_2 (\beta_2 > 0)$ accounts jointly for the overhead in downlink transmission and the ratio of output to input bits offloaded to the cloud [13]. When computation offloading takes place, the total execution time t_c can be expressed as

$$t_c = t_{\mathrm{U}} + \tau_c + t_{\mathrm{D}}. \tag{5}$$

In (5), τ_c is the execution time in the cloud and given by $\tau_c = \frac{\alpha(1-\lambda)I}{f_c}$, where f_c is the computational speed of cloud. Here, f_c is fixed for the duration of application execution [13], [27]. In addition, t_{U} and t_{D} denote the uplink and downlink transmission delay, which can be expressed as $t_{\mathrm{U}} = \frac{\beta_1(1-\lambda)I}{R_{\mathrm{U}}}$ and $t_{\mathrm{D}} = \frac{\beta_2(1-\lambda)I}{R_{\mathrm{D}}}$, respectively. Thus, the energy consumption of SMD E_c is expressed as

$$E_c = (P_0 + k_t P_t)\, t_{\mathrm{U}} + P_r t_{\mathrm{D}}, \tag{6}$$

where P_0, k_t, and P_r refer to the static power consumption, efficient factor of power amplifier, and receive power consumption, respectively.

3) Total Cost: Since parallel computing is considered, the latency to execute the application $L(f_l, P_t, \lambda)$ can be given by

$$L(f_l, P_t, \lambda) = \max\{t_l, t_c\}. \tag{7}$$

Besides, the whole energy consumption of SMD $E(f_l, P_t, \lambda)$ can be expressed as

$$E(f_l, P_t, \lambda) = E_l + E_c. \tag{8}$$

D. Problem Formulation

1) Energy Consumption Minimization Problem (ECM): In order to prolong the battery lifetime, it is useful to minimize the overall energy consumption of SMD while guaranteeing the latency requirement of application. The energy minimization problem is formulated as follows:

$$\begin{aligned}
\mathbf{P1}: \ \min_{f_l, P_t, \lambda}\ & E(f_l, P_t, \lambda) \\
s.t.\ & \mathrm{C1}: L(f_l, P_t, \lambda) \le L_{\max}, \\
& \mathrm{C2}: 0 \le \lambda \le 1, \\
& \mathrm{C3}: 0 \le P_t \le P_{t_{\max}}, \\
& \mathrm{C4}: 0 \le f_l \le f_{l_{\max}},
\end{aligned}$$

where $E\left(P_t, f_{l,}, \lambda\right)$ is given by

$$\begin{aligned}
E(f_l, P_t, \lambda) = \alpha I k \lambda f_l^2 + (P_0 + k_t P_t) \frac{\beta_1 (1-\lambda) I}{W_{\mathrm{U}} \log_2 (1 + P_t a)} \\
+ P_r \frac{\beta_2 (1-\lambda) I}{R_{\mathrm{D}}},
\end{aligned} \tag{9}$$

where $a = \frac{d^{-\upsilon}|h_1|^2}{\mathrm{N}_0}$. Besides, t_c can be expressed as

$$t_c = \frac{\beta_1 (1-\lambda) I}{W_{\mathrm{U}} \log_2\left(1 + \frac{P_t d^{-\upsilon}|h_1|^2}{\mathrm{N}_0}\right)} + \frac{\alpha(1-\lambda)I}{f_c} + \frac{\beta_2(1-\lambda)I}{R_{\mathrm{D}}}. \tag{10}$$

In **P1**, C1 reflects the latency constraint. C2 specifies the domain of λ. C3 and C4 are the maximum transmit power and computational speed constraints imposed by radio interface and CPU, respectively.[3] We can see both $E(f_l, P_t, \lambda)$ and t_c in C1 are nonconvex. Therefore, **P1** is a nonconvex problem, which is challenging to be solved [31]. Note that using DVS technology, the SMD can adjust its computational speed according to the optimization results.

[3]Note that f_l is restricted to a finite set of values in practice. If we model it as a discrete variable, the offloading problem will be formulated as a mixed-integer optimization problem, which is NP-hard in general. Due to the difficulty in obtaining its optimal solution, it is hard to provide insights into the optimal policy structures. Therefore, we model it as a continuous variable in our paper. Accordingly, the optimal solution could be served as a performance upper-bound of realistic offloading strategies. Moreover, the computational speed of SMD is assumed to be a continuous variable in many researches [19], [30].

2) Latency Minimization Problem (LM): In situations where the SMD has a stringent requirement on energy consumption while application is delay-sensitive, it is preferred to use limited energy to, as far as possible, shorten the latency to execute the application, which can be formulated as follows:

$$\begin{aligned}\mathbf{P2}:\ &\min_{f_l,P_t,\lambda}\ L\left(f_l,P_t,\lambda\right)\\ &s.t.\ \text{C5}: E\left(f_l,P_t,\lambda\right)\le E_{\max},\\ &\qquad \text{C2, C3, C4},\end{aligned}$$

where C5 reflects the energy constraint. Due to the nonconvexity of both objective and constraints, **P2** is a nonconvex problem. Additionally, it is also a nonsmooth optimization problem due to the nonsmoothness of objective function $L\left(f_l,P_t,\lambda\right)$.

Remark 1: Not only the LM problem but also the ECM problem is suitable for delay-sensitive applications. Delay-sensitive applications are characterized by their bounded end-to-end delay requirements [32], [33], which can be guaranteed in the ECM problem.

III. Energy-Optimal Partial Offloading Control Scheme

In this section, the feasibility of the energy minimization problem is first analyzed. Then, we propose an algorithm to solve this nonconvex problem by transforming it to a convex problem. At last, we discuss two special cases and extend the problem to a multiple cloud servers scenario.

A. Feasibility Analysis

In order to guarantee that the feasible region of f_l is not empty, we have

$$\lambda\le\frac{L_{\max}f_{l_{\max}}}{\alpha I}\triangleq\lambda_{\text{u}} \tag{11}$$

according to C1 and C4. Similarly, to ensure that the feasible region of P_t is not empty, from C1 and C3, we can obtain

$$\lambda\ge1-\frac{L_{\max}}{I\left(\frac{\beta_1}{W_{\text{U}}\log_2\left(aP_{t_{\max}}+1\right)}+\frac{\alpha}{f_c}+\frac{\beta_2}{R_{\text{D}}}\right)}\triangleq\lambda_{\text{l}}. \tag{12}$$

Taking C2 into consideration, we define

$$\lambda_{\max}=\min\left\{\lambda_{\text{u}},1\right\}, \tag{13}$$

$$\lambda_{\min}=\max\left\{\lambda_{\text{l}},0\right\}. \tag{14}$$

To make the feasible region of λ be not empty, $\lambda_{\min}\le\lambda_{\max}$ should hold, which is the equivalent of

$$L_{\max}\ge L_{\max}^{\text{par}}, \tag{15}$$

where $L_{\max}^{\text{par}}$ is given by

$$L_{\max}^{\text{par}}=\frac{1}{\frac{f_{l_{\max}}}{\alpha I}+\frac{1}{I\left(\frac{\beta_1}{W_{\text{U}}\log_2\left(aP_{t_{\max}}+1\right)}+\frac{\alpha}{f_c}+\frac{\beta_2}{R_{\text{D}}}\right)}}. \tag{16}$$

Based on the analysis above, we conclude that **P1** is feasible only if $L_{\max}\ge L_{\max}^{\text{par}}$, which implies that only the applications with $L_{\max}\ge L_{\max}^{\text{par}}$ can be supported in partial offloading.

Remark 2: In full offloading, when the application is totally processed in the SMD, $\frac{\alpha I}{f_{l_{\max}}}\le L_{\max}$ should hold. Otherwise, we have $I\left(\frac{\beta_1}{W_{\text{U}}\log_2\left(aP_{t_{\max}}+1\right)}+\frac{\alpha}{f_c}+\frac{\beta_2}{R_{\text{D}}}\right)\le L_{\max}$. Therefore, only the applications with $L_{\max}\ge L_{\max}^{\text{full}}$ can be supported in full offloading, where $L_{\max}^{\text{full}}$ is given by

$$L_{\max}^{\text{full}}=\min\left\{\frac{\alpha I}{f_{l_{\max}}},I\left(\frac{\beta_1}{W_{\text{U}}\log_2\left(aP_{t_{\max}}+1\right)}+\frac{\alpha}{f_c}+\frac{\beta_2}{R_{\text{D}}}\right)\right\}. \tag{17}$$

Comparing (16) with (17), we can see $L_{\max}^{\text{par}}\le L_{\max}^{\text{full}}$. Therefore, only partial offloading scheme can support these applications with $L_{\max}\in[L_{\max}^{\text{par}},L_{\max}^{\text{full}})$.

B. Optimal Solution

We optimally solve **P1** by transforming the original problem to an one-dimensional problem. Before we elaborate the details of the proposed algorithm, we first give a lemma which is the basis of this algorithm.

Lemma 1: We always have

$$\inf_{x,y}f\left(x,y\right)=\inf_{x}\tilde{f}\left(x\right),$$

where $\tilde{f}\left(x\right)=\inf_y f\left(x,y\right)$.

Proof: See [31]. ■

Lemma 1 tells us that we could minimize a function by first minimizing over some of the variables, and then minimizing over the remaining ones. With Lemma 1, we could solve **P1** by minimizing over f_l, P_t, and λ, sequentially.

In **P1**, we discover that $E\left(f_l,P_t,\lambda\right)$ increases monotonically with the increase of f_l. Besides, from C1, we have $t_l\le L_{\max}$, which yields $f_l\ge\frac{\alpha\lambda I}{L_{\max}}$. Therefore, the optimal f_l can be derived in closed-form as follows:

$$f_l^*\left(\lambda\right)=\frac{\alpha\lambda I}{L_{\max}}. \tag{18}$$

Substituting (18) into **P1**, we can simplify the original optimization problem to **P3** by reducing the number of variables to two. Specifically,

$$\begin{aligned}\mathbf{P3}:\ &\min_{P_t,\lambda}\ E\left(P_t,\lambda\right)\\ &s.t.\ \text{C6}: t_c\le L_{\max},\\ &\qquad \text{C7}: 0\le\lambda\le\lambda_{\max},\\ &\qquad \text{C3},\end{aligned}$$

where $E\left(P_t,\lambda\right)$ is given by

$$E\left(P_t,\lambda\right)=\frac{k\left(\alpha I\right)^3}{L_{\max}^2}\lambda^3+\beta_1\left(1-\lambda\right)I\underbrace{\frac{P_0+k_tP_t}{W_{\text{U}}\log_2\left(1+P_ta\right)}}_{f\left(P_t\right)}+P_r\frac{\beta_2\left(1-\lambda\right)I}{R_{\text{D}}}. \tag{19}$$

Next, we minimize $E\left(P_t,\lambda\right)$ by first minimizing over P_t and then over λ. In order to obtain the optimal P_t, we should analyze the feature of $f\left(P_t\right)$, which is revealed in the following lemma.

Lemma 2: The function

$$f\left(P_t\right) = \frac{P_0 + k_t P_t}{W_{\mathrm{U}} \log_2\left(1 + P_t a\right)}$$

is unimodal.

Proof: First, we can obtain that $f(x) = x\left(P_0 + k_t \frac{2^{\frac{1}{W_{\mathrm{U}} x}} - 1}{a}\right)$ is convex w.r.t. x, $x \geq 0$, since its Hessian matrix is positive semidefinite. Hence, if the function $f(x)$ has a minimum, this minimum is unique. Next, let $P_t = \frac{2^{\frac{1}{W_{\mathrm{U}} x}} - 1}{a}$, we can represent $f(x)$ as $f\left(\frac{1}{W_{\mathrm{U}} \log_2(1 + a P_t)}\right)$ versus P_t, i.e., $f(P_t)$. In this procedure, all we do is to apply a change of variable, which is monotonous. Therefore, it is still true to say if $f(P_t)$ has a minimum, this minimum is unique. In other words, the function $f(P_t)$ is unimodal. ∎

Thanks to Lemma 2, the optimal P_t will be chosen among three points, i.e., the two extreme points of set $\boldsymbol{S}$ and the peak point of $f(P_t)$. Here, $\boldsymbol{S}$ denotes the feasible region of P_t. From C6, we can obtain the following result

$$P_t \geq \frac{2^{\frac{\beta_1}{W_{\mathrm{U}}\left(\frac{L_{\max}}{(1-\lambda) I} - \frac{\alpha}{f_c} - \frac{\beta_2}{R_{\mathrm{D}}}\right)}} - 1}{a} \triangleq P_{t_{\min}}(\lambda). \tag{20}$$

According to C3 and (20), we can express the optimal P_t in closed-form as

$$P_t^*(\lambda) = \begin{cases} P_{t_{\min}}(\lambda), & P_{t_{\min}}(\lambda) > \hat{P}_t, \\ \hat{P}_t, & P_{t_{\min}}(\lambda) \leq \hat{P}_t \leq P_{t_{\max}}, \\ P_{t_{\max}}, & \hat{P}_t > P_{t_{\max}}, \end{cases} \tag{21}$$

where $\hat{P}_t$ denotes the transmit power that minimizes $f(P_t)$, i.e., $\nabla f(P_t) \mid_{P_t = \hat{P}_t} = 0$. Note that $\hat{P}_t$ is not related to λ. Using (21), **P3** can be rewritten as a simplified one-dimensional problem of λ as follows:

$$\begin{aligned} \mathbf{P4}: \ & \min_{\lambda} \ E(\lambda) \\ & s.t. \ \mathrm{C8}: \lambda_{\min} \leq \lambda \leq \lambda_{\max}, \end{aligned}$$

where

$$\begin{aligned} E(\lambda) = & \frac{k(\alpha I)^3}{L_{\max}^2} \lambda^3 + f\left(P_t^*(\lambda)\right) \beta_1 (1-\lambda) I \\ & + P_r \frac{\beta_2 (1-\lambda) I}{R_{\mathrm{D}}}. \end{aligned} \tag{22}$$

So far we have simplified the original optimization problem **P1** into an one-dimensional problem **P4**. However, it is difficult to prove the convexity of $E(\lambda)$ directly due to the item $f\left(P_t^*(\lambda)\right) \beta_1 (1-\lambda) I$. Fortunately, we can prove the convexity by transforming **P3** and using the following lemma.

Lemma 3: If f is convex in (x, y), and C is a convex nonempty set, then the function

$$g(x) = \inf_{y \in C} f(x, y)$$

is convex in x, provided $g(x) > -\infty$ for some x. The domain of g is the projection of dom f *on its $x-$coordinates, i.e.,* dom $g = \left\{x \mid (x, y) \in \mathrm{dom}\ f \text{ for some } y \in C\right\}$.

Proof: See [31]. ∎

Algorithm 1 Energy-Optimal Partial Computation Offloading (EPCO)

```
1: Obtain the latency constraint $L_{\max}$ and other system settings, then calculate $L_{\max}^{\mathrm{par}}$
2: if $L_{\max} < L_{\max}^{\mathrm{par}}$ then
3:     Drop this application or update the system to support it
4: else
5:     Using bisection algorithm to solve P4 and obtain $\lambda^*$
6: end if
7: Calculate $f_l^*$ using (18)
8: Calculate $P_t^*$ using (21)
```

This lemma provides us with a method to prove the convexity of a function. Motivated by it, we make the following efforts.

Let

$$x = \frac{1-\lambda}{\log_2\left(1 + P_t a\right)}, \tag{23}$$

then the constraint on x is $x \in \left[\frac{1-\lambda}{\log_2\left(1 + P_{t_{\max}} a\right)}, +\infty\right)$. Hence, we can reformulate **P3** in a more compact form as

$$\begin{aligned} \mathbf{P5}: \ & \min_{x, \lambda} \ E(x, \lambda) \\ & s.t. \ \mathrm{C9}: t_c(x, \lambda) \leq L_{\max}, \\ & \qquad \mathrm{C10}: x \geq \frac{1-\lambda}{\log_2\left(1 + P_{t_{\max}} a\right)}, \\ & \qquad \mathrm{C7}, \end{aligned}$$

where the objective is

$$\begin{aligned} E(x, \lambda) = & \frac{k(\alpha I)^3}{L_{\max}^2} \lambda^3 + \frac{\beta_1 I}{W_{\mathrm{U}}} x \left[P_0 + \frac{k_t}{a}\left(2^{\frac{1-\lambda}{x}} - 1\right)\right] \\ & + P_r \frac{\beta_2 (1-\lambda) I}{R_{\mathrm{D}}}, \end{aligned} \tag{24}$$

and the offloading latency is

$$t_c(x, \lambda) = \frac{\beta_1 I}{W_{\mathrm{U}}} x + \frac{\alpha (1-\lambda) I}{f_c} + \frac{\beta_2 (1-\lambda) I}{R_{\mathrm{D}}}. \tag{25}$$

Since Hessian matrix of $E(x, \lambda)$ is positive semidefinite, $E(x, \lambda)$ is convex in (x, λ) within the triangular region specified by C6 and $x \in \left[\frac{1-\lambda}{\log_2\left(1 + P_{t_{\max}} a\right)}, +\infty\right)$. Additionally, C9 is a linear function of x and λ. Therefore, **P5** is a convex optimization problem, which minimizes a convex function $E(x, \lambda)$ over a convex set $\mathcal{C}$ [31]. Define $E_1(\lambda) = \inf_{x \in \mathcal{C}} E(x, \lambda)$, thanks to Lemma 3, we can see that $E_1(\lambda)$ is convex w.r.t. λ. Since $E(\lambda)$ in **P4** is the same as $E_1(\lambda)$, **P4** is also a convex problem. Besides, it is an one-dimensional problem. Hence, some simple algorithms (e.g., 0.618 and bisection) could be used to obtain its globally optimal solution. Without loss of generality, we choose the bisection method.

By now, we have solved the original problem **P1**. The pseudo code of this method is given in Algorithm 1. Besides, bisection algorithm in line 5 could be found in [31].

Remark 3: Here, we provide the corresponding quantization method based on the structure of ECM. Assuming that for a set of files to be processed for virus scan, the possible partition set is $\Omega = \{\lambda_1, \lambda_2, \cdots \lambda_{\mathrm{M}}\}$. We define $\lambda_{c1} = \arg \min_{\lambda_i \in \Omega, \lambda_i \le \lambda^*} \{\lambda^* - \lambda_i\}$ and $\lambda_{c2} = \arg \min_{\lambda_i \in \Omega, \lambda_i \ge \lambda^*} \{\lambda_i - \lambda^*\}$. Since $E(\lambda)$ is a convex function, we can choose the optimal offloading ratio in practice λ_p^* through comparing $E(\lambda_{c1})$ and $E(\lambda_{c2})$. Specifically, $\lambda_p^* = \lambda_{c1}$ if $E(\lambda_{c1}) \le E(\lambda_{c2})$. Otherwise, $\lambda_p^* = \lambda_{c2}$. Note that if λ_{c1} or λ_{c2} do not exist, we let $E(\lambda_{c1}) = +\infty$ or $E(\lambda_{c2}) = +\infty$.

C. Analysis of Special Cases

In this subsection, we provide an analysis of some special cases of the problem aforementioned. This analysis gives practical guidelines for partial communication offloading. Moreover, a key different conclusion from other works (e.g., [13]) due to DVS technology is derived.

1) Optimality of Local Execution: Here, we give the necessary and sufficient condition under which the SMD prefers to process the application locally. Specifically, $\lambda = 1$ should be feasible and $\frac{dE(\lambda)}{d\lambda} |_{\lambda=1} \le 0$ due to the convexity of $E(\lambda)$. The first condition means that the SMD should have enough computation capacity to support the application with latency requirement $L_{\max}$, i.e.,

$$f_{l_{\max}} \ge \frac{\alpha I}{L_{\max}}. \tag{26}$$

The second condition holds if

$$\frac{dE(\lambda)}{d\lambda} |_{\lambda=1} = \frac{3k(\alpha I)^3}{L_{\max}^2} - f\left(P_t^*(1)\right)\beta_1 I - \frac{P_r \beta_2 I}{R_{\mathrm{D}}} \le 0, \tag{27}$$

where $f(P_t)$ and $P_t^*(1)$ have been given in Lemma 2 and (21), respectively.

2) Optimality of Total Offloading: Similarly, the necessary and sufficient condition under which total offloading is the optimal decision is: i) $\lambda = 0$ is feasible; ii) $\frac{dE(\lambda)}{d\lambda} |_{\lambda=0} \ge 0$. Different from [13], where total offloading could be optimal under some conditions, we come to a different conclusion with DVS technology, which is given as follows:

Theorem 1: When the SMD has the capability of DVS, total offloading could not be the optimal strategy.

Proof: Intuitively, the SMD prefers to offload its computation when the channel condition is very good. Considering $a \to +\infty$, $\hat{P}_t = +\infty$. According to (21), $P_t^*(\lambda) = P_{t_{\max}}$. Therefore,

$$\frac{dE(\lambda)}{d\lambda} |_{\lambda=0} = -\beta_1 I f\left(P_{t_{\max}}\right) - \frac{P_r \beta_2 I}{R_{\mathrm{D}}} < 0, \tag{28}$$

which violates the second condition mentioned above. Hence, total offloading can not be the optimal strategy. ■

D. Extension to Multiple Cloud Servers

In this subsection, we briefly extend the simple case described above to a multiple cloud servers scenario, where a set of cloud servers, i.e., FAPs can process the application for the SMD. Specifically, the SMD sends data to the most suitable FAP if an offloading decision is made. Then this FAP distributes the data to each FAP in the femto-cloud. Here, we aim to optimize the computation distribution among femto-cloud as well as the user association to minimize the energy consumption of SMD, which is hard to be tackled. Therefore, we divide this problem into two subproblems and solve them one by one. One is to find the optimal computation distribution for a given associated FAP. The other is to choose the most suitable associated FAP.

1) Subproblem One: Here, we consider point-to-point communication between the severing FAP and other FAPs in the femto-cloud [34], [35]. Therefore, the cloud latency L_c, which is due to the communication latency between FAPs and the computation latency at each FAP, is the maximum latency experienced with the FAPs of the femto-cloud. It can be written as

$$L_c = \max_{n=\{1,\cdots,N\}} \left\{ \frac{\alpha w_n}{f_n} + \delta_{Tx,bh}(n) + \delta_{Tx,bh}^r(n) \right\}, \tag{29}$$

where N, w_n, f_n, $\delta_{Tx,bh}(n)$, and $\delta_{Tx,bh}^r(n)$ denote the number of FAPs in femto-cloud, allocated computation bits to FAP n, computational speed of FAP n, one way communication latency from the associated FAP to FAP n, and that for the reverse way, respectively. Without loss of generality, we denote the index of the associated FAP as 1. Therefore, $\delta_{Tx,bh}(1)$ and $\delta_{Tx,bh}^r(1)$ both are zero. Additionally, we consider the fiber communication as the transmission technology between FAPs, which guarantees rapid interaction among femto-cloud[4] Due to the super-high throughput of fiber communication, $\delta_{Tx,bh}(n)$ and $\delta_{Tx,bh}^r(n)$ are load independent [36]. Therefore, L_c can be approximated by

$$L_c = \max \left\{ \frac{\alpha w_1}{f_1}, \max_{n=\{2,\cdots,N\}} \left\{ \frac{\alpha w_n}{f_n} + 2\delta \right\} \right\}, \tag{30}$$

where δ refers to the load independent latency. Using (30), The offloading latency t_{cm} could be expressed as

$$t_{cm} = t_{\mathrm{U}} + L_c + t_{\mathrm{D}}. \tag{31}$$

Accordingly, the original problem **P1** can be rewritten as

$$\begin{aligned} \mathbf{P6}: \quad & \min_{f_l, P_t, \lambda, w_n} \quad E(f_l, P_t, \lambda) \\ & s.t. \ \mathrm{C11}: \max\{t_l, t_{cm}\} \le L_{\max}, \\ & \quad \mathrm{C12}: \sum_{n=1}^{N} w_n = (1-\lambda) I, \\ & \quad \mathrm{C13}: w_n \ge 0, \\ & \quad \mathrm{C2}, \mathrm{C3}, \mathrm{C4}, \end{aligned}$$

where $E\left(P_t, f_l, \lambda\right)$ is the same as (9). From C11, we have

$$t_{\mathrm{U}} + t_{\mathrm{D}} \le L_{\max} - L_c. \tag{32}$$

Observing (32), we can see that reducing L_c can expand the feasible region on (f_l, P_t, λ) space, which consequently reduces the optimal value of **P6**. Hence, instead of directly solving **P6**, we first investigate the following problem

$$\begin{aligned} \mathbf{P7}: \ & \min_{w_n} L_c \\ & s.t. \ \mathrm{C12}, \mathrm{C13}. \end{aligned}$$

[4]Note that the analysis below is applicable for wireless communication between FAPs if fiber communication is not available.

Algorithm 2 Energy-Optimal Partial Computation Offloading in Multiple Cloud Servers Scenario (EPCOMCSS)

```
1: Initialize the optimal value of P6 $V^*$
2: for $n = 1$ to $N$ do
3:     Calculate $L'_{\max}$ according to (35)
4:     Use Algorithm 1 to solve P8 and obtain the optimal solution $\left(f_{l_n}, P_{t_n}, \lambda_n\right)$ and the optimal value $V_n$
5:     if $V_n \le V^*$ then
6:         Set $V^* = V_n$, $P_t^* = P_{t_n}$, $f_l^* = f_{l_n}$, and $\lambda^* = \lambda_n$
7:     end if
8: end for
9: Calculate $w_n^*$ according to (33)
```

Easily, we can obtain the optimal solution of **P7** as below,

$$w_n^* = \begin{cases} \dfrac{(1-\lambda) I f_1 + \frac{2\delta f_1}{\alpha}\sum_{n=2}^N f_n}{\sum_{n=1}^N f_n}, & n = 1 \\ \dfrac{f_n}{\sum_{n=2}^N f_n}\left[(1-\lambda) I - w_1\right], & n \ge 2. \end{cases} \tag{33}$$

Accordingly, the optimal value of **P7** L_c^* can be expressed as

$$L_c^* = \frac{\alpha(1-\lambda) I}{\sum_{n=1}^N f_n} + \frac{2\delta\sum_{n=2}^N f_n}{\sum_{n=1}^N f_n}. \tag{34}$$

Substituting (34) and (18) into **P6**, we can simplify the problem as follows:

$$\begin{aligned} \textbf{P8}: \ & \min_{P_t,\lambda} E(P_t,\lambda) \\ & s.t.\ \text{C14}: t_{\text{U}} + \frac{\alpha(1-\lambda) I}{\sum_{n=1}^N f_n} + t_{\text{D}} \le L'_{\max}, \\ & \quad \text{C3}, \text{C7}, \end{aligned}$$

where $E(P_t,\lambda)$ is given by (19), and $L'_{\max}$ is given by

$$L'_{\max} = L_{\max} - \frac{2\delta\sum_{n=2}^N f_n}{\sum_{n=1}^N f_n}. \tag{35}$$

Since **P8** has the same structure as **P3**, it could be efficiently solved with Algorithm 1.

2) Subproblem Two: Since the size of femto-cloud is usually limited, the exhaustive search method can be used to solve this problem. Specifically, an optimal energy consumption value can be obtained for any given associated FAP by solving subproblem one. Through comparing these values, we find the minimum and choose the corresponding FAP as the associated FAP.

Remark 4: The associated FAP may be not the one with the best channel condition, since the computational speeds of FAPs also affect the result, which is reflected in Section V-B and Section V-D. Specifically, larger computational speed of the associated FAP, i.e., f_1, larger $L'_{\max}$, which provides a larger feasible region of **P8**. Consequently, a smaller optimal value could be obtained.

After solving these two subproblems, ECM in a multiple cloud servers scenario is handled. The pseudo code of this method is given in Algorithm 2.

IV. Latency-Optimal Partial Offloading Control Scheme

In this section, we first present the feasibility analysis and then propose an algorithm to solve the nonconvex and nonsmooth problem **P2**. Finally, we extend this problem to a multiple cloud servers scenario and propose the corresponding algorithm.

A. Feasibility Analysis

Since **P2** aims at minimizing the latency under energy consumption constraint (i.e., $E(f_l, P_t, \lambda) \le E_{\max}$) and other three constraints (i.e., C2, C3, and C4), the feasibility problem of **P2** is the equivalent of solving the following problem:

$$\begin{aligned} \textbf{P9}: \ & \min_{f_l,P_t,\lambda} E(f_l,P_t,\lambda) \\ & s.t.\ \text{C2}, \text{C3}, \text{C4}. \end{aligned}$$

If the optimal value of **P9**, i.e., $E\left(f_l^*, P_t^*, \lambda^*\right)$ satisfies $E\left(f_l^*, P_t^*, \lambda^*\right) \le E_{\max}$, then **P2** is feasible; otherwise **P2** is infeasible. Note that the feasible set of **P2** is nonempty, since we can always find a feasible f_l when setting λ as one to meet the energy consumption constraint.

B. Optimal Solution

From Section II, we know that **P2** is a nonconvex and nonsmooth problem. Motivated by the difficulties of handling this problem, we propose a suboptimal algorithm in this section. The basic idea is to construct a non-increasing objective sequence, which converges to a locally optimal solution of **P2**. We elaborate the details of the proposed algorithm in the sequel.

First, we introduce a new variable t to transform the originally nonsmooth problem **P2** to a smooth one as follows:

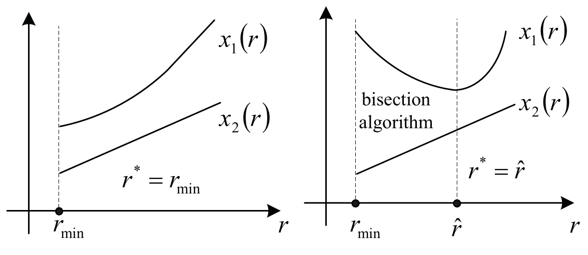
$$\begin{aligned} \textbf{P10}: \ & \min_{f_l,P_t,\lambda,t} t \\ & s.t.\ \text{C15}: L(f_l,P_t,\lambda) \le t, \\ & \quad \text{C2}, \text{C3}, \text{C4}, \text{C5}. \end{aligned}$$

By setting $r = \frac{1}{\log_2(1+P_t a)}$, we rewrite **P10** as

$$\begin{aligned} \textbf{P11}: \ & \min_{f_l,r,\lambda,t} t \\ & s.t.\ \text{C16}: \alpha I\lambda - f_l t \le 0, \\ & \quad \text{C17}: \left(\frac{\beta_1}{W_{\text{U}}} r + \frac{\alpha}{f_c} + \frac{\beta_2}{R_{\text{D}}}\right)(1-\lambda) I - t \le 0, \\ & \quad \text{C18}: E(f_l,r,\lambda) - E_{\max} \le 0, \\ & \quad \text{C19}: r \ge \frac{1}{\log_2\left(1 + a P_{t_{\max}}\right)} \triangleq r_{\min}, \\ & \quad \text{C2}, \text{C4}, \end{aligned}$$

where the first item in C18 could be expressed as

$$\begin{aligned} E(f_l,r,\lambda) = {} & \alpha k I\lambda f_l^2 + \frac{P_r \beta_2 (1-\lambda) I}{R_{\text{D}}} \\ & + \frac{\beta_1 (1-\lambda) I}{W_{\text{U}}} \underbrace{r\left[P_0 + \frac{k_t}{a}\left(2^{\frac{1}{r}} - 1\right)\right]}_{g(r)}. \end{aligned} \tag{36}$$



(a) $r_{\min} \geq \hat{r}$. (b) $r_{\min} < \hat{r}$.

Fig. 2. An illustration of r^* in several cases.

Observing **P11**, we can find that **P11** becomes a linear program of λ and t when we fix f_l and r, while it becomes a convex program of f_l and r when we fix λ and t. Therefore, based on the univariate search technique [21], we could solve **P11** by using alternating minimization between the linear program and the convex program aforementioned. Specifically, we first give a feasible solution of **P11**, e.g., $\left(f_{l_0}, r_0, \lambda_0, t_0\right)$ and solve the linear program to obtain (λ_1, t_1) with the given $\left(f_{l_0}, r_0\right)$. Based on (λ_1, t_1), we then solve the convex problem to obtain $\left(f_{l_1}, r_1\right)$. This process is continued until the relative difference between the objective values in two sequential iterations becomes less than a pre-defined error tolerance threshold ϵ.

Note that with the given $\left(\lambda_j, t_j\right)$, where $j, (j \geq 1)$ denotes the iterative number, the convex problem is a feasibility problem, which has a constant objective. To obtain a non-increasing objective sequence, we should construct a new objective. Here, we adopt the original expression of latency as the new objective, and thus write the new optimization as follows:

$$\mathbf{P12}: \min_{f_l, r} \max \left\{ \underbrace{\frac{\alpha I \lambda_j}{f_l}}_{x_1(r)}, \underbrace{\left(\frac{\beta_1 r}{W_{\mathrm{U}}} + \frac{\alpha}{f_c} + \frac{\beta_2}{R_{\mathrm{D}}}\right)\left(1-\lambda_j\right) I}_{x_2(r)} \right\}$$
$$s.t.\ \mathrm{C20}: E\left(f_l, r, \lambda_j\right) \leq E_{\max}, \quad \mathrm{C4, C19},$$

where the left-hand term of C20 is given by (36).

The optimal r is illustrated in Fig. 2. Obviously, $x_2(r)$ is a monotonically increasing function of r. In order to analytically obtain the optimal r, we should analyze the monotone property of $x_1(r)$.[5] Since $g(r)$ is a strictly convex function of r, it has only one global optimal solution $\hat{r}$.

- If $r_{\min} \geq \hat{r}$, $g(r)$ monotonically increases in $\left[r_{\min}, +\infty\right)$. Via C20, we can see that a larger $g(r)$ results in a smaller f_l, thus leading to a larger $x_1(r)$. Hence, $x_1(r)$ is a monotonically increasing function of r. As shown in Fig. 2a, both $x_1(r)$ and $x_2(r)$ monotonically increase with the increment of r in $\left[r_{\min}, +\infty\right)$, the optimal r should be $r_{\min}$.
- If $r_{\min} < \hat{r}$, $g(r)$ monotonically decreases in $\left[r_{\min}, \hat{r}\right]$ and monotonically increases in $\left[\hat{r}, +\infty\right)$. As shown in Fig. 2b, when $r \in \left[\hat{r}, +\infty\right)$, the optimal r is $\hat{r}$ due to the same reason stated above. Therefore, we can reduce the search region from $[r_{\min}, +\infty]$ to $\left[r_{\min}, \hat{r}\right]$. In $\left[r_{\min}, \hat{r}\right]$, $x_1(r)$ monotonically decreases, while $x_2(r)$ monotonically increases. Hence, we can use the bisection algorithm to find the optimal r [31].

[5]Term $\frac{\alpha I \lambda_j}{f_l}$ is affected by r via C20. Hence, it is reasonable to name $\frac{\alpha I \lambda_j}{f_l}$ as $x_1(r)$.

Algorithm 3 Analytically Solve **P12** (AS) Algorithm

```
1: Based on the channel state and $P_{t_{\max}}$, calculate $r_{\min}$ and $\hat{r}$
2: if $r_{\min} \geq \hat{r}$ then
3:   Set $r_j = r_{\min}$
4:   Set $f_{l_j} = \min\left\{\hat{f}_l, f_{l_{\max}}\right\}$, where $\hat{f}_l$ is the solution of $E\left(f_l, r_j, \lambda_j\right) = E_{\max}$
5:   Based on $f_{l_j}, r_j$, calculate the objective of P12
6: else
7:   Given tolerance $\varepsilon > 0$, let $l = r_{\min}$, and $u = \hat{r}$
8:   while $u - l \geq \varepsilon$ do
9:     Set $r_j = \frac{(l+u)}{2}$
10:    Set $f_{l_j} = \min\left\{\hat{f}_l, f_{l_{\max}}\right\}$, where $\hat{f}_l$ is the solution of $E\left(f_l, r_j, \lambda_j\right) = E_{\max}$
11:    Calculate $x_1\left(r_j\right)$ and $x_2\left(r_j\right)$
12:    if $x_1\left(r_j\right) > x_2\left(r_j\right)$ then
13:      Set $l = r_j$
14:    else if $x_1\left(r_j\right) < x_2\left(r_j\right)$ then
15:      Set $u = r_j$
16:    else
17:      Break
18:    end if
19:  end while
20:  Based on $f_{l_j}, r_j$, calculate the objective of P12
21: end if
```

Algorithm 4 Latency-Optimal Partial Computation Offloading (LPCO)

```
1: Set iteration number $j = 1$, initialize with a feasible $\left(f_{l_0}, r_0, \lambda_0, t_0\right)$
2: Obtain $(\lambda_1, t_1)$ and the corresponding objective value $t_1$ by solving the linear program, which fixes $f_l$ and $r$ as $f_{l_0}$ and $r_0$, respectively
3: Based on $(\lambda_1, t_1)$, use Algorithm 3 to solve P12 and obtain $\left(f_{l_1}, r_1\right)$
4: while $\frac{|t_j - t_{j-1}|}{t_{j-1}} > \epsilon$ do
5:   $j \leftarrow j + 1$
6:   Update $\left(\lambda_j, t_j\right)$ and the corresponding objective value $t_j$ by solving the linear program, which fixes $f_l$ and $r$ as $f_{l_{j-1}}$ and $r_{j-1}$, respectively
7:   Based on $\left(\lambda_j, t_j\right)$, use Algorithm 3 to solve P12 and obtain $\left(f_{l_j}, r_j\right)$
8: end while
```

The pseudo code of the method is represented in Algorithm 3. Further, we can solve **P11** to obtain latency-optimal partial computation offloading strategy by using the algorithm described in Algorithm 4, whose convergence is stated in Theorem 2.

Theorem 2: If **P10** *is feasible for the initial setting* $\left(f_{l_0}, r_0, \lambda_0, t_0\right)$, *the convergence of Algorithm 4 is guaranteed.*

Proof: Assuming that **P11** has a nonempty domain for f_l, r, λ, and t, we can obtain a smaller or equal value, i.e., $t_j \leq t_{j-1}$ through optimizing (λ, t) and (f_l, r) alternately. Therefore, Algorithm 4 yields a non-increasing objective sequence, which is clearly bounded below by a value larger than zero and converges to the stationary point. ■

Remark 5: Here, we still take the virus scan as an example to present the quantization method. Assuming that the possible partition set for a set of files is $\Omega = \{\lambda_1, \lambda_2, \cdots \lambda_M\}$. Define $\lambda_{c1} = \arg \min_{\lambda_i \in \Omega, \lambda_i \leq \lambda^*} \{\lambda^* - \lambda_i\}$ and $\lambda_{c2} = \arg \min_{\lambda_i \in \Omega, \lambda_i \geq \lambda^*} \{\lambda_i - \lambda^*\}$. With Algorithm 4, we obtain λ^*, which may not fall into Ω. However, we can achieve the near-optimal value using the probabilistic ratio [29]. Specifically, $\Pr(\lambda^* = \lambda_{c1}) = 1 - \Pr(\lambda^* = \lambda_{c2}) = p$, where $p = \arg \min_{0 \leq p \leq 1} [p t_c(\lambda_{c1}) + (1-p) t_l(\lambda_{c2})]$.

C. Extension to Multiple Cloud Servers

In this subsection, we study the LM problem in the multiple cloud servers scenario described in Subsection III-D. We aim to derive the optimal computation distribution among femto-cloud and the user association to minimize the execution latency of application, which is hard to be solved. Therefore, we divide this problem into two subproblems and solve them one by one. One is to find the optimal computation distribution for a given associated FAP. The other is to choose the most suitable associated FAP. For the second subproblem, we can use the exhaustive search method described in Subsection III-D. Next, we focus on the first subproblem.

Based on the analysis given in Section III-D, the original problem **P2** can be rewritten as

$$\mathbf{P13}: \min_{f_l, P_t, \lambda, w_n} \max\{t_l, t_U + L_c + t_D\}$$
$$s.t.\ \text{C2, C3, C4, C5, C12, C13.}$$

Observing **P13**, we find that the optimal w_n is the one that minimize L_c. In other words, we can still solve **P7** before solving **P13**. Therefore, the optimal computation distribution among multiple FAPs is the same as that derived in Section III-D.

Substituting (34) and (18) into **P13**, we can simplify the problem as follows:

$$\mathbf{P14}: \min_{f_l, P_t, \lambda} \max\left\{t_l, t_U + \frac{\alpha(1-\lambda)I}{\sum_{n=1}^{N} f_n} + t_D + t'\right\}$$
$$s.t.\ \text{C2, C3, C4, C5,}$$

where $t' = \frac{2\delta \sum_{n=2}^{N} f_n}{\sum_{n=1}^{N} f_n}$ is a constant, which is independent of optimization variables. Since **P14** has the same structure as **P2**, it could be efficiently solved with the LPCO algorithm.

Remark 6: Note that we can always decouple **P7** from **P6** and **P13** to obtain the optimal computation distribution among multiple FAPs. This is because we do not account for the energy consumption of femto-cloud.

After solving these two subproblems, LM problem in a multiple cloud servers scenario is handled. The pseudo code of this method is given in Algorithm 5.

Algorithm 5 Latency-Optimal Partial Computation Offloading in Multiple Cloud Servers Scenario (LPCOMCSS)

```
1: Initialize the optimal value of P13 $V^*$
2: for $n = 1$ to $N$ do
3:     Use Algorithm 4 to solve P14 and obtain the optimal
       solution $(f_{l_n}, P_{t_n}, \lambda_n)$ and the optimal value $V_n$
4:     if $V_n \leq V^*$ then
5:         Set $V^* = V_n$, $P_t^* = P_{t_n}$, $f_l^* = f_{l_n}$, and $\lambda^* = \lambda_n$
6:     end if
7: end for
8: Calculate $w_n^*$ according to (33)
```

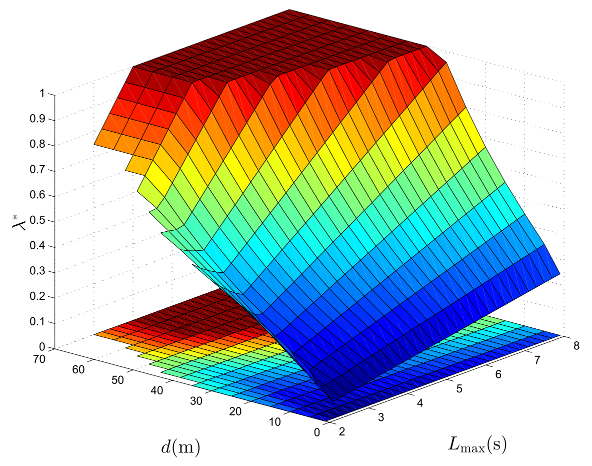
V. Simulation Results

This section provides some simulation results to illustrate the performance of the proposed algorithms. We set $k = 10^{-26}$ so that the energy consumption is consistent with the measurements in [37]. Besides, we let $\alpha = 40$ to fit the computing features in [13]. The remaining parameters are taken as follows: $W_U = W_D = 10$ MHz [13], $P_0 = 0.4$ W [13], $k_t = 18$ [13], $\beta_1 = 1$ [13], $\beta_2 = 0.2$ [13], $P_F = 0.1$ W [38], $P_r = 0.4$ W [39], $P_{t_{\max}} = 0.1$ W [28], $f_{l_{\max}} = 4 \times 10^8$ cycles/s [37], $f_c = 8 \times 10^8$ cycles/s [13], and $\delta = 15$ ms [36].

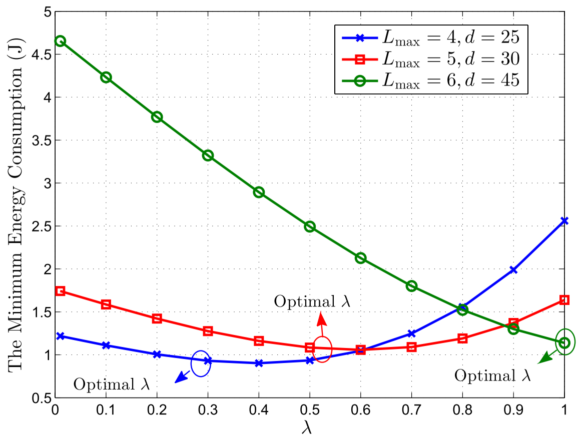
A. Performance of EPCO

Fig. 3 shows the optimal ratio λ^* obtained by EPCO algorithm and proves its optimality. Fig. 3a shows the optimal ratio λ^* versus d and $L_{\max}$. As d increases, sending data through wireless channel consumes more energy. Therefore, λ^* increases and finally arrives to $\frac{f_{l\max} L_{\max}}{\alpha I}$ as stated in C7. Note that the original optimization problem is feasible in the projection area shown in Fig. 3a. Moreover, the optimal λ^* is always greater than zero, which verifies the results given in Theorem 1. Fig. 3b shows the minimum energy consumption of SMD as a function of λ under several simulation settings. As shown in Fig. 3b, the minimum energy consumption can be obtained by adopting the optimal ratio λ^*. Moreover, the optimal ratio is consistent with that shown in Fig. 3a, which verifies the optimality of EPCO algorithm.

Fig. 4 shows the admission probability (i.e., the probability that the application with latency requirement $L_{\max}$ can be supported by the system) of the full offloading (FO) and the proposed EPCO. Specifically, the admission probability performance with respect to d in the case of $L_{\max} = 3$ s is displayed in Fig. 4a. Note that the applications with $L_{\max} = 3$ s cannot be supported only by the SMD, which can be obtained by (11). In other words, offloading is necessary under this condition. From Fig. 4a, we can see as d increases, the admission probability reduces in both schemes because the communication cost increases with the growth of d. Further, the admission probability is reduced to zero since offloading costs so much energy that the SMD cannot afford. Moreover, compared with FO, EPCO could obtain higher application admission probability, since it makes full use of the computation resources at both sides. Fig. 4b presents the

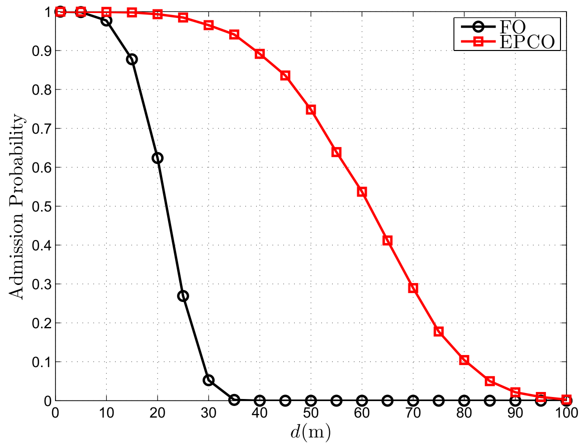


(a) λ^* vs. d and $L_{\max}$.

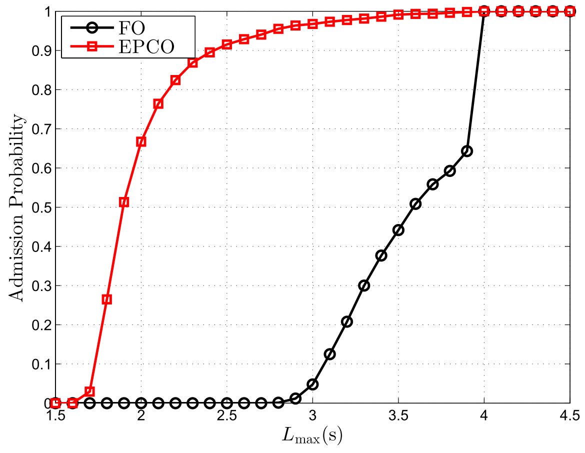


(b) The minimum energy consumption v.s. λ.

Fig. 3. λ^* and its optimality.



(a) AP v.s. d.



(b) AP v.s. $L_{\max}$.

Fig. 4. Admission probability (AP) performance.

admission probability versus $L_{\max}$. In Fig. 4b, we can see that the admission probability in both schemes grow with the increase of $L_{\max}$. Moreover, when $L_{\max}$ is larger than 4 s, the value of admission probability is equal to one. This is due to the fact that the application with $L_{\max} \geqslant 4$ s can be solely supported by the SMD. In addition, as shown in Fig. 4b, only the proposed EPCO scheme can support the application when $L_{\max} \in [1.7, 2.8]$, which verifies the benefit of partial offloading in terms of enlarging admission probability.

Fig. 5 evaluates the minimum energy consumption of SMD using several schemes versus distance d. As expected, EPCO outperforms the other three schemes, since it combines the advantages of Partial Offloading (PO) and DVS technology. Especially, EPCO surpasses both the PO with $f_{l_1} = f_{l_{\max}}$ scheme and PO with $f_{l_2} = 0.3 f_{l_{\max}}$ scheme, which exhibits the superiority of DVS technology. Furthermore, EPCO outmatches the FO with optimal f_l scheme when d falls in about [15, 45], which verifies the benefit of PO. This is because that data can be processed parallelly in EPCO scheme. Within a given latency requirement, less bits will be executed in the SMD. Hence, the SMD can use a slower computational speed to save more energy. Similarly, the SMD can choose more suitable transmit power to save energy for the offloaded bits. Next, we explain some interesting phenomena shown in this figure. Firstly, we observe that when d is small (less than about 10 m), all these schemes perform almost the same. This is because the channel is so good that the SMD offloads almost all the computation. Therefore, almost all the energy is spent on data exchanging, which increases with the increasing d. Secondly, the minimum energy consumption in the PO with $f_{l_2} = 0.3 f_{l_{\max}}$ scheme increases until the delay constraint cannot be satisfied for large d. The reason is that the SMD cannot support this application all by itself when using f_{l_2} and it has to offload. As d increases, the communication consumes

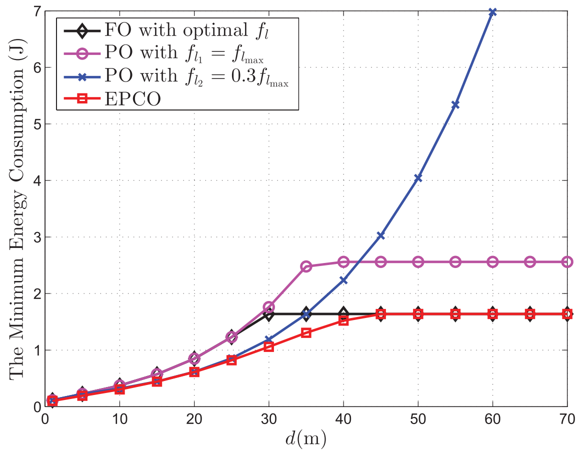


Fig. 5. The minimum energy consumption vs. d.

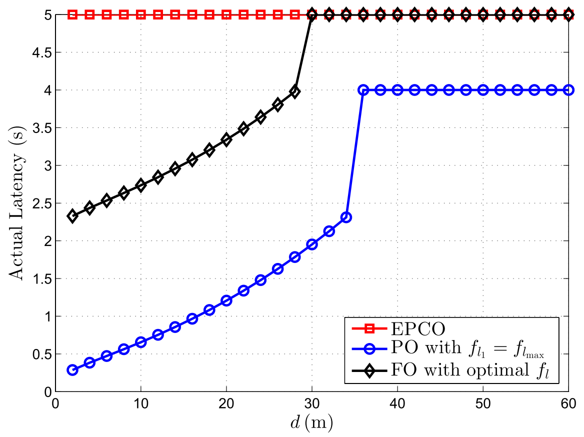


Fig. 6. The actual latency vs. d under $L_{\max} = 5$ s.

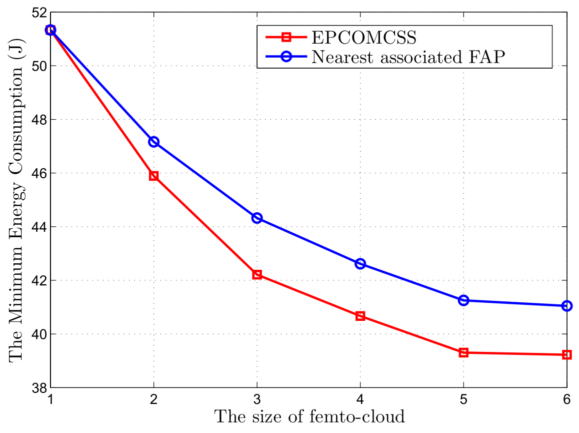


Fig. 7. The minimum energy consumption v.s. N.

more energy. Finally, the other three schemes saturate at the value that equals the energy spent on total execution at the SMD. The reason is that transmission consumes more energy than computing directly at the SMD when d is larger than some value, and hence the SMD prefers to execute the application all by itself. In other words, the conditions under which local execution is optimal are met. At saturation, the minimum energy consumption value in FO with optimal f_l scheme equals that in EPCO, since both schemes use DVS technology to minimize the local energy consumption. Without using DVS, PO with $f_{l_1} = f_{l_{\max}}$ scheme saturates at a larger value. Especially, EPCO surpasses PO with $f_{l_1} = f_{l_{\max}}$ scheme by about 36% in terms of energy consumption.

Fig. 5 shows that EPCO outperforms other schemes in terms of energy consumption. Next, we explain this phenomenon from the perspective of latency via Fig. 6. Fig. 6 presents the actual latency under EPCO, PO with fixed f_l, and FO with optimal f_l scheme, respectively. The actual latency in EPCO is always equal to $L_{\max}$. This is because that once it is below $L_{\max}$, we can choose a lower f_l to further reduce the energy consumption, which results in the actual latency always being $L_{\max}$ no matter under what channel conditions. On the contrary, the actual latency in PO with $f_l = f_{l_{\max}}$ and that in FO with optimal f_l scheme are not always equal to $L_{\max}$. The reasons are as follows: In PO with $f_l = f_{l_{\max}}$ scheme, the SMD cannot adaptively adjust the computational speed to fully utilize $L_{\max}$. Additionally, a larger uplink transmission time may not lead to a less energy consumption due to P_0. Therefore, the latency constraint is inactive. In other words, the actual latency is not equal to $L_{\max}$. Finally, the PO with $f_l = f_{l_{\max}}$ scheme saturates at 4 s, which is the time required by total execution at the SMD. Similarly, in FO with optimal f_l scheme, the SMD chooses total offloading when d is small while local execution when d is large. When the SMD offloads its computation, the latency constraint may be inactive as explained above. However, in our proposed scheme EPCO, the SMD can utilize $L_{\max}$ to save energy when local execution is preferred, and thus the actual latency is equal to $L_{\max}$ when d is large, since the SMD has the capability of DVS.

Fig. 5 and Fig. 6 indicate that EPCO can fully utilize $L_{\max}$ to save energy, which is especially suitable for delay-constrained applications. Taking online game as the example, once $L_{\max}$ is met, the video interface will be smooth, thus satisfying the mobile players. Additionally, it is more important to prolong battery lifetime for the mobile players in such situations.

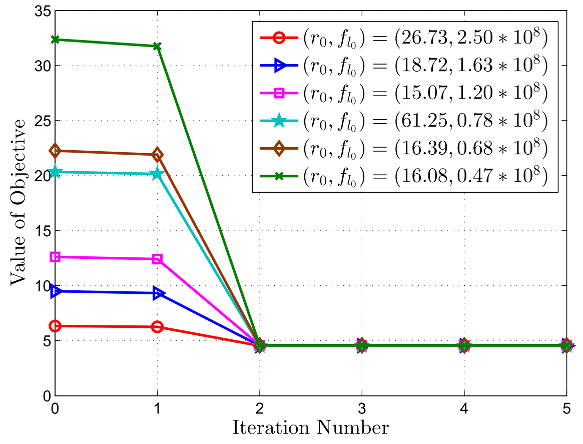
B. Performance of EPCOMCSS

Fig. 7 shows the energy performance versus the size of femto-cloud in the multiple cloud servers scenario. Here, we set $L_{\max} = 0.4$ s. The reason is that it is more meaningful to consider the multiple cloud servers scenario for the applications with stringent latency, since they requires more computation resource. From this figure, we can see that the minimum energy consumption decreases with the increment of N, which means more energy saving could be achieved in the multiple cloud servers scenario. The reasons are as follows: As N increases, richer computational capability of femto-cloud is available. Besides, increasing N leads to greater selection diversity gain of multiple FAPs. We can also see that the optimal association achieves lower energy consumption by considering both channel conditions and computation capability of FAP.

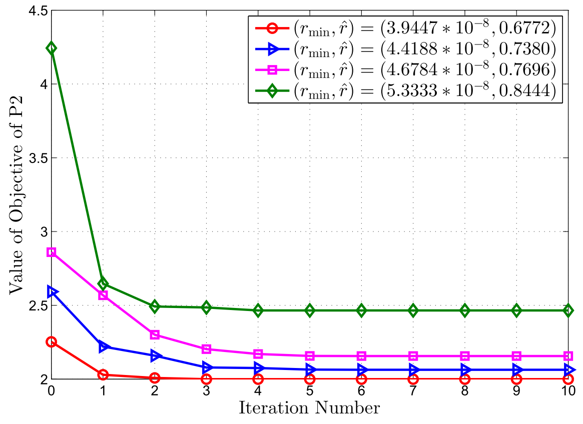
C. Performance of LPCO

In Fig. 8a, we plot the convergence evolution of the outer loop of LPCO with different initial points. It is observed that it always converges fast, which validates Theorem 2. Here, either solving the linear problem once or solving the convex problem once is termed one iteration. To show the overall convergence of LPCO, we further display the convergence evolution of its inner loop, i.e., Algorithm 3 in Fig. 8b. We observe that it has a fast convergence rate and converges typically in a few steps. Thus, LPCO is cost efficient in the computational complexity. In addition, a different channel condition a leads to a different $(r_{\min}, \hat{r})$. Fig. 8b shows that the minimal latency decreases as channel condition a increases, since good channel conditions can reduce the cost of communication, e.g., latency.

Fig. 9 shows the minimum latency under several schemes as a function of d. Here, we adopt such a benchmark, which works like exhaustive search method (LES). Specifically, we first uniformly choose $1000 \times 1000 \times 1000$ (P_t, f_l, λ) points in $(0, P_{t_{\max}}) \times (0, f_{l_{\max}}) \times (0, 1)$ region. Then we pick up the feasible points and calculate the corresponding objectives. Through



(a) Convergence performance of the outer loop of LPCO.



(b) Convergence performance of the inner loop of LPCO, i.e., Algorithm 3.

Fig. 8. Convergence performance of LPCO.

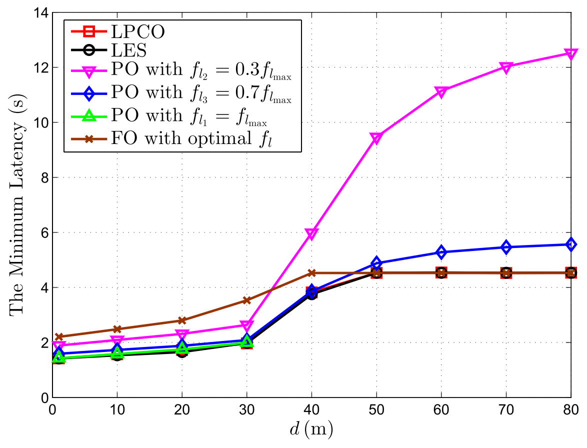


Fig. 9. The minimum latency v.s. d.

comparing these objectives, we could find the "optimal" value. Fig. 9 shows that LPCO performs almost the same as LES, which verifies the superior performance of LPCO. Moreover, through comparing LPCO, PO with $f_{l_2} = 0.3 f_{l_{\max}}$, PO with $f_{l_3} = 0.7 f_{l_{\max}}$, and PO with $f_{l_4} = f_{l_{\max}}$, we can see lower latency can be obtained if the SMD has the capability of DVS, since the SMD can more intelligently manage offloading via DVS technology. Besides, LPCO outmatches FO with optimal f_l, since the SMD can intelligently utilize the resources in the

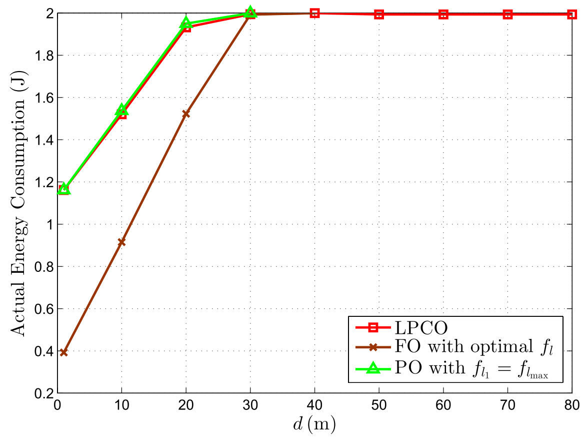


Fig. 10. The actual energy consumption v.s. d.

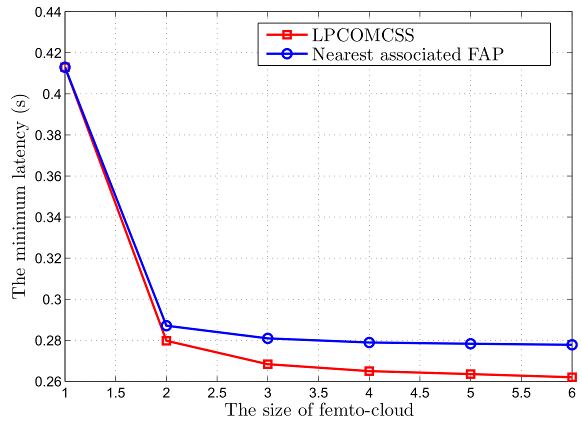


Fig. 11. The minimum latency v.s. N.

SMD and cloud. In Fig. 9, the minimum latency in the six schemes increases with the increasing d. This is because that as d increases, offloading through wireless channel becomes costly, thus leading to more and more data processed locally. Due to the energy consumption constraint, the SMD has to reduce f_l, and thus leads to an increasing latency. When all the data within the application is locally processed, the minimum latency is independent of d. Moreover, it remains at $\min\left\{f_{l_{\max}}, \sqrt{\frac{E_{\max}}{\alpha k I}}\right\}$. Note that the null values indicate that the application can not be supported by the system.

Fig. 10 shows the actual energy consumption v.s. d under LPCO, FO with optimal f_l, and PO with $f_{l_1} = f_{l_{\max}}$ scheme, respectively. Different from that the actual latency in EPCO is always equal to $L_{\max}$ due to using DVS, the actual energy consumption in LPCO is not always equal to $E_{\max}$. The reason is straightforward. Although we can choose a larger f_l to reduce latency once $E_{\max}$ constraint is inactive, we cannot arbitrarily increase f_l due to constrained computational speed of SMD, i.e., $f_{l_{\max}}$, which is verified by that the actual energy consumption of LPCO is almost the same as that of PO with $f_{l_1} = f_{l_{\max}}$.

D. Performance of LPCOMCSS

Fig. 11 shows the minimum latency versus the size of femto-cloud in the multiple cloud servers scenario. From this figure, we can see that the minimum latency decreases with the increment of N. This is because when N increases, richer computational capability of femto-cloud is available. Besides, greater selection diversity gain of multiple FAPs could be obtained. We can also see that the optimal association achieves lower latency by considering both channel conditions and computation capability of FAP.

VI. Conclusions

In this paper, we have investigated partial computation offloading with DVS technology in mobile edge computing and formulated two optimization problems, namely, the ECM problem and the LM problem. To address the ECM problem, we have designed the EPCO algorithm to transform the original problem and obtained the globally optimal solutions in closed-form except for λ. Through the analysis of some special cases, we have got the conditions under which local execution is optimal and achieved a conclusion that total offloading could not be optimal when the SMD has the capability of DVS. Moreover, a multiple cloud servers scenario has been addressed, where the optimal computation distribution among clouds as well as the optimal user association have been derived. Then we proposed LPCO algorithm to solve the LM problem which can achieve good performance. Similarly, we studied the LM problem in a multiple cloud servers scenario. Finally, extensive simulations verified the advantages of the proposed algorithms with respect to energy consumption, latency, and admission probability.

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, 2013.

[2] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: Computation offloading as the bridge," *IEEE Netw.*, vol. 27, no. 5, pp. 28–33, Sep./Oct. 2013.

[3] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[4] 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.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. EuroSys*, Salzburg, Austria, Apr. 2011, pp. 301–314.

[6] 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.

[7] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "CONCERT: A cloud-based architecture for next-generation cellular systems," *IEEE Wireless Commun.*, vol. 21, no. 6, pp. 14–22, Dec. 2014.

[8] FP7 European Project. (2012). *Distributed Computing, Storage and Radio Resource Allocation Over Cooperative Femtocells (TROPIC)*. [Online]. Available: http://www.ict-tropic.eu/

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

[10] 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.

[11] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Proc. IEEE ICC*, Budapest, Hungary, Jun. 2013, pp. 728–732.

[12] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *Proc. IEEE SPAWC*, Darmstadt, Germany, Jun. 2013, pp. 26–30.

[13] 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.

[14] 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.

[15] P. Di Lorenzo, S. Barbarossa, and S. Sardellitti. (2016). "Joint optimization of radio resources and code partitioning in mobile edge computing." [Online]. Available: http://arxiv.org/abs/1307.3835

[16] 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.

[17] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Mar. 2013.

[18] G. Qu, "What is the limit of energy saving by dynamic voltage scaling?" in *Proc. IEEE/ACM ICCAD*, San Jose, CA, USA, Nov. 2001, pp. 560–563.

[19] 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.

[20] Y. Wang *et al.*, "Energy-optimal partial computation offloading using dynamic voltage scaling," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 2695–2700.

[21] G. S. G. Beveridge and R. S. Schechter, *Optimization: Theory and Practice*. New York, NY, USA: McGraw-Hill, 1970.

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

[23] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, to be published.

[24] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[25] S. Chen, Y. Wang, and M. Pedram, "A semi-Markovian decision process based control method for offloading tasks from mobile devices to the cloud," in *Proc. IEEE GLOBECOM*, Atlanta, GA, USA, Dec. 2013, pp. 2885–2890.

[26] M. Molina, O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. IEEE PIMRC*, Washington, DC, USA, Sep. 2014, pp. 1093–1098.

[27] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *Proc. IEEE ICC*, London, U.K., Jun. 2015, pp. 5529–5534.

[28] S. Barbarossa, P. Di Lorenzo, and S. Sardellitti, "Computation offloading strategies based on energy minimization under computational rate constraints," in *Proc. IEEE EuCNC*, Sydney, NSW, Australia, Jun. 2014, pp. 1–5.

[29] X. Wang, W. Chen, and Z. Cao, "ARCOR: Agile rateless coded relaying for cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 60, no. 6, pp. 2777–2789, Jul. 2011.

[30] W. Zhang, Y. Wen, J. Cai, and D. O. Wu, "Toward transcoding as a service in a multimedia cloud: Energy-efficient job-dispatching algorithm," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2002–2012, Jun. 2014.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

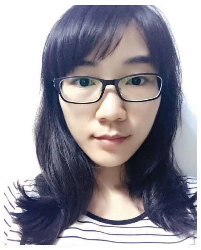
[32] D. T. T. Nga, M.-G. Kim, and M. Kang, "Delay-guaranteed energy saving algorithm for the delay-sensitive applications in IEEE 802.16e systems," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, pp. 1339–1347, Nov. 2007.

[33] S. Chimmanee, "PACS metric based on regression for evaluating end-to-end QoS capability over the Internet for telemedicine," in *Proc. IEEE ICOIN*, Bangkok, Thailand, Jan. 2013, pp. 359–364.

[34] ÓJ. Oueis, E. C. Strinati, and S. Barbarossa, "Small cell clustering for efficient distributed cloud computing," in *Proc. IEEE PIMRC*, Washington, DC, USA, Sep. 2014, pp. 1474–1479.

[35] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. IEEE VTC Spring*, Glasgow, Scotland, May 2015, pp. 1–6.

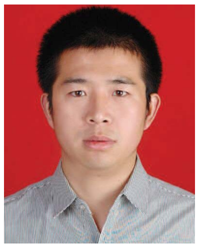
[36] J. Oueis, E. Calvanese-Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on distributed cloud computing," in *Proc. IEEE WCNCW*, Istanbul, Turkey, Apr. 2014, pp. 12–17.
[37] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX HotCloud*, Boston, MA, USA, Jun. 2010, pp. 4–11.
[38] A. Damnjanovic *et al.*, "A survey on 3GPP heterogeneous networks," *IEEE Wireless Commun.*, vol. 18, no. 3, pp. 10–21, Jun. 2011.
[39] A. R. Jensen, M. Lauridsen, P. Mogensen, T. B. Sørensen, and P. Jensen, "LTE UE power consumption model: For system level energy and performance optimization," in *Proc. IEEE VTC Fall*, Quebec City, QC, Canada, Sep. 2012, pp. 1–5.



Yanting Wang received the B.S. degree in communications engineering from Xidian University, Xi'an, China. She is currently pursuing the Ph.D. degree with the State Key Laboratory of ISN. Her research interests include computation offloading, caching, applications of convex optimization theory, and heterogeneous networks.

Min Sheng (M'03–SM'16) received the M.Eng and Ph.D. degrees in Communication and Information Systems from Xidian University, Shaanxi, China, in 1997 and 2000, respectively. She is currently a Full Professor with the Broadband Wireless Communications Laboratory, School of Telecommunication Engineering, Xidian University. Her general research interests include mobile ad hoc networks, wireless sensor networks, wireless mesh networks, third generation (3G)/fourth generation (4G) mobile communication systems, dynamic radio resource management (RRM) for integrated services, cross-layer algorithm design and performance evaluation, cognitive radio and networks, cooperative communications, and medium access control (MAC) protocols. She has published 2 books and over 50 papers in refereed journals and conference proceedings. She is a member of the IEEE. She was the New Century Excellent Talents in University by the Ministry of Education of China, and obtained the Young Teachers Award by the Fok Ying-Tong Education Foundation, China, in 2008.

Xijun Wang (M'12) received the B.S. degree (high honors) in communications engineering from Xidian University, Xi'an, China, in 2005, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. Since 2012, he has been with the School of Telecommunications Engineering, Xidian University, where he is currently an Associate Professor. He visited the Singapore University of Technology and Design, Singapore, from 2015 to 2016. His current research interests include spectrum sharing, LTE unlicensed, cognitive radios, and heterogeneous networks. He has served as a Technical Program Co-Chair of the Wireless Communications Systems Symposium of the IEEE/CIC ICCC 2016, and a Publicity Chair of the IEEE/CIC ICCC 2013. He is a Reviewer for several IEEE journals and has been recognized as an Exemplary Reviewer of the IEEE WIRELESS COMMUNICATIONS LETTERS in 2014. He was a recipient of the best paper award at the IEEE/CIC ICCC 2013.



Liang Wang received the B.S. degree in telecommunications engineering and the Ph.D. degree in communication and information systems from Xidian University in 2009 and 2015, respectively. He is currently a Lecturer with the School of Computer Science, Shaanxi Normal University. His research interests focus on dynamic spectrum access in cognitive radio networks, energy-efficient transmission, applications of convex optimization theory, and robust design in wireless communications networks.

Jiandong Li (SM'05) received the B.E., M.S., and Ph.D. degrees in communications engineering from Xidian University, Xi'an, China, in 1982, 1985, and 1991, respectively. He has been a faculty member of the School of Telecommunications Engineering with Xidian University since 1985, where he is currently a Professor and Vice Director of the Academic Committee of the State Key Laboratory of Integrated Service Networks. He was a Visiting Professor with the Department of Electrical and Computer Engineering, Cornell University, from 2002 to 2003. He served as the General Vice Chair for ChinaCom 2009 and the TPC Chair of the IEEE ICCC 2013. He was awarded as a Distinguished Young Researcher from NSFC and a Changjiang Scholar from the Ministry of Education, China, respectively. His major research interests include wireless communication theory, cognitive radio, and signal processing.