Energy-Efficient Computation Offloading in Mobile Edge Computing Systems with Uncertainties

Tianxi Ji, Changqing Luo, Lixing Yu, Qianlong Wang, Siheng Chen, Arun Thapa, and Pan Li

Abstract—Computation offloading is indispensable for mobile edge computing (MEC). It uses edge resources to enable intensive computations and save energy for resource-constrained devices. Existing works generally impose strong assumptions on radio channels and network queue sizes. However, practical MEC systems are subject to various uncertainties rendering these assumptions impractical. In this paper, we investigate the energy-efficient computation offloading problem by relaxing those common assumptions and considering intrinsic uncertainties in the network. Specifically, we minimize the worst-case expected energy consumption of a local device when executing a time-critical application modeled as a directed acyclic graph. We employ the extreme value theory to bound the occurrence probability of uncertain events. To solve the formulated problem, we develop an ϵ-bounded approximation algorithm based on column generation. The proposed algorithm can efficiently identify a feasible solution that is less than $(1+\epsilon)$ of the optimal one. We implement the proposed scheme on an Android smartphone and conduct extensive experiments using a real-world application. Experiment results corroborate that it will lead to lower energy consumption for the client device by considering the intrinsic uncertainties during computation offloading. The proposed computation offloading scheme also significantly outperforms other schemes in terms of energy saving.

Index Terms—Mobile edge computing, computation offloading, network uncertainties, energy efficiency.

I. INTRODUCTION

THE breakthroughs of hardware and software technologies have resulted in the surge of various mobile applications, such as real-time online gaming and anywhere anytime online social interactions. Mobile devices (e.g., smartphones and wearable devices) executing these applications are usually computation-constrained and battery-limited [1], which can hinder the development of mobile applications. To address this issue, academia and industry propose to employ moible edge computing (MEC) to let mobile devices offload their computations to edge servers with abundant resources in the proximity [2], [3]. As a result, mobile devices are able to support computation-intensive and energy-hungry mobile applications in MEC systems.

So far, researchers have conducted extensive works on energy-efficient computation offloading schemes [4]–[16]. Specifically, some works are based on physical layer design of wireless communications [5], [9]. Some designed schemes from the perspective of cross-layer design [6], [7], [17]. Some focused on designing offloading schemes for time-critical applications [10], [12]–[15], [18]. Previous studies generally impose strong assumptions on communication channels and network queue sizes; they develop computation offloading schemes based on specific radio channels and network queue models. For example, Zhang et al. [13] proposed the one-climb policy to offload computations under independent and identically distributed (i.i.d.) stochastic channels. Geng et al. [4] proposed a critical path based solution to allocate computation tasks to a mobile multi-core device and an edge server without considering dynamic network queues.

However, a practical MEC system is always subject to intrinsic uncertainties like unknown communication channels and network queue size. Factors like weather, obstacles, movements can easily affect radio channels [19]. For example, it has been shown that the varying locations of the local mobile transmitter and receiver can lead to uncertainties in channel law as well as arbitrarily varying communication bit rates and energy consumption [20]. The network queue size is also highly dependent on data volume, network condition, arrival/processing methods, etc., [21]. Recent studies have shown that sudden burst of serving requests can overflow the edge servers' resource and cause MEC failure [22], [23]. Consequently, it is impossible to use approximated mathematical models to accurately capture the underlying dynamics of radio channel and network queue size [24]. Since the strong assumptions on radio channels and network queue size cannot hold in practical MEC systems, simply applying computation offloading schemes developed based on the two assumptions can cause improper parameter deployment at the wireless link layer and even upper layers, which incurs performance degradation. This is because the parameter deployment at the wireless link layer and upper layers, e.g., transmission and reception power, channel allocation, and routing, are highly related to the radio channels and network queue size. Particularly, in Section VII-D, we will empirically corroborate the fact that strong assumptions on channel conditions and queuing delay will result in much higher energy consumption at the client device in computation offloading.

In light of this, we are motivated to investigate the computation offloading problem by taking into account intrinsic

T. Ji and P. Li are with the Department of Electrical, Computer, and Systems Engineering, Case Western Reserve University, Cleveland, OH, 44106 USA (e-mail: txj116@case.edu; lipan@case.edu).

C. Luo is with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: cluo@vcu.edu).

L. Yu is with School of Information Science and Engineering, Yunnan University, Yunan, 650500 China (e-mail: yulixing@ynu.edu.cn).

Q. Wang is with Department of Computer and Information Sciences, Towson University, Towson, MD 21252 (e-mail:qwang@towson.edu).

S. Chen is with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240 China (e-mail: sihengc@sjtu.edu.cn).

A. Thapa is with the Department of Electrical Engineering, Tuskegee University, Tuskegee, AL 36088 USA (e-mail: athapa@tuskegee.edu).

uncertainties in practical MEC systems. We model the mobile application as a directed acyclic graph (DAG), and handle the execution dependency by explicitly considering the parent and children sets of each computation module. To efficiently solve the formulated the energy-efficient computation offloading problem, we develop a column generation based ϵ-bounded algorithm with theoretical optimality guarantee. The main contributions of this paper are summarized as follows:

- We study energy-efficient computation offloading by lifting strong assumptions on communication channels and network queue size imposed by previous studies.
- We design an energy-efficient computation offloading scheme for executing a mobile application modeled as a DAG in a practical MEC system with uncertainties. Particularly, we employ the extreme value theory to bound the occurrence probability of uncertain events.
- We formulate the energy-efficient computation offloading problem that is subject to the application execution time and develop an efficient column generation based algorithm to solve it. In particular, we also provide an ϵ-bounded approximate solution with theoretical guarantee on the optimality of the proposed scheme.
- We propose computation offloading principles for mobile applications with only sequential or parallel module dependency in practical MEC systems with uncertainties.
- We implement our proposed offloading scheme on an Android platform, and conduct extensive experiments by executing a real-world application, i.e., Smart Diagnosis. Experiment results corroborate that we can reduce the energy consumption of the client device by explicitly considering the intrinsic uncertainties in computation offloading, and also demonstrate that our proposed scheme significantly outperforms other state-of-the-art computation offloading schemes in terms of energy saving.

Roadmap. We introduce the related works in Section II. Section III describes the considered system model. We present the process of handling the randomness caused by uncertainties of MEC systems in Section IV, followed by the formulation of the energy-efficient computation offloading problem in Section V. We develop the column generation based algorithm to solve the formulated problem, and provide computation offloading principles for special mobile applications in Section VI. We show our experiment results in Section VII, and conclude the paper in Section VIII.

II. RELATED WORKS

Quite a few works have studied the problem of energy-efficient computation offloading in MEC systems [3]–[7], [9], [10], [12]–[15], [25]. From the communication perspective, these existing studies can be generally categorized into the physical layer design based, the cross-layer design based, and the time-critical application based. We discuss them as follows.

Physical layer design based schemes. This line of works focus on the impact of electronic circuit transmission on offloading decisions and energy consumption. For example, Geng et al. [4] propose a critical path based solution to recursively check the task modules and move them to the right

CPU cores of a multicore mobile device to save its energy consumption. Wang et al. [5] take advantage of dynamic voltage scaling technology to adjust the transmission power at the physical layer in the course of computation offloading. Sardellitti et al. [9] jointly optimize the radio resources, constellation size, and CPU cycles per second of the mobile device to minimize the energy consumption of local device.

Cross-layer design based schemes. This category of works attend to devise offloading systems by exploiting the resources in different layers in MEC systems. For example, Barbarossa et al. [17] propose a joint framework encompassing a fading channel depending on the number of antennas and packet retransmission strategies to determine the joint allocation of radio resources and computation tasks. Zhang et al. [6] incorporate the multi-access characteristics of the 5G heterogeneous network to jointly optimize offloading and radio resource allocation. Yu et al. [7] minimize the energy consumption of local execution by jointly investigating the subcarrier allocation for offloaded tasks and local CPU time allocation.

Time-critical mobile application based schemes. These works concentrate on the development of proper delay-sensitive scheduling mechanism to meet the quality-of-service requirements of mobile users in computation offloading. For example, Hermes [10], a fully polynomial time approximation scheme, minimizes the application execution latency while meeting the prescribed resource utilization constraints. Jia et al. [12] propose a heuristic programming partition scheme to maximize the parallelism so as to minimize the completion time of an application. Zhang et al. [13] propose a one-climb-policy to offload application with only sequential dependency while meeting a time deadline. Mahmoodi et al. [14] apply integer programming to develop a wireless aware joint scheduling and computation offloading to maximize energy saving for devices while satisfying the execution deadline. Liu et al. [15] propose a latency-optimal offloading scheme by considering the combination of the queueing state in the task buffer, the execution state of the device's processing unit and transmission unit. Meng et al. [26] use an infinite horizon average cost Markov decision process to characterize the interactions between local device and edge server, and develop a delay-optimal multilevel water-filling computation offloading solution. Hou et al. [25] consider computation offloading for latency-sensitive applications in Internet of Vehicles.

All these previous studies impose strong assumptions on communication channels and network queue size. However, in practical MEC systems, radio channels and network queue size are usually dynamic and the patterns of the dynamics are unable to be captured accurately [24]. Particularly, channel status can also be very unpredictable even during the offloading of task modules of an application. As a result, those assumptions cannot hold in real world applications. Although some works, e.g., [10], adapt an online optimization scheme to continuously probe the radio channels of the unknown dynamic environments, it cannot consider potential extreme cases sufficiently and will introduce extra communication overhead. Therefore, to develop an energy-efficient computation offloading scheme for a practical MEC system is still an open issue.

III. System Model

A. Mobile Application Model

We consider a mobile application that is composed of mutually dependent computation task modules (i.e., procedures/components), which need to be executed in a specific order. Thus, a mobile application can generally be modeled as a directed acyclic graph (DAG) $G = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ and $\mathcal{E}$ are the sets of nodes (i.e., task modules) and directed edges (i.e., indicating the dependency between two task modules), respectively. This modeling has been widely adopted in the computation offloading literature, such as, [2], [10], [11], [13], [16], which capture the inter-dependency among different task modules using directed acyclic call-graph. In practice, the partition of application into task modules can be obtained using task profilers [27], [28] or manually decided by the mobile user. In Fig. 1, we show an example of a DAG modeled real-world mobile mobile application called Smart Diagnosis.
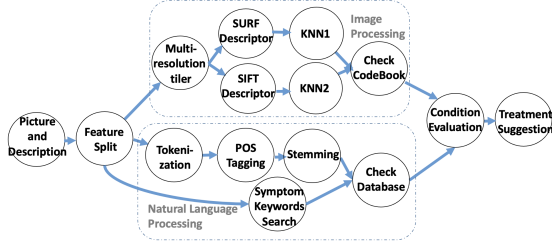

Fig. 1. An example of a DAG-modeled mobile application, Smart Diagnosis.

A task module $n \in \mathcal{N}$ of a mobile application is defined by ω_n, where ω_n is the computation workload (in CPU cycles) of module n. The dependency between two task modules is defined by $o_{n,k}$, for $k \in \mathcal{C}(n)$, where $o_{n,k}$ is the output data size (in bits) from n to k and $\mathcal{C}(n)$ is the set of child nodes of module n. Likewise, we also denote by $\mathcal{M}(n)$ the set of parent nodes of module n. Note that, ω_n's and $o_{n,k}$'s can be inferred by task profilers [27], [28] before running any offloading scheme. Table I lists the frequently used notations.

B. MEC System Model

We consider a typical MEC system, in which task modules of a mobile application are offloaded over uncertain radio channels to edge servers. A task module is waiting in a queue before it is transmitted over a wireless channel. In addition, to support computation offloading, the edge server assigns virtual machines to execute task modules offloaded by client devices. We also take into account a time-slotted MEC system; the time is divided into slots with equal duration Δ, and the discrete time period $((t-1)\Delta, t\Delta]$ is referred to as slot t. To guarantee the quality of service, the entire mobile application is considered to complete before T timeslots.

C. Computation Execution in the MEC System

1) The Location of Executing A Task Module: To model the module execution location, we first define a binary variable $\mathbf{x}_{n,t}^p$ to indicate whether module n terminates its execution at the tth slot at a client device (i.e., $p = 0$) or at an edge server

(i.e., $p = 1$). $\mathbf{x}_n^0 \in \{0,1\}^T$ and $\mathbf{x}_n^1 \in \{0,1\}^T$ are 1-sparse vectors that recording values of $\mathbf{x}_{n,t}^0$ and $\mathbf{x}_{n,t}^1$, respectively. As a result, we have $c_n = \sum_{t=1}^T \mathbf{x}_{n,t}^0 = 1$ and $s_n = \sum_{t=1}^T \mathbf{x}_{n,t}^1 = 1$ for executing task module n at a client device and an edge server, respectively. Since a task module can be executed at either a client device or the edge server, we have

$$c_n + s_n = 1, \forall n \in \mathcal{N}. \tag{1}$$

Particularly, for practical mobile applications, the first and last task module are usually used for initializing a mobile application and displaying computation results [2]. Both of them need to be done at the client. Hence,

$$c_1 = 1 \quad \text{and} \quad c_N = 1. \tag{2}$$

2) Module Execution Completion: The execution completion time of a module n can be calculated by $\sum_{p=0}^1 \sum_{t=1}^T t\mathbf{x}_{n,t}^p$. Due to a time-critical mobile application, the completion time of the last task module is subject to

$$\sum_{t=1}^T t\mathbf{x}_{N,t}^0 \leq T. \tag{3}$$

3) Task Module Execution Dependency: Due to the interaction among task modules, the completion of executing a task module depends on its parent modules. Specifically, the starting time to execute a module n should be no earlier than the time when its parent module m ($\forall m \in \mathcal{M}(n)$) completes execution plus the time for possible output data queuing and transmission between the client device and the edge server. Thus, we can model the module execution dependency by

$$\sum_{p=0}^1 \sum_{t=1}^T t\mathbf{x}_{m,t}^p + c_m s_n \frac{Q_{m,n}^u + o_{m,n}}{R_u} + s_m c_n \frac{Q_{m,n}^d + o_{m,n}}{R_d}$$
$$\leq \sum_{p=0}^1 \sum_{t=1}^T t\mathbf{x}_{n,t}^p - s_n \frac{\omega_n}{f_s} - c_n \frac{\omega_n}{f_c}, \forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}, \tag{4}$$

where $Q_{m,n}^u$ (resp. $Q_{m,n}^d$) is a random variable presenting the existing queue lengths of the out-going buffers at a client device (resp. edge server) when transmitting $o_{m,n}$ over uncertain wireless channels, f_c and f_s are the CPU frequencies of the client device and virtual machine, respectively, and R_u (resp. R_d) is a random variable presenting the bit rates for uploading data to (resp. downloading data from) the edge server. For instance, $\frac{Q_{m,n}^u + o_{m,n}}{R_u}$ calculates the overall time of transmitting the output data of size $o_{m,n}$ and the existing buffered data of size $Q_{m,n}^u$ to the edge server under a random transmission rate R_u. Based on the Shannon-Hartley theorem, the bit rates R_u and R_d can be obtained by $R = B \log_2(1 + SNR)$, where B is the bandwidth and SNR is the nondeterministic signal-to-noise ratio. Note that in (4) we do not have any assumptions on the queuing models (e.g., Poisson distributed task arrival rates and exponential distributed job processing time assumed in the M/M/1 model [29]), and the uncertainties introduced by queue length and bit rates will be addressed by the extreme value theory in the following section.

TABLE I
FREQUENTLY USED NOTATIONS.

Notations	Descriptions	Notations	Descriptions
$\mathcal{N}$	set of task modules in an application	$c_n \in \{0,1\}$	whether or not module n is executed at local device
$\mathcal{E}$	set of dependency connections among modules	$s_n \in \{0,1\}$	whether or not module n is offloaded to edge server
ω_n	workload (in CPU cycles) of task module $n \in \mathcal{N}$	$Q^u_{m,n}$	uncertain existing queue lengths at the client device
$o_{n,k}$	output data size (in bits) from module n and to k	$Q^d_{m,n}$	uncertain existing queue lengths at the edge server
$\mathcal{C}(n)$	set of children nodes of module n	f_c (f_s)	CPU frequencies at local device (edge server)
$\mathcal{M}(n)$	set of parent nodes of module n	R_u (R_d)	uncertain uploading (downloading) bit rates
Δ	duration of a time slot in the MEC system	P_u (P_d)	uncertain uploading (downloading) power consumption
T	completion deadline of the mobile application	ϵ_m	upper bound of extreme event occurrence probability
$\mathbf{x}^p_{n,t}$	whether or not module n terminates execution in	ϵ	approximation factor of the optimal solution
$\in \{0,1\}$	tth slot at client device $p=0$ (or edge $p=1$)	Ψ	worst-case expected energy consumption of local device

IV. HANDLING THE UNCERTAINTIES

Due to the uncertain radio channels and network queue length, $Q^u_{m,n}, Q^d_{m,n}, R_u$, and R_d in (4) are random variables. To deal with the uncertainties, we propose to apply the extreme value theory and bound the time for the data queuing and transmission in the worst case. In so doing, we can draw broad conclusions about computation offloading in MEC systems without relying on specific case of radio channel and queuing model (e.g., the block-fading channel, constant or Poisson distributed queue size). The reason is that the extreme value theory is a technique describing the unusual rather than the usual, even if the network parameters change over time, their values will still be upper-bounded by their extreme values with a large probability. By explicitly accounting for the uncertainties, our methodology is also superior than the classical ones, e.g., [29], which applies large deviation theory to handle rare events in the network.

Specifically, we consider the Generalized Extreme Value (GEV) distribution instead of its specific cases commonly used by existing works [30]–[33] when applying the extreme value theory to wireless radio channels. This is because these methods need to choose the most appropriate special case of the GEV distribution, and the followed statistic inferences do not allow any flexibility once the special case is determined.

We start with the case that a client device offloads module n to the edge server, i.e., $s_n = 1$, $c_n = 0$, and $\mathbf{x}^0_n = \mathbf{0}$, and rewrite (4) as

$$c_m \frac{Q^u_{m,n} + o_{m,n}}{R_u} \leq \sum_{t=1}^{t=T} t\mathbf{x}^1_{n,t} - \sum_{p=0}^{1}\sum_{t=1}^{t=T} t\mathbf{x}^p_{m,t} - \frac{\omega_n}{f_s}, \quad (5)$$

$\forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}$. Define random variable $V = \frac{Q^u_{m,n} + o_{m,n}}{R_u} \in [0, +\infty)$, which is the quotient between two random variables. According to [34], one generic way to derive the distribution of quotient from the joint distribution of the two involved random variables (i.e., $Q^u_{m,n} + o_{m,n}$ and R_u in our case) is by integration of the following form $f_V(v) = \int |r| f_{Q,R}(vr, r) dr$, where $f_{Q,R}(vr, r)$ is the joint PDF of $Q^u_{m,n} + o_{m,n}$ and R_u, v and r denotes one pair of instance of random variable V and R_u, and the multiplication of u and v represent one instance of the size of the buffered data ($Q^u_{m,n}$) plus the size of $o_{m,n}$.

The common way of handling the randomness is to bound the expected time of the data transmission and queuing by taking the expected value of V. However, due to no prior knowledge of the distributions of $Q^u_{m,n}$ and R_u, it is impossible to obtain $f_V(v)$ in practice. Moreover, only relying on the expected value is insufficient for ultra-reliable low latency communication (URLLC) applications, as it might fail to quantify some extreme events in the wireless communication systems [33], [35]. To address this issue, we consider the kth order statistic, i.e., $V_{(k)} = \max_{1 \leq i \leq k} V_i$, to bound the time of data transmission and queuing in the worst case. $\{V_1, V_2, \cdots V_k\}$ are samples drawn from $f_V(v)$. Thus, we can rewrite (5) as

$$c_m s_n V_{(k)} \leq \sum_{t=1}^{t=T} t\mathbf{x}^1_{n,t} - \sum_{p=0,t=1}^{p=1,t=T} t\mathbf{x}^p_{m,t} - s_n \frac{\omega_n}{f_s},$$

$\forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}$, and bound the probability that $V_{(k)}$ exceeds its upper bound by

$$\Pr(c_m V_{(k)} \geq b^{c,s}_{m,n}) \leq \epsilon_m, \forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}, \quad (6)$$

where $b^{c,s}_{m,n} = \sum_{t=1}^{t=T} t\mathbf{x}^1_{n,t} - \sum_{p=0,t=1}^{p=1,t=T} t\mathbf{x}^p_{m,t} - \frac{\omega_n}{f_s}$ and $\epsilon_m \ll 1$. To characterize the PDF of $V_{(k)}$, we resort to the extreme value theory and explore the Generalized Fisher-Tippett-Gnedenko theorem to bound the worst-case data transmission time and queuing delay in the MEC system.

Theorem 1 (Generalized Fisher-Tippett-Gnedenko Theorem [36]): Let $V_i, 1 \leq i \leq k$, be a sequence of random variables with a common PDF as $f_V(v)$, and $V_{(k)} = \max_{1 \leq i \leq k} V_i$. If there exist two sequences of constants $a_k \in \mathcal{R}^+$ and $b_k \in \mathcal{R}$ such that $\Pr(\frac{V_{(k)} - b_k}{a_k} \leq z) \approx G(z)$ as $k \rightarrow \infty$ for a non-degenerate distribution function G, where $\approx$ stands for "asymptotic to". Then G is a member of the GEV family with a cumulative distribution function (CDF) as

$$G(z) = \begin{cases} \exp\left\{ -[1 + \xi(\frac{z-\mu}{\sigma})^{-1/\xi}]\right\}, & \text{if } \xi \neq 0, \\ \exp\left\{ -exp[-\frac{z-\mu}{\sigma}]\right\}, & \text{if } \xi = 0, \end{cases}$$

defined over the set $\{z : 1 + \xi(z - \mu)/\sigma > 0\}$, where $\mu \in \mathcal{R}$, $\xi \in \mathcal{R}$, and $\sigma \in \mathcal{R}^+$ are the location parameter, shape parameter, and scale parameter, respectively.

Remark 1: In particular, $G(z)$ becomes the Gumbel distribution, Fréchet distribution, and Weibull distribution, when $\xi = 0$, $\xi > 0$, and $\xi < 0$, respectively. The three distributions are all members of GEV family distribution and can be written in the form of $G(z)$ (with different parameters, i.e., μ, ξ, and σ). Theorem 1 establishes the CDF in terms of

$\Pr(\frac{V_{(k)} - b_k}{a_k} \leq z)$ instead of $\Pr(V_{(k)} \leq z)$. This is because as k increases the corresponding probability density function may degenerate to a point mass (page 46 of [36]). It can be avoided by allowing a linear renormalization of the block maximum $V_{(k)}$, i.e., $\frac{V_{(k)} - b_k}{a_k}$, and the renormalized variable is asymptotically distributed as $G(z)$ as k increases. Note that if the renormalization of a variable follows a GEV family distribution, the variable itself also follows a GEV family distribution with different parameters (page 49 of [36]), i.e., the CDF of $V_{(k)}$ takes the same form as $\Pr(\frac{V_{(k)} - b_k}{a_k} \leq z)$ (i.e., $G(z)$), but with different μ, ξ, and σ.

Based on Remark 1, to bound the probability (in (6)) by ϵ_m, we can bound the corresponding extreme quantile of $V_{(k)}$, i.e., $z^u_{\epsilon_m}$. Then, we have $\Pr(V_{(k)} \geq z^u_{\epsilon_m}) = \epsilon^u_m$. Thus, (6) becomes

$$c_m z^u_{\epsilon_m} \leq b^{c,s}_{m,n}, \forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}. \tag{7}$$

Note that $b^{c,s}_{m,n}$ is determined by the binary variable $\mathbf{x}^p_n$, and $z^u_{\epsilon_m}$ is a constant given the probability ϵ_m and a GEV distribution. By inverting the function of G (which can also describe the CDF of $V_{(k)}$, but with different μ, ξ, and σ), the extreme quantile is

$$z^u_{\epsilon_m} = \begin{cases} \mu - \dfrac{\sigma}{\xi}[1 - \{-\log(1 - \epsilon_m)\}^{-\xi}], & \text{if } \xi \neq 0, \\ \mu - \sigma \log\{-\log(1 - \epsilon_m)\}, & \text{if } \xi = 0. \end{cases} \tag{8}$$

Parameters of the GEV distribution $\mathbf{\Theta} = (\xi, \sigma, \mu)$ can be inferred via the maximum likelihood estimation (MLE), which will be elaborated in Section VII-B.

Now, we consider n is executed at the client device, i.e., $c_n = 1, s_n = 0$ and $\mathbf{x}^1_n = \mathbf{0}$. We can rewrite (4) as

$$s_m \frac{Q^d_{m,n} + o_{m,n}}{R_d} \leq \sum_{t=1}^{t=T} t\mathbf{x}^0_{n,t} - \sum_{p=0,t=1}^{p=1,t=T} t\mathbf{x}^p_{m,t} - c_n \frac{\omega_n}{f_c},$$

$\forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}$. We notice that $\frac{Q^d_{m,n} + o_{m,n}}{R_d}$ is also a random variable. Thus, we also apply the kth order statistics and the extreme value theory to obtain

$$s_m z^d_{\epsilon_m} \leq b^{s,c}_{m,n}, \forall m \in \mathcal{M}(n), \forall n \in \mathcal{N}. \tag{9}$$

where $z^d_{\epsilon_m}$ is the extreme quantile and $b^{s,c}_{m,n} = \sum_{t=1}^{t=T} t\mathbf{x}^0_{n,t} - \sum_{p=0,t=1}^{p=1,t=T} t\mathbf{x}^p_{m,t} - \frac{\omega_n}{f_c}$ is the nondeterministic upper bound.

V. PROBLEM FORMULATION

We consider to minimize a client device's energy consumption under the constraint of the completion time of the mobile application. Specifically, a client device's energy consumption for task execution and data transmission can be given by

$$E = \sum_n c_n \kappa \omega_n f_c^2 + P_u \sum_{m \in \mathcal{M}(n)} \sum_n c_m s_n \frac{o_{m,n}}{R_u} + P_d \sum_{m \in \mathcal{M}(n)} \sum_n s_m c_n \frac{o_{m,n}}{R_d}, \tag{10}$$

where κ is a constant related to the hardware architecture of the client device [2], [28], P_u and P_d are random variables represent the power consumption of the device for transmitting data to and receiving data from the server. Due to the uncertain

radio channels, the client device's energy consumption is nondeterministic as well.

To design a robust computation offloading scheme for transmitting offloaded computation over uncertain radio channels, we first formulate the energy consumption in the worst case:

$$\bar{E} = \sum_n c_n \kappa \omega_n f_c^2 + \sum_{m \in \mathcal{M}(n)} \sum_n c_m s_n o_{m,n} J_{(k)} + \sum_{m \in \mathcal{M}(n)} \sum_n s_m c_n o_{m,n} H_{(k)}, \tag{11}$$

where $J_{(k)} = \max_{1 \leq i \leq k} J_i$ and $H_{(k)} = \max_{1 \leq i \leq k} H_i$ are the k-th order statistics. $\{J_1, J_2, \cdots, J_k\}$ and $\{H_1, H_2, \cdots, H_k\}$ are the sample sets drawn from $f_{P_u/R_u}(p_u/r_u)$ and $f_{P_d/R_d}(p_d/r_d)$, respectively (by following statistical conventions, P_u/R_u and P_d/R_d denote the random variables, and p_u/r_u and p_d/r_d are the instances). Then, we define the expected energy consumption of the client device in the worst case as follows:

$$\Psi = \mathbb{E}_{\substack{J_{(k)} \sim GEV(j_{(k)}) \\ H_{(k)} \sim GEV(h_{(k)})}} (\bar{E}) \sum_n c_n \kappa \omega_n f_c^2 + \sum_{m \in \mathcal{M}(n)} \sum_n \left(c_m s_n o_{m,n} \mathbb{E}_{J_{(k)} \sim GEV(j_{(k)})}(J_{(k)}) + s_m c_n o_{m,n} \mathbb{E}_{H_{(k)} \sim GEV(h_{(k)})}(H_{(k)}) \right), \tag{12}$$

where $GEV(j_{(k)})$ and $GEV(h_{(k)})$ are the inferred probability distributions of $J_{(k)}$ and $H_{(k)}$, respectively. The expected value of GEV distributed variable can be calculated as [36]

$$\mathbb{E}_{X \sim GEV(x)}(X) = \begin{cases} \mu + \sigma(g_1 - 1)/\xi, & \text{if } \xi \neq 0, \xi < 1, \\ \mu + \sigma\gamma, & \text{if } \xi = 0, \\ \infty, & \text{if } \xi \geq 1, \end{cases} \tag{13}$$

where $g_1 = \Gamma(1 - \xi)$ and γ is Euler's constant. For notational simplicity, we have $\theta_u = \mathbb{E}(J_{(k)})$ and $\theta_d = \mathbb{E}_{H_{(k)} \sim GEV(h_{(k)})}(H_{(k)})$.

Therefore, we formulate the energy efficient computation offloading problem as

$$\text{MP}: \min_{\mathbf{x}^p_n \in \{0,1\}^T, \ p \in \{0,1\}} \Psi \tag{14}$$
$$\text{s. t.} \quad (1), (2), (3), (7), (9).$$

VI. A COLUMN GENERATION BASED EFFICIENT ϵ-BOUNDED APPROXIMATION ALGORITHM

The formulated optimization problem in (14) is a quadratically constrained binary quadratic programming. In particular, the decision variables are $2N$ 1-sparse binary vectors in $\{0, 1\}^T$, resulting in a total of 2^{2NT} degrees of freedom. This makes the formulated problem computationally-intractable. To address this issue, we propose an ϵ-bounded approximate algorithm based on the column generation (CG) technique that is usually used to solve linear or nonlinear programming in an iterative way [37] and is widely used in wireless communication [38], [39]. Particularly, our proposed scheme can find the ϵ-bounded approximate offloading policy, and especially the optimal policy when setting $\epsilon = 0$.

Specifically, we consider MP in (14) as a master problem, and solve it via starting with a restricted master problem (i.e., called RMP) which minimizes Ψ given only a fraction of the columns. We can add a new column into RMP only if

it has been determined by a price problem (i.e., called PP) to be profitable for further reducing Ψ. To be more specific, at each iteration, PP first determines whether any column $(\mathbf{x}_n^p)$ uninvolved in RMP can lead to a negative reduced cost, and then the one with the most negative reduced cost will be added into RMP. The iteration terminates at or satisfyingly close to the optimal solution.

A. Restricted Master Problem

The general RMP that minimizes Ψ by considering only a subset of all decision columns is

$$\text{RMP}: \quad \min_{\mathbf{x}_n^p \in \mathcal{S}_I} \quad \widetilde{\Psi} \tag{15}$$
$$\text{s. t.} \quad (1), (2), (3), (7), (9),$$

where $\mathcal{S}_I$ is a subset of all columns considered in MP, denoted as $\mathcal{S} = \{\mathbf{x}_1^0\} \cup \{\mathbf{x}_N^0\} \cup \{\mathbf{x}_n^p | \mathbf{x}_n^p \in \{0,1\}^T, n \in \{2, \cdots, N-1\}, p \in \{0,1\}\}$, and $\widetilde{\Psi}$ is the expected worst case energy consumption calculated with respect to $\mathcal{S}_I$. At the same time, we set all the elements in set $\mathcal{S}_I^c = \mathcal{S} \backslash \mathcal{S}_I$ (the complement set of $\mathcal{S}_I$) to be zero vectors. Given a feasible offloading policy determined by $\{\mathbf{x}_n^p, n \in \mathcal{N}, p \in \{0,1\}\}$, $\widetilde{\Psi}$ becomes deterministic. Thus, RMP is reduced to a feasibility checking problem. Specifically, with a provided offloading policy, we need to check whether there exists at least one feasible solution in the set defined by the constraints in (15).

Without loss of generality (w.l.o.g.), we can start with a special RMP in which all computation task modules are assigned to the client device ($c_n = 1, s_n = 0, \forall n \in \mathcal{N}$). As a result, $\mathcal{S}_I = \{\mathbf{x}_n^0 | \forall n \in \mathcal{N}\}$ and $\widetilde{\Psi} = \sum_n \kappa \omega_n f_c^2$. Then, one feasible solution for this special RMP is $\{\mathbf{x}_{n,t}^0 = 1, \text{if } t = \sum_{i=1}^n \lceil \frac{\omega_n}{f_c} \rceil, \mathbf{x}_n^1 = \mathbf{0} | n \in \mathcal{N}\}$.

Since RMP involves only a subset of $\mathcal{S}$ used by MP, i.e., $\mathcal{S}_I \subset \mathcal{S}$, $\widetilde{\Psi}$ serves as an upper bound on the optimal result of MP [37]. By introducing more columns to RMP, the upper bound may be lowered down further. Therefore, we need to determine which column is the most profitable to MP in terms of negative reduced cost, and when the optimal result of RMP is exactly the same as or satisfyingly close to the optimal result of MP. This is achieved by solving the corresponding price problem, described in the following.

B. Adding A New Column to RMP via PP

In each iteration, when RMP is solved, we need to check whether adding any new column can lead to better solution to MP. In particular, for each $\mathbf{x}_n^1$ in $\mathcal{S}_I^c$ (i.e., $\mathcal{S} \backslash \mathcal{S}_I$), we need to examine whether it has a negative reduced cost. The reduced cost ζ_n for a column $\mathbf{x}_n^1 \in \mathcal{S}_I^c$ is, according to [37],

$$\zeta_n = \sum_{m \in \mathcal{M}(n)} c_m s_n o_{m,n} \theta_u + \sum_{k \in \mathcal{C}(n)} s_n c_k o_{n,k} \theta_d$$
$$- \sum_{m \in \mathcal{M}(n)} \pi_{m,n} b_{m,n}^{c,s} - \sum_{k \in \mathcal{C}(n)} \pi_{n,k} b_{n,k}^{s,c}, \tag{16}$$

where $\pi_{m,n}$'s and $\pi_{n,k}$'s are the Lagrangian dual optimal solution corresponding to constraints (7) and (9) in (15).

Then, we need to find the column that can produce the most negative reduced cost. Hence, the column to be generated and added to RMP is obtained by solving the price problem:

$$\text{PP}: \quad \min_{\mathbf{x}_n^1 \in \mathcal{S}_I^c} \quad \zeta_n$$
$$\text{s. t.} \quad c_m s_n z_{\epsilon_m}^u \leq b_{m,n}^{c,s}, \forall m \in \mathcal{M}(n), \tag{17}$$
$$s_n c_k z_{\epsilon_m}^d \leq b_{n,k}^{s,c}, \quad \forall k \in \mathcal{C}(n).$$

In PP, the new column $\mathbf{x}_n^1$ is embedded in $b_{m,n}^{c,s}$ and $b_{n,k}^{s,c}$. This is because all execution location indicators c_m's, s_n's and c_k's have been determined by solving RMP. Thus, PP is a binary integer programming (BIP). Denote by r^* the optimal solution to PP. If $r^* \geq 0$, then no column can generate negative reduced cost, and the current solution to RMP is the optimal solution to MP. Otherwise we add to RMP the column with the most negative reduced cost identified by (17).

We first discuss the feasible solution to RMP with the added new column and solve PP in the next subsection. Recall that the objective function in (15) becomes deterministic under a new offloading policy. The new feasible solution can be obtained by setting

$$\mathbf{x}_g^0 = \mathbf{0} \quad \text{and} \quad \mathbf{x}_{g,t}^1 = 1, \exists_{=1} t \in \{t_{min}, \cdots, t_{max}\}, \tag{18}$$

where g is the node index of the new column identified by (17), $\exists_{=1}$ means "exists and choose one", $t_{min} = \max_{m \in \mathcal{M}(n)} \sum_{p=0}^{p=1} \sum_{t=1}^{t=T} t \mathbf{x}_{m,t}^p + c_m s_n o_{m,n} \theta_u + \lceil \frac{\omega_n}{f_s} \rceil$ and $t_{max} = \min_{k \in \mathcal{C}(n)} \sum_{p=0}^{p=1} \sum_{t=1}^{t=T} t \mathbf{x}_{k,t}^p - s_n c_k o_{n,k} \theta_d - \lceil \frac{\omega_k}{f_c} \rceil$ are constants, indicating the range of the execution completion time of node g at the server. $t_{min} \leq t_{max}$ is guaranteed when solving PP.

C. Solving PP

To solve PP, we need to evaluate the reduced cost of every $\mathbf{x}_n^1 \in \mathcal{S}_I^c$ by solving $N-2$ BIP's and checking their sign. Yet, solving multiple BIP's is time-consuming when a DAG has hundreds or thousands of nodes. To circumvent this problem, we formulate a new price problem (NPP):

$$\text{NPP}: \quad \min_{\lambda_n, \ \mathbf{x}_n^1, \ n \in \mathcal{S}_I^c} \quad r = \sum_{n \in \mathcal{S}_I^c} \lambda_n \zeta_n$$
$$\text{s. t.} \quad c_m s_n z_{\epsilon_m}^u \leq \sum_{n \in \mathcal{S}_I^c} \lambda_n b_{m,n}^{c,s}, \forall m \in \mathcal{M}(n),$$
$$s_n c_k z_{\epsilon_m}^d \leq \sum_{n \in \mathcal{S}_I^c} \lambda_n b_{n,k}^{s,c}, \quad \forall k \in \mathcal{C}(n), \tag{19}$$
$$\sum_{n \in \mathcal{S}_I^c} \lambda_n = 1, \quad \lambda_n \in \{0,1\},$$

where the decision variables are λ_n's and $\mathbf{x}_n^1$'s (absorbed in $b_{m,n}^{c,s}$, $b_{n,k}^{s,c}$ and ζ_n). (19) is equivalent to (17), because in (19) only a single λ_n can be activated (equals to 1) in both the objective function and the constraints, and the number of λ_n's equals to the cardinality of $\mathcal{S}_I^c$ in (17). Although NPP formulated in (19) is a quadratic integer programming with quadratic constraints, which is generally more complicated than (17), we can solve it efficiently by further decomposing (19) into two separate problems, including a column selection

(CS) problem with the fixed $b_{m,n}^{c,s}$ and $b_{n,k}^{s,c}$ and a completion time decision (TD) problem with the selected columns.

W.l.o.g., we can initially set $b_{m,n}^{c,s} = z_{\epsilon_m}^u$ and $b_{n,k}^{s,c} = z_{\epsilon_m}^d$, and optimize (19) over λ_n as

$$\text{CS}: \quad \min_{\lambda_n,\ n \in \mathcal{S}_I^c} \quad r = \sum_{n \in \mathcal{S}_I^c} \lambda_n \zeta_n$$
$$\textbf{s. t.} \quad \sum_{n \in \mathcal{S}_I^c} \lambda_n = 1, \quad \lambda_n \in \{0, 1\}. \tag{20}$$

Then (19) is reduced to a trivial problem, i.e., 0-1 knapsack problem with the maximum weight capacity $W = 1$. The solution to CS is $\lambda_l = 1$, for $l = \arg\min_n \zeta_n$.

Next, we fix λ_n's, and formulate the TD problem as follows:

$$\text{TD}: \quad \min_{\mathbf{x}_l^1,\ l \in \mathcal{S}_I^c} \quad r = \lambda_l \zeta_l$$
$$\textbf{s. t.} \quad c_m s_l z_{\epsilon_m}^u \leq \lambda_l b_{m,l}^{c,s}, \forall m \in \mathcal{M}(l),$$
$$s_l c_k z_{\epsilon_m}^d \leq \lambda_l b_{l,k}^{s,c}, \quad \forall k \in \mathcal{C}(l), \tag{21}$$
$$\lambda_l = 1.$$

TD is a BIP problem with the only decision variable $\mathbf{x}_l^1$ (absorbed in $b_{m,l}^{c,s}$, $b_{l,k}^{s,c}$ and ζ_l). By relaxing $\mathbf{x}_l^1$ as $0 \leq \mathbf{x}_{l,t}^1 \leq 1$, TD can be relaxed to a linear programming that can be easily solved by polynomial interior algorithm [37]. Then, we set one element with the largest value in the solution vector as $\mathbf{x}_{l,t}^1 = 1$, and other elements as $\mathbf{x}_{l,t'}^1 = 0$ ($t' \neq t$).

We use $\underline{r}^*$ to denote the solution of NPP achieved by solving CS and relaxed TD iteratively, and summarize the above process of solving NPP in Algorithm 1.

Algorithm 1 Solving NPP

Input: Dual solution $\pi_{m,n}$'s and $\pi_{n,k}$'s of RMP, $\mathcal{S}_I^c$, maximum iteration number max_iter_num, $iter\_num = 0$
Output: $\mathbf{x}_g^1 \in \mathcal{S}_I^c$ (column to be added to RMP)

1: **while** $iter\_num < max\_iter\_num$ or $\underline{r}^*$ changes **do**
2:     Initialize $b_{m,n}^{c,s}$'s as $z_{\epsilon_m}^u$, $b_{n,k}^{s,c}$ as $z_{\epsilon_m}^d$;
3:     Given $b_{m,n}^{c,s}$'s and $b_{n,k}^{s,c}$'s, solve CS by setting $\lambda_l = 1$, for $l = \underset{n}{\text{argmin }} \zeta_n$;
4:     Given $\lambda_l = 1$, relax $0 \leq \mathbf{x}_{l,t}^1 \leq 1$ and solve the relaxed TD by polynomial interior algorithm;
5:     Set the maximum $\mathbf{x}_{l,t}^1$ to 1 and other $\mathbf{x}_{l,t}^1$'s as 0;
6:     Evaluate $\underline{r}^*$;
7:     $iter\_num = iter\_num + 1$;
8: **end while**
9: set $g = l$, return $\mathbf{x}_g^1$.

D. ϵ-Bounded Approximate Solution

The solver developed in Algorithm 1 does not achieve the optimal solution to PP, i.e., r^*. However, when the result of NPP, i.e., $\underline{r}^*$, is larger than or equal to 0, i.e., $r^* \geq \underline{r}^* \geq 0$, MP can still be solved optimally. Even if $\underline{r}^* < 0$, we can still find an ϵ-bounded approximate solution to MP. First, we define the ϵ-bounded approximate solution as follows.

Definition 1: The ϵ-Bounded Approximate Solution. Let $0 \leq \epsilon < 1$ be the predefined parameter and Ψ^* be the optimal result. Then a solution Ψ is called the ϵ-bounded approximate solution if it satisfies $\Psi^* \leq \Psi \leq (1+\epsilon)\Psi^*$.

We can have the following lemma on determining the ϵ-bounded approximate offloading policy.

Lemma 1: Denote by Ψ_l and Ψ_u the lower and upper bounds on the optimal result Ψ^* of MP, respectively. Then, the ϵ-bounded approximate solution can be obtained when no new column $(\mathbf{x}_n^1)$ can be found by solving PP, or the iteration stops at $\underline{r}^* \geq 0$, or $\frac{\Psi_l}{\Psi_u} \geq \frac{1}{1+\epsilon}$.

Proof: When $\frac{\Psi_l}{\Psi_u} \geq \frac{1}{1+\epsilon}$, we have $\Psi_u \leq (1+\epsilon)\Psi_l \leq (1+\epsilon)\Psi^*$. Then, given any result Ψ between the lower and upper bounds, i.e., $\Psi_l \leq \Psi \leq \Psi_u$, we can have $\Psi^* \leq \Psi \leq \Psi_u \leq (1+\epsilon)\Psi^*$, which is the ϵ-bounded approximate solution by Definition 1. Besides, when $\underline{r}^* \geq 0$ or no new column can be found by solving PP, as discussed before, the obtained solution is the optimal solution and hence an ϵ-bounded approximate solution as well ($\epsilon = 0$). ∎

The optimal result of RMP at each iteration serves as an upper bound of MP, i.e., Ψ_u. A lower bound can be obtained by setting $\Psi_l = \Psi_u + \mathcal{K}r^* \leq \Psi^*$ according to [37], where r^* is the optimal solution to PP, and $\mathcal{K} \geq \sum_n s_n$ holds for the optimal solution to RMP. Since we do not actually obtain r^* with the decomposition of NPP into CS and relaxed TD, the lower bound can be set to $\Psi_u + \mathcal{K}\underline{r}^*$, which is less than $\Psi_u + \mathcal{K}r^*$ and hence Ψ^*. Additionally, because $\underline{r}^*$ is negative, Ψ_l may be negative as well. Thus, in practice, we set $\Psi_l = \max\{0, \Psi_u + \mathcal{K}\underline{r}^*\}$.

At each iteration, by solving RMP and NPP we can obtain a new pair of lower and upper bounds. By evaluating their ratio, i.e., $\frac{\Psi_l}{\Psi_u}$, we can determine whether we have obtained an ϵ-bounded approximate solution. The process to find an ϵ-bounded approximate offloading solution for offloading a DAG modeled application is summarized in Algorithm 2.

Algorithm 2 An ϵ-bounded approximate solution for DAG-modeled application offloading

Input: initialize ϵ, $\mathcal{S}_I$, $\Psi_l = 0$, $\Psi_u = \infty$, $\underline{r}^* = \infty$
Output: offloading decision c_n's and s_n's

1: **while** PP generates a new column, or $\frac{\Psi_l}{\Psi_u} < \frac{1}{1+\epsilon}$, or $\underline{r}^* < 0$ **do**
2:     Solve RMP under current $\mathcal{S}_I$ by checking the feasibility, obtain Ψ_u and the dual optimal solutions $\pi_{m,n}$'s and $\pi_{n,k}$'s;
3:     Solve NPP using Algorithm 1, and obtain new column $\mathbf{x}_g^1$ according to (18);
4:     Obtain the result of NPP, i.e., $\underline{r}^*$;
5:     Update $\mathcal{S}_I = \mathcal{S}_I \cup \mathbf{x}_g^1$, set $\mathbf{x}_g^0 = \mathbf{0}$, $\mathcal{S}_I^c = \mathcal{S} \backslash \mathcal{S}_I$;
6:     $\Psi_l = \Psi_u + \mathcal{K}\underline{r}^*$;
7: **end while**
8: **for** $n \in \{1, \cdots, N\}$ **do**
9:     $c_n = \sum_t \mathbf{x}_{n,t}^0$; $s_n = \sum_t \mathbf{x}_{n,t}^1$;
10: **end for**

E. Computational Complexity Analysis

It is obvious that the computational complexity is closely related to the DAG's structure, and the number of parent modules of a node is clearly upper bounded by N. Since Algorithm 2 solves the formulated problem iteratively, we assume that it solves the problem by K iterations. There are a total of $N-2$ task modules that can be offloaded, so $K \leq N-2$. At each iteration, we need to solve both the RMP and NPP. As a result, we can analyze the computational complexity as follows. First,

solving the RMP is reduced to feasibility checking, which requires the computational complexity of $O(NT)$. Second, the NPP includes CS and TD. Solving CS takes the $\mathcal{O}(N|\mathcal{S}_I^C|)$ computational complexity, where $|\mathcal{S}_I^C|$ decreases by one at each iteration. Solving the TD by the polynomial interior algorithm requires the computational complexity of $\mathcal{O}(n^3)$, where n is the number of variables. Moreover, $\mathbf{x}_l^1$ with T dimensions is the only decision variable, which means we have $n = T$. Therefore, we can conclude that the computational complexity of solving MP in (14) by the proposed scheme based on column generation is $\mathcal{O}\big(K(NT + N^2 + T^3)\big)$. Computational complexity also implicitly depends on ϵ through the parameter K. The reason is that K depends on if $\frac{\Psi_L}{\Psi_u} < \frac{1}{1+\epsilon}$ holds (i.e., one of the condition on obtaining an ϵ-bounded approximate solution discussed in Lemma 1). As a result, by choosing different values of ϵ, the value of K also varies.

F. Offloading Mobile Applications with Special Structures

Some mobile applications may have simple structures, and can be modeled as DAGs with only sequential or parallel task module dependency [13], [40]. In this section, we provide principles to conduct computation offloading for these special mobile applications.
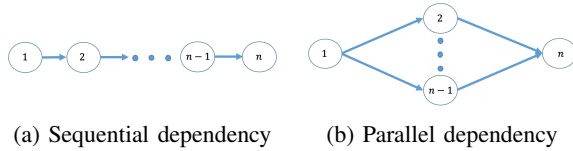

(a) Sequential dependency   (b) Parallel dependency

Fig. 2. DAGs with only sequential or parallel task module dependency.

1) Case I: Applications with Only Sequential Module Dependency: For applications that can be modeled as a DAG with only sequential dependency, i.e., Fig. 2(a), we develop an efficient way to offload its computations. First, we arrive at a theorem about the optimal computation offloading policy for executing applications with only sequential dependency under uncertain radio channel and network queues.

Theorem 2: Under uncertain radio channel with queuing delay, the optimal offloading policy for executing applications with only sequential dependency only migrates computations once from the client device to the edge server if needed.

Proof: W.l.o.g., suppose an application has N sequential modules, and two subsequences, i.e., modules from u to v and p to q, are migrated to the server for execution, where $1 < u < v < p < q < N$. Then, the expected worst-case energy consumption for this offloading policy is $\Psi_1 = \sum_{n \in S_1} c_n \kappa \omega_n f_c^2 + P_u(o_{u-1} + o_{p-1})\theta_u + P_d(o_{v+1} + o_{q+1})\theta_d$, where $S_1 = \big\{\{2, \cdots, u-1\} \cup \{v+1, \cdots, p-1\} \cup \{q+1, \cdots, N-1\}\big\}$. If modules from $v+1$ to $p-1$ are executed at the server instead, then the expected worst-case energy consumption for the new policy is $\Psi_2 = \sum_{n \in S_2} c_n \kappa i_n \omega_n f_c^2 + P_u o_{u-1}\theta_u + P_d o_{q+1}\theta_d$, where $S_2 = \big\{\{2, \cdots, u-1\} \cup \{q+1, \cdots, N-1\}\big\}$. Obviously, we have $\Psi_2 < \Psi_1$, thus offloading only one subsequence, i.e., migrating only once, provides a better solution. ∎

With Theorem 2, we can have a more efficient way to solve PP for applications with only sequential dependency. Since the offloaded modules should be consecutive in a DAG with only sequential dependency, once we find one column with negative reduced cost, we can keep adding its neighboring module with increasing hops as the new column until the reduced cost is positive. Then the subsequence (each node has a negative reduced cost) is migrated to the server for execution.

2) Case II: Applications with Only Parallel Dependency: We now consider DAG with only parallel dependency, i.e., Fig. 2(b), We also have a theorem on the computation offloading policy for executing such applications.

Theorem 3: Under uncertain radio channel with queuing delay, the optimal offloading policy for executing applications with only parallel dependency is the one that offloads all modules whose energy consumption for execution is larger than the expected worst-case energy consumption for data transmission, i.e., $\kappa \omega_n f_c^2 > P_u \theta_u o_1 + P_d \theta_d o_n, n \in \mathcal{N}$.

Proof: Suppose the minimum expected worst-case energy consumption without the nth module is $\Psi_{\mathcal{N}/n}^*$. If n is executed locally, then $\Psi = \Psi_{\mathcal{N}/n}^* + \kappa \omega_n f_c^2$, else if n is executed remotely, then $\Psi = \Psi_{\mathcal{N}/n}^* + P_u \theta_u o_1 + P_d \theta_d o_n$. Hence, offloading a module that consumes higher energy by local execution than data transmission will result in lower Ψ. ∎

VII. Experimental Results

Now, we will describe the experiment setup, discuss parameter estimation for the considered GEV distributions, evaluate the performance of the proposed computation offloading scheme, empirically corroborate the performance degradation caused by simple assumptions on network condition and queues, and compare with the state-of-the-art mechanism. Furthermore, we will also study the scalability of the proposed scheme, and investigate the offloading performance on simulated applications with sequential or parallel dependence.

A. Experimental Setup

We conduct extensive experiments including both testbed experiments (Section VII-C, VII-D, and VII-E) and simulations (Section VII-F and VII-G) to evaluate the performance of our proposed scheme. Specifically, we establish a MEC architecture in which four servers (provided by the Accipiter System[1]), each having 2.4GHz CPU with 6 cores, are deployed in a $40 \times 70 m^2$ floor area in an office building. The servers are directly connected to a switch. The client devices used in the experiments are all Motorola moto G^4 with 1.5GHz CPU. In Fig. 3, we provide the floor plan of the environment along with the edge server used in the testbed experiments.

We execute a real-world mobile application, i.e., Smart Diagnosis visualized in Fig. 1. We implement this application on Android devices and encapsulate the proposed scheme as an API that can be called when the application is invoked. We set the competition time $T = 0.5 \times 10^4$ milliseconds (approximately the average execution time of a mobile device running a computer vision application), and $\Delta = 1$ millisecond.

[1] https://www.accipitersystems.com/

Fig. 3. Floor plan of the experiment environment. The edge server is located at the second office room (indicated by the red arrow), top right corner.

To measure the communication parameters in the MEC system, we first execute Smart Diagnosis locally on several mobile devices, randomly walk in the considered area with varying speeds, make video phone calls, and run programs at these devices to generate random matrices with varying sizes, offload them to the servers to calculate the singular values, then receive the results. At the same time, we also use available Android tools, i.e., tPacketCapture, WiFi SNR, and PETrA [41], to record the statistics of the queue length, the channel SNR, and the client device's energy consumption, respectively. Specifically, tPacketCapture captures out-going and incoming packets using VpnService provided by Android OS, and the captured data are saved as PCAP files and further analyzed using Wireshark. WiFi SNR records the varying SNR, from which we calculate the corresponding bit rates using Shannon-Hartley theorem given the predetermined bandwidth for data transmission. PETrA provides the client device's energy consumption profile in real-time when executing applications. In Table II, we show some statistics of the out-going and incoming packet size (i.e., P_u and P_d), uploading and downloading data rates (i.e., R_u and R_d), and uploading and downloading power consumption (i.e., P_u and P_d).

TABLE II
STATISTICS OF COLLECTED SAMPLES.

parameters	mean	standard deviation
P_u (×10^5 bit)	1.01	0.09
P_d (×10^5 bit)	1.32	0.01
R_u (×10^5 bit/s)	4.5	0.5
R_d (×10^5 bit/s)	13.1	1.1
P_u (mW)	0.14	0.04
P_d (mW)	0.16	0.06

B. Parameter Estimation for GEV Distribution

Based on the statistics collected in Section VII-A, we calculate the communication parameters, i.e., $z_{\epsilon_m}^u$, $z_{\epsilon_m}^d$, θ_u and θ_d, which involve the inference of GEV distribution of $V_{(k)}$, $J_{(k)}$, and $H_{(k)}$. Since all of them follow the same process, we only show the process of inferring $V_{(k)}$. We use tPacketCapture and WIFI SNR to capture a series of independent samples of real-time queue length and SNR, and then form a set $\{V_1, V_2, \cdots, V_N\}$, where $V_n = \frac{Q_n^u + o_n}{R_u}$, o_n is a generated matrix data size measured in bits, Q_n^u is the queue length when transmitting o_n to the edge server and is determined based on the time tag when the matrix transmission command is invoked and executed, and R_u is the corresponding bit rate. The samples are blocked into $\frac{N}{k}$ sequences of length k, for

some large values of k, generating a series of block maxima, $V_{(k)_1}, V_{(k)_2}, \cdots, V_{(k)_\frac{N}{k}}$, to which the GEV distribution for $V_{(k)}$ in the case of transmitting data to the edge server is fitted. Please refer to [36] for the details of using MLE to find the GEV distribution given series of block maxima. In the testbed experiment, we set k (the order of statistics) as 1500, because by the end of executing the Smart Diagnosis, each device can generate about 1500 random matrices, transmit them to the server, and receive the singular values. The data collection period is repeated for 100 times , which means $\frac{N}{k} = 100$.

Fig. 4 shows the histogram of the block maximas (indicated by the blue bars) and the inferred GEV distributions (indicated by the red curves). Specifically, by fixing $\epsilon_m^u = \epsilon_m^d = 0.1$, according to (8), we calculate the 90% quantile of $f_{V_k}(v_k)$ for transmitting data to the edge server and client device as $z_{\epsilon_m}^u = 0.349$ and $z_{\epsilon_m}^d = 0.107$, respectively. By applying (13), we calculate $\theta_u = 4.81 \times 10^{-4}$ and $\theta_d = 1.11 \times 10^{-5}$. The parameters used for executing Smart Diagnosis in the MEC system are summarized in Table III.
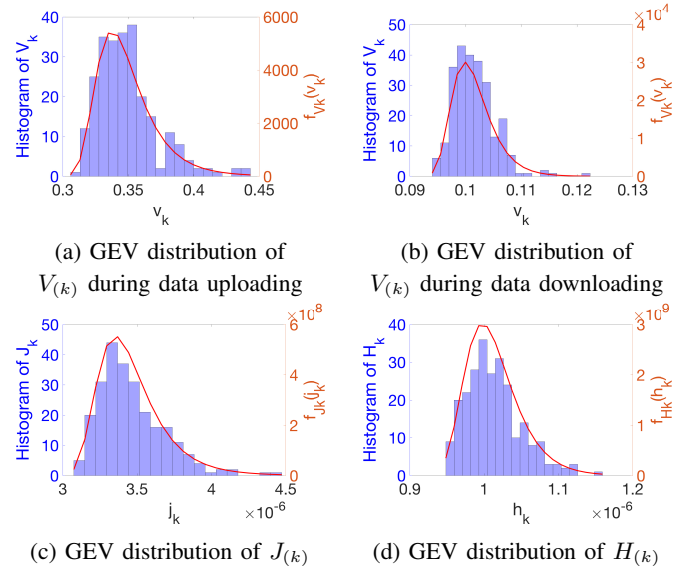

(a) GEV distribution of $V_{(k)}$ during data uploading   (b) GEV distribution of $V_{(k)}$ during data downloading

(c) GEV distribution of $J_{(k)}$ (d) GEV distribution of $H_{(k)}$

Fig. 4. Inferred GEV distributions.

C. Performance of the Proposed Scheme

To evaluate the efficacy of our proposed scheme, we calculate the offloading percentage and the energy consumption of Smart Diagnosis using different computation offloading decisions obtained under varying approximation parameter (ϵ), while fixing the probabilities (ϵ_m^u and ϵ_m^d) of the extreme case events happening during the period of data uploading and downloading.

We show the results in Fig. 5, where the blue dotted line represents the offloading percentage and the red dotted line is the energy consumption. Specifically, Fig. 5(a) and (b) are the results of setting $\epsilon_m^d = \epsilon_m^u = 0.01$ and $\epsilon_m^d = \epsilon_m^u = 0.1$, respectively.[2] We see that our scheme is very robust under

---

[2]In practice, we can choose large values of ϵ_m^d and ϵ_m^u if there are multiple mobile devices utilizing the computation resource at the edge, because in this case, the probability of experiencing high queuing size and low communication rate will be higher.

TABLE III
THE SUMMARY OF EXPERIMENT SETTINGS.

f_c	f_s	κ	$z_{\epsilon_m}^u$	$z_{\epsilon_m}^d$	T	Δ	θ_u	θ_d
1.5GHz	2.4GHz	10^{-24}mW/Hz^3	0.349s	0.107s	0.5×10^4ms	1	4.81×10^{-4}mW·s/bit	1.11×10^{-5}mW·s/bit

(a) Fix $\epsilon_m^d = \epsilon_m^u = 0.01$. (b) Fix $\epsilon_m^d = \epsilon_m^u = 0.1$.

(c) Fix $\epsilon = 0.03, \epsilon_m^d = 0.01$. (d) Fix $\epsilon = 0.03, \epsilon_m^u = 0.01$.
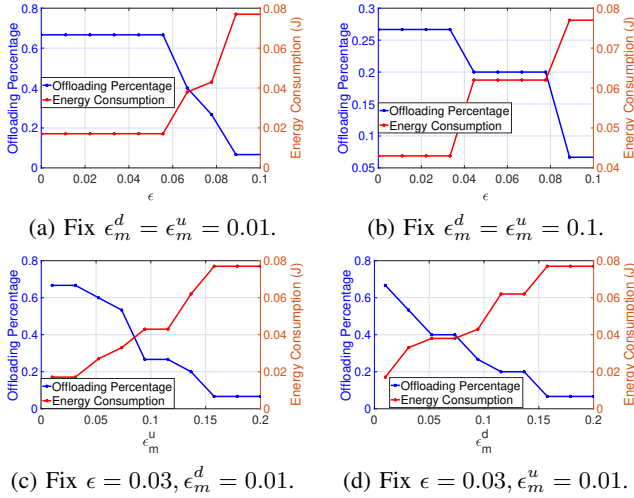
Fig. 5. Offloading percentage and energy consumption under various approximation parameter and queueing delay.

various ϵ. For example, in Fig. 5(a), when $\epsilon_m^d = \epsilon_m^u = 0.01$, the offloading policy is optimal as long as the approximation parameter does not exceed 0.05. This is because the client device favors an energy-friendly computation offloading policy that offloads more computing nodes, and when the extreme case events happen with a lower probability, the radio channel is able to support more data transferring between the client device and the edge server. Even increasing the probability that extreme case events occur, we can still obtain an optimal offloading policy when $0 \le \epsilon \le 0.03$ as shown in Fig. 5(b). Based on the obtained optimal offloading decision for Smart Diagnosis when $\epsilon = 0.03, \epsilon_m^d = \epsilon_m^u = 0.1$, modules named "SURF Description", "SIFT Description", "KNN1", "KNN2", "Tokenization", and "POS Tagging" in Fig. 1 are offloaded.

We further evaluate the impact of queuing delay on the performance by varying probabilities ϵ_m^u and ϵ_m^d, and show the results in Fig. 5 (c) and (d). We can see that for both cases, the offloading percentage decreases and Ψ increases with the increasing values of ϵ_m^u and ϵ_m^d. The reason is that as ϵ_m^u and ϵ_m^d increase, the channel conditions and queuing delay cause the increase in data transmission time, thus the proposed scheme decreases the number of modules to be offloaded.

D. Performance Degradation Caused by Simple Network Condition and Queueing Assumptions

Having demonstrated the efficacy of the proposed scheme, we now empirically corroborate that it will lead to higher energy consumption for the client device if we simply adopt strong assumptions on the network conditions or the queuing delays when deciding which task module to offload. Specifically, we consider the offloading decisions for the Smart

Diagnosis obtained under the following scenarios: (i) a block-fading communication channel, and (ii) constant queuing size of the out-going buffers at the client device and server.

Scenario (i). The block-fading channel assumes that the channel condition does not change over the duration of application execution, thus, according to the Shannon-Hartley theorem, the client device can have constant uploading and downloading bit rates (i.e., R_u and R_d). As a result, the unknown conditions when making computation offloading decisions are $Q_{m,n}^u$, $Q_{m,n}^d$ (cf. (4)), P_u, and P_d (cf. (10)). The offloading decision can still be obtained by the developed column generation based algorithm by considering constant R_u and R_d when fitting the required GEV distributions in Section VII-B.

(a) $R_d = 22.7$MB/s, vary R_u (b) $R_u = 10.7$MB/s, vary R_d
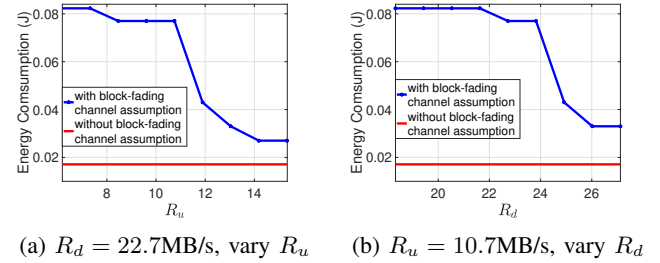
Fig. 6. Energy consumption comparison between offloading decisions obtained with and without the block-fading channel assumption.

In Fig. 6 (a) and (b), we present the empirical energy consumption of the client device when executing Smart Diagnosis using the offloading decisions obtained with the block-fading channel assumption (the blue curves). We also plot the energy consumption obtained without assuming block-fading channel (the red lines) for comparison. In this experiment, we fix $\epsilon_m^d = \epsilon_m^u = 0.01$ and $\epsilon = 0.05$. More specifically, in Fig. 6 (a), we fix R_d to be the average value computed using SNR measured by WiFiSNR in Section VII-A, and vary R_u in the range of $\mathbb{E}[R_u] \pm 4.5$MB/s, where $\mathbb{E}[R_u] = 10.7$MB/s is obtained from Section VII-A. Similarly, in Fig. 6 (b), we fix R_u to be the average value obtained in Section VII-A, and vary R_d in the range of $\mathbb{E}[R_d] \pm 4.5$MB/s, where $\mathbb{E}[R_d] = 22.7$MB/s is also obtained from Section VII-A. Clearly, by considering the uncertainty in downloading and uploading bit rates, i.e., without the strong assumptions on the communication channel, the offloading decisions achieved by our proposed scheme can always lead to much lower empirical energy consumption. The reason is that under the block-fading channel assumption, when R_u or R_d is low, the offloading decision will assign more task module at the client device, which increases its energy consumption. On the other hand, when R_u or R_d is high, the offloading decision will assign more task module to the server. However, in practice, it may take the client device more time to transmit or receive the data if the data rates drop in the course of the application execution, because we

carry the mobile device and walk at various speed during all the experiments. Similar phenomenon has also been shown in [20], i.e., the varying locations of the local mobile transmitter and receiver can lead to uncertainties in channel law as well as arbitrarily varying communication bit rates.

Scenario (ii). In this case, the unknown conditions in our computation offloading scheme are R_u, R_d, P_u, and P_d. The offloading decision is obtained using the column generation based algorithm by considering the constant $Q_{m,n}^u$ and $Q_{m,n}^d$.

Fig. 8. Comparison with state-of-the-art offloading approaches.

Fig. 9. Evaluation of the scalability of the proposed computation offloading algorithm.

(a) $Q_{m,n}^d = 3.3$KB, vary $Q_{m,n}^u$ (b) $Q_{m,n}^u = 5.4$KB, vary $Q_{m,n}^d$
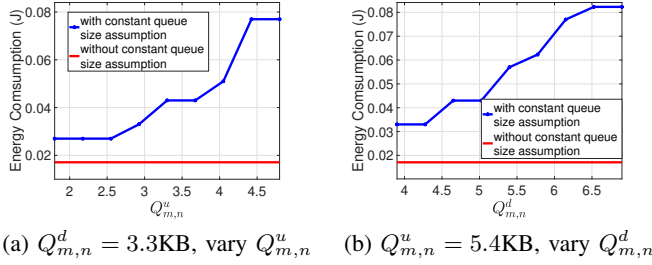
Fig. 7. Energy consumption comparison between offloading decisions obtained with and without the constant queuing size assumption.

In Fig. 7 (a) and (b), we present the empirical energy consumption of the client device when executing Smart Diagnosis using the offloading decisions obtained with the constant queuing size assumption (the blue curves). Additionally, the energy consumption (the red lines) obtained without such assumption is shown for comparison. In this experiment, we still fix $\epsilon_m^d = \epsilon_m^u = 0.01$ and $\epsilon = 0.05$. Particularly, in Fig. 7 (a), we fix $Q_{m,n}^d$ to be the average value measured by tPacketCapture in Section VII-A, and vary $Q_{m,n}^u$ in the range of $\mathbb{E}[Q_{m,n}^u] \pm 1.5$KB, where $\mathbb{E}[Q_{m,n}^u] = 3.3$KB is obtained from Section VII-A. Similarly, in Fig. 7 (b), we fix $Q_{m,n}^u$ to be the average value obtained in Section VII-A, and vary $Q_{m,n}^d$ in the range of $\mathbb{E}[Q_{m,n}^d] \pm 1.5$MB/s, where $\mathbb{E}[Q_{m,n}^d] = 5.4$MB/s is also obtained from Section VII-A. Clearly, by considering the uncertainty of the queuing size, i.e., without the constant queuing size assumption, the offloading decisions achieved by the proposed scheme can always lead to much lower empirical energy consumption. Similar analysis also applies to this scenario; under the assumption of constant queuing size at the out-going buffer of the client device and the server, when $Q_{m,n}^u$ or $Q_{m,n}^d$ is low, the offloading decision will assign more task modules to the server to save local energy consumption, but it may take the client device more time to transmit or receive the data, which incur extra communication energy, if the queuing size increase during the application execution. The reason is that sudden burst of serving requests can overflow the edge servers' resource and may even cause MEC failure [22], [23]. On the other hand, if $Q_{m,n}^u$ or $Q_{m,n}^d$ is high, the offloading decision will assign more task modules to the local device and increase its energy consumption.

E. Performance Improvement over Existing Schemes

We compare our proposed algorithm with two state-of-the-art offloading approaches, i.e., Hermes [10] and JSCO [14] discussed in Section II. The experiment are conducted
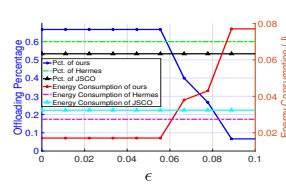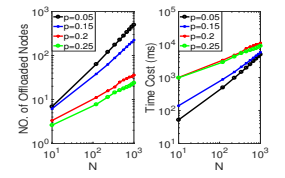
by varying ϵ when $\epsilon_m^d = \epsilon_m^u = 0.01$, and the offloading percentage and energy consumption are calculated for all algorithms. The results are shown in Fig. 8. We see that our scheme can always lead to the lowest energy consumption by offloading more computation modules as long as the approximation parameter ϵ is less than 0.05. Particularly, when $\epsilon \leq 0.05$, our scheme can save more than 50% energy for the local device, as the energy consumption of executing Smart Diagnosis using Hermes, JSCO, and our scheme are about $0.03J$, $0.028J$, and $0.014J$, respectively. The reason is that JSCO fails to consider uncertainties in dynamic radio channels with queueing delay, and Hermes introduces extra energy consumption due to communication overhead caused by continuously probing the channel.

F. The Scalability of The Proposed Scheme

To explore the scalability of the proposed scheme, we measure the number of offloaded nodes when running simulated DAGs with different sizes. Specifically, we simulate DAGs with layer-by-layer structure, where a random number of nodes is generated for each layer and edges are added between nodes across layers with a given probability p. In this experiment, the computation workload and output data size of each module in simulated DAGs are all attributed to half-normal distributions (nonnegative support) with different parameters. Besides, the communication parameters used in this simulation experiment are the same as those in Table III.

We vary the number of nodes in a DAG from 10^1 to 10^3 and consider four connecting probabilities, i.e., $p \in \{0.05, 0.15, 0.2, 0.25\}$. In the left panel of Fig. 9, we plot the number of offloaded nodes versus the size of DAG when having $\epsilon = 0.03$ and $\epsilon_m^u = \epsilon_m^d = 0.01$. We see that by fixing a DAG size, i.e., $N = 10^2$, the number of offloaded modules decreases with the increasing p. This is because the denser the DAG, the more dependency among modules, and offloading will cost more energy in dense DAGs than that in sparse ones. Moreover, we also observe that by fixing a connecting probability, i.e., $p = 0.05$, the number of offloaded nodes grows linearly with the size of DAG. Since the proposed scheme identifies one column at each iteration, it scales linearly with the graph size. This indicates that our scheme is computationally efficient. In the right panel of Fig. 9, we also plot the time cost of running our proposed algorithm on these simulated DAGs. We can see that the computation time also scales linearly with the graph size, and it suggests that our proposed algorithm can work in real-world MEC system.

G. Offloading Applications with Special Dependency

In this section, to corroborate the findings in Theorem 2 and 3, we also simulate special applications with only sequential or parallel dependency. Specifically, we consider an application composed of 14 task modules. The computation workload and output data size for each task module are sampled from Poisson distributions with $\lambda_1 = 10^6$Hz and $\lambda_2 = 12$kb.

Fig. 10 shows the offloading decision of task modules with only sequential dependency. In particular, we compare the offloading decisions by considering two approximation parameters, i.e., $\epsilon = 0$ and $\epsilon = 0.03$, and for each ϵ, we consider the effect of ϵ_m^d and ϵ_m^u on the offloading policy. In Fig. 10, we use the blue dotted line to show the offloading policy under $\epsilon_m^d = \epsilon_m^u = 0.01$, use the red circled line to show the offloading policy under $\epsilon_m^d = \epsilon_m^u = 0.1$, and use 1 or 0 to indicate a particular module is offloaded or not. We can see that in both cases, the offloading policies for sequential DAG only offload one subsequence. Besides, when the extreme events happen with low probabilities, the offloading policy will offload longer subsequence. This is because the better radio conditions, the less task modules will be executed locally.
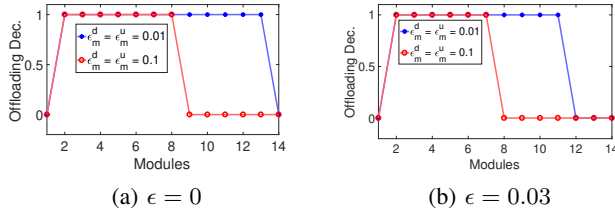

(a) $\epsilon = 0$ (b) $\epsilon = 0.03$

Fig. 10. Offloading decision for sequential dependency DAG.

(a) $\epsilon = 0$, $\epsilon_m^d = \epsilon_m^u = 0.01$ (b) $\epsilon = 0.03$, $\epsilon_m^d = \epsilon_m^u = 0.1$
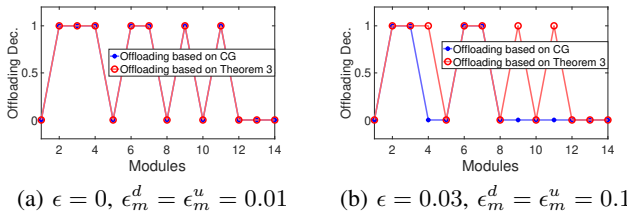
Fig. 11. Offloading decision for parallel dependency DAG.

Fig. 11 shows the offloading decision of all tasks with only parallel dependency. We compare the offloading decisions by considering two sets of approximation parameter and probabilities, i.e., $\epsilon = 0$, $\epsilon_m^d = \epsilon_m^u = 0.01$ and $\epsilon = 0.03$, $\epsilon_m^d = \epsilon_m^u = 0.1$. In Fig. 11, the blue dotted lines indicate the offloading policy decided by running our column generation based algorithm, the red circled lines indicate the policy decided by Theorem 3. We can see that in Fig. 11(a), the obtained offloading polices are identical for both methods, but in Fig. 11(b), the policy decided by Theorem 3 offloads more nodes than that decided by the algorithm. The reason is that when the approximation parameter is large, e.g., $\epsilon = 0.03$ in Fig. 11(b), the solution to the proposed algorithm is an ϵ-bounded approximate solution, while the solution given by Theorem 3 is the optimal offloading decision. This implies that when applications have only parallel dependency, we can directly

obtain the optimal offloading policy by evaluating the energy consumption and expected worst-case energy consumption for data transmission of each task module.

Additionally, we also conduct experiments to offload the simulated DAGs with special dependency structures under the uncertainty MEC system described in Section VII-A. The communication parameters are still set as those listed in Table III. In Table IV, we compare the energy consumption achieved by our scheme, under different choice of ϵ, ϵ_m^d, and ϵ_m^u, with Hermes and JSCO. Clearly, our developed scheme can still achieve the lowest local energy consumption, which suggests again that by explicitly addressing the uncertainties involved in computation offloading using the extreme value theory, we can make offloading decisions that are close to the optimal ones (i.e., when $\epsilon = 0$).

TABLE IV
COMPARISON OF ENERGY CONSUMPTION CAUSED BY OUR ALGORITHM AND OTHER SCHEMES ON DAG WITH SPECIAL DEPENDENCY.

	sequential DAG		parallel DAG	
	$\epsilon = 0$	$\epsilon = 0.03$	$\epsilon = 0$	$\epsilon = 0.03$
$\epsilon_m^d = \epsilon_m^u = 0.01$	0.011 J	0.014 J	0.017 J	0.018 J
$\epsilon_m^d = \epsilon_m^u = 0.1$	0.016 J	0.018 J	0.020 J	0.022 J
	0.019 J by Hermes		0.022 J by Hermes	
	0.020 J by JSCO		0.024 J by JSCO	

VIII. CONCLUSIONS

In this paper, we investigate the problem of energy-efficient computation offloading in a practical MEC systems. In contrast to existing works which usually impose strong assumptions on communication channels and network queues, we remove these assumptions and address the uncertainties in the MEC system. First, we handle the uncertainties and bound the occurrence probability of extreme events using extreme value theory. Then, we formulate the expected energy consumption in the worst case when executing time-sensitive applications. Next, since the formulated optimization problem is a quadratically constrained binary quadratic programming, we develop an ϵ-bounded approximation algorithm based on column generation technique to solve it. In addition, we also tailor the proposed computation offloading scheme to accommodate special applications with only sequential or parallel task module dependency. We implement the proposed scheme on the Android platform and conduct extensive experiments by executing a real-world mobile application. Experiment results corroborate the fact that it will lead to lower energy consumption by explicitly consider the uncertainties in communication channels and network queues, and show that our scheme outperform state-of-the-art schemes in terms of energy saving.

REFERENCES

[1] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing (POD'96)*, May, 1996.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv*, vol. 1701, 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3142685, IEEE Transactions on Wireless Communications

13

[3] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[4] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *IEEE Conference on Computer Communications (INFOCOM'18)*. Honolulu, HI, USA, 2018.

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

[6] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896 – 5907, 2016.

[7] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *IEEE Global Communications Conference (GLOBECOM'16)*. Washington, DC, USA, 2016.

[8] S. Salinas, C. Luo, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale quadratic programs," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIACCS'16)*. Xi'an, China, May, 2016.

[9] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, 2015.

[10] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[11] C. Luo, S. Salinas, M. Li, and P. Li, "Energy-efficient autonomic offloading in mobile edge computing," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing (DASC'17)*. Orlando, FL, USA, November 6-10, 2017.

[12] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *IEEE Computer Communications Workshops (INFOCOM WKSHPS'14)*. Toronto, ON, Canada, April 27-May 2, 2014.

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

[14] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, pp. 301 – 313, April-June, 2019.

[15] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *IEEE International Symposium on Information Theory (ISIT'16)*. Barcelona, Spain, July 10-15, 2016.

[16] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[17] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *2013 Future Network & Mobile Summit*. Lisboa, Portugal, July 3-5, 2013.

[18] T. Ji, Y. Guo, Q. Wang, X. Wang, and P. Li, "Economy: Point clouds-based energy-efficient autonomous navigation for uavs," *IEEE Transactions on Network Science and Engineering*, 2021.

[19] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[20] A. Lapidoth and P. Narayan, "Reliable communication under channel uncertainty," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2148–2177, 1998.

[21] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

[22] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138–147, May, 2017.

[23] C. S. M. Babou, D. Fall, S. Kashihara, I. Niang, and Y. Kadobayashi, "Home edge computing (hec): Design of a new edge computing technology for achieving ultra-low latency," in *Edge Computing – EDGE 2018*. Springer International Publishing, 2018, pp. 3–17.

[24] B. Ezio, "The impact of uncertain channel models on wireless communication," *Journal of Communications and Information Networks*, vol. 1, no. 1, 2016.

[25] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined iov," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097–7111, 2020.

[26] X. Meng, W. Wang, Y. Wang, V. K. Lau, and Z. Zhang, "Closed-form delay-optimal computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4653–4667, 2019.

[27] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *Consumer Communications & Networking Conference (CCNC'17)*. Las Vegas, NV, USA, January 8-11, 2017.

[28] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, pp. 1–7, 2010.

[29] A. J. Ganesh, N. O'Connell, and D. J. Wischik, *Big queues*. Springer, 2004.

[30] C. Liu, Y. Shu, J. Liu, and O. W. Yang, "Application of extreme value theory to the analysis of wireless network traffic," in *IEEE International Conference on Communications (ICC'17)*. Glasgow, UK, January 24-28 2007.

[31] W. Xiao, S. Agarwal, D. Starobinski, and A. Trachtenberg, "Reliable rateless wireless broadcasting with near-zero feedback," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1924–1937, 2012.

[32] B. Swapna, A. Eryilmaz, and N. B. Shroff, "Throughput-delay analysis of random linear network coding for wireless broadcasting," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6328–6341, 2013.

[33] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Globecom Workshops (GC Wkshps'17)*. Singapore, December 4-8 2017.

[34] J. Curtiss, "On the distribution of the quotient of two chance variables," *The Annals of Mathematical Statistics*, vol. 12, no. 4, pp. 409–421, 1941.

[35] M. K. Abdel-Aziz, C.-F. Liu, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-reliable low-latency vehicular networks: Taming the age of information tail," in *IEEE Global Communications Conference (GLOBECOM'18)*. Abu Dhabi, United Arab, December 9-13, 2018.

[36] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An introduction to statistical modeling of extreme values*. Springer, 2001, vol. 208.

[37] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[38] M. Li, P. Li, X. Huang, Y. Fang, and S. Glisic, "Energy consumption optimization for multihop cognitive cellular networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 358–372, 2014.

[39] M. Li, S. Salinas, P. Li, X. Huang, Y. Fang, and S. Glisic, "Optimal scheduling for multi-radio multi-channel multi-hop cognitive cellular networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 1, pp. 139–154, 2014.

[40] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proceedings of the 9th international conference on Mobile systems, applications, and services (Mobisys'11)*. Washington, DC, USA, June 28-July 1, 2011.

[41] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Software-based energy profiling of android apps: Simple, efficient and reliable?" in *IEEE 24th international conference on software analysis, evolution and reengineering (SANER'17)*. Klagenfurt, Austria, February 20-24, 2017.