Offloading Decision in Edge Computing for Continuous Applications under Uncertainty

Wei Chang*†, Yang Xiao†, Wenjing Lou†, Guochu Shou*
*Beijing Key Laboratory of Network System Architecture and Convergence, School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China
†Virginia Polytechnic Institute and State University, VA, USA
Email: weichang@bupt.edu.cn, {xiaoy, wjlou}@vt.edu, gcshou@bupt.edu.cn

***Abstract*— Edge computing (EC) is an emerging paradigm to push sufficient computation resources towards the network edge, improving application performance significantly by offloading applications to the edge computing node. We investigate continuous application offloading decision in EC, for which it is uncertain how users operate continuous applications and how long continuous applications last before completion. That means some characteristics of continuous applications, e.g., the number of user operations, the uploading and downloading data size for offloading computation of each user operation, and the number of central processing unit (CPU) cycles required to execute computation of each user operation, are unknown when making offloading decision. In this scenario, an energy consumption constrained average response time minimization problem among multiple users for continuous applications under uncertainty is formulated. To tackle this problem, we propose the Response Time-Improved Offloading algorithm with Energy Constraint (RTIOEC) to make offloading decision with fewer characteristics of applications. The evaluation results show that the RTIOEC algorithm achieves comparatively short average response time of continuous applications while satisfying the energy consumption constraint with a predefined upper bound of violation probability. Our results demonstrate the practicality of the RTIOEC algorithm in offloading decision in EC for continuous applications under uncertainty.**

***Index Terms*—Edge computing, uncertainty, chance constrained programming, multi-dimensional knapsack problem, dynamic programming.**

I. Introduction

THE enormous popularity of smart user equipment, e.g., smartphones, tablets and wearable devices, has motivated numerous novel applications, which have become an indispensable part of everyday life over recent years. These novel applications, such as augmented reality (AR), virtual reality (VR) and real-time game, are typically computation-intensive and delay-sensitive, which poses stringent requirements, especially on delay [1]. However, it is a great challenge for smart user equipment (UE) to support these applications because the user equipment is resource-constrained, equipped with limited computation and battery capacity due to the limited physical size [2]. The tension between computation-intensive and delay-sensitive applications and resource-constrained user equipment have a negative effect on the development of these novel applications [2] [3].

Edge computing (EC) has emerged as a promising technique to tackle the aforementioned problem by offloading computation-intensive and delay-sensitive applications from user equipment to the edge computing node [4]. The computation resources in EC are deployed at the network edge, such as at base stations (BSs), adjacent to BSs or adjacent to access points (APs), which are in close proximity to users and more computationally capable than that at user equipment [5] [6]. The proximity and sufficiency of computation resources at the edge computing node lead to short transfer time and short execution time, respectively. Therefore, EC is a promising solution to reduce delay significantly, meeting the stringent requirement of these novel applications [7] [8]. Additionally, EC can potentially prolong the battery lifetime of user equipment because the computation-intensive applications, which demand high energy consumption, can be executed at the edge computing node instead of user equipment [9] [10]. Due to its outstanding performances of offloading applications with low delay and low energy consumption of user equipment, EC plays a key role in the development of computation-intensive and delay-sensitive applications.

Despite the aforementioned advantages, the performance of offloading applications in EC is affected by multiple factors, including characteristics of applications and user equipment, wireless channel condition, computation capacity at the edge computing node and user requirements. To take full benefits of EC for improving the performance, e.g., reducing delay and energy consumption of user equipment, the offloading decision scheme should be well-designed while taking the aforementioned factors into account. Prior works have made extensive efforts in this area to make offloading decision efficiently [11] [12] [13]. In these works, it is typically assumed that some characteristics of applications, e.g., the uploading and downloading data size for offloading and the number of central processing unit (CPU) cycles required for execution, are exactly known when making offloading decision with the schemes [14] [15] [16].

Such assumption, however, does not hold for continuous applications. Different from the applications with only one user operation such as facial recognition, a continuous application involves multiple subsequent user operations and the user continues interacting with the application after the offloading decision is made, for which the application characteristics are unknown when making offloading decision. User operation in this paper refers to one single operation which involves computation during the execution of a continuous application, such as one movement of the hero for a real-time game or rendering one frame for AR. Generally, when making offloading decision for continuous applications, the number of user operations, the uploading and downloading data size for offloading computation of each user operation, and the number of CPU cycles required to execute computation of each user operation are unknown [17]. For instance, when a user visits a museum with an AR application or plays a real-time game, it is uncertain how the user will operate the continuous application and how long the visit or the real-time game will last before the visit or the game finishes. Therefore, the parameters presented above are unknown when making offloading decision. For this reason, previous offloading decision schemes assuming that these application characteristics are known are not applicable for continuous applications.

Notable continuous applications include AR, VR, real-time game, autonomous driving and navigation. These continuous applications have greatly enriched our daily life, e.g., VR providing immersive experience [18], autonomous driving offering safer transportation [19] and navigation aiding indoor robots [20]. However, to the best of our knowledge, little effort has been made to tackle the challenging problem of designing offloading decision schemes in EC for continuous applications under uncertainty. Here the uncertainty of continuous applications refers to the unknown parameters presented above when making offloading decision. In this paper, we propose a solution to this challenge, that is, making offloading decision with fewer characteristics of applications. To solve this optimization problem under uncertainty, we adopt chance constrained programming. First introduced by Charnes and Cooper [21], Chance constrained programming falls into a class of stochastic programming which takes the randomness in input parameters into consideration. The main feature of chance constrained programming is that the constraint under uncertainty is satisfied with a predefined probability. The advantages of chance constrained programming over other approaches to solve optimization problems under uncertainty have been demonstrated by extensive applications [22] [23]. In this paper, the optimization problem under uncertainty is transformed into a deterministic one based on chance constrained programming.

A. Main Contributions

The main contributions of this paper can be summarized as follows:

- We investigate the computation offloading decision in EC for continuous applications under uncertainty. In the multi-user scenario, an energy consumption constrained average response time minimization problem for continuous applications under uncertainty is formulated. Notably, the number of user operations, the uploading and downloading data size for offloading computation of each user operation, and the number of CPU cycles required to execute computation of each user operation are unknown when making offloading decision.
- The Response Time-Improved Offloading algorithm with Energy Constraint (RTIOEC) is designed to solve the continuous application offloading decision problem, which is applicable under uncertainty. To make offloading decision with fewer characteristics of applications, the RTIOEC algorithm first decomposes this problem under uncertainty into two sub-problems. The first sub-problem, inheriting the uncertainty, is transformed into a solvable deterministic optimization problem based on chance constrained programming. The second sub-problem is transformed into a three-dimensional knapsack problem and solved by dynamic programming.
- The experimental results and analysis are provided to evaluate the performance of the RTIOEC algorithm. The average response time of continuous applications decreases significantly with the RTIOEC algorithm while saving the energy of use equipment. The upper bound of the violation probability of the energy consumption constraint under uncertainty is controllable and can be specified by tuning the risk level. Additionally, we reveal the impacts of the parameters of continuous applications on offloading with the RTIOEC algorithm in EC.

B. Related Work

Due to the potential benefits mentioned above, application offloading in EC has attracted significant attention over recent years. To further improve the performance of application offloading, extensive efforts have been made to minimize delay. Ren et al. [24] derive the closed-form expression of minimum delay and propose an offloading algorithm jointly optimizing computation and communication resource management to decrease the weighted-sum delay in a multi-user EC scenario. Taking both transfer time and execution time into account, Fan et al. [25] design an application aware workload allocation scheme to minimize the delay of applications. Ketykó et al. [26] propose a general offloading model in EC considering completion time and investigate the performance of a heuristic algorithm and an exact algorithm in this scenario. Elbamby et al. [27] construct a clustering method to group users based on mutual interests and spatial proximity and solve the offloading problem in EC as a matching game to minimize completion time under the reliability constraint. Ni et al. [28] develop a resource allocation strategy for offloading in EC considering the completion time of applications and the price cost. The authors also propose algorithms to predict the completion time.

The energy consumption of user equipment is also a main concern for offloading in EC and has been jointly studied with the delay of applications. To reduce the completion time and application failure under energy constraint, Mao et al. [29] design an effective computation offloading strategy in a green EC system. Wang et al. [17] formulate an energy constrained delay minimization problem and a delay constrained energy consumption minimization problem, respectively. An optimal offloading algorithm leveraging on univariate search technique is proposed to tackle the former problem and the latter one is transformed into a convex problem to find the optimal solution. Liu et al. [30] formulate an average completion time minimization problem with the average power constraint at user equipment and design a search algorithm to efficiently obtain the optimal solution of the problem. Based on a game theoretical approach, Chen et al. [1] propose a distributed algorithm to make offloading decision and quantify its efficiency ratio in terms of reducing delay and saving energy of user equipment. Dinh et al. [14] present semidefinite relaxation-based algorithms to achieve the tradeoff between the energy consumption of user equipment and the delay of applications. Sardellitti et al. [31] formulate the computation offloading problem with the objective of minimizing the energy consumption under delay constraint and present an iterate algorithm with successive convex approximation technique to obtain the solution.

Besides the extensive research efforts that assume the characteristics of applications are known, some studies pay particular attention to offloading in the presence of unknown parameters, especially the arrival of tasks. To optimize offloading schedule in this scenario, Li et al. [32] propose a resource allocation scheme in EC to improve the throughput and reduce delay. Xu et al. [33] formulate the offloading problem as a Markov decision process and design an efficient reinforcement learning-based algorithm to reduce delay and operational cost. To generate offloading schedule in EC, Lyu et al. [34] convert the stochastic optimization problem to deterministic optimization based on a perturbed Lyapunov technique and solve it as a knapsack problem. Zhang et al. [35] propose a predictive combination-mode scheme in EC to reduce the offloading cost. Mao et al. [36] present an online offloading algorithm based on Lyapunov optimization in multi-user EC system to minimize energy consumption.

Different from the previous work assuming that the characteristics of applications are known or just taking the randomness of task arrivals into consideration, we investigate the offloading decision algorithm among multiple users in EC for continuous applications under uncertainty, in other words, making offloading decision with much fewer characteristics of applications.

II. Stochastic System Model

We consider a general EC system with multiple users and an edge computing node as shown in Fig. 1. The set of users requesting offloading continuous applications to the edge computing node is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$, where N is the total number of users. Each user in the set has a continuous application that can be offloaded to the edge computing node. As described previously, some characteristics of continuous applications are unknown when making offloading decision. To represent these uncertain characteristics, they are modeled as random variables. Therefore, the continuous application of user i is denoted by $F_i \triangleq (S_i,\ M_i,\ \boldsymbol{b_{i,up}},\ \boldsymbol{b_{i,down}},\ \boldsymbol{c_i})$, where M_i, $\boldsymbol{b_{i,up}}$, $\boldsymbol{b_{i,down}}$ and $\boldsymbol{c_i}$ are random variables. S_i is the type of continuous application F_i. In this paper, different

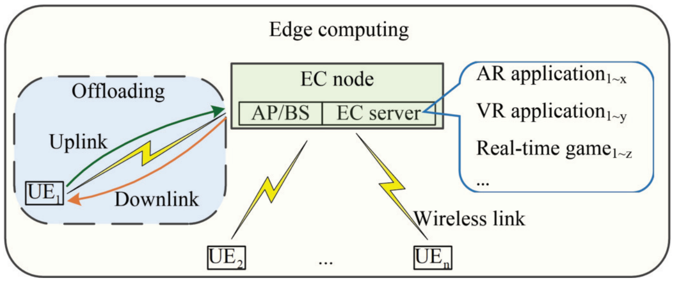


Fig. 1. An illustration of offloading continuous applications with multiple users.

continuous applications that are based on the same technology are considered of different types. M_i is a random variable representing the number of user operations of continuous application F_i. As the description in Section I, a user operation in this paper is one single operation which involves computation during the execution of a continuous application, e.g., a movement of the hero for a real-time game and rendering one frame for AR. The random vectors $\boldsymbol{b_{i,up}} \triangleq (b_{i1,up}, b_{i2,up}, \ldots, b_{iM_i,up})$ and $\boldsymbol{b_{i,down}} \triangleq (b_{i1,down}, b_{i2,down}, \ldots, b_{iM_i,down})$ are the uploading and downloading data size for offloading computation of all the user operations of continuous application F_i. $\boldsymbol{c_i} \triangleq (c_{i1}, c_{i2}, \ldots, c_{iM_i})$ is also a random vector representing the number of CPU cycles required to execute computation of all the user operations of continuous application F_i. Although random variables M_i, $\boldsymbol{b_{i,up}}$, $\boldsymbol{b_{i,down}}$, and $\boldsymbol{c_i}$ are unknown when making offloading decision, the probability distributions governing the variables can be obtained by the previous data of applications in the same type. The offloading decision algorithm proposed in this paper focuses on this uncertain scenario, i.e., making offloading decision with fewer characteristics of applications.

The set of possible execution locations is denoted as $\mathcal{L} = \{local, offloading\}$. $local$ represents the continuous application is executed at the user equipment, i.e., locally. $offloading$ represents the application is executed at the edge computing node. It is assumed that both the user equipment and the edge computing node can support the continuous application, which is a common assumption in similar works [1] [29] [37]. Additionally, the application is atomic and sequentially dependent. Therefore, only full offloading is applicable in this paper.

A. Local Execution

If continuous application F_i is executed locally, the time for executing the jth user operation is given as

$$t_{ij}^{loc} = \frac{c_{ij}}{f_i^{loc}}, \tag{1}$$

where f_i^{loc} denotes the computation capacity of user i, i.e., CPU cycles per second. As mentioned above, the number of CPU cycles required to execute computation of each user operation is unknown and modeled as a random variable when making offloading decision. This is different from other applications and a difficulty for the offloading decision scheme design.

According to circuit theories and the assumption in [17], [29] and [38] that the user equipment is low-power and operates at low voltage, the power consumption for executing continuous application F_i locally is modeled as in [39] and [40], i.e., denoted as

$$p_i^{loc} = \kappa_i^{loc}\left(f_i^{loc}\right)^3, \tag{2}$$

where κ_i^{loc} is the effective switched capacitance of user i, which depends on the chip architecture at the user equipment.

According to (1) and (2), the total energy consumed by the user equipment for executing continuous application F_i locally is denoted as

$$\tilde{e}_i^{loc} = p_i^{loc} * \sum_{j=1}^{M_i} t_{ij}^{loc} = \left(f_i^{loc}\right)^2 \kappa_i^{loc} \sum_{j=1}^{M_i} c_{ij}. \tag{3}$$

B. Edge Computing Node Execution

If a continuous application is offloaded to the edge computing node by user i, the incurred data transmission is bidirectional. In the uplink, user i uploads input parameters to the edge computing node. In the downlink, user i downloads computation result from the edge computing node. Considering the frequency division duplex mode, the uplink and downlink transmission rate for offloading continuous application F_i are denoted as

$$R_{i,up} = W_{i,up}^{off} log_2 \left[1 + \frac{P_{i,T} g_{i,up} (d_0/d_i)^{\varphi}}{N_o W_{i,up}^{off}}\right], \tag{4}$$

$$R_{i,down} = W_{i,down}^{off} log_2 \left[1 + \frac{P_{i,F} g_{i,down} (d_0/d_i)^{\varphi}}{N_o W_{i,down}^{off}}\right], \tag{5}$$

where $W_{i,up}^{off}$ and $W_{i,down}^{off}$ are the uplink and downlink channel bandwidth allocated to user i for offloading, respectively. In this paper, the fixed bandwidth allocation is adopted [15] [41]. $P_{i,T}$ is the transmit power of user i for uploading input parameters, which can be adjusted for saving energy of user equipment [15] [42]. $P_{i,F}$ is the transmit power of the AP at the edge computing node. $g_{i,up}(d_0/d_i)^{\varphi}$ and $g_{i,down}(d_0/d_i)^{\varphi}$ are uplink and downlink channel gain and are approximated by the average value. $g_{i,up}$ and $g_{i,down}$ are the uplink and downlink channel fading coefficients, respectively. d_0 is the reference distance. d_i is the distance from user i to the AP. φ is the path loss exponent. N_o is the density of white Guassian noise power.

The transfer time of uploading for the jth user operation of continuous application F_i is given as

$$t_{ij,up}^{off} = \frac{b_{ij,up}}{R_{i,up}}. \tag{6}$$

Similarly, the transfer time of downloading for the jth user operation of continuous application F_i is defined as

$$t_{ij,down}^{off} = \frac{b_{ij,down}}{R_{i,down}}. \tag{7}$$

If a continuous application is offloaded to the edge computing node, the total energy consumed by the user equipment is just for uploading and downloading. Because the execution of the continuous application is at the edge computing node, which does not consume any energy of the user equipment. The total energy consumed by the user equipment for offloading continuous application F_i to the edge node is given as

$$\tilde{e}_i^{off} = (P_{i,0} + k_i P_{i,T}) \sum_{j=1}^{M_i} t_{ij,up}^{off} + P_{i,r} \sum_{j=1}^{M_i} t_{ij,down}^{off}, \tag{8}$$

where $P_{i,0} + k_i P_{i,T}$ is the total transmit power for uploading. $P_{i,0}$ accounts for the constant circuit power of user i, which includes the power used by filter and digital-to-analog converter. k_i is the efficient factor of the power amplifier for user i. $P_{i,r}$ is the receive power of user i.

III. Problem Formulation

A. Performance Metric: Average Response Time

Many applications to be offloaded are delay-sensitive, which motivates extensive works on minimizing the completion time of applications in EC, e.g., [24], [29] and [30]. However, completion time is not a valid metric for continuous application performance. The reason is that the completion time of a continuous application mainly depends on the continuous interaction between user and application and other factors instead of offloading decision scheme. For example, the completion time of a real-time game mainly depends on when the user exits the game or when the game is over. In this paper, the average response time for continuous applications over all users is introduced as the system performance metric. The response time is defined as the average completion time of user operations for a continuous application. It has great impact on improving quality-of-experience for delay-sensitive applications. For example, if a user plays table tennis with AR, the response time, i.e., the average completion time of rendering one frame, directly impacts when the user sees the track change after batting the table tennis ball.

For local execution, the completion time of a user operation is just the execution time at the user equipment. Therefore, the response time of executing continuous application F_i locally is given as

$$t_i^{loc} = \frac{1}{M_i} \sum_{j=1}^{M_i} t_{ij}^{loc} . \tag{9}$$

For edge computing node execution, the completion time of a user operation is the sum of the execution time at the edge computing node and the time of data transmission between the user equipment and the AP. Therefore, the response time of executing continuous application F_i at the edge computing node is denoted as

$$t_i^{off} = \frac{1}{M_i} \sum_{j=1}^{M_i} \left(\frac{c_{ij}}{f_i^{off}} + \frac{b_{ij,up}}{R_{i,up}} + \frac{b_{ij,down}}{R_{i,down}} \right), \tag{10}$$

where f_i^{off} is the computation capacity allocated to user i at the edge computing node, which depends on the contract subscribed by user i from the edge cloud service providers [1]. Here the delay caused by the data transmission between the AP and the edge computing server is not taken into account. Because it can be ignored compared to the delay of execution and transmission between user equipment and the AP [15] [43].

B. Energy Consumption Constrained Average Response Time Minimization Problem

Continuous applications to be offloaded are computation-intensive, which demand high energy consumption. Most of the user equipment is battery-powered and has limited battery lifetime [2]. Therefore, energy consumption constraint should be considered for prolonging the battery lifetime of user equipment. In this paper, this constraint is captured by a predefined energy threshold θ_i, which depends on the type of continuous application S_i and the user requirement. Specifically, θ_i indicates the energy saving requirement for offloading continuous application F_i. If the energy of user equipment saved by offloading is greater than or equal to θ_i, it is feasible that user i offloads the continuous application. Otherwise, the continuous application is executed locally.

Consequently, the energy consumption constrained average response time minimization problem among multiple users for continuous applications under uncertainty is formulated as:

$$\begin{aligned}
\mathbf{P_1} \quad & \min_{I_i^{loc}, I_i^{off}, P_{i,T}} \sum_{i=1}^{N} \left(I_i^{loc} t_i^{loc} + I_i^{off} t_i^{off} \right) / N \\
s.t.\ & \mathrm{C1}: \ \tilde{e}_i^{loc} - \tilde{e}_i^{off} \geq \theta_i, \ \forall i \in \mathcal{N}, \\
& \mathrm{C2}: \ I_i^{loc} + I_i^{off} = 1, \forall i \in \mathcal{N}, \\
& \mathrm{C3}: \ \sum_{i=1}^{N} I_i^{off} W_{i,up}^{off} \leq W_{up,max}, \\
& \mathrm{C4}: \ \sum_{i=1}^{N} I_i^{off} W_{i,down}^{off} \leq W_{down,max}, \\
& \mathrm{C5}: \sum_{i=1}^{N} I_i^{off} f_i^{off} \leq f_{max}^{off}, \\
& \mathrm{C6}: \ P_{i,T} \in [0, p_{i,max}], \forall i \in \mathcal{N}, \\
& \mathrm{C7}: \ I_i^{loc} \in \{0, \ 1\}, \ \forall i \in \mathcal{N}, \\
& \mathrm{C8}: \ I_i^{off} \in \{0, \ 1\}, \ \forall i \in \mathcal{N}.
\end{aligned} \tag{11}$$

I_i^{loc} is an indicator function with $I_i^{loc} = 1$ if continuous application F_i is executed locally and $I_i^{loc} = 0$ otherwise. Similarly, $I_i^{off} = 1$ indicates that continuous application F_i is offloaded to the edge computing node. $W_{up,max}$ and $W_{down,max}$ are the total uplink and downlink channel bandwidth, respectively. f_{max}^{off} is the total computation capacity at the edge computing node. $p_{i,max}$ is the maximum transmit power of user i.

In $\mathbf{P_1}$, C1 is the energy consumption constraint, which involves uncertainty. C2 reflects that the continuous application is executed either at the user equipment or at the edge computing node. C3 and C4 are the uplink and downlink channel bandwidth constraints, respectively. C5 is the computation capacity constraint of the edge computing node. C6 is the transmit power constraint of user i. C7 and C8 reflects the feasible set of I_i^{loc} and I_i^{off}, respectively.

It is worth emphasizing that several inputs are unknown and modeled as random variables in the objective function and C1. To provide rich modeling flexibility, the value of θ_i can be positive, 0, and even negative. When the response time is of much more significance than the energy consumption or the applications are extremely delay-sensitive, e.g., Tactile Internet applications [44], the value of θ_i can be set to be 0 or negative.

IV. Response Time-Improved Offloading Algorithm with Energy Constraint

A. Decomposing $\mathbf{P_1}$ into Two Sub-problems

In $\mathbf{P_1}$, the presence of inter-dependent variables and unknown parameters pose a challenge to solving the problem directly. Instead, we decompose $\mathbf{P_1}$ equivalently into two sub-problems, namely $\mathbf{SP_1}$ and $\mathbf{SP_2}$, to obtain the transmit power for uploading and execution locations separately. From $\mathbf{P_1}$, it can be seen that the execution location of continuous application F_i depends on the transmit power of user i and execution locations of the other applications due to C3, C4 and C5. However, the transmit power of user i for uploading does not depend on execution locations of the other applications with the given decision that user i will offload continuous application F_i to the edge computing node. That means if we assume continuous application F_i will be offloaded to the edge computing node, the transmit power of user i, i.e., variable $P_{i,T}$, can be decoupled from the execution locations of the other applications, i.e., other variables in $\mathbf{P_1}$. Therefore, we can first obtain the transmit power of user i for uploading with the assumption that user i will offload continuous application F_i to the edge computing node. After that, $P_{i,T}$ is excluded from the variables of $\mathbf{P_1}$ and the actual execution locations can be obtained based on the solution of transmit power for each user. Therefore, $\mathbf{P_1}$ can be equivalently decomposed into two sub-problems, i.e., $\mathbf{SP_1}$ to obtain the transmit power for uploading and $\mathbf{SP_2}$ to obtain the execution locations. Finally, we can obtain the solution to $\mathbf{P_1}$ by combining the results of $\mathbf{SP_1}$ and $\mathbf{SP_2}$.

B. $\mathbf{SP_1}$: Solution for $P_{i,T}$

According to $\mathbf{P_1}$ and the analysis above, the sub-problem of obtaining the transmit power to upload input parameters for each user with the assumption that the user will offload the continuous application to the edge computing node is formulated as

$$\begin{aligned}
\mathbf{SP_1} \quad & \min_{P_{i,T}} \frac{1}{M_i} \sum_{j=1}^{M_i} \left(\frac{c_{ij}}{f_i^{off}} + \frac{b_{ij,up}}{R_{i,up}} + \frac{b_{ij,down}}{R_{i,down}} \right) \\
& s.t. \ \mathrm{C1}, \ \mathrm{C6},
\end{aligned} \tag{12}$$

wherein (10) is substituted into the objective function, i.e., t_i^{off}, to facilitate the analysis. It is worth mentioning that I_i^{loc} and I_i^{off} are excluded from the variables of $\mathbf{SP_1}$ because continuous application F_i is assumed to be offloaded to the edge computing node here.

The sub-problem $\mathbf{SP_1}$ involves uncertainty in the objective function and the constraint, which is a stochastic programming problem. As mentioned above, much of the difficulty to solve $\mathbf{P_1}$ arises from the uncertainty. To obtain the transmit power of user i, the objective function of $\mathbf{SP_1}$ is first converted into a deterministic form by replacing the response time with its expected value. The expected value of response time for continuous application F_i is given by

$$\begin{aligned} E(t_i^{off}) =& E\left(\frac{1}{M_i}\sum_{j=1}^{M_i}\left(\frac{c_{ij}}{f_i^{off}}+\frac{b_{ij,up}}{R_{i,up}}+\frac{b_{ij,down}}{R_{i,down}}\right)\right) \\ =& \frac{E(c_{ij})}{f_i^{off}}+\frac{E(b_{ij,up})}{R_{i,up}}+\frac{E(b_{ij,down})}{R_{i,down}}, \end{aligned} \tag{13}$$

where $E(\alpha)$ is the expected value of α.

After the replacement of the objective function, the uncertainty of this sub-problem only remains in C1. Next C1 is adapted to convert $\mathbf{SP_1}$ into a solvable deterministic optimization problem. Based on chance constrained programming, the constraint C1 under uncertainty is replaced by a probabilistic constraint to satisfy C1 with a specified high probability, denoted as

$$\text{C9}: \ \text{Prob}\left\{\tilde{e}_i^{loc}-\tilde{e}_i^{off}\geq\theta_i\right\}\geq 1-\epsilon,\ \forall i\in\mathcal{N}, \tag{14}$$

where Prob $\{\alpha\}$ denotes the probability of the event α [45]. ϵ is the predefined upper bound of the probability for C1 to be violated, i.e., the risk level [22] [46]. Therefor, $1-\epsilon$ is the probability level to satisfy C1, which is usually high.

Obviously, the uploading data size for offloading is positive. We substitute (3), (6), (7) and (8) into (14) and rewrite C9 as

$$\begin{aligned} & Prob\left\{\frac{k_iP_{i,T}+P_{i,0}}{R_{i,up}}\leq\right. \\ & \left.\frac{\kappa_i^{loc}\left(f_i^{loc}\right)^2\sum\limits_{j=1}^{M_i}c_{ij}-\theta_i-\frac{\sum\limits_{j=1}^{M_i}b_{ij,down}}{R_{i,down}}P_{i,r}}{\sum\limits_{j=1}^{M_i}b_{ij,up}}\right\}\geq 1-\epsilon,\ \forall i\in\mathcal{N}, \end{aligned} \tag{15}$$

where, as mentioned above, $b_{ij,up}$, $b_{ij,down}$ and c_{ij} are not known when making offloading decision, raising the uncertainty and difficulty of the problem. For notation convenience, we define $\sum\limits_{j=1}^{M_i}b_{ij,up}\triangleq\tilde{b}_{i,up}$, $\sum\limits_{j=1}^{M_i}b_{ij,down}\triangleq\tilde{b}_{i,down}$ and $\sum\limits_{j=1}^{M_i}c_{ij}\triangleq\tilde{c}_i$.

To convert the probabilistic constraint (15) into a deterministic form, we perform the following transformations from (16) to (19). First, we denote the bulky right-hand side of the probability in (15) by $\mathcal{G}$. That is

$$\begin{aligned} \mathcal{G}&\triangleq g\left(\tilde{b}_{i,up},\tilde{b}_{i,down},\tilde{c}_i\right) \\ &=\frac{\kappa_i^{loc}\left(f_i^{loc}\right)^2\tilde{c}_i-\theta_i-\frac{\tilde{b}_{i,down}}{R_{i,down}}P_{i,r}}{\tilde{b}_{i,up}}. \end{aligned} \tag{16}$$

Then, based on chance constrained programming, we need to obtain the cumulative distribution function of $\mathcal{G}$. Since there are three random variables in (16), the cumulative distribution function of $\mathcal{G}$ can be obtained by triple integral, which is given as

$$F\left(\mathcal{G}\right)=\iiint\limits_{\Omega}f_i\left(\tilde{b}_{i,up},\tilde{b}_{i,down},\tilde{c}_i\right)d\tilde{b}_{i,up}d\tilde{b}_{i,down}d\tilde{c}_i \tag{17}$$

where $f_i\left(\tilde{b}_{i,up},\tilde{b}_{i,down},\tilde{c}_i\right)$ is the joint probability density function (PDF) of $\tilde{b}_{i,up}$, $\tilde{b}_{i,down}$ and $\tilde{c}_i$, which can be obtained by the previous data of applications in the same type. Ω is the domain defined by $\left\{g\left(\tilde{b}_{i,up},\tilde{b}_{i,down},\tilde{c}_i\right)\leq\mathcal{G},\tilde{b}_{i,up}\geq 0,\tilde{b}_{i,down}\geq 0,\tilde{c}_i\geq 0\right\}$ on the three-dimensional Euclidean space because the uploading and downloading data size and the number of CPU cycles required for execution are obviously positive. As described in Section III, the value of θ_i can be positive, 0 and even negative. For notation convenience, we define $\hat{b}_{i,down}\triangleq max\left\{0,-\frac{\theta_i+\mathcal{G}\tilde{b}_{i,up}}{P_{i,r}/R_{i,down}}\right\}$ to invert (17) to a unified form. Then, $F\left(\mathcal{G}\right)$ can be expressed as

$$\begin{aligned} F\left(\mathcal{G}\right)=&\int_0^{+\infty}\int_{\hat{b}_{i,down}}^{+\infty}\int_0^{\frac{\theta_i+\frac{P_{i,r}}{R_{i,down}}\tilde{b}_{i,down}+\mathcal{G}\tilde{b}_{i,up}}{\kappa_i^{loc}\left(f_i^{loc}\right)^2}} \\ & f_i\left(\tilde{b}_{i,up},\tilde{b}_{i,down},\tilde{c}_i\right)d\tilde{c}_id\tilde{b}_{i,down}d\tilde{b}_{i,up}. \end{aligned} \tag{18}$$

Therefore, the deterministic form of (15) is given by

$$\text{C10}: \frac{k_iP_{i,T}+P_{i,0}}{R_{i,up}}\ \leq F^{-1}\left(\epsilon\right),\forall i\in\mathcal{N}, \tag{19}$$

where F^{-1} is the inverse function of F.

With the derivation based on chance constrained programming for C1, the deterministic equivalent of $\mathbf{SP_1}$ is given by

$$\begin{aligned} \mathbf{SP}_1^{\star}\ \min_{P_{i,T}}&\left(\frac{E\left(c_{ij}\right)}{f_i^{off}}+\frac{E\left(b_{ij,up}\right)}{R_{i,up}}+\frac{E\left(b_{ij,down}\right)}{R_{i,down}}\right) \\ s.t.&\ \text{C6},\ \text{C10}. \end{aligned} \tag{20}$$

Substituting (4) into (19), the function of the left-hand term of C10 is given as

$$h\left(P_{i,T}\right)=\frac{k_iP_{i,T}+P_{i,0}}{W_{i,up}^{off}log_2\left[1+\frac{P_{i,T}g_{i,up}(d_0/d_i)^{\varphi}}{N_oW_{i,up}^{off}}\right]}. \tag{21}$$

The unimodality of function $h\left(P_{i,T}\right)$ can be conveniently proven by calculating its first derivative (Please refer to Appendix). Specifically, it is strictly decreasing on interval $(0,p_{i,T}^0)$ and is strictly increasing on interval $(p_{i,T}^0,+\infty)$, where $p_{i,T}^0$ represents the extreme point. In the objective function of $\mathbf{SP}_1^{\star}$, only term $\frac{1}{R_{i,up}}$ is related to variable $P_{i,T}$. And $R_{i,up}$ is a strictly increasing function of variable $P_{i,T}$ on interval $(0,+\infty)$ according to Eq. (4). That means the objective function of $\mathbf{SP}_1^{\star}$ is a strictly decreasing function of variable $P_{i,T}$ on interval $(0,+\infty)$. Therefore, the solution to transmit power of user i for uploading is unique and can be obtained by solving $\mathbf{SP}_1^{\star}$ if the intersection of C6 and the solution set of C10 is a nonempty set. In this case, it is feasible that user i offloads the continuous application to the edge computing node with the obtained transmit power for uploading. However, the intersection of C6 and the solution set of C10 may be empty in some cases. For example, the uploading and downloading data size for offloading is large and the number of CPU cycles required for execution is small. With the given parameters, any value of the $P_{i,T}$ on the interval of C6 is not available to satisfy C1 with the specified high probability in this case. For that, continuous application F_i cannot be offloaded to the edge computing node.

C. $\mathbf{SP_2}$: *Offloading Decision*

If any value of the $P_{i,T}$ is not available to offload continuous application F_i with the constraints of $\mathbf{SP}_1^\star$, continuous application F_i will be executed locally. For user i, if the transmit power for uploading can be obtained by solving $\mathbf{SP}_1^\star$, the offloading needs to satisfy an additional response time constraint. The constraint is that the response time of continuous application F_i executed at the edge computing node is shorter than that of local execution. Because it is obvious that for minimizing the average response time, continuous application F_i will be executed locally if user i can not reduce the response time of continuous application F_i by offloading. The set of these users, of whom the continuous applications can only be executed locally due to the two properties, is denoted by $\mathcal{N}_1$. For these users, we have the functions $I_i^{loc} = 1$ and $I_i^{off} = 0$ to indicate the actual execution locations of their continuous applications. The other users, i.e., the ones who satisfy the constraints of $\mathbf{SP}_1^\star$ and can reduce the response time by offloading, are the candidate users to offload their continuous applications. Such set of users, denoted by $\mathcal{N}_2$, can be obtained by solving $\mathbf{SP}_1^\star$ and comparing the response time of executing the continuous application at different locations. The cardinality of $\mathcal{N}_2$ is denoted by N_2.

Due to the constraints of the total uplink channel bandwidth, the total downlink channel bandwidth and the total computation capacity at the edge computing node, it may be impossible for all the users in $\mathcal{N}_2$ to offload their continuous applications. Therefore, the second sub-problem for solving $\mathbf{P_1}$ is to obtain the execution locations of the continuous applications for users in $\mathcal{N}_2$. Because the continuous applications of users in $\mathcal{N}_1$ can only be executed locally, only the execution locations of the continuous applications for users in $\mathcal{N}_2$ can be optimized to reduce the average response time. Similar to $\mathbf{SP}_1^\star$, the response time is also replaced by the expected value. The response time of executing continuous application F_i at the edge computing node is given as (13). The response time of executing continuous application F_i locally is given by

$$E\left(t_i^{loc}\right) = E\left(\frac{1}{M_i}\sum_{j=1}^{M_i} t_{ij}^{loc}\right) = \frac{E\left(c_{ij}\right)}{f_i^{loc}}. \tag{22}$$

Then, the sub-problem is formulated as

$$\begin{aligned}
\mathbf{SP_2}\ \ & \min_{I_i^{loc}, I_i^{off}} \sum_{i=1}^{N_2} \left(I_i^{loc} E(t_i^{loc}) + I_i^{off} E(t_i^{off})\right)/N_2 \\
& s.t.\ \text{C11}: \ I_i^{loc} + I_i^{off} = 1, \forall i \in \mathcal{N}_2, \\
& \quad \text{C12}: \ \sum_{i=1}^{N_2} I_i^{off} W_{i,up}^{off} \le W_{up,max}, \\
& \quad \text{C13}: \ \sum_{i=1}^{N_2} I_i^{off} W_{i,down}^{off} \le W_{down,max}, \\
& \quad \text{C14}: \ \sum_{i=1}^{N_2} I_i^{off} f_i^{off} \le f_{max}^{off}, \\
& \quad \text{C15}: \ I_i^{loc} \in \{0,\ 1\},\ \forall i \in \mathcal{N}_2, \\
& \quad \text{C16}: \ I_i^{off} \in \{0,\ 1\},\ \forall i \in \mathcal{N}_2.
\end{aligned} \tag{23}$$

It is worthwhile to note that the transmit power to upload input parameters for each user in $\mathcal{N}_2$ is excluded from the variables of $\mathbf{SP_2}$ because it has been obtained by solving $\mathbf{SP_1}$. Additionally, $\mathbf{SP_2}$ does not involve uncertainty, which can be solved as a deterministic optimization problem.

From $\mathbf{SP_2}$, we can see that the objective of minimizing the expected value of the average response time can be converted into maximize the expected value of the total response time reduced by offloading compared to that of local execution. Therefore, the equivalent of $\mathbf{SP_2}$ is given by

$$\begin{aligned}
\mathbf{SP}_2^\star\ \ & \max_{I_i^{off}} \sum_{i=1}^{N_2} \left[I_i^{off}\left(E(t_i^{loc}) - E(t_i^{off})\right)\right] \\
& s.t.\ \text{C12},\ \text{C13},\ \text{C14},\ \text{C16}.
\end{aligned} \tag{24}$$

The problem $\mathbf{SP}_2^\star$ is essentially a three-dimensional knapsack problem. The continuous application of each user in set $\mathcal{N}_2$ is an item which has a three-dimensional size, i.e., the uplink channel bandwidth, the downlink channel bandwidth and the computation capacity allocated at the edge computing node. The value of an item is the expected response time reduction of the continuous application for offloading. We solve $\mathbf{SP}_2^\star$ by dynamic programming in the following, which can obtain the optimal solution of a three-dimensional knapsack problem [47]. The expected response time reduction of offloading the continuous application for user i is first expressed as

$$V_i = E(t_i^{loc}) - E(t_i^{off}). \tag{25}$$

To break down the three-dimensional knapsack problem into a sequence of steps based on dynamic programming, we define the maximum expected response reduction at the x-th ($0 \le x \le N_2$) step as

$$\begin{aligned}
& V_x\left(W_{up}, W_{down}, f^{off}\right) \triangleq \max\Bigg\{\sum_{i=1}^{x} I_i^{off} V_i \ | \\
& \sum_{i=1}^{x} I_i^{off} \le W_{up,NO.}, \sum_{i=1}^{x} I_i^{off} \le W_{down,NO.}, \\
& \sum_{i=1}^{x} f_{i,NO.}^{off} \le \lfloor f_{\max}^{off}/f_{unit}^{off} \rfloor \Bigg\},
\end{aligned} \tag{26}$$

where $W_{up,NO.}$ and $W_{down,NO.}$ are the number of uplink and downlink channels based on the bandwidth allocation. $\lfloor \alpha \rfloor$ denotes the floor function of α. f_{unit}^{off} is the unit of computation resources provided by the edge cloud service providers for users to subscribe. $f_{i,NO.}^{off}$ represents the number of units that user i subscribes, which is an integer. As a special case, we define $V_0\left(W_{up}, W_{down}, f^{off}\right) \triangleq 0$. The recursive function to obtain $V_x\left(W_{up}, W_{down}, f^{off}\right)$ is given as

$$V_x\left(W_{up}, W_{down}, f^{off}\right) = \begin{cases} V_{x-1}\left(W_{up}, W_{down}, f^{off}\right), if\ 0 \le f^{off} < f^{off}_{i,NO.}, \\ \max\left\{V_{x-1}\left(W_{up}, W_{down}, f^{off}\right), V_{x-1}\left(W_{up}-1,\right.\right. \\ \left.\left. W_{down}-1, f^{off} - f^{off}_{i,NO.}\right) + V_x\right\}, \\ if\ f^{off}_{i,NO.} \le f^{off} \le \lfloor f^{off}_{\max}/f^{off}_{unit} \rfloor. \end{cases} \tag{27}$$

We can determine solution I_i^{off} corresponding to the procedure of obtaining $V_{N_2}\left(W_{up}, W_{down},\ f^{off}\right)$. Then, indicator function I_i^{loc} is calculated according to C11.

D. RTIOEC Algorithm

As mentioned above, the RTIOEC algorithm first decomposes $\mathbf{P_1}$ into two sub-problems, i.e., $\mathbf{SP_1}$ and $\mathbf{SP_2}$. After that, the process of the RTIOEC algorithm is divided into two stages, each of which solves a sub-problem. The goal in the first stage is to obtain the transmit power of each user for uploading with the assumption that each user will offload its continuous application to the edge computing node. The sub-problem $\mathbf{SP_1}$ inherits the uncertainty of $\mathbf{P_1}$, which is the main difficulty to solve both $\mathbf{P_1}$ and $\mathbf{SP_1}$. To solve the problem, $\mathbf{SP_1}$ is transformed into a solvable deterministic optimization problem based on chance constrained programming, i.e., $\mathbf{SP}_1^\star$. By solving $\mathbf{SP}_1^\star$, we can see if it is feasible that user i offloads the continuous application to the edge computing node with the constraints. And the transmit power of user i for uploading is obtained if the offloading of user i is feasible. In the second stage, the actual execution location is obtained based on the transmit power for uploading obtained in the first stage. The sub-problem $\mathbf{SP_2}$ is transformed into a three-dimensional knapsack problem, i.e., $\mathbf{SP}_2^\star$, which is solved by dynamic programming. The detailed steps of the RTIOEC algorithm are given in Algorithm 1.

Algorithm 1 RTIOEC algorithm

```
Input: $N$, $W_{up,max}$, $W_{down,max}$, $f^{off}_{max}$, $\epsilon$, $f_i^{loc}$, $f_i^{off}$, $W^{off}_{i,up}$, $W^{off}_{i,down}$, $P_{i,F}$, $P_{i,0}$, $P_{i,r}$, $p_{i,max}$, $g_{i,up}$, $g_{i,down}$, $d_0$, $d_i$, $\varphi$, $N_o$, $\kappa_i$, $k_i$, $\theta_i$, $E\left(b_{ij,up}\right)$, $E\left(b_{ij,down}\right)$, $E\left(c_{ij}\right)$, $f_i\left(\tilde{b}_{i,up}, \tilde{b}_{i,down}, \tilde{c}_i\right)$
Output: $I_i^{loc}$, $I_i^{off}$, $P_{i,T}$
1: $\mathcal{N}_2 \leftarrow \varnothing$
2: for $i \leftarrow 1$ to $N$ do
3:     Initialize $\mathbf{SP_1}$
4:     Calculate $F\left(\mathcal{G}\right)$ according to (5) and (18)
5:     Obtain C10 according to (19)
6:     Covert $\mathbf{SP_1}$ into $\mathbf{SP}_1^\star$
7:     Solve problem $\mathbf{SP}_1^\star$ to obtain the $P_{i,T}$
8:     if the $P_{i,T}$ of problem $\mathbf{SP}_1^\star$ exists then
9:         Calculate $E(t_i^{off})$ according to (13)
10:        Calculate $E(t_i^{loc})$ according to (22)
11:        if $E(t_i^{off}) < E(t_i^{loc})$ then
12:            $\mathcal{N}_2 \leftarrow \mathcal{N}_2 \cup \{i\}$
13:        end if
14:    else
15:        $I_i^{loc} \leftarrow 1$, $I_i^{off} \leftarrow 0$
16:    end if
17: end for
18: $W_{up} \leftarrow 0$, $W_{down} \leftarrow 0$, $f^{off} \leftarrow 0$, $V_0\left(W_{up}, W_{down}, f^{off}\right) \leftarrow 0$
19: With $P_{i,T}$, calculate $V_i$ according to (13), (22) and (25)
20: for $x \leftarrow 1$ to $N_2$ do
21:     for $W_{up} \leftarrow 1$ to $W_{up,NO.}$ do
22:         for $W_{down} \leftarrow 1$ to $W_{down,NO.}$ do
23:             for $f^{off} \leftarrow 1$ to$\lfloor f^{off}_{\max}/f^{off}_{unit} \rfloor$ do
24:                 Calculate $V_x\left(W_{up}, W_{down}, f^{off}\right)$ according to (27)
25:             end for
26:         end for
27:     end for
28: end for
29: for $x \leftarrow N_2$ to 1 do
30:     if $V_x(W_{up}, W_{down}, f^{off}) = V_{x-1}(W_{up}-1, W_{down}-1, f^{off} - f^{off}_{i,NO.}) + V_x$, then
31:         $I_x^{off} \leftarrow 1$, $I_x^{loc} \leftarrow 0$, $W_{up} \leftarrow W_{up}-1$, $W_{down} \leftarrow W_{down}-1$, $f^{off} \leftarrow f^{off} - f^{off}_{i,NO.}$
32:     end if
33: end for
```

In the EC system, the RTIOEC algorithm can be deployed and integrated at the edge computing node to make offloading decision for users in the coverage. The users first send the offloading requests to the edge computing node, which contain the parameters of users, i.e., computation capacity, location, effective switched capacitance, efficient factor of power amplifier, maximum transmit power, receive power, constant circuit power and type of the continuous application. After that, the edge computing node inquires the other parameters for offloading and makes offloading decision with the RTIOEC algorithm. The offloading decision is then sent to the user equipment. Finally, based on the offloading decision, the user executes the continuous application locally or offloads it to the edge computing node.

Fig. 2 shows an example of making offloading decision for continuous applications by the RTIOEC algorithm with the same mean but different distributions of random variables. By the definition in Section II, F_i denotes the continuous application of user i, for $i = 1, 2, ..., N$. To demonstrate the impact of PDFs on the offloading decision, here the random variables across different continuous applications have the same mean but potentially different PDFs. And we assume that the fixed parameters are the same for all applications. The PDF of random variables for F_N is flatter than that for F_1. That means the actual values of random variables for F_1 are more squeezed and the constraint under uncertainty of F_1 is more likely to be satisfied. Consequently, the RTIOEC algorithm tends to offload F_1 to the edge computing node while F_N is executed at the user equipment, although the random variables of them have the same mean.

The computational complexity of the RTIOEC algorithm mainly comes from two parts—solving $\mathbf{SP}_1$ for each user and solving $\mathbf{SP}_2$. The computational com-

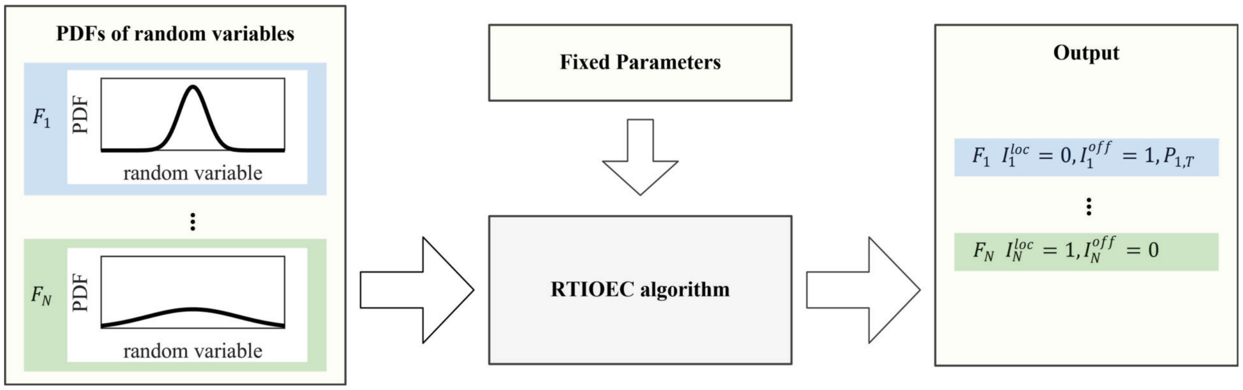


Fig. 2. An illustration of making offloading decision for continuous applications by the RTIOEC algorithm.

plexity of solving $\mathbf{SP_1}$ is $O(N)$ because the operations in Line 3-7 only involve basic arithmetic operations. It takes $O(N_2 W_{up,NO}.W_{down,NO}.\lfloor f_{\max}^{off}/f_{unit}^{off}\rfloor)$ computational time to solve $\mathbf{SP}_2$ by dynamic programming [48]. Because $\mathcal{N}_2$ is a subset of $\mathcal{N}$, the computational complexity of solving $\mathbf{SP}_2$ by dynamic programming is $O(NW_{up,NO}.W_{down,NO}.\ \lfloor f_{\max}^{off}/f_{unit}^{off}\rfloor)$ in the worst case. Therefore, the computational complexity of the RTIOEC algorithm can be obtained by adding them up, i.e., $O(NW_{up,NO}.W_{down,NO}.\lfloor f_{\max}^{off}/f_{unit}^{off}\rfloor)$. The number of users in an AP/BS coverage and satisfying the constraints is not large. And the number of channels and the total computation capacity at the edge computing node are usually limited [15]. Considering the low computational complexity, the RTIOEC algorithm is applicable to make offloading decision for continuous applications.

V. Experiments and Performance Evaluation

In this section, we perform experiments to evaluate the performance of the RTIOEC algorithm. In our experiments, parameters are set in accordance with prior works for meaningful evaluation. The number of users requesting offloading continuous applications to the edge computing node is set to be 20 [10]. The AP has a coverage of 50 meters (m) and the distance from each user to the AP is uniformly distributed within $(0, 50)$ m [49]. The computation capacity of the user equipment is assigned randomly from set $\{0.5, 0.6, ..., 1.0\}$ GHz [10]. The effective switched capacitance of users is 10^{-27} [38]. The efficient factor of power amplifier is set as 18 based on the user equipment [17] [50]. As is described above, we consider the frequency division duplex mode in this paper and set the maximum transmit power and the receive power of users as 0.1 Watt (W) and 0.4 W, respectively [17] [51]. The transmit power of the AP at the edge computing is 0.1 W [52]. The constant circuit power of users is 0.4 W [50]. The uplink and downlink channel bandwidth allocated to a user for offloading is 1 MHz. The uplink and downlink channel fading coefficients are -40 dB. The reference distance is 1 m [29]. The density of white Guassian noise power is set to be -174dBm/Hz [38]. The path loss exponent is 4 [29]. The total uplink and downlink channel bandwidth are both 10 MHz. The total computation capacity at the edge computing node is assumed to be 20 GHz based on the actual resource state of the Nokia Airframe Cloud Server architecture [53]. The computation capacity allocated to each user at the edge computing node is randomly from $\{1.5, 2.0, 2.5\}$ GHz. The number of user operations, the uploading and downloading data size for offloading computation of each user operation, and the number of CPU cycles required to execute computation of each user operation are considered as normal distribution. The means of them are 20000, 50 kbits, 50 kbits and 50 Megacycles, respectively [54]. And the ratio of variance to squared mean is 0.1. The risk level, i.e., the predefined upper bound of the violation probability, is set to be 5%. The parameters of the experiments are set as the description above unless otherwise specified. We repeat experiments 1000 times and show the average results.

In this paper, the RTIOEC algorithm is compared with four schemes, namely local computing (local), Dynamic Programming algorithm based on the Mean of Random Variables (DPMRV), Dynamic Programming algorithm based on the Actual Value of Random Variables (DPAVRV) and Computation Offloading algorithm based on Game Theory (COGT) proposed in [1]. The local computing scheme is that each user only executes its continuous application locally. The DPMRV algorithm is that the random variables, i.e., M_i, $b_{ij,up}$, $b_{ij,down}$ and c_{ij}, are replaced by their mean values, which transforms the problem into a deterministic problem rather than a problem under uncertainty. And the offloading decision made by the DPMRV algorithm is based on dynamic programming. The DPAVRV algorithm is that making offloading decision based on dynamic programming algorithm with the actual value of random variables. Although it is impractical to obtain the actual value of the random variables when making offloading decision, the DPAVRV algorithm can be regarded as the benchmark in this paper. Because it achieves the optimal performance, that is, minimizing the average response time without violating any uncertain constraint. The COGT algorithm is that making offloading decision based on the game theoretical approach admitting the Nash equilibrium, for which the weight of each user is assigned randomly from set $\{1, 2, ..., 5\}$.

Fig. 3 shows the relationship between the violation probability of the energy consumption constraint and the risk level

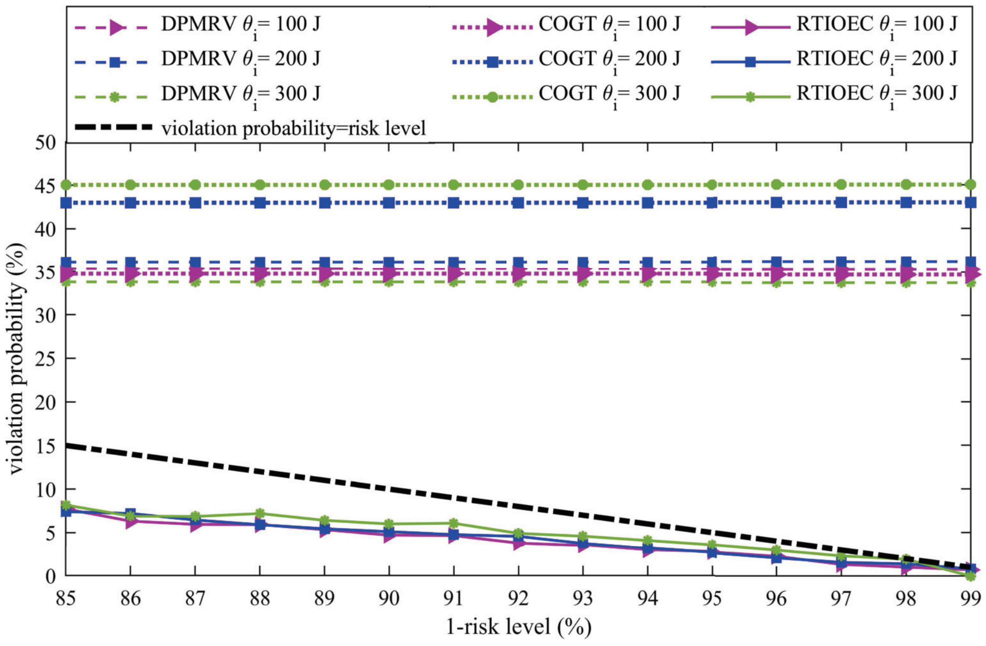


Fig. 3. The violation probability of the constraints with different risk levels.

by varying the energy saving requirement of each offloading user. We observe that the violation probability by the RTIOEC algorithm decreases with the risk level and is consistently much lower than that by the DPMRV algorithm and the COGT algorithm. This is due to the fact that the violation probability by the RTIOEC algorithm is constrained by the risk level. While the violation probability by the DPMRV algorithm and the COGT algorithm is not related to the risk level and out of control. Therefore, the DPMRV algorithm and the COGT algorithm are not applicable to solve the offloading decision problem under uncertainty. In addition, the violation probability by the RTIOEC algorithm stays below the risk level, since the energy consumed by user equipment for offloading is also constrained by the maximum transmit power of users (constraint C6). With the property, the upper bound of the violation probability can be controlled and specified by the RTIOEC algorithm, which is critical to solve the problem under uncertainty and satisfy the energy saving requirement for most users.

Fig. 4 shows the violation probability of the constraints with different ratios of variance to squared mean. For different energy saving requirements, the violation probability by the RTIOEC algorithm increases as the ratio of variance to squared mean increases in the first half of the stage. Different from the consistently high violation probability by the DPMRV algorithm and the COGT algorithm, the violation probability by the RTIOEC algorithm shows different trends in the second half of the stage for different energy saving requirements. For a low energy saving requirement, e.g., 100 J, the violation probability increases slightly because the constraint of risk level is active for more users in this case. When the energy saving requirement is high, e.g., 200 J and 300 J, the violation probability by the RTIOEC algorithm decreases noticeably and approaches 0 for low risk levels. The reason is that when the energy consumption constraint is stringent, few users can satisfy the energy consumption constraint and offload continuous applications to the edge computing node, leading to a low violation probability by the RTIOEC algorithm. Besides, the violation probability by the RTIOEC algorithm stays below the risk level and is far lower than that by the DPMRV algorithm and the COGT algorithm even that the random variables are distributed with large variance. Therefore, offloading by the RTIOEC algorithm can satisfy the energy saving requirement while tolerating wider random variable distributions.

Fig. 5 shows the average energy consumed by user equipment versus different energy saving requirements of each offloading user. The average energy consumption barely fluctuates at the beginning because the loose constraint has little impact on offloading at this stage. Compared to the DPMRV algorithm, the DPAVRV algorithm and the COGT algorithm, the RTIOEC algorithm takes a more conservative approach to offloading continuous applications for considering the uncertainty. Therefore, the average energy consumption by the RTIOEC algorithm decreases and reaches the minimum earlier than that by the other three algorithms. As the energy saving requirement continues rising, the average energy consumption increases, and finally stabilizes around that of local execution when almost no user can satisfy the energy consumption constraint and offload continuous applications. Since the RTIOEC algorithm takes into account the uncertainty of application characteristics in order to achieve low violation probability, the average energy consumption may not be the lowest in some cases. However, the significantly lower violation probability means that more user requirements are satisfied. And offloading continuous applications by the RTIOEC algorithm can generally save the energy of user equipment compared to executing all the continuous applications locally.

Fig. 6 reveals the impact of energy saving requirement

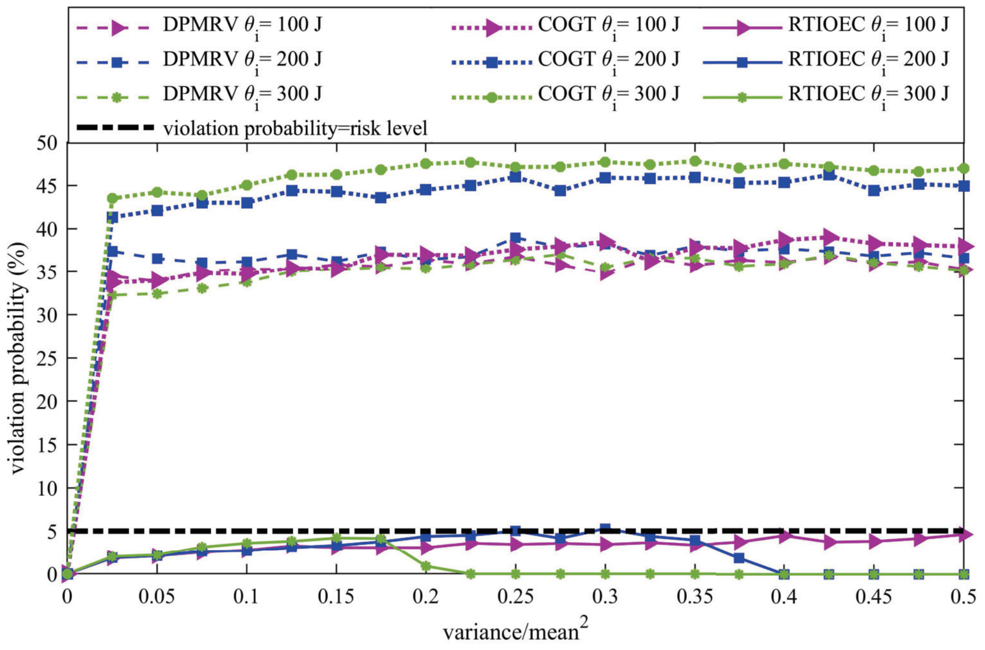


Fig. 4. The violation probability of the constraints with different ratios of variance to squared mean.

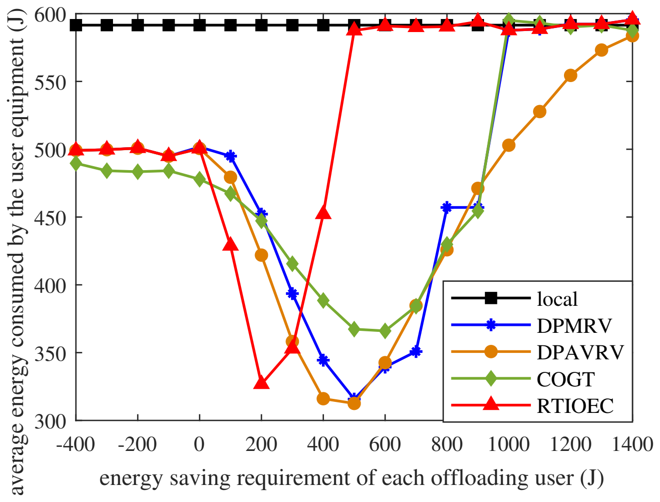


Fig. 5. The average energy consumed by user equipment with different energy saving requirements of each offloading user.

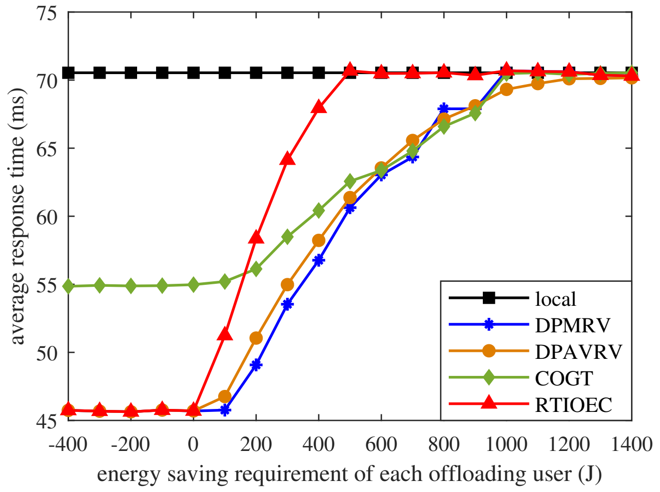
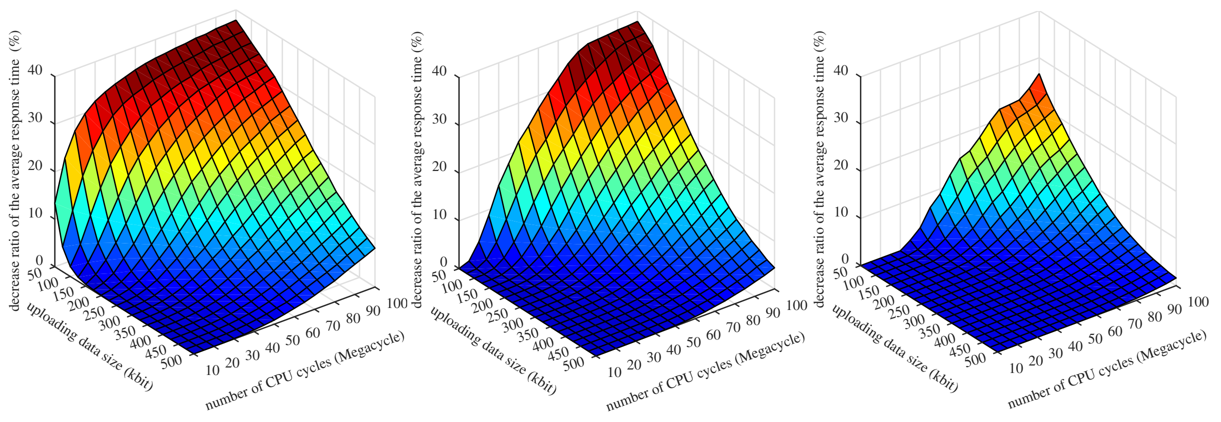


Fig. 6. The average response time with different energy saving requirements of each offloading user.

of each offloading user on average response time. We observe that the average response time by the four offloading schemes is significantly lower than that of local execution, especially when the energy saving requirement is low. For instance, the average response time by the RTIOEC algorithm is approximately 35% lower than that of local execution when the energy saving requirement is 0 J. Although the average response time by the RTIOEC algorithm is higher than that by the other three offloading schemes when the energy saving requirement is high, the RTIOEC algorithm has the following advantages. As discussed previously, the violation probability by the RTIOEC algorithm is far lower and can be specified, which is important in actual implementation. Meanwhile the violation probability by the DPMRV algorithm and the COGT algorithm is out of control, for which the two algorithms are not applicable to solve the problem under uncertainty. For the DPAVRV algorithm, it is the idealistic benchmark and not practical because it is not likely to obtain the actual value of the random variables when making offloading decision. The average response time increases at the middle stage with the energy saving requirement increasing. This is due to two reasons: The offloading users upload the input parameters with a smaller transmit power for satisfying more stringent energy consumption constraint with the specified risk level, which leads to longer transfer time. Additionally, few users can satisfy the energy consumption constraint and offload continuous applications with the energy saving requirement increasing. The average response time fluctuates slightly at the



(a) $\theta_i = -100$ J (b) $\theta_i = 100$ J (c) $\theta_i = 300$ J

Fig. 7. The decrease ratio of the average response time with different uploading and downloading data size and different number of CPU cycles for the computation of a user operation.

beginning and finally stabilizes around the result of executing all the continuous applications locally for the same reasons as that of the average energy consumed by user equipment shown in Fig. 5.

Fig. 7 shows the decrease ratio of the average response time with different uploading and downloading data size and different number of CPU cycles for the computation of a user operation. In the experiments, the downloading data size is set the same as that of uploading. As the description in Section III, the negative value of θ_i in the experiments means that the energy consumed by user equipment for offloading can be greater than that of local execution in the scenario that the response time is of much more significance than the energy consumption or the applications are extremely delay-sensitive. Because the average response time naturally increases as the number of CPU cycles required for execution increases, the decrease ratio of the average response time is introduced to be a unified metric for quantifying the efficiency of the RTIOEC algorithm. As shown in Fig. 7, the decrease ratio of the average response time increases as the number of CPU cycles increases and the uploading and downloading data size decreases. The reason is that the larger number of CPU cycles required to execute computation leads to more significant decrease of execution time due to the sufficient computation resources in EC. And the smaller uploading and downloading data size leads to shorter transfer time and lower energy consumed by user equipment. It is also observed that the decrease ratio of the average response time is high with low energy saving requirement. Because low energy saving requirement indicates that the offloading users can upload the input parameters with high transmit power to reduce transfer time, and more users may satisfy the energy consumption constraint and offload continuous applications to the edge computing node.

VI. Conclusion and Future Work

In this paper, we investigate the computation offloading decision in EC for continuous applications under uncertainty. Due to the uncertainty in user operations, the number of user operations, the uploading and downloading data size for offloading computation of each user operation, and the number of CPU cycles required to execute computation of each user operation are unknown when making offloading decision. An energy consumption constrained average response time minimization problem among multiple users is formulated in this scenario. To tackle the minimization problem, the RTIOEC algorithm is proposed to make offloading decision with fewer characteristics of applications. We carry out experiments to evaluate the performance of the RTIOEC algorithm and compare the RTIOEC algorithm with four offloading decision schemes. The results show that the average response time of continuous applications decreases significantly with the RTIOEC algorithm while satisfying the energy consumption constraint with the predefined upper bound of the violation probability. That means the violation probability of the energy consumption constraint under uncertainty is controllable and the energy saving requirement to prolong the battery lifetime of user equipment can be satisfied for most users. Therefore, the RTIOEC algorithm with the salient features is applicable under uncertainty, meeting the challenge of making offloading decision in EC for continuous applications.

By pushing sufficient computation resources towards the network edge, EC potentially improve the application performance significantly, e.g., delay and energy consumption of user equipment. Offloading continuous applications in EC is still an ongoing research and many challenges remain to be solved. For future work, we will look into the impact of user mobility, adopt dynamic bandwidth and computation resource management and take the energy consumed by edge computing server into consideration. Other extensions are to implement partial offloading and extend the work to the scenario with multiple edge computing nodes.

Appendix
Unimodality Proof of $h(P_{i,T})$

Considering the proposed scenario, all the parameters in $h(P_{i,T})$ are obviously positive in which we obtain the proof. For notation convenience, we define $r_i \triangleq \frac{g_{i,up}(d_0/d_i)^{\varphi}}{N_o W_{i,up}^{off}}$. The first derivative of $h\left(P_{i,T}\right)$ is given as

$$h^{'}\left(P_{i,T}\right)=\frac{k_i\left(1+r_iP_{i,T}\right)\ln\left(1+r_iP_{i,T}\right)-r_i\left(k_iP_{i,T}+P_{i,0}\right)}{\ln 2W_{i,up}^{off}\left(1+r_iP_{i,T}\right)\left[\log_2{}^{\left(1+r_iP_{i,T}\right)}\right]^2} \tag{28}$$

The proof of monotonicity is equivalent to prove the numerator of $h^{'}\left(P_{i,T}\right)$ is negative on interval $\left(0,p_{i,T}^0\right)$ and is positive on interval $\left(p_{i,T}^0,+\infty\right)$ because the denominator is positive on both intervals. We first denote the numerator of $h^{'}\left(P_{i,T}\right)$ by $Z\left(P_{i,T}\right)$. That is

$$Z\left(P_{i,T}\right)\triangleq k_i\left(1+r_iP_{i,T}\right)\ln\left(1+r_iP_{i,T}\right)-r_i\left(k_iP_{i,T}+P_{i,0}\right) \tag{29}$$

The first derivative of $Z\left(P_{i,T}\right)$ is given by $Z^{'}\left(P_{i,T}\right)=k_ir_i\ln\left(1+r_iP_{i,T}\right)$. Obviously, $Z^{'}\left(P_{i,T}\right)$ is positive on interval $\left(0,+\infty\right)$, which means that $Z\left(P_{i,T}\right)$ is strictly increasing on $\left(0,+\infty\right)$. Additionally, we have $Z\left(0\right)=-r_iP_{i,0}<0$ and $Z\left(+\infty\right)>0$. We denote the zero point of $Z^{'}\left(P_{i,T}\right)$ as $p_{i,T}^0$. $Z\left(P_{i,T}\right)$ is negative on internal $\left(0,p_{i,T}^0\right)$ and is positive on interval $\left(p_{i,T}^0,+\infty\right)$. Therefore, $h(P_{i,T})$ is strictly decreasing on interval $\left(0,p_{i,T}^0\right)$ and is strictly increasing on interval $\left(p_{i,T}^0,+\infty\right)$. That means $h(P_{i,T})$ is unimodal.

Acknowledgment

Chang and Shou's work is supported by National Natural Science Foundation of China (Grant No. 61471053), Beijing Laboratory of Advanced Information Networks and the 111 project (Grant No. B17007). Xiao and Lou's work was supported in part by US National Science Foundation under grants CNS-1800650, and by the Virginia Commonwealth Cyber Initiative (CCI).

References

[1] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.

[2] M. Othman, S. A. Madani, S. U. Khan *et al.*, "A survey of mobile cloud computing application models," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 393–413, 2013.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv:1701.01090*, 2017.

[5] "Mobile edge computing," Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_ paper_v%12018-09-14.pdf, ETSI White Paper, [Online].

[6] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2017.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] J. Xue and G. Shou, "How far can optical access networks support in multi-access edge computing for low delay?" in *Optical Fiber Communication Conference*. Optical Society of America, 2018, pp. Tu2G–4.

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

[10] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774–777, 2017.

[11] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[13] J. Liu, G. Shou, Q. Wang, Y. Liu, Y. Hu, and G. Zhigang, "Load-balanced service function chaining in edge computing over fiwi access networks for internet of things," in *unpublished*, 2019.

[14] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[15] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[16] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[17] 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.

[18] E. Bastug, M. Bennis, M. Médard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 110–117, 2017.

[19] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[20] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[21] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[22] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[23] S. Li, Y. Huang, C. Li, B. Jalaian, S. Russell, Y. T. Hou, W. Lou, and B. MacCall, "A real-time solution for underlay coexistence with channel uncertainty," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[24] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.

[25] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[26] I. Ketykó, L. Kecskés, C. Nemes, and L. Farkas, "Multi-user computation offloading as multiple knapsack problem for 5g mobile edge computing," in *2016 European Conference on Networks and Communications (EuCNC)*. IEEE, 2016, pp. 225–229.

[27] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *2017 European conference on networks and communications (EuCNC)*. IEEE, 2017, pp. 1–6.

[28] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *ieee internet of things journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[30] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1451–1455.

[31] 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, pp. 89–103, 2015.

[32] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, M. K. Khan, M. Ni, and G. Zhu, "Joint admission control and resource allocation in edge computing for internet of things," *IEEE Network*, vol. 32, no. 1, pp. 72–79, 2018.

[33] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on*

Cognitive Communications and Networking, vol. 3, no. 3, pp. 361–373, 2017.

[34] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[35] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[36] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[37] L. Yang, H. Zhang, X. Li, H. Ji, and V. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 6, pp. 2762–2773, 2018.

[38] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[39] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2-3, pp. 203–221, 1996.

[40] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "The energy/frequency convexity rule: Modeling and experimental validation on mobile devices," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2013, pp. 793–803.

[41] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[42] S. Li, Y. T. Hou, W. Lou, B. Jalaian, S. Russell, and B. MacCall, "Optimal power control with channel uncertainty in ad hoc networks," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*. IEEE, 2019, pp. 652–657.

[43] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, 2017.

[44] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[45] A. Prékopa, *Stochastic programming*. Springer Science & Business Media, 2013, vol. 324.

[46] S. Li, Y. Huang, C. Li, B. A. Jalaian, Y. T. Hou, and W. Lou, "Coping uncertainty in coexistence via exploitation of interference threshold violation," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 71–80.

[47] P. Gilmore and R. E. Gomory, "The theory and computation of knapsack functions," *Operations Research*, vol. 14, no. 6, pp. 1045–1074, 1966.

[48] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, 2004.

[49] T. Q. Quek, G. de La Roche, I. Güvenç, and M. Kountouris, *Small cell networks: Deployment, PHY techniques, and resource management*. Cambridge University Press, 2013.

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

[51] 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 *2012 IEEE Vehicular Technology Conference (VTC Fall)*. IEEE, 2012, pp. 1–5.

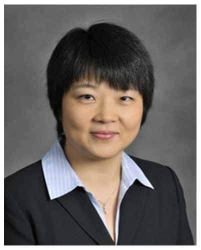
[52] A. Damnjanovic, J. Montojo, Y. Wei, T. Ji, T. Luo, M. Vajapeyam, T. Yoo, O. Song, and D. Malladi, "A survey on 3gpp heterogeneous networks," *IEEE Wireless communications*, vol. 18, no. 3, pp. 10–21, 2011.

[53] "Nokia airframe data center solution executive summary," Available: http://networks.nokia.com/sites/default/files/document lndcs3xecutive_su mmary_290515rm.pdf, Nokia, Tech. Rep., 2015 [Online].

[54] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

Wei Chang is currently pursuing the Ph.D. degree with School of Information and Communication Engineering at Beijing University of Posts and Telecommunications. Prior to beginning her Ph.D. studies, she received the B.E. degree in communication engineering from Tianjin University in 2014. She has been a Visiting Ph.D. Student with the Virginia Polytechnic Institute and State University since 2018. Her research interests are mainly edge computing, network planning and cyber security.

Yang Xiao is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering at Virginia Tech, supervised by Prof. Wenjing Lou. He received his B.S. degree from the School of Electrical and Information Engineering at Shanghai Jiao Tong University and M.S. degree from the Electrical Engineering and Computer Science Department at University of Michigan, Ann Arbor. His research interests lie in network security, blockchain, and IoT security.



Wenjing Lou is the W. C. English Endowed Professor of Computer Science at Virginia Tech and a Fellow of the IEEE. She holds a Ph.D. in Electrical and Computer Engineering from the University of Florida. Her research interests cover many topics in the cybersecurity field, with her current research interest focusing on wireless networks, privacy protection in machine learning systems, and security and privacy problems in the Internet of Things (IoT) systems. Prof. Lou is a highly cited researcher by the Web of Science Group. She received the Virginia Tech Alumni Award for Research Excellence in 2018, which is the highest university level faculty research award. She received the INFOCOM Test-of-Time paper award in 2020. She is the TPC chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She is the Steering Committee Chair of IEEE CNS conference series, steering committee member of IEEE INFOCOM conference and IEEE Transactions on Mobile Computing. She served as a program director at US National Science Foundation (NSF) from 2014 to 2017.

Guochu Shou is a professor with School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research interests include access network, edge computing, fiber and wireless network virtualization, network construction and routing, mobile internet and applications.