International Journal of Parallel, Emergent and Distributed Systems

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/gpaa20

Non-clairvoyant and randomised online task offloading in mobile edge computing

Keqin Li

INTERNATIONAL JOURNAL OF PARALLEL, EMERGENT AND DISTRIBUTED SYSTEMS
2022, VOL. 37, NO. 4, 413–424
https://doi.org/10.1080/17445760.2021.2024536

Non-clairvoyant and randomised online task offloading in mobile edge computing

Keqin Li

Department of Computer Science, State University of New York, New Paltz, NY, USA

ABSTRACT

In this paper, we consider non-clairvoyant task offloading for random tasks in mobile edge computing within the framework of combinatorial optimisation. For offline non-clairvoyant task offloading, we propose a non-clairvoyant task offloading algorithm, which is able to determine a task offloading strategy without knowing the amount of computation and communication of any task. For online non-clairvoyant task offloading, we propose a randomised online task offloading algorithm, which is able to make an offloading decision for an arrival task without knowing anything about future tasks and other tasks. For both algorithms, we analyse the probability of certain performance guarantee. We also demonstrate numerical data. To the best of the author's knowledge, this is the first paper which considers both offline and online non-clairvoyant task offloading in mobile edge computing, together with analytical results on performance guarantee with high probability.

ARTICLE HISTORY
Received 7 December 2021
Accepted 28 December 2021

KEYWORDS
Mobile edge computing; non-clairvoyant task offloading; randomised online task offloading

1. Introduction

1.1. Motivation

Task offloading (i.e. computation offloading) in mobile edge computing has been extensively investigated by many researchers in recent years (see [1–6] for comprehensive surveys). Task offloading has been studied as a combinatorial optimisation problem, i.e. to minimise the total execution time of a set of tasks [7]. In virtually all existing studies, for a single task to be offloaded, it is assumed the amount of computation and communication of the task is known in advance; for a collection of tasks to be offloaded, it is assumed that the collection of tasks are all available to a task offloading algorithm.

In real applications, we encounter the following two challenges. The first challenge is that the amount of computation and communication of a task may not be predictable, due to variable input/output data and uncertain execution paths. In such a situation, a task offloading algorithm needs to make an offloading decision without information of the amount of computation and communication of a task. This is similar to non-clairvoyant task scheduling in parallel and distributed systems [8]. The second challenge is that the information of a set of task may not be entirely available, since tasks may arrive dynamically at different times. In such a situation, a task offloading algorithm should make an offloading decision for an arrival task immediately without the knowledge of future tasks. This is similar to online non-clairvoyant task scheduling in parallel and distributed systems [9].

Unfortunately, there has been little study for non-clairvoyant task offloading in mobile edge computing, either online or offline. It is worth to mention that task offloading in the framework of Lyapunov

CONTACT Keqin Li lik@newpaltz.edu Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

optimisation [10] is not entirely online, since in each time slot, the information of all tasks in that time slot is known. Furthermore, the optimisation goal is not to minimise the total execution time of a list of tasks, but some performance measure averaged over all time slots. There are also online task offloading algorithms based on deep reinforcement learning, whose primary goal is to optimally adjust task offloading decisions according to various time-varying conditions [11]. All these studies are neither for non-clairvoyant task offloading, nor for combinatorial optimisation.

1.2. Contributions

In this paper, we consider non-clairvoyant task offloading for random tasks in mobile edge computing within the framework of combinatorial optimisation.

- Offline non-clairvoyant task offloading – We propose a non-clairvoyant task offloading algorithm, which is able to determine a task offloading strategy without knowing the amount of computation and communication of any task.
- Online non-clairvoyant task offloading – We propose a randomised online task offloading algorithm, which is able to make an offloading decision for an arrival task without knowing anything about future tasks and other tasks.

For both algorithms, we analyse the probability of certain performance guarantee. We also demonstrate numerical data. To the best of the author's knowledge, this is the first paper which considers both offline and online non-clairvoyant task offloading in mobile edge computing, together with analytical results on performance guarantee with high probability.

The rest of the paper is organised as follows. In Section 2, we present our task offloading model in mobile edge computing. In Section 3, we consider non-clairvoyant task offloading. In Section 4, we consider randomised online task offloading. In Section 5, we conclude the paper.

2. A task offloading model

In this section, we present our task offloading model in mobile edge computing.

Throughout the paper, $\boldsymbol{E}(X)$ is the expectation of a random variable X; $\mathrm{Var}(X)$ is the variance of a random variable X; $\boldsymbol{P}(\cdot)$ is the probability of an event.

We consider task offloading of one *user equipment* (UE) in a multiple *mobile edge cloud* (MEC) servers environment. There are M MEC servers: MEC_1, MEC_2,…, MEC_M. For convenience, the UE is treated as MEC_0. The UE has computation speed s_0, and MEC_j has computation speed s_j, measured by number of billion instructions (BI) per second, where $1 \le j \le M$. The communication speed between the UE and MEC_j is c_j, measured by number of million bits (MB) per second, where $1 \le j \le M$.

The UE has a set S of N random tasks: $S = \{\tau_1, \tau_2, \ldots, \tau_N\}$. Each task is $\tau_i = (r_i, d_i)$, where r_i is the amount of computation, measured by number of BI, and d_i is the amount of communication, measured by number of MB, for all $1 \le i \le N$. The r_i's are independent and identically distributed (i.i.d.) random variables with mean $\boldsymbol{E}(r_i) = \mu_r$ and variance $\mathrm{Var}(r_i) = \sigma_r^2$. The d_i's are i.i.d. random variables with mean $\boldsymbol{E}(d_i) = \mu_d$ and variance $\mathrm{Var}(d_i) = \sigma_d^2$.

A *task offloading strategy* is to divide the set S of tasks into $M+1$ disjoint subsets: $S_0, S_1, \ldots, S_M$, where $S_0 \cup S_1 \cup \cdots \cup S_M = S$. Tasks in S_0 are executed on the UE, and tasks in S_j are executed on MEC_j, for all $1 \le j \le M$. Let a subset be $S_j = \{\tau_{j,1}, \tau_{j,2}, \ldots, \tau_{j,N_j}\}$, with $N_j = |S_j|$, for all $0 \le j \le M$.

A task $\tau_{0,k}$, where $1 \le k \le N_0$, which is executed on the UE, has random execution time $t_{0,k} = r_{0,k}/s_0$, measured by seconds. The $t_{0,k}$'s are i.i.d. random variables with mean $\boldsymbol{E}(t_{0,k}) = \mu_r/s_0$ and variance $\mathrm{Var}(t_{0,k}) = \sigma_r^2/s_0^2$.

A task $\tau_{j,k}$, where $1 \le k \le N_j$, $1 \le j \le M$, which is executed on the MEC_j, has random execution time $t_{j,k} = r_{j,k}/s_j + d_{j,k}/c_j$, measured by seconds. The $t_{j,k}$'s are i.i.d. random variables with mean $\boldsymbol{E}(t_{j,k}) = \mu_r/s_j + \mu_d/c_j$ and variance $\mathrm{Var}(t_{j,k}) = \sigma_r^2/s_j^2 + \sigma_d^2/c_j^2$.

MEC_j has some existing workload W_j, measured by seconds, with mean $\boldsymbol{E}(W_j)$ and variance $\mathrm{Var}(W_j)$, for all $1 \le j \le M$.

3. Non-clairvoyant task offloading

In this section, we consider non-clairvoyant task offloading.

3.1. Problem

In *non-clairvoyant task offloading*, the information of the r_i's and the d_i's are not available. A task offloading strategy, i.e. $S_0, S_1, \ldots, S_M$, is determined without any knowledge of the r_i's, the d_i's, and the W_j's, except some of their statistical data and properties such as μ_r, μ_d, and $\boldsymbol{E}(W_j)$. Given a set S of N tasks, the problem is to find $S_0, S_1, \ldots, S_M$, such that the total execution time of the N tasks is minimised.

3.2. Algorithm

Our non-clairvoyant task offloading algorithm works as follows. The algorithm simply divides S into $M+1$ disjoint subsets $S_0, S_1, \ldots, S_M$, with $|S_j| = N_j$, for all $0 \le j \le M$. The values of the N_j's are determined below.

The total execution time of all tasks in S_0 is a random variable

$$T_0 = t_{0,1} + t_{0,2} + \cdots + t_{0,N_0},$$

with mean

$$\boldsymbol{E}(T_0) = N_0 \left(\frac{\mu_r}{s_0} \right),$$

and variance

$$\mathrm{Var}(T_0) = N_0 \left(\frac{\sigma_r}{s_0} \right)^2.$$

The total execution time of all tasks in S_j is a random variable

$$T_j = t_{j,1} + t_{j,2} + \cdots + t_{j,N_j} + W_j,$$

with mean

$$\boldsymbol{E}(T_j) = N_j \left(\frac{\mu_r}{s_j} + \frac{\mu_d}{c_j} \right) + \boldsymbol{E}(W_j),$$

and variance

$$\mathrm{Var}(T_j) = N_j \left(\frac{\sigma_r^2}{s_j^2} + \frac{\sigma_d^2}{c_j^2} \right) + \mathrm{Var}(W_j),$$

for all $1 \le j \le M$.

The N_j's are determined in such a way that the UE and all the MEC's complete their workload in about the same time, i.e.

$$\boldsymbol{E}(T_0) = \boldsymbol{E}(T_1) = \cdots = \boldsymbol{E}(T_M) = \tilde{T},$$

that is,

$$N_0\left(\frac{\mu_r}{s_0}\right) = N_j\left(\frac{\mu_r}{s_j} + \frac{\mu_d}{c_j}\right) + \boldsymbol{E}(W_j) = \tilde{T},$$

for all $1 \le j \le M$. The above equality implies that

$$N_0 = \frac{\tilde{T}}{\mu_r/s_0},$$

and

$$N_j = \frac{\tilde{T} - \boldsymbol{E}(W_j)}{\mu_r/s_j + \mu_d/c_j},$$

for all $1 \le j \le M$. Since

$$N_0 + \sum_{j=1}^{M} N_j = N,$$

that is,

$$\frac{\tilde{T}}{\mu_r/s_0} + \sum_{j=1}^{M} \frac{\tilde{T} - \boldsymbol{E}(W_j)}{\mu_r/s_j + \mu_d/c_j} = N,$$

we get

$$\tilde{T}\left(\frac{1}{\mu_r/s_0} + \sum_{j=1}^{M} \frac{1}{\mu_r/s_j + \mu_d/c_j}\right) = N + \sum_{j=1}^{M} \frac{\boldsymbol{E}(W_j)}{\mu_r/s_j + \mu_d/c_j},$$

which gives

$$\tilde{T} = \left(N + \sum_{j=1}^{M} \frac{\boldsymbol{E}(W_j)}{\mu_r/s_j + \mu_d/c_j}\right)\left(\frac{1}{\mu_r/s_0} + \sum_{j=1}^{M} \frac{1}{\mu_r/s_j + \mu_d/c_j}\right)^{-1},$$

and

$$N_0 = \frac{1/(\mu_r/s_0)}{1/(\mu_r/s_0) + \sum_{j=1}^{M} 1/(\mu_r/s_j + \mu_d/c_j)}\left(N + \sum_{j=1}^{M} \frac{\boldsymbol{E}(W_j)}{\mu_r/s_j + \mu_d/c_j}\right),$$

and

$$N_j = \frac{1/(\mu_r/s_j + \mu_d/c_j)}{1/(\mu_r/s_0) + \sum_{j=1}^{M} 1/(\mu_r/s_j + \mu_d/c_j)}\left(N - \frac{\boldsymbol{E}(W_j)}{\mu_r/s_0}\right),$$

for all $1 \le j \le M$.

Remark 3.1: We assume that

$$N > \frac{\boldsymbol{E}(W_j)}{\mu_r/s_0},$$

that is,

$$\boldsymbol{E}(W_j) < N(\mu_r/s_0),$$

for all $1 \le j \le M$. If

$$\boldsymbol{E}(W_j) \ge N(\mu_r/s_0),$$

for some $1 \le j \le M$, we simply exclude MEC_j for task offloading, since the existing workload W_j on MEC_j is too heavy.

Our *Non-clairvoyant Task Offloading Algorithm* (NCTOA) is formally presented in Algorithm 1.

Algorithm 1 Non-clairvoyant Task Offloading Algorithm (NCTOA).

Input: A set S of N random tasks: $S = \{\tau_1, \tau_2, \ldots, \tau_N\}$.
Output: A partition of S into $M+1$ disjoint subsets: $S_0, S_1, \ldots, S_M$, where $S_0 \cup S_1 \cup \cdots \cup S_M = S$.

for $(j = 0; j \le M; j{+}{+})$ **do** (1)
  $S_j \leftarrow$ any subset of S with N_j tasks; (2)
  $S \leftarrow S - S_j$; (3)
end do (4)

3.3. Analysis

The total execution time of all the N tasks in S is $T = \max\{T_0, T_1, \ldots, T_M\}$. Since $\tilde{T}$ is considered as the best achievable time for completing the N tasks, we compare T with $\tilde{T}$.

The following theorem gives a performance guarantee for non-clairvoyant task offloading.

Theorem 3.1: *Our non-clairvoyant task offloading algorithm guarantees*

$$\boldsymbol{P}\left(T \le \tilde{T} + bB\right) \ge \left(1 - \frac{1}{b^2}\right)^{M+1},$$

for all $b > 0$, where

$$B = \max\left\{\sqrt{N_0}\left(\frac{\sigma_r}{s_0}\right), \max_{1\le j\le M}\left\{\sqrt{N_j\left(\frac{\sigma_r^2}{s_j^2} + \frac{\sigma_d^2}{c_j^2}\right) + \text{Var}(W_j)}\right\}\right\}.$$

Proof: Chebyshev's inequality [12, p. 389] states that for any random variable X, we have

$$\boldsymbol{P}\left(|X - \boldsymbol{E}(X)| \geq b\sqrt{\mathrm{Var}(X)}\right) \leq \frac{1}{b^2},$$

and equivalently,

$$\boldsymbol{P}\left(|X - \boldsymbol{E}(X)| \leq b\sqrt{\mathrm{Var}(X)}\right) \geq 1 - \frac{1}{b^2},$$

for all $b > 0$. By applying Chebyshev's inequality to T_j, we obtain

$$\boldsymbol{P}\left(|T_j - \tilde{T}| \leq b\sqrt{\mathrm{Var}(T_j)}\right) \geq 1 - \frac{1}{b^2},$$

for all $0 \leq j \leq M$. Furthermore, we can have

$$\boldsymbol{P}\left(T_j \leq \tilde{T} + b\sqrt{\mathrm{Var}(T_j)}\right) \geq \boldsymbol{P}\left(|T_j - \tilde{T}| \leq b\sqrt{\mathrm{Var}(T_j)}\right) \geq 1 - \frac{1}{b^2},$$

for all $0 \leq j \leq M$. Let

$$B = \max_{0\leq j\leq M}\left\{\sqrt{\mathrm{Var}(T_j)}\right\},$$

which is actually the value given in the theorem. Then, we get

$$\boldsymbol{P}\left(T_j \leq \tilde{T} + bB\right) \geq \boldsymbol{P}\left(T_j \leq \tilde{T} + b\sqrt{\mathrm{Var}(T_j)}\right) \geq 1 - \frac{1}{b^2},$$

for all $0 \leq j \leq M$. Consequently, we obtain

$$\boldsymbol{P}\left(T \leq \tilde{T} + bB\right) = \prod_{j=0}^{M}\boldsymbol{P}\left(T_j \leq \tilde{T} + bB\right) \geq \left(1 - \frac{1}{b^2}\right)^{M+1},$$

where we notice that the T_j's are independent random variables. ■

More accurate performance analysis is possible for specific probability distributions. For instance, let us assume that the r_i's, the d_i's, and the W_j's are all normal random variables. Note that we implicitly assume that the distribution in the range $(-\infty, 0)$ is extremely small and negligible. Let X be a standard normal random variable with mean 0 and variance 1, whose probability density function is

$$f_X(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2},$$

and whose cumulative distribution function is

$$F_X(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-y^2/2}\,\mathrm{d}y.$$

The following theorem gives the probability $\boldsymbol{P}(T \leq (1+\gamma)\tilde{T})$ for normal random variables.

Theorem 3.2: *If the r_i's, the d_i's, and the W_j's are all normal random variables, we have*

$$\boldsymbol{P}\left(T \leq (1+\gamma)\tilde{T}\right) = \Phi\left(\frac{\gamma\tilde{T}}{\sqrt{N_0}(\sigma_r/s_0)}\right)\prod_{j=1}^{M}\Phi\left(\frac{\gamma\tilde{T}}{\sqrt{N_j(\sigma_r^2/s_j^2 + \sigma_d^2/c_j^2) + \mathrm{Var}(W_j)}}\right),$$

where $\gamma > 0$.

Proof: It is clear that for all $0 \le j \le M$, T_j is a linear combination of independent normal random variables, which is still a normal random variable with mean $\boldsymbol{E}(T_j)$ and variance $\text{Var}(T_j)$ [12, pp. 213, 357]. It is well known that

$$X = \frac{T_j - \boldsymbol{E}(T_j)}{\sqrt{\text{Var}(T_j)}}$$

is a standard normal random variable, that is,

$$\boldsymbol{P}(T_j \le \tau) = \boldsymbol{P}\left(X \le \frac{\tau - \boldsymbol{E}(T_j)}{\sqrt{\text{Var}(T_j)}}\right) = \Phi\left(\frac{\tau - \boldsymbol{E}(T_j)}{\sqrt{\text{Var}(T_j)}}\right) = \Phi\left(\frac{\tau - \tilde{T}}{\sqrt{\text{Var}(T_j)}}\right).$$

Therefore, we obtain

$$\boldsymbol{P}\left(T \le (1+\gamma)\tilde{T}\right) = \prod_{j=0}^{M} \boldsymbol{P}\left(T_j \le (1+\gamma)\tilde{T}\right) = \prod_{j=0}^{M} \Phi\left(\frac{\gamma\tilde{T}}{\sqrt{\text{Var}(T_j)}}\right),$$

where $\gamma > 0$. The theorem is proved by substituting $\text{Var}(T_j)$ into the last equation for all $0 \le j \le M$. ■

3.4. Numerical data

We now demonstrate some numerical data.

We use the following parameter setting. We consider a mobile computing environment with $M = 5$ MECs. The computation speed of the UE is $s_0 = 2.0$ BI/s. The computation speed of MEC_j is $s_j = 3.0 + 0.1(j-1)$ BI/s, for all $1 \le j \le M$. The communication speed of MEC_j is $c_j = 5.0 + 0.2(j-1)$ MB/s, for all $1 \le j \le M$. The random tasks have the following parameters: $\mu_r = 1.5$ BI, $\sigma_r = 0.3$ BI, $\mu_d = 2.5$ MB, and $\sigma_d = 0.4$ MB. The existing workload on MEC_j has mean $\boldsymbol{E}(W_j) = 2.0 + 0.2(j-1)$ s and variance $\text{Var}(W_j) = (0.3 + 0.05(j-1)\text{ s})^2$, for all $1 \le j \le M$.

In Figure 1, we show the probability $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ (actually, the lower bound given in Theorem 3.1) vs. N, for $\gamma = 0.2, 0.4, 0.6, 0.8, 1.0$. In doing so, we set $b = \gamma\tilde{T}/B$, so that $T \le \tilde{T} + bB$ is equivalent to $T \le (1+\gamma)\tilde{T}$. It is easily observed that $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ increases as γ and N increase. Even for reasonable values of γ and N, $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ is already very high. For instance, when $N = 100$, $\boldsymbol{P}(T \le 2\tilde{T})$ is 0.98937.

In Figure 2, we show the probability $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ given in Theorem 3.2 vs. N, for $\gamma = 0.05, 0.10, 0.15, 0.20, 0.25$. It is observed that $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ is already very high even for small values of γ. For instances, when $\gamma = 0.25$, $\boldsymbol{P}(T \le 1.25\tilde{T})$ is 0.88493 for $N = 10$, and is almost 1 for $N \ge 30$.

4. Randomised online task offloading

In this section, we consider randomised online task offloading.

4.1. Problem

In *online task offloading*, the N tasks are given as a list $L = (\tau_1, \tau_2, \ldots, \tau_N)$. Tasks are not all available in the begin, and may arrive dynamically. A task offloading strategy, i.e. a partition of L into $M+1$ disjoint sublists $L_0, L_1, \ldots, L_M$, is determined without the information of the entire list. An arrival task should be assigned to the UE or offloaded to an MEC immediately, without the knowledge of future tasks. Tasks must be considered in the given order (i.e. online task offloading). Furthermore, the information of the r_i's and the d_i's are not available (i.e. non-clairvoyant task offloading). Given a list L of N tasks, the problem is to find $L_0, L_1, \ldots, L_M$, such that the total execution time of the N tasks is minimised. Note that not only there is no information of future arrival tasks, but also an online task offloading algorithm does not know the value of N, yet still needs to minimise the total execution time of all the N tasks.

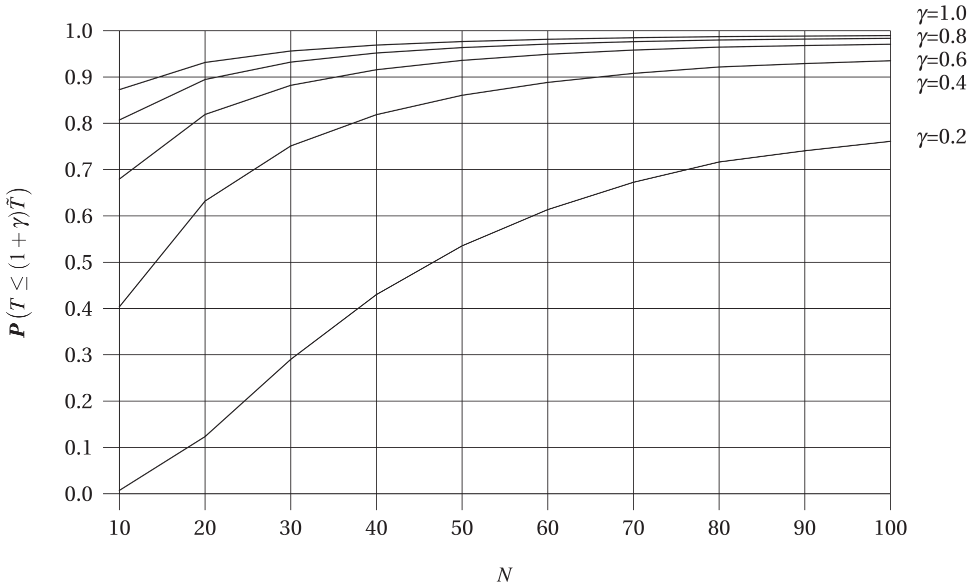

Figure 1. $\boldsymbol{P}(T \leq (1+\gamma)\tilde{T})$ in Theorem 3.1 vs. N.

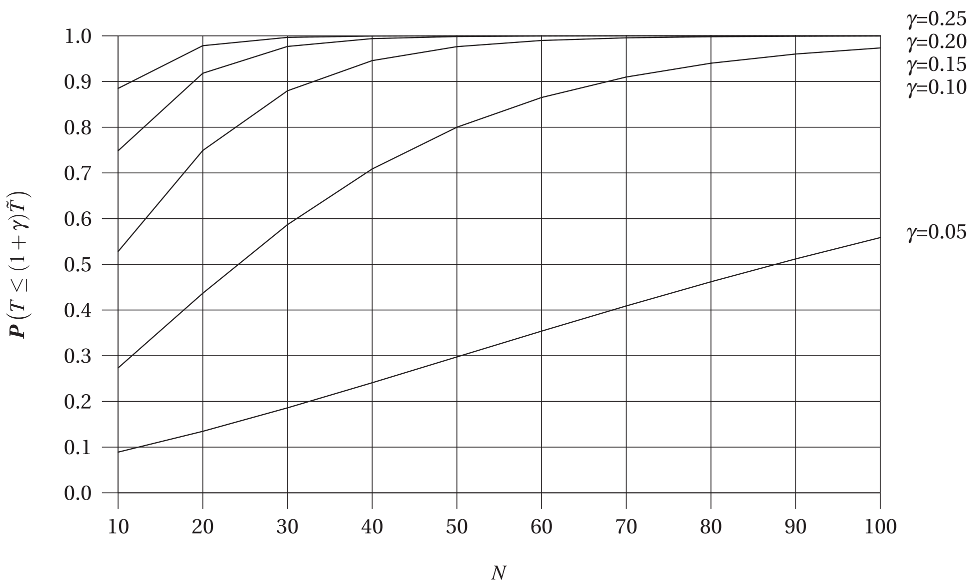

Figure 2. $\boldsymbol{P}(T \leq (1+\gamma)\tilde{T})$ in Theorem 3.2 vs. N.

4.2. Algorithm

Our randomised online task offloading algorithm works as follows. For each task τ_i, the tasks is assigned to MEC$_j$ (i.e. put into L_j) with probability $p_j = N_j/N$ (i.e. randomised task offloading), where N_j is given in Section 3, for all $0 \leq j \leq M$.

It is noticed that the p_j's depend on N. However, if $N \to \infty$, we have $p_j \to p'_j$, where

$$p'_0 = \frac{1/(\mu_r/s_0)}{1/(\mu_r/s_0) + \sum_{j=1}^{M} 1/(\mu_r/s_j + \mu_d/c_j)},$$

and

$$p'_j = \frac{1/(\mu_r/s_j + \mu_d/c_j)}{1/(\mu_r/s_0) + \sum_{j=1}^{M} 1/(\mu_r/s_j + \mu_d/c_j)},$$

for all $1 \le j \le M$. In real applications, when N is large, we can use p'_j as an approximation of p_j, which is independent of N.

Our *Randomised Online Task Offloading Algorithm* (ROTOA) is formally presented in Algorithm 2.

Algorithm 2 Randomised Online Task Offloading Algorithm (ROTOA).

Input: A list L of N random tasks: $L = (\tau_1, \tau_2, \ldots, \tau_N)$.
Output: A partition of L into $M+1$ disjoint sublists: $L_0, L_1, \ldots, L_M$.

for ($i = 1; i \le N; i$++) **do** (1)
  pick an index j randomly from $\{0, 1, 2, \ldots, M\}$, where j is selected with probability p_j; (2)
  append τ_i to L_j; (3)
  remove τ_i from L; (4)
end do (5)

Let N'_j be the number of tasks assigned to MEC$_j$, which is a random variable. Clearly, $N'_j = y_1 + y_2 + \cdots + y_N$, where the y_i's are i.i.d. Bernoulli random variables with probability p_j and $q_j = 1 - p_j$. It is well known that N'_j is a binomial random variable with mean $\boldsymbol{E}(N'_j) = Np_j = N_j$ and variance $\mathrm{Var}(N'_j) = Np_jq_j = N_jq_j$ ([12, p. 146]).

4.3. Analysis

The main difficulty in analysing the randomised online task offloading algorithm is that the N'_j's (and therefore, the T_j's) are not independent of each other. It is easy to see that the N'_j's follow a multinomial distribution, with a probability mass function as

$$\boldsymbol{P}(N'_0 = k_0, N'_1 = k_1, \ldots, N'_M = k_M) = \binom{N}{k_0, k_1, \ldots, k_M} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M},$$

which is

$$\boldsymbol{P}(N'_0 = k_0, N'_1 = k_1, \ldots, N'_M = k_M) = \frac{N!}{k_0!k_1!\cdots k_M!} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M},$$

where $k_0 + k_1 + \cdots + k_M = N$, and $p_0 + p_1 + \cdots + p_M = 1$.

Theorem 4.1: *For our randomised online task offloading algorithm, we have*

$$\boldsymbol{P}(T \le \tau) = \sum_{k_0+k_1+\cdots+k_M=N} \frac{N!}{k_0!k_1!\cdots k_M!} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M} \prod_{j=0}^{M} \boldsymbol{P}\left(T_j \le \tau | N'_j = k_j\right),$$

for all $\tau > \tilde{T}$.

Proof: Note that for fixed $k_0, k_1, \ldots, k_M$, the T_j's are independent random variables. This means that

$$\boldsymbol{P}(T \le \tau | N'_0 = k_0, N'_1 = k_1, \ldots, N'_M = k_M) = \prod_{j=0}^{M} \boldsymbol{P}\left(T_j \le \tau | N'_j = k_j\right).$$

Hence, we have

$$\begin{aligned}&\boldsymbol{P}(T \le \tau)\\&= \sum_{k_0+k_1+\cdots+k_M=N} \boldsymbol{P}(N'_0 = k_0, N'_1 = k_1, \ldots, N'_M = k_M)\boldsymbol{P}(T \le \tau | N'_0 = k_0, N'_1 = k_1, \ldots, N'_M = k_M),\end{aligned}$$

which is

$$\boldsymbol{P}(T \le \tau) = \sum_{k_0+k_1+\cdots+k_M=N} \binom{N}{k_0, k_1, \ldots, k_M} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M} \prod_{j=0}^{M} \boldsymbol{P}\left(T_j \le \tau | N'_j = k_j\right).$$

The last equation is essentially the theorem. ■

It is hard to find $\boldsymbol{P}(T_j \le \tau | N'_j = k_j)$. Fortunately, for normal random variables, we are able to calculate the probability $\boldsymbol{P}(T \le \tau)$ for our randomised online task offloading algorithm, as shown in the following theorem.

Theorem 4.2: *If the r_i's, the d_i's, and the W_j's are all normal random variables, we have*

$$\begin{aligned}&\boldsymbol{P}(T \le \tau)\\&= \sum_{k_0+k_1+\cdots+k_M=N} \frac{N!}{k_0! k_1! \cdots k_M!} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M}\\&\quad \times \Phi\left(\frac{\tau - k_0(\mu_r/s_0)}{\sqrt{k_0}(\sigma_r/s_0)}\right) \prod_{j=1}^{M} \Phi\left(\frac{\tau - (k_j(\mu_r/s_j + \mu_d/c_j) + \boldsymbol{E}(W_j))}{\sqrt{k_j(\sigma_r^2/s_j^2 + \sigma_d^2/c_j^2) + \mathrm{Var}(W_j)}}\right),\end{aligned}$$

for all $\tau > \tilde{T}$.

Proof: Under the condition that $N'_0 = k_0$, the total execution time of all tasks in S_0 is a random variable

$$T_0 = t_{0,1} + t_{0,2} + \cdots + t_{0,k_0},$$

with mean

$$\boldsymbol{E}(T_0 | N'_0 = k_0) = k_0 \left(\frac{\mu_r}{s_0}\right),$$

and variance

$$\mathrm{Var}(T_0 | N'_0 = k_0) = k_0 \left(\frac{\sigma_r}{s_0}\right)^2.$$

Similarly, under the condition that $N'_j = k_j$, the total execution time of all tasks in S_j is a random variable

$$T_j = t_{j,1} + t_{j,2} + \cdots + t_{j,k_j} + W_j,$$

with mean

$$\boldsymbol{E}(T_j | N'_j = k_j) = k_j \left(\frac{\mu_r}{s_j} + \frac{\mu_d}{c_j}\right) + \boldsymbol{E}(W_j),$$

and variance

$$\mathrm{Var}(T_j|N_j' = k_j) = k_j\left(\frac{\sigma_r^2}{s_j^2} + \frac{\sigma_d^2}{c_j^2}\right) + \mathrm{Var}(W_j),$$

for all $1 \le j \le M$. Furthermore, since T_j is a normal random variable, we have

$$\boldsymbol{P}(T_j \le \tau|N_j' = k_j) = \Phi\left(\frac{\tau - \boldsymbol{E}(T_j|N_j' = k_j)}{\sqrt{\mathrm{Var}(T_j|N_j' = k_j)}}\right),$$

for all $0 \le j \le M$. By Theorem 4.1, we get

$$\boldsymbol{P}(T \le \tau) = \sum_{k_0+k_1+\cdots+k_M=N} \frac{N!}{k_0!k_1!\cdots k_M!} p_0^{k_0} p_1^{k_1} \cdots p_M^{k_M} \prod_{j=0}^{M} \Phi\left(\frac{\tau - \boldsymbol{E}(T_j|N_j' = k_j)}{\sqrt{\mathrm{Var}(T_j|N_j' = k_j)}}\right).$$

The theorem is proved by substituting $\boldsymbol{E}(T_j|N_j' = k_j)$ and $\mathrm{Var}(T_j|N_j' = k_j)$ into the last equation for all $0 \le j \le M$. ■

4.4. Numerical data

We now demonstrate some numerical data.

We use the same parameter setting as Subsection 3.4.

In Figure 3, we show the probability $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ given in Theorem 4.2 vs. N, for $\gamma = 0.30, 0.35, 0.40, 0.45, 0.50$. It is observed that $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ is noticeably lower than that in Theorem 3.2 due to increased randomness in the randomised online task offloading algorithm. However, for reasonable values of γ and N, $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ is reasonably high. For instance, when $\gamma = 0.5$ and $N = 100$, $\boldsymbol{P}(T \le 1.5\tilde{T})$ is 0.90126.

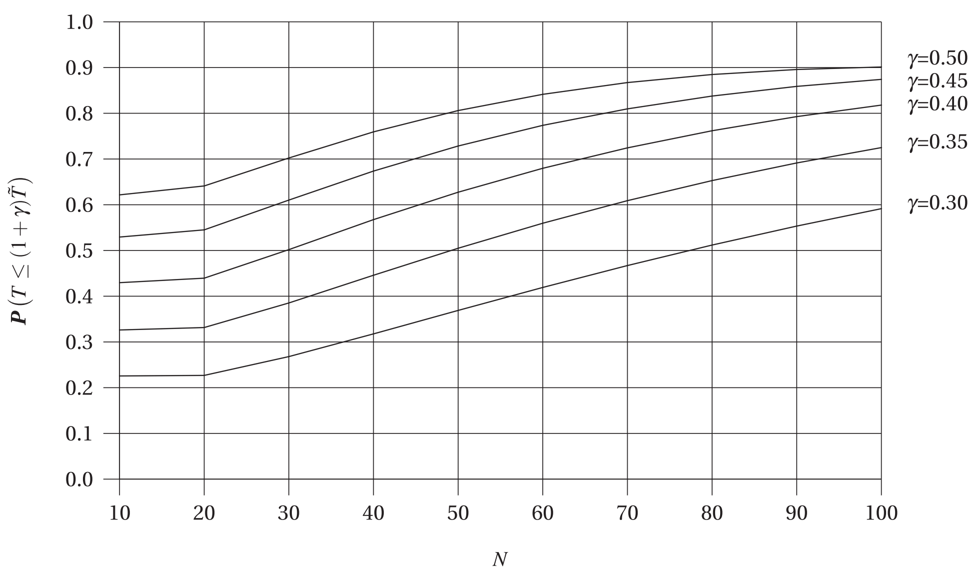

Figure 3. $\boldsymbol{P}(T \le (1+\gamma)\tilde{T})$ in Theorem 4.2 vs. N.

5. Conclusions

We have considered both offline and online non-clairvoyant task offloading for random tasks in mobile edge computing. We have proposed algorithms for both types of task offloading and analysed their performance by providing high probability of performance guarantee. These algorithms should be very useful in real mobile edge computing applications, where there is little information about task computation and communication requirement and/or there is need for immediate task offloading without waiting for further tasks.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] Bhattacharya A, De P. A survey of adaptation techniques in computation offloading. J Netw Comput Appl. 2017;78:97–115.
[2] Khan MA. A survey of computation offloading strategies for performance improvement of applications running on mobile devices. J Netw Comput Appl. 2015;56:28–40.
[3] Kumar K, Liu J, Lu Y-H, et al. A survey of computation offloading for mobile systems. Mobile Netw Appl. 2013;18(1):129–140.
[4] Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun Surveys & Tutorials. 2017;19(3):1628–1656.
[5] Shan X, Zhi H, Li P, et al. A survey on computation offloading for mobile edge computing information. IEEE BigDataSecurity/HPSC/IDS; Omaha, NE, USA: 3–5 May 2018. p. 248–251.
[6] Shiraz M, Sookhak M, Gani A, et al. A study on the critical analysis of computational offloading frameworks for mobile cloud computing. J Netw Comput Appl. 2015;47:47–60.
[7] Li K. Heuristic computation offloading algorithms for mobile users in fog computing. ACM Trans Embedded Comput Syst. 2021;20(2):1–28.
[8] Li K. An average-case analysis of online non-clairvoyant scheduling of independent parallel tasks. J Parallel Distrib Comput. 2006;66(5):617–625.
[9] Li K. Non-clairvoyant scheduling of independent parallel tasks on single and multiple multicore processors. J Parallel Distrib Comput. 2019;133:210–220.
[10] Ding Y, Li K, Liu C, et al. Short- and long-term cost and performance optimization for mobile user equipments. J Parallel Distrib Comput. 2021;150:69–84.
[11] Huang L, Bi S, Zhang Y-JA. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Trans Mobile Comput. 2020;19(11):2581–2593.
[12] Ross S. A first course in probability. 4th ed. New York: Macmillan College Publishing Company; 1994.