Al-Habob, A. A., Ibrahim, A., Dobre, O. A. & Armada, A. G. (2020). Collision-Free Sequential Task Offloading for Mobile Edge Computing. *IEEE Communications Letters*, 24(1), pp. 71–75.

DOI: 10.1109/lcomm.2019.2948179

Collision-free Sequential Task Offloading for Mobile Edge Computing

Author 1, Author 2, Author 3, and Author 4

***Abstract*—In this paper, a collision-free sequential task offloading scheme to multiple mobile-edge computing servers is proposed. The problem is formulated as a multi-objective optimization of latency and offloading failure probability. An exact solution technique is developed to obtain a benchmark optimal solution for the problem. A more computationally efficient heuristic algorithm is additionally developed, whose sub-optimal solution yields a performance close to optimal. Simulation results illustrate that the proposed offloading scheme can effectively reduce both latency and offloading failure probability.**

***Index Terms*—Mobile edge computing, sequential task offloading, ultra-reliable low-latency task offloading.**

I. Introduction

In the last decade, an exponential growth of mobile applications has been witnessed, leading to the need of running computational-intensive tasks on mobile devices. Although recent mobile devices are equipped with more powerful processors, even these may not be able to handle tasks requiring intensive computations in a short time. Moreover, mobile devices are constrained by capacity-limited battery, and task computing consumes a large portion of the battery power. This paradigm limits the devices' ability to run applications that require intensive processing.

Mobile-edge computing (MEC) is considered as a promising solution that provides cloud-like computing and avoids offloading the tasks to distant centralized servers [1]–[3]. In MEC, to take advantage of the servers diversity on the network edges, the task can be offloaded to multiple MEC servers cooperatively to further reduce the latency [4], [5]. From the mobile device point of view, the important requirements of task offloading are low energy consumption, low offloading error, and low latency, which mandates ultra-reliable and low-latency task offloading. Two main offloading schemes can be considered, namely parallel offloading and sequential offloading. For the former, the sub-tasks are offloaded simultaneously to the servers over orthogonal communication channels [4]. For the latter, the sub-tasks are offloaded in a time sequential manner to servers over a shared communication channel. A sequential task offloading framework was proposed in [6]. In this framework, a user equipment (UE) segments a task into sub-tasks and offloads them to multiple servers in sequence; the failure probabilities and latency caused by uplink transmission are studied.

This paper tackles a more realistic scenario of sequential task offloading, in which the feedback from the servers to the UE is considered. Including the feedback into the sequential task offloading framework imposes the possibility of transmission collision between the feedback of a given server and the uplink/downlink transmission to/of another server. The problem is formulated as a weighted-sum multi-objective optimization problem of latency and offloading failure probability. When compared with the product used in [6], this formulation enables us to develop an exact technique that guarantees an optimal solution and to control the importance (weight) of one objective over the other. Moreover, even if an exact scheme is hypothetically developed for the product cost function, it would not guarantee the optimality if the weight values are changed. This is mainly because the weights would come as powers for the product terms rather than coefficients as in the weighted sum. A more computationally-efficient heuristic solution is developed for the formulated optimization problem.

The rest of this paper is organized as follows. Section II illustrates the system and communication models. Section III formulates the optimization problem. An exact method for obtaining a benchmark optimal solution is proposed in Section IV. Section V develops a heuristic approach for obtaining good quality sub-optimal solutions with low computational effort. Section VI presents simulation results, and Section VII concludes the paper.

II. System and Communication Models

A. System Model

A delay-sensitive and computationally-intensive task ($\mathcal{T}$) is offloaded by an UE to a set $\mathcal{S} = \{s_i\}_{i=1}^{N}$ of N MEC servers. Each server is equipped with a central processing unit (CPU) that helps offload and compute the UE's task. We use a tuple $\mathcal{T} = \{U, D, C\}$ to represent the task $\mathcal{T}$ in which U is the size of the input data (in bits), D is the output computed result (in bits), and C is the required CPU cycles. The output computed result is modeled as $D = \beta U$, where β ($\beta > 0$) is the output to input ratio of the task [5]. The number of CPU cycles C is modeled as $C = \alpha U$, where α ($\alpha > 0$) depends on the task's computational complexity [2]. A server s_i is represented by a tuple $s_i = \{Ru_i, Rd_i, f_i\}$, where Ru_i is the uplink data rate, Rd_i is the downlink data rate, and f_i is the computational speed of the CPU in the i-th server.

To select the servers' offloading sequence and obtain an ordered set of servers $\mathcal{S}^*$, we assign a weight w_i to each server and sort the servers in an ascending order of w_i. We study a weighting scheme which reflects the computational capabilities and quality of transmission links:

$$w_i = \frac{U}{Ru_i} + \frac{\alpha U}{f_i} + \frac{\beta U}{Rd_i}. \tag{1}$$

The task $\mathcal{T}$ can be divided into M ($M \leq N$) non-overlapping sub-tasks and distributed to servers. We define the task allocation vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_N]^{\mathsf{T}}$, such that

$\sum_{i=1}^{N} \eta_i = 1$ and $\eta_i \geq 0$ is the portion of the task that is offloaded to the i-th server. Task partitioning causes overhead, and consequently, we introduce δ ($\delta \geq 1$) to represent the ratio of the transmitted data size to the original task data size due to segmentation overhead. We use a tuple $\tau_i = \{u_i(\boldsymbol{\eta}), c_i(\boldsymbol{\eta}), d_i(\boldsymbol{\eta})\}$ to represent the sub-task that is offloaded to the i-th server in which $u_i(\boldsymbol{\eta}) = \eta_i \delta U$ is the size of the sub-task's input data (in bits), $c_i(\boldsymbol{\eta}) = \eta_i \alpha U$ is the required CPU cycles, and $d_i(\boldsymbol{\eta}) = \eta_i \beta U$ is the output computed result (in bits).

B. Communication Model

The UE offloads the sub-tasks using the entire channel bandwidth in a sequential technique. The end-to-end delay consists of two delay components: (1) *Task transmission delay*: the uplink and downlink transmission delay of the i-th sub-task can be expressed as $\mathcal{D}u_i(\boldsymbol{\eta}) = \frac{u_i(\boldsymbol{\eta})}{Ru_i}$ and $\mathcal{D}d_i(\boldsymbol{\eta}) = \frac{d_i(\boldsymbol{\eta})}{Rd_i}$, respectively. In sequential task offloading, the uplink transmission of the i-th sub-task will not start until the uplink transmission of all the previous $(i-1)$ sub-tasks is finished. In other words, the waiting time of the i-th sub-task is $\mathcal{W}_i(\boldsymbol{\eta}) = \sum_{j=1}^{i-1} \mathcal{D}u_j(\boldsymbol{\eta})$ and $\mathcal{W}_1(\boldsymbol{\eta}) = 0$. (2) *Computing delay*: The CPU in the i-th server computes the sub-task with a computational speed f_i (in cycles per second). Consequently, the computing delay in the i-th server is $\mathcal{D}c_i(\boldsymbol{\eta}) = \frac{c_i(\boldsymbol{\eta})}{f_i}$. The delay for completing the i-th sub-task can be expressed as

$$\mathcal{D}_i(\boldsymbol{\eta}) = \mathcal{W}_i(\boldsymbol{\eta}) + \mathcal{D}u_i(\boldsymbol{\eta}) + \mathcal{D}c_i(\boldsymbol{\eta}) + \mathcal{D}d_i(\boldsymbol{\eta}), \quad (2)$$

and the total latency for completing the task is given by

$$\mathcal{L}(\boldsymbol{\eta}) = \max_{\forall i \in \mathcal{S}} \{\mathcal{D}_i(\boldsymbol{\eta})\}. \quad (3)$$

The transmission failure probability of offloading the i-th sub-task can be written as

$$P_i(\boldsymbol{\eta}) = P_i^u(\boldsymbol{\eta}) + \left[1 - P_i^u(\boldsymbol{\eta})\right] P_i^d(\boldsymbol{\eta}), \quad (4)$$

where $P_i^u(\boldsymbol{\eta})$ is the uplink transmission error defined as [6]:

$$P_i^u(\boldsymbol{\eta}) = 1 - (1-q)^{\frac{u_i(\boldsymbol{\eta})}{\varrho_i^u}}, \quad (5)$$

and $P_i^d(\boldsymbol{\eta})$ is the downlink transmission error, similarly defined as:

$$P_i^d(\boldsymbol{\eta}) = 1 - (1-q)^{\frac{d_i(\boldsymbol{\eta})}{\varrho_i^d}}, \quad (6)$$

with ϱ_i^u and ϱ_i^d as the uplink and downlink transport block size, respectively, and q as the targeted block error rate. Consequently, the task offloading failure probability can be expressed as

$$\begin{aligned} P(\boldsymbol{\eta}) &= 1 - \prod_{i=1}^{N} \left(1 - P_i(\boldsymbol{\eta})\right) \\ &= 1 - (1-q)^{U \sum_{i=1}^{N} \eta_i \left(\frac{\delta}{\varrho_i^u} + \frac{\beta}{\varrho_i^d}\right)}. \end{aligned} \quad (7)$$

III. Problem Formulation

Offloading the task to fewer participating servers with better communication channels reduces the task offloading failure probability. On the other hand, a small number of participating servers increases the latency. Moreover, the servers with good channel quality are not always the best in terms of the computational speed. Our objective is to minimize $\mathcal{L}(\boldsymbol{\eta})$ and $P(\boldsymbol{\eta})$ simultaneously. Hence, the problem is a multi-objective optimization problem. To tackle the trade-off between $\mathcal{L}(\boldsymbol{\eta})$ and $P(\boldsymbol{\eta})$, we consider the weighted sum method [7]. Keeping in mind that $\mathcal{L}(\boldsymbol{\eta})$ and $P(\boldsymbol{\eta})$ have different orders of magnitude and ranges, they should be transformed such that they have similar ranges [8]. We define the latency-reliability cost function as

$$\Psi(\boldsymbol{\eta}) = \lambda \frac{\mathcal{L}(\boldsymbol{\eta})}{L} + (1-\lambda) \frac{P(\boldsymbol{\eta})}{E}, \quad (8)$$

where

$$L = \max_{\forall i \in \mathcal{S}} \{\frac{\delta U}{Ru_i} + \frac{\alpha U}{f_i} + \frac{\beta U}{Rd_i}\}, \quad (9)$$

and

$$E = \max_{\forall i \in \mathcal{S}} \{1 - (1-q)^{\frac{\delta U}{\varrho_i^u}} (1-q)^{\frac{\beta U}{\varrho_i^d}}\} \quad (10)$$

are the highest values of $\mathcal{L}(\boldsymbol{\eta})$ and $P(\boldsymbol{\eta})$, respectively,[1] and $0 \leq \lambda \leq 1$ is the relative weight. If both latency and failure probability are of equal importance, then $\lambda = 0.5$.

Let us assume that the first M servers in $\mathcal{S}^*$ contribute to the task offloading; a transmission collision occurs if at least one of the following two events happens: (1) One of the contributing servers finishes its sub-task computational and sends back the results while the task uplink offloading has not finished yet; (2) Two or more contributing servers send back the results simultaneously. To avoid such events and guarantee non-overlapping transmissions, we introduce the following constraints

$$\mathcal{D}u_1(\boldsymbol{\eta}) + \mathcal{D}c_1(\boldsymbol{\eta}) \geq \sum_{j=1}^{M} \mathcal{D}u_j(\boldsymbol{\eta}) \Rightarrow \mathcal{D}c_1(\boldsymbol{\eta}) \geq \sum_{j=2}^{M} \mathcal{D}u_j(\boldsymbol{\eta}) \quad (11)$$

and

$$\mathcal{D}u_i(\boldsymbol{\eta}) + \mathcal{D}c_i(\boldsymbol{\eta}) \geq \mathcal{D}c_{i-1}(\boldsymbol{\eta}) + \mathcal{D}d_{i-1}(\boldsymbol{\eta}), \forall\ 2 \leq i \leq M. \quad (12)$$

Constraint (11) guarantees that the first contributing server sends the feedback after all the other contributing servers finish their uplink transmissions. Constraint (12) guarantees that any subsequent contributing server sends its feedback after the previous server finishes the downlink transmission. A collision-free task allocation is shown in Fig. 1.

We introduce the server contribution decision variable vector, $\boldsymbol{\gamma} = [\gamma_i]_{1 \times N}$, where the binary decision variable γ_i is defined as

$$\gamma_i = \begin{cases} 1, & \text{if } s_i \text{ is contributing to the task offloading} \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

[1] L and E represent the latency and error encountered in offloading the entire task to the server that causes the highest latency and the server that causes the highest offloading error, respectively.

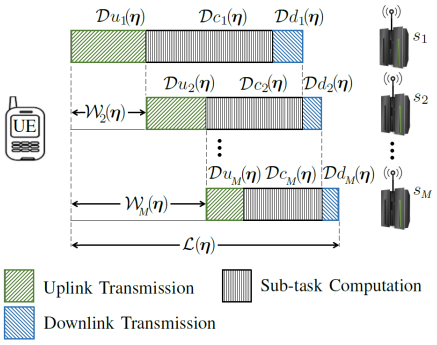


Figure 1: Sequential and collision-free task makespan of offloading the task to M servers.

Consequently, the number of contributing servers $M = \sum_{i=1}^{N} \gamma_i$ and the optimization problem is formulated as

$$\mathbf{P1} \min_{\boldsymbol{\eta},\boldsymbol{\gamma}} \quad \Psi(\boldsymbol{\eta}), \tag{14a}$$

$$\text{s.t.} \quad \mathcal{D}c_1(\boldsymbol{\eta}) \geq \sum_{j=2}^{N} \mathcal{D}u_j(\boldsymbol{\eta}), \tag{14b}$$

$$\mathcal{D}u_i(\boldsymbol{\eta}) + \mathcal{D}c_i(\boldsymbol{\eta}) \geq \gamma_i \left(\mathcal{D}u_{i-1}(\boldsymbol{\eta}) + \mathcal{D}c_{i-1}(\boldsymbol{\eta})\right), \tag{14c}$$

$$\forall\ 2 \leq i \leq N,$$

$$\eta_i \leq \gamma_i \leq \eta_i U,\ \forall i \in \mathcal{S}^*, \tag{14d}$$

$$\gamma_i \geq \gamma_{i+1},\ \forall 1 \leq i \leq N-1, \tag{14e}$$

$$\sum_{i=1}^{N} \eta_i = 1, \tag{14f}$$

$$\eta_i \geq 0,\ \forall i \in \mathcal{S}^*, \tag{14g}$$

$$\gamma_i \in \{0,1\},\ \forall i \in \mathcal{S}^*. \tag{14h}$$

Constraints (14b) and (14c) guarantee collision-free task offloading. In (14c), if server s_i is contributing to task offloading, i.e., $\gamma_i = 1$, the size of sub-task τ_i should be chosen such that (12) is satisfied to guarantee non-overlapping transmissions. Constraint (14d) ensures that if server s_i receives no sub-task, then it should not be selected as a contributing server. It also guarantees that $\gamma_i = 1$ only if server s_i is contributing to the task offloading. Constraint (14e) guarantees the offloading sequence and server priority by ensuring that the allowable $\boldsymbol{\gamma}$ vectors can never have $\gamma_i = 0$ and $\gamma_{i+1} = 1$ for any two consecutive elements. Therefore, $\boldsymbol{\gamma}$ guarantees that the first M servers in $\mathcal{S}^*$ contribute in the offloading. Consequently, we have only N feasible possibilities of $\boldsymbol{\gamma}$, i.e., $\boldsymbol{\gamma}^1 = [1,0,0,...0]^T$, $\boldsymbol{\gamma}^2 = [1,1,0,...0]^T$, $\boldsymbol{\gamma}^3 = [1,1,1,0,...0]^T$ up to $\boldsymbol{\gamma}^N = [1,1,1,1,...1]^T$. Constraints (14f) and (14g) guarantee the offloading of the whole task. The optimization problem in (14) is a non-convex mixed integer non-linear program which cannot be directly solved by the convex optimization techniques. The next section introduces an optimal solution to the problem **P1**.

IV. Benchmark Optimal Solution Algorithm

In this section, we propose an exact algorithm to find an optimal solution for **P1**. For a given fixed $\boldsymbol{\gamma} = \hat{\boldsymbol{\gamma}}$, it is important to note that:

$$\mathcal{L}(\boldsymbol{\eta}) = \mathcal{D}_{\hat{M}}(\boldsymbol{\eta}), \tag{15}$$

where $\hat{M} = \sum_{j=1}^{N} \hat{\gamma}_j$, and hence, for iterations $1 \leq i \leq N$, we solve the following *non-linear program* (NLP):

$$\mathbf{P2} \min_{\boldsymbol{\eta}} \quad \lambda \frac{\mathcal{D}_{\hat{M}}(\boldsymbol{\eta})}{L} + (1-\lambda)\frac{P(\boldsymbol{\eta})}{E}, \tag{16a}$$

$$\text{s.t.} \quad \text{(14b), (14c), (14d), (14f) and (14g).} \tag{16b}$$

The constraint sets (14b), (14c), (14d), (14f) and (14g) in **P2** are linear and yield a polyhedron feasible region. It is straightforward to show that $P(\boldsymbol{\eta})$ in (16a) is concave, since it can be re-written in the form $1 - \exp\left[\mathbf{a}^T\boldsymbol{\eta}\right]$, where $\mathbf{a} = [a_i]_{1\times N}$ with $a_i = \ln(1-q)\, U\left(\frac{\delta}{\varrho_i^u} + \frac{\beta}{\varrho_i^d}\right)$. Since $\exp(.)$ is known to be a convex function, then a composition with an affine mapping is also convex, i.e., $\exp\left[\mathbf{a}^T\boldsymbol{\eta}\right]$ is convex [9, Section 3.2.2] and following this directly, $P(\boldsymbol{\eta})$ is concave. The linear term $\lambda\frac{\mathcal{D}_{\hat{M}}(\boldsymbol{\eta})}{L}$ is both concave and convex, while $(1-\lambda)\frac{P(\boldsymbol{\eta})}{E}$ is concave; hence the sum of the two terms is concave. Consequently, the optimal solution lies in one of the extreme points (vertices) of the polyhedron [10, Section 7.8], [11, Section 3.2]. Algorithm 1 is designed to obtain the optimal solution of **P1**, in which the main steps are described as follows. The decision vector $\boldsymbol{\gamma}$ is set in each iteration i to $\boldsymbol{\gamma}^i = \sum_{j=1}^{i} \mathbf{e}_j$, where $\mathbf{e}_j$ is an N-element vector whose j-th element is the only non-zero element and equal to 1. In the i-th iteration, the feasible set of solutions for $\boldsymbol{\eta}$ is a polyhedron $\mathcal{P}_i$ whose set of vertices $\mathcal{V}_i$ are all enumerated using primal-dual polytope method [12]. For each vertex $\mathbf{v}_i^k \in \mathcal{V}_i$, $k = 1,2,\ldots,|\mathcal{V}_i|$ (where $|\bullet|$ is the cardinality of a set), the corresponding task allocation vector is obtained as $\boldsymbol{\eta} = \left[\mathbf{v}_i^{k\,T}, \mathbf{0}_{1\times(N-i)}\right]^T$ and the objective function is evaluated. The algorithm terminates after examining the vertices of N polyhedrons and returns $(\boldsymbol{\eta}^*, \boldsymbol{\gamma}^*)$ optimal for **P1**. It is worth mentioning that identifying all vertices of a polyhedron is NP hard [13]. The next section introduces a more computationally-efficient heuristic solution to the problem **P1**.

V. Heuristic Solution

In this section, a sub-optimal solution is proposed to solve **P1**. Our strategy is described as follows. For a given $\mathcal{S}^*$ and M contributing servers, the collision-free task allocation that provides minimum latency can be achieved by replacing the inequalities in (11) and (12) by equalities. After simple algebraic manipulations, we obtain the following set of equations

$$\begin{aligned}
\eta_1\frac{\alpha}{f_1} &= \eta_2\frac{\delta}{Ru_2} + \sum_{j=3}^{M}\eta_j\frac{\delta}{Ru_j},\\
\eta_2\frac{\alpha}{f_2} &= \sum_{j=3}^{M}\eta_j\frac{\delta}{Ru_j} + \eta_1\frac{\beta}{Rd_1},\\
&\cdots\\
\eta_M\frac{\alpha}{f_M} &= \sum_{j=1}^{M-1}\eta_j\frac{\beta}{Rd_j}.
\end{aligned} \tag{17}$$

Algorithm 1 Optimum Solution Algorithm.

Input: U, N, $\mathcal{S}$, δ, α, β, and λ;
Calculate L and E using (9) and (10), respectively;
Obtain $\mathcal{S}^*$ by sorting s_i in $\mathcal{S}$ in an ascending order of corresponding w_i;
$\Psi^* \leftarrow +\infty$; $\boldsymbol{\gamma}^0 \leftarrow [\mathbf{0}_{1\times N}]^T$;
for $i = 1$ to N **do**
  $\boldsymbol{\gamma}^i = \boldsymbol{\gamma}^{i-1} + \mathbf{e}_i$;
  $\mathcal{V}_i \leftarrow$ enumerates the vertices of polyhedron of constraints (14b), (14c), (14d), (14f) and (14g);
  for $k = 1$ to $|\mathcal{V}_i|$ **do**
    $\boldsymbol{\eta} = \left[\mathbf{v}_i^k, \mathbf{0}_{1\times(N-i)}\right]^T$;
    Evaluate $\Psi(\boldsymbol{\eta})$;
    if $\Psi^* > \Psi(\boldsymbol{\eta})$
      $\Psi^* \leftarrow \Psi(\boldsymbol{\eta})$; $\boldsymbol{\eta}^* \leftarrow \boldsymbol{\eta}$; $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^i$;
    end if
  end for
end for
Return Ψ^*, $\boldsymbol{\eta}^*$, and $\boldsymbol{\gamma}^*$.

Further straightforward manipulations of (17) yield

$$\begin{aligned}
\left(\frac{\alpha}{f_1}+\frac{\beta}{Rd_1}\right)\eta_1 &= \left(\frac{\delta}{Ru_2}+\frac{\alpha}{f_2}\right)\eta_2,\\
\left(\frac{\alpha}{f_2}+\frac{\beta}{Rd_2}\right)\eta_2 &= \left(\frac{\delta}{Ru_3}+\frac{\alpha}{f_3}\right)\eta_3,\\
&\cdots,\\
\left(\frac{\alpha}{f_{M-1}}+\frac{\beta}{Rd_{M-1}}\right)\eta_{M-1} &= \left(\frac{\delta}{Ru_M}+\frac{\alpha}{f_M}\right)\eta_M.
\end{aligned} \tag{18}$$

Based on (18), (14f), and (14g), the sub-optimal collision-free task allocation results as:

$$\eta_i^* = \begin{cases} \left(1+\sum_{i=2}^{M}\frac{\prod_{j=1}^{i}\left(\frac{\alpha}{f_j}+\frac{\beta}{Rd_j}\right)}{\prod_{j=2}^{i}\left(\frac{\delta}{Ru_j}+\frac{\alpha}{f_j}\right)}\right)^{-1}, & i=1\\ \frac{\prod_{j=1}^{i}\left(\frac{\alpha}{f_j}+\frac{\beta}{Rd_j}\right)}{\prod_{j=2}^{i}\left(\frac{\delta}{Ru_j}+\frac{\alpha}{f_j}\right)}\eta_1^*, & 2\le i\le M\\ 0, & M<i\le N. \end{cases} \tag{19}$$

To tackle the latency-reliability trade-off, the number of contributing servers M varies from 1 to N. Using the sorted servers and (19), Algorithm 2 finds $\boldsymbol{\eta}$ that minimizes the latency-reliability cost function heuristically.

Calculating $\boldsymbol{\eta}$ according to (19) requires $\mathcal{O}(N^2)$ product operations and evaluating the objective function requires $\mathcal{O}(N)$ product operations in (7) and $\mathcal{O}(N)$ **max(.)** operations in (3). Consequently, the computational complexity of finding a solution using Algorithm 2 is $\mathcal{O}(N^2)$ product and **max(.)** operations. Therefore, this algorithm is suitable for online implementation.

VI. Simulation Results

In this section, for the numerical results, we consider the LTE system configuration with 20 MHz channel bandwidth and 100 resource blocks. We assume that the signal-to-noise ratio (SNR) at the servers and the UE is uniformly distributed in the interval $[0, 30]$ dB. The modulation and coding scheme (MCS) is adjusted dynamically to guarantee that q does not exceed 10^{-7} and the transport block size in each link is calculated based on the SNR-MCS mapping [6], [14]. The computational speed of the servers is uniformly distributed in the interval $[1\times10^7, 10\times10^7]$ cycles per second. The task size U is set to 1 Mbits, $\delta = 1$, $\alpha = 1900/8$ cycles per bit, and $\beta = 0.2$. These parameters and $\lambda = 0.5$ are considered in the following results, unless otherwise stated.

Algorithm 2 Heuristic Solution Algorithm.

Input: U, N, $\mathcal{S}$, δ, α, β, and λ;
Calculate L and E using (9) and (10), respectively;
Obtain $\mathcal{S}^*$ by sorting s_i in $\mathcal{S}$ in an ascending order of corresponding w_i;
$\Psi^* \leftarrow +\infty$;
$\boldsymbol{\gamma}^0 \leftarrow [\mathbf{0}_{1\times N}]^T$;
for $M = 2$ to N **do**
  $\boldsymbol{\gamma}^M = \boldsymbol{\gamma}^{M-1} + \mathbf{e}_M$;
  Calculate $\boldsymbol{\eta}$ according to (19);
  Evaluate $\Psi(\boldsymbol{\eta})$;
  if $\Psi^* > \Psi(\boldsymbol{\eta})$ and (14d) satisfied
    $\Psi^* \leftarrow \Psi(\boldsymbol{\eta})$; $\boldsymbol{\eta}^* \leftarrow \boldsymbol{\eta}$; $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^M$;
  end if
end for
Return Ψ^*, $\boldsymbol{\eta}^*$ and $\boldsymbol{\gamma}^*$.

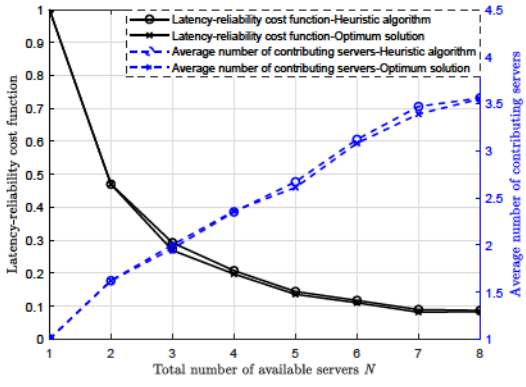
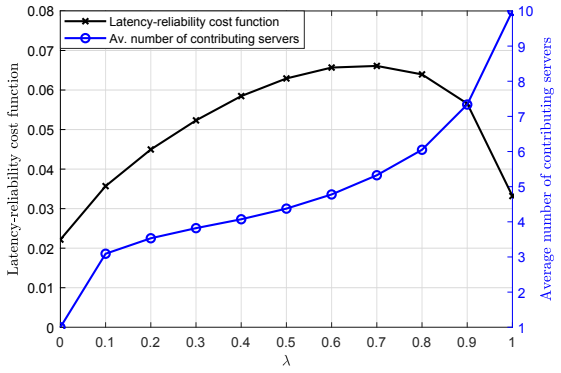


Figure 2: The effect of the number of available servers N.

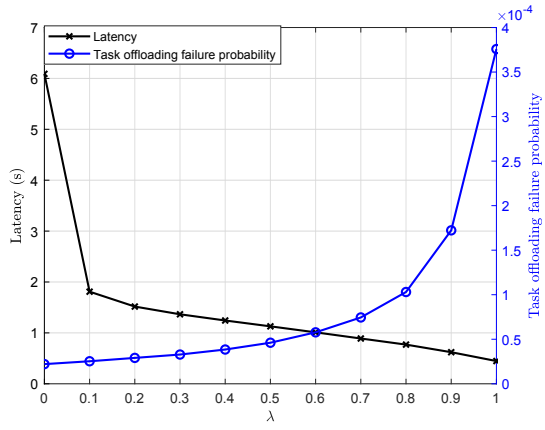
Figure 2 illustrates the performance of the optimal and sub-optimal solutions versus the number of available servers N. It is seen that the latency-reliability cost function decreases as N increases and the proposed sub-optimal solution achieves near-optimal performance. It is clear that the number of the contributing servers increases as the the number of available servers increases.

The effect of the latency-reliability relative weight λ on the cost function and the number of contributing servers is presented in Fig. 3a. It is noticed that as λ increases, the number of the contributing servers increases and the cost function is concave in λ. To get more insight into the behavior of the cost function, Fig. 3b shows the corresponding latency

and offloading failure probability. It is clear that for low values of λ, the algorithm minimizes the offloading failure probability and as λ approaches 1, it minimizes the latency.



(a) Latency-reliability cost function.



(b) Latency and offloading failure probability.

Figure 3: The effect of the relative weight λ with $N = 10$ servers.

Weighted Sum vs. Latency-Reliability Product Cost Function: Here, the proposed heuristic solution is utilized to compare the performance of the proposed framework for the weighted sum latency-reliability cost function in (8) and the latency-reliability product cost function which is defined in [6] as $\Psi(\boldsymbol{\eta}) = \mathcal{L}(\boldsymbol{\eta}) P(\boldsymbol{\eta})$. It is important to mention that for a weighted sum objective function, we can easily choose to improve either the delay or the reliability by controlling λ without any change in the structure of the problem or in the solution algorithm. This is not the case for the product objective function.

The latency and offloading failure probability of both weighted sum with $\lambda = 0.5$ and latency-reliability product cost functions are illustrated in Fig. 4. It is clear that for $\lambda = 0.5$, the weighted sum achieves lower latency. On the other hand, the latency-reliability product cost function achieves lower offloading failure probability.

VII. Conclusion

This paper proposed a sequential task offloading scheme that guarantees collision-free offloading to multiple MEC servers. The problem was formulated as a weighted-sum multi-objective optimization problem of latency and offloading failure probability, which enables to develop an exact technique that guarantees the optimal solution. A more computationally-efficient heuristic algorithm was developed for online implementation. Simulation results illustrated that the proposed offloading scheme can effectively reduce both latency and offloading failure probability.

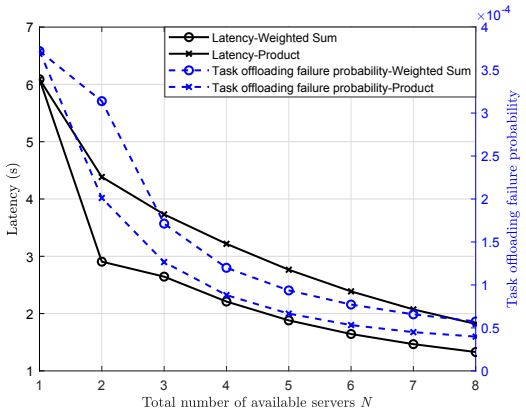


Figure 4: Latency and offloading failure probability of weighted sum with $\lambda = 0.5$ and latency-reliability product cost functions.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

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

[3] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, First quarter 2014.

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

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

[6] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, no. 99, pp. 12 825–12 837, Feb. 2018.

[7] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, Apr. 2004.

[8] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 853–862, Jun. 2010.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[10] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. Siam, 2014, vol. 19.

[11] W. L. Winston, M. Venkataramanan, and J. B. Goldberg, *Introduction to Mathematical Programming*. Thomson/Brooks/Cole Duxbury; Pacific Grove, CA, 2003, vol. 1.

[12] D. Bremner, K. Fukuda, and A. Marzetta, "Primal—dual methods for vertex and facet enumeration," *Discrete & Computational Geometry*, vol. 20, no. 3, pp. 333–357, Oct. 1998.

[13] M. E. Dyer, "The complexity of vertex enumeration methods," *Mathematics of Operations Research*, vol. 8, no. 3, pp. 381–402, 1983.

[14] M. Mezzavilla, M. Miozzo, M. Rossi, N. Baldo, and M. Zorzi, "A lightweight and accurate link abstraction model for the simulation of LTE networks in NS-3," in *Proc. 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2012, pp. 55–60.