International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 4, August 2020, pp. 3757~3766
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i4.pp3757-3766

Time and resource constrained offloading with multi-task in a mobile edge computing node

Mohamed El Ghmary, Youssef Hmimz, Tarik Chanyour, Mohammed Ouçamah Cherkaoui Malki
SidiMohamed Ben Abdellah University, FSDM, LIIAN Labo, Morocco

Article Info

Article history:

Received Aug 4, 2019
Revised Dec 21, 2019
Accepted Jan 11, 2020

Keywords:

Computation offloading
Mobile edge computing
Processing time optimization
Simulated annealing

ABSTRACT

In recent years, the importance of the mobile edge computing (MEC) paradigm along with the 5G, the Internet of Things (IoT) and virtualization of network functions is well noticed. Besides, the implementation of computation-intensive applications at the mobile device level is limited by battery capacity, processing capabalities and execution time. To increase the batteries life and improve the quality of experience for computationally intensive and latency-sensitive applications, offloading some parts of these applications to the MEC is proposed. This paper presents a solution for a hard decision problem that jointly optimizes the processing time and computing resources in a mobile edge-computing node. Hence, we consider a mobile device with an offloadable list of heavy tasks and we jointly optimize the offloading decisions and the allocation of IT resources to reduce the latency of tasks' processing. Thus, we developped a heuristic solution based on the simulated annealing algorithm, which can improve the offloading rate and reduce the total task latency while meeting short decision time. We performed a series of experiments to show its efficiency. Finally, the obtained results in terms of full-time treatrement are very encouraging. In addition, our solution makes offloading decisions within acceptable and achievable deadlines.

Corresponding Author:

Mohamed El Ghmary,
Department of Computer Science,
SidiMohamed Ben Abdellah University,
FSDM, LIIAN Labo, Fez, Morocco.
Email: mohamed.elghmary@usmba.ac.ma

1. INTRODUCTION

Smart Mobile devices are changing day by day in the marketplace. These devices are compact and small, thing that makes convenient to use. There are currently billions of connected objects in the world (IoT) [1]. Mobile devices typically have limited resources, such as limited battery power and local CPU capacity, and may therefore suffer from an unsatisfactory computing experience. Mobile edge computing is emerging as a promising remedy. By offloading computing tasks to physically close MEC servers, the quality of the computing experience, such as device power consumption and turnaround time, could be greatly improved [2, 3]. As a result, the execution time of these tasks can be effectively reduced by offloading heavy tasks to an MEC node.

Computationally demanding mobile applications, such as face recognition, language processing and online gaming, have been developing fast and increasingly outgrowing the limited capabilities of devices [4]. Typical characteristics of MEC include low latency, proximity, high bandwidth, mobility support and location awareness [5]. There is only a number of works that have jointly optimized the offloading decisions and resource allocation of multiple devices, typically for delay-tolerant services [6, 7]. In [7], both offline and online approaches were proposed for the joint optimization, where a single task was offloaded to the MEC server while the others were executed locally. In [7], the allocations of computational resources and

transmission bandwidths were optimized by exploiting semi-definite programming, and the offloading decisions were generated through randomized rounding. In [6], a heuristic scheme based on a submodular optimization method was proposed to jointly optimize offloading decisions and resource allocations for delay-tolerant tasks. More recent studies have been focused on either offloading decisions or resource allocation among multiple devices [8-12], with no allocation of transmission or computational resources. In [12], under limited cloud resources, both online and offline algorithms were developed to partition tasks for multiple devices. In [9], using queueing theory, offloading decisions were formulated as a non-cooperative game in a three-tier MEC architecture consisting of mobile devices, cloudlets and distant cloud. Mobile edge Computing is a form of computer architecture [13-15]. It is about processing the data at the periphery of the network directly where they are generated. Rather than transfering the data generated by devices connected to the IoT to the cloud or the data center. It can offer nearby customized services that require good transmission bandwidth, additional data storage and processing. As illustrated in Figure 1, MEC can augment mobile devices' capabilities by offloading [3, 16, 17] some parts of their heavy applications via wireless access to a resource-rich edge node, and then effectively reduces their power consumptions [18]. The first works which dealt with the offloading within MEC environments were interested in studying the multi-user single-task case. In these scenarios, a user only targets to offload only a unique task. They studied resource allocation while they optimize the average task duration [19]. Actually, an application is generally partitioned into multiple tasks, and the offloading decisions must concern each of them. Consequently, even a single user has to handle simultaneously multiple tasks. Therefore, the offloading decision parameters should be selected according to a multi-task scenario.

Figure 1. Mobile edge computing illustration

Recently, the authors of [20] studied a single-user multi-task offloading senario by optimizing radio resources and local frequency. They did not consider the local energy availability nor the remote server's frequency. Besides, they consider tasks with the same deadline T^d. In this work, we study the general multi-task offloading senario where we introduce the control of the available local energy, and consider the edge server's frequency as a decision parameter in our optimization problem. Moreover, we consider a general setting where each offloadable task has to be executed within its specific deadline t_i^{max}. According to our vision, we can prolong the battery life of the mobile device by considering the amount of its available power, and reduce the tasks' processing time by adjusting the edge server's frequency. Subsequently, we have formulated an optimization problem that minimizes the processing time by jointly deciding the local and edge computing frequencies, as well as the offloading decisions. Due to its combinatorial nature and after its decomposition, we propose a heuristic solution based on a simulated annealing algorithm to jointly decide the tasks' offloading and the allocation of computing resources. The objective is to minimize the processing time via the offloading by considering the tasks' latency constraints and a threshold of available processing time. The remainder of this paper is organized as follows : the system's model and the optimization problem formulation are presented in Section 2. In Section 3, we present our method to solve the optimization problem. In section 4 we present the simulation results and their discussion. Finally, Section 5 concludes the paper.

2. SYSTEM MODEL AND PROBLEM FORMULATION

2.1. System model

Figure 2 Shows a single smart mobile device (SMD) containing an offloadable multi-task list. In this work, we plan to study the behavior of the offloading process for a multi-task SMD in an edge environment, while we optimize computation resources available at the edge server as well as at the mobile device. Particularly, the available energy at the SMD for tasks execution is limited. Besides, in the context of offloading, some pieces of a computationally intensive application are divided into multiple mutually independent offloadable tasks [21-24]. Therefore, according to the available computational and radio resources, some tasks are pick-up from the resulting tasks list to be offloaded to the edge servers for computing. The others are performed locally on the SMD itself. The execution of the whole list must happen within the time limit of the application. Additionally, it is assumed that the SMD concurrently performs computation and wireless transmission. For all these considerations, we derive a mathematical processing time model that considers three main decisions: the offloading decision for each task, the local execution frequency of the SMD, and the server execution frequency at the edge. Then, we formulate an processing time problem.

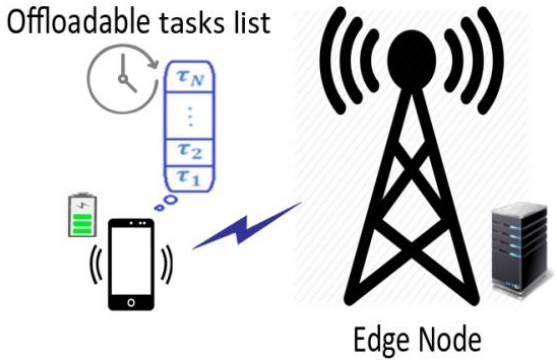

Figure 2. System model illustration

Practically, the SMD is connected to an edge node (EN), and is intended to offload a set of independent tasks by the mean of an edge access point (EAP). Additionally, the wireless channel conditions between the SMD and the wireless access point are not considered in this work. Moreover, at the time of the offloading decision and the transmission of the offloadable tasks, the uplink rate r is assumed almost unchanged.

As shown in Figure 2., the considered smart mobile device contains N independent tasks denoted as $\tau \triangleq \{\tau_1, \tau_2, \dots, \tau_N\}$. In addition, these tasks are assumed to be computationally intensive and delay sensitive and have to be completed. Each task τ_i can be processed either locally or at the edge. It represents an atomic input data task that cannot be divided into sub-tasks. Moreover, it is characterized by the following three parameters $\tau_i \triangleq \langle d_i, \lambda_i, t_i^{max} \rangle$. The first one denoted d_i[bits] identifies the amount of the input parameters and program codes to transfer from the user's local device to the edge server. The second one denoted λ_i [cycles] specifies the workload referring to the computation amount needed to accomplish the processing of this task. The third parameter t_i^{max} refers to the required maximum latency for this task.

The execution nature decision for a task τ_i either locally or by offloading to the edge server is denoted x_i where $x_i \in \{0; 1\}$. $x_i = 1$ indicates that the SMD has to offload τ_i to the edge server, and $x_i = 0$ indicates that τ_i is locally processed. From this point, all time expressions are given in *Seconds*, and energy consumptions are given in *Joule*. Then, if the SMD locally executes task τ_i, the completion time of its local execution is $t_i^L = \frac{\lambda_i}{f_L}$. So, for all tasks, we have:

$$t^L = \sum_{i=1}^{N} (1 - x_i) \frac{\lambda_i}{f_L}. \quad (1)$$

additionally, the corresponding energy consumption is given by: $e_i^L = k_L . f_L^2 . \lambda_i$ [11]. Hence, the total energy consumption while executing all tasks that were decided to be locally executed in the SMD is given by:

$$e^L = k_L . f_L^2 . \sum_{i=1}^{N} \lambda_i (1 - x_i) \quad (2)$$

if task τ_i is offloaded to the edge node, the offloading process completion time is: $t_i^O = t_i^{Com} + t_i^{Exec} + t_i^{Res}$, where t_i^{Com} is the time to transmit the task to the EAP, and it is given by $t_i^{Com} = \frac{d_i}{r}$. t_i^{Exec} is the time to execute the task τ_i at the EN, and it can be formulated as $t_i^{Exec} = \frac{\lambda_i}{f_S}$. t_i^{Res} is the time to receive the result out from the edge node. Because the data size of the result is usually ignored compared to the input data size, we ignore this relay time and its energy consumption as adopted by [25]. Hence, for the τ_i task $t_i^O = x_i\left(\frac{d_i}{r} + \frac{\lambda_i}{f_S}\right)$, and for all tasks, we have:

$$t^O = \sum_{i=1}^{N} x_i\left(\frac{d_i}{r} + \frac{\lambda_i}{f_S}\right). \quad (3)$$

So, the energy consumption of the communication process can be obtained by multiplying the resulting transmission period by the transmission undertaken power p^T, and the rest of the execution period by the idle mode power p^I. Thus, this energy is:

$$e^C = \frac{p^T \sum_{i=1}^{N} x_i d_i}{r}. \quad (4)$$

Finally, given the offloading decision vector $\mathbb{X}$ for all tasks, the local execution frequency $\boldsymbol{f_L}$ of the SMD, and the server execution frequency $\boldsymbol{f_S}$ at the edge, the total execution time for the SMD is composed of its local execution time, the communication time as well as the execution time at the EN, and it is given by:

$$T(\mathbb{X}, f_L, f_S) = t^L + t^O. \quad (5)$$

Then, according to (1) and (3) and if we note $\Lambda = \sum_{i=1}^{N} \lambda_i$, the total execution time can be formulated as:

$$T(\mathbb{X}, f_L, f_S) = \left\{\frac{\Lambda}{f_L} - \frac{\sum_{i=1}^{N} \lambda_i x_i}{f_L} + \frac{\sum_{i=1}^{N} d_i x_i}{r} + \frac{\sum_{i=1}^{N} \lambda_i x_i}{f_S}\right\}. \quad (6)$$

2.2. Problem formulation

In this section, we present our optimization problem formulation that aims to minimize the overall execution time in the offloading process. Initially, to prepare the problem's data we start with an initial sorting of the tasks list $\tau \triangleq \{\tau_1, \tau_2, \dots, \tau_N\}$ according to their deadlines t_i^{max}. Hence, the tasks execution order within the SMD or the edge server in the final solution must fulfill the initial order for both cases. Accordingly, the obtained problem is formulated as:

$$\boldsymbol{\mathcal{P}1}: \min_{\{x, f_L, f_S\}} \left\{\frac{\Lambda}{f_L} - \frac{\sum_{i=1}^{N} \lambda_i x_i}{f_L} + \frac{\sum_{i=1}^{N} d_i x_i}{r} + \frac{\sum_{i=1}^{N} \lambda_i x_i}{f_S}\right\},$$

$$\text{s.t. } (C_{1.1}) \quad x_i \in \{0; 1\}; \qquad i \in [\![1; N]\!];$$

$$(C_{1.2}) \quad F_L^{min} \le f_L \le F_L^{max};$$

$$(C_{1.3}) \quad 0 < f_S \le F_S ;$$

$$(C_{1.4}) \quad t_i^L = \frac{(1-x_i)}{f_L} \sum_{k=1}^{i} \lambda_k (1 - x_k) \le t_i^{max}; \qquad i \in [\![1; N]\!];$$

$$(C_{1.5}) \quad t_i^O = x_i \sum_{k=1}^{i} x_k \left(\frac{d_k}{r} + \frac{\lambda_k}{f_S}\right) \le t_i^{max} ; \qquad i \in [\![1; N]\!];$$

$$(C_{1.6}) \quad e^L + e^C = k_L . f_L^2 . \sum_{i=1}^{N} \lambda_i (1 - x_i) + \frac{p^T}{r} \sum_{i=1}^{N} d_i x_i \le E^{max}.$$

In this work, each one of the available tasks can be either executed locally or offloaded to the edge node. Thus, every feasible offloading decision solution has to satisfy the above constraints. The constraint $(C_{1.1})$ refers to the offloading decision variable x_i for task τ_i which equals 0 or 1. The second constraint $(C_{1.2})$indicates that the allocated variable local frequency $f_L$belongs to a priori fix interval given by $[F_L^{min}, F_L^{max}]$. Similarly, the allocated variable remote edge server frequency $f_S$belongs to the interval $[0, F_S^{max}]$ in constraint $(C_{1.3})$. The constraint $(C_{1.4})$ shows that the execution time of each decided local task must satisfy its deadline t_i^{max}. Similarly, in constraint $(C_{1.5})$, the offloading time of each decided offloadable task must satisfy the same deadline t_i^{max}. The final constraint $(C_{1.6})$ imposes that the total local execution energy must not exceed the tolerated given amount E^{max}. This constraint is important especially for SMDs with critical battery.

3. PROBLEM RESOLUTION

In this section, we will introduce how we derive our solution from the obtained optimization problem.

3.1. Problem decomposition

In our proposed model, the offloading decision vector for all the tasks is denoted $\mathbb{X}$. Let define the vector that contains the offloadable tasks' identifiers:

$$\mathbb{X}_1 = \{i \in \mathbb{X} \ / \ x_i = 1\} \quad (7)$$

$$\mathbb{X}_0 = \{i \in \mathbb{X} \ / \ x_i = 0\} \quad (8)$$

Additionally, we define: $\Lambda_i = \sum_{k=1}^{i} \lambda_i$, $\Lambda_i^1 = \sum_{k=1}^{i} x_i\lambda_i$, $D_i = \sum_{k=1}^{i} d_i$, $D_i^1 = \sum_{k=1}^{i} x_i d_i$.

Also, given the decision vector $\mathbb{X}_1$, constraint $(C_{1.4})$ for a local task can be reformulated as $\frac{\Lambda_i - \Lambda_i^1}{t_i^{max}} \leq f_L;\ \forall i \in [\![1; N]\!]$. Finally, it is equivalent to one constraint: $\max_i \left\{\frac{\Lambda_i - \Lambda_i^1}{t_i^{max}}\right\} \leq f_L$. Likewise, constraint $(C_{1.5})$ for an offloadable task means $\frac{D_i^1}{r} + \frac{\Lambda_i^1}{f_S} \leq t_i^{max}$ $(\forall i \in [\![1; N]\!])$. So $\frac{D_i^1}{r}$ and $\frac{\Lambda_i^1}{f_S}$ must be strictly less than t_i^{max} $(\forall i \in [\![1; N]\!])$; particularly $\min_i \left\{t_i^{max} - \frac{D_i^1}{r}\right\} > 0$. In this case constraints $(C_{1.5})$ can be reformulated as $\frac{\Lambda_i^1}{t_i^{max} - \frac{D_i^1}{r}} \leq f_S;\ \forall i \in [\![1; N]\!]$. Finally, it is equivalent to one constraint: $\max_i \left\{\frac{\Lambda_i^1}{t_i^{max} - \frac{D_i^1}{r}}\right\} \leq f_S$.

Similarly, constraint $(C_{1.6})$ means $k_L.f_L^2.(\Lambda_N - \Lambda_N^1) + \frac{p^T D_N^1}{r} \leq E^{max}$. So $k_L.f_L^2.(\Lambda_N - \Lambda_N^1)$ and $\frac{p^T D_N^1}{r}$ must be strictly less than E^{max}. In this case constraint $(C_{1.6})$ can be reformulated as $f_L \leq \sqrt{\frac{E^{max} - \frac{p^T D_N^1}{r}}{k_L(\Lambda_N - \Lambda_N^1)}}$. For ease of use, let note:

$$f_L^- = \max_i \left\{\frac{\Lambda_i - \Lambda_i^1}{t_i^{max}}\right\}; \quad (9)$$

$$f_L^+ = \sqrt{\frac{E^{max} - \frac{p^T D_N^1}{r}}{k_L(\Lambda_N - \Lambda_N^1)}}; \quad (10)$$

$$f_S^- = \max_i \left\{\frac{\Lambda_i^1}{t_i^{max} - \frac{D_i^1}{r}}\right\}. \quad (11)$$

Thus, for a given offloading decision vector $\mathbb{X}$, we get the following optimization sub-problem:

$$\boldsymbol{P2}(\mathbb{X}): \min_{\{f_L, f_S\}} \left\{\frac{\Lambda_N - \Lambda_N^1}{f_L} + \frac{D_N^1}{r} + \frac{\Lambda_N^1}{f_S}\right\},$$

$$\text{s.t.} \quad (C_{2.1}) \quad F_L^{min} \leq f_L \leq F_L^{max};$$

$$(C_{2.2}) \quad f_L^- \leq f_L ;$$

$$(C_{2.3}) \quad f_S^- \leq f_S \leq F_S ;$$

$$(C_{2.4}) \quad k_L f_L^2(\Lambda_N - \Lambda_N^1) + \frac{p^T D_N^1}{r} \leq E^{max}.$$

Considering the continuous variables f_L and f_S, problem P2 is a continuous multi-variable optimization problem. The objective function $T(\mathbb{X}, f_L, f_S) = \frac{\Lambda_N - \Lambda_N^1}{f_L} + \frac{D_N^1}{r} + \frac{\Lambda_N^1}{f_S}$ can be decomposed into the following two independent functions $T_1(f_L)$ and $T_2(f_S)$ where $T_1(f_L) = \frac{\Lambda_N - \Lambda_N^1}{f_L}$ and $T_2(f_S) = \frac{D_N^1}{r} + \frac{\Lambda_N^1}{f_S}$. Moreover, given the disjunction between constraints $(C_{2.1}), (C_{2.2})$ and $(C_{2.4})$ on the one hand, and $(C_{2.3})$ in problem P2 on the other hand, this last can be equivalently decomposed into the following two independent optimization sub-problems.

$$\mathcal{P}3.1(\mathbb{X}): \min_{\{f_L\}} \left\{ T_1(f_L) = \frac{\Lambda_N - \Lambda_N^1}{f_L} \right\}$$

$$\text{s.t.} \quad (C_{3.1.1}) \qquad F_L^{min} \le f_L \le F_L^{max};$$

$$(C_{3.1.2}) \qquad f_L^- \le f_L \le f_L^+.$$

$$\mathcal{P}3.2(\mathbb{X}): \min_{\{f_S\}} \left\{ T_2(f_S) = \frac{D_N^1}{r} + \frac{\Lambda_N^1}{f_S} \right\}$$

$$\text{s.t.} \quad (C_{3.2.1}) \qquad f_S^- \le f_S \le F_S.$$

3.2. Problems resolution

For the $\mathcal{P}3.1$ problem, the objective function $T_1(f_L)$ is a strictly increasing continuous function according to its variable f_L. Hence, by taking into consideration the obtained constraints $(C_{3.1.1})$ and $(C_{3.1.2})$, we can derive the following function's optimum f_L^* given by:

$$f_L^* = \begin{cases} 0 & \text{if} \quad \mathbb{X} = \mathbb{X}_1 \\ \emptyset & \text{if} \quad E^{max} \le \frac{p^T D_N^1}{r} \text{ or } f_L^- > F_L^{max} \text{ or } f_L^+ < F_L^{min} \text{ or } f_L^- > f_L^+ \\ F_L^{max} & \text{if } f_L^+ > F_L^{min} \\ f_L^+ & \text{otherwise} \end{cases} \tag{12}$$

for the $\mathcal{P}3.2$ problem, the objective function $T_2(f_S)$ is strictly decreasing w.r.t. the variable f_S. So, by taking into consideration the $(C_{3.2.1})$ constraint, we can derive the following function's optimum f_S^* given by:

$$f_S^* = \begin{cases} \emptyset & \text{if} \quad \min_i \left\{ t_i^{max} - \frac{D_i^1}{r} \right\} \le 0 \text{ or } f_S^- > F_S \\ F_S & \text{otherwise} \end{cases} \tag{13}$$

3.2.1. The processing time determination

Similarly, given an offloading decision vector $\mathbb{X}$ the following algorithm uses the first algorithm to determine the minimal processing time:

Algorithm 1: Processing time calculation

Input: The list τ of N sub-tasks, offloading policy $\mathbb{X}$.
Output: $T(\mathbb{X}, f_L, f_S)$.

```
Calculate (f_L, f_S) using X according to (12) and (13);
if f_L = ∅ or f_S = ∅ then
    return ∞;
else
    Calculate T(X, f_L, f_S) according to (14);
    return T(X, f_L, f_S) ;
end if
```

3.3. Proposed solutions

Next, the problem relies on determining the optimal offloading decision vector $\mathbb{X}$ that gives the optimal processing time. However, to iterate over all possible combinations of a list of N binary variables, the time complexity is exponential (the exhaustive search over all possible solutions requires 2^N iterations). Subsequently, the total time complexity of the whole solution (including Algorithm 1) is $O(2^N)*O(1)=O(2^N)$ that is not practical for large values of N. In the following, we propose a low complexity approximate algorithm to solve this question.

3.3.1. Brute force search solution

For comparison purpose, we introduce the Brute Force Search method for feasible small values of N. This method explores all cases of offloading decisions and saves the one with the minimum processing time as well as its completion time. Now, the next algorithm summarizes the brute force search solution.

Algorithm 2: Brute Force Search Offloading

Input: The list τ of N sub-tasks;
Output: the offloading policy $\mathbb{X}^*$.

Initialize: minTime=∞**;**

1: $\mathbf{i \leftarrow 1}$;
2: **while** $\mathbf{i \leq 2^N - 1}$ **do**
3: Use the N bits representation of integer i to build the policy $\mathbb{X}$**;**
4: Call **Algorithm 2** to get newTime using τ and $\mathbb{X}$;
5: **if** $\mathbf{newTime < minTime}$ **then**
6: $\mathbf{minTime \leftarrow newTime}$ **;**
7: $\mathbb{X}^* \leftarrow \mathbb{X}$;
8: **end if**
9: $\mathbf{i \leftarrow i + 1}$;
10: **end while**
11: **return** $\mathbb{X}^*$ **;**

3.3.2. Simulated annealing offloading decision solution

We propose a method based on simulated annealing (SA) as the second solution. In the field of optimization, the SA technique has been adopted as a heuristic solution and in particular for difficult problems. To improve a solution, it uses an iterative random solution variation. Interested readers may consult the following works [26] and [27] for more details about this issue. Some references dealing with the offloading in cloud environments [7, 28, 29] use tasks' workload density defined as $\omega_i = \frac{\lambda_i}{d_i}$[cycle / bit] as a priority factor to decide the tasks' offloading. Additionally, the generated tasks are generally with different workload densities. Moreover, if two tasks are given with a slightly different data sizes, the one that consumes less processing time is the one given by the smallest cycles' count. Besides, with almost the same cycles' count, the one that consumes less offloading processing time is the one given by the smallest data size. In both cases, the task with the highest workload density is favorable for offloading (provided to have an offloading energy gain compared to the local execution and not to exceed its execution deadline). On the other hand, a task with a high workload density often has a large number of cycles. Its local execution is generally very expensive and thus makes its offloading often very favorable. In this context, we introduce a workload density threshold ω_T such that: tasks with $\omega_i > \omega_T$ are more favorable to be offloaded. The others are executed locally or offloaded with a proportional probability to their computational densities. Those with small densities are favorable for local execution, and those with high densities are favorable to be offloaded. Accordingly, if we note $\omega_{min} = \min_i\{\omega_i\}$, $\omega_{max} = \max_i\{\omega_i\}$ and the middle of the interval $[\omega_{max}, \omega_{min}]$ as $\underline{\omega_T} = (\omega_{max} + \omega_{min})/2$ then ω_T can be chosen such that $\underline{\omega_T} \leq \omega_T < \omega_{max}$.

In our proposed second solution, which we denote simulated annealing offloading decision (SAOD), we adopted the following general threshold probability:

$$p = e^{-\Delta T_i / T_0} \quad (14)$$

where T_0 is the initial temperature constant. ΔT_i is the solutions' processing time variation while changing the task i state. Then, in each stage of our solution and with the intention to avoid local optimums, random solutions with poor processing time performance are accepted in line with a certain probability threshold. Accordingly, the next algorithm summarizes our heuristic solution.

Algorithm 3: Simulated Annealing Offloading Decision

Input: The list τ of N sub-tasks,T_0, CF, ε, ω_T;
Output: the offloading policy $\mathbb{X}^*$.
Initialize: a random policy $\mathbb{X}$;
1: Call **Algorithm 1** to calculate oldTime using τ and $\mathbb{X}$;
2: minTime=∞;
3: **while** $T_0 > \varepsilon$ **do**
4: **for each** i in τ **do**
5: **if** $\omega_i > \omega_T$ **then**
6: **if** task i not in $\mathbb{X}_1$ **then**
7: add i to $\mathbb{X}_1$;
8: **end if**
9: **else if** $\omega_T - \omega_i \geq (\omega_T - \omega_{min}) * \text{random}(0,1)$ **then**

```
            if task i in 𝕏1 then
                move i from 𝕏1 to 𝕏0 ;
            end if
            else if task i in 𝕏0 then
                move i from 𝕏0 to 𝕏1 ;
            end if
        end if
    end if
    Update 𝕏 using the new 𝕏1;
    Call Algorithm 1 to get newTime using τ and 𝕏;
    if newTime≠ ∞ then
        ΔTi = newTime − oldTime
        if ΔTi < 0 then
            oldTime=newTime;
            if newTime<minTime then
                minTime =newTime ; 𝕏* = 𝕏 ;
            end if
        else
            Calculate p according to (14);
            if e^(−ΔTi/T0) > random(0,1) then
                oldTime = newTime;
            else
                Put back i to its original set;
            end if
        end if
    end if
  end for
  T0= T0*CF
end wile
return 𝕏* ;
```

In this algorithm, ***random (0,1)*** is a function's call that generates a random number in [0,1].

4. RESULTS AND DISCUSSION

4.1. Simulation setup

The presented results in this work are averaged for 100 time executions. We implement all the algorithms on the C++ language. The transmission bandwidth between the mobile device node and remote edge server is set tor = 100Kb/s. The local CPU frequency $f_L$of the mobile device will be optimized between $F_L^{min} = 1$MHz and $F_L^{max} = 60$MHz. The CPU frequency of the remote edge server node will be optimized under the value $F_S = 6$GHz. The deadlines t_i^{max} are uniformly defined from 0.5s to 2s. The threshold energy E^{max} is uniformly chosen in $[0.6\ ,\ 0.8] * \Lambda . k_L . (F_L^{max})^2$.Additionally, the data size of each one of the N tasks is assumed to be in [30,300] Kb. For the cycle amount of each task, it is assumed to belong to [60,600]MCycles. The idle power and transmission power are set to be $p^I = 0.01$ Watt and $p^T = 0.1$ Watt respectively. For the energy efficiency coefficients, we set $k_L = 10^{-26}$and $k_S = 10^{-29}$. For the simulated annealing method, the following parameter values are adopted: factor = 0.5, ε= 0.3, $\varepsilon_{min} = 0.1$ and $\varepsilon_{max} = 0.4$, T0 = 200, Δt = 0.02, CF=0.5.

4.2. Performance analysis

We present our results in terms of average decision time and average tasks' processing time. We start by studying the average tasks' processing time for each method. Thus, we carried an experiment where we vary the number of tasks parameter between 2 and 26 tasks.

4.2.1. The processing time

For each method, we start by studying the average tasks' processing time. Thus, we carried an experiment where we vary the number of tasks parameter between 2 and 26. The experiment's results are depicted in the following two figures. Figure 3 represents the obtained results for both based solution (BFS) and Simulated Annealing based solution (SAOD). It shows a small distance between the curves representing

the realized averaged tasks' processing time. Accordingly, the differences between the optimal BFS method and the SAOD method vary from 0.00% to 2.15%.

4.2.2. The average execution time

Now Figure 4 depicts the average of the execution time in ms to get the offloading decisions for both schemes. While the tasks count N is between 2 and 26, it clearly shows the exponential variation of the BFS execution time w.r.t. N. Additionally, the SAOD curve illustrates a slightly stable running time that starts from 0.2ms for N=2 and reaches 0.62ms for N=21.

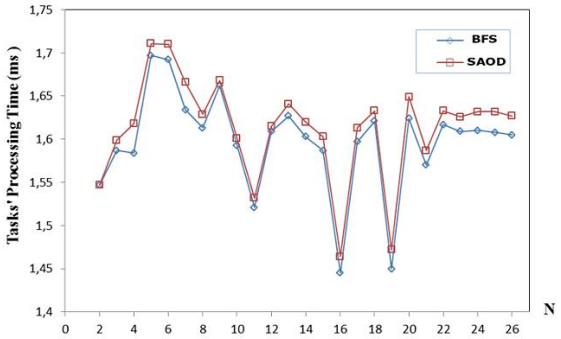

Figure 3. Tasks' processing time for N between 2 and 26

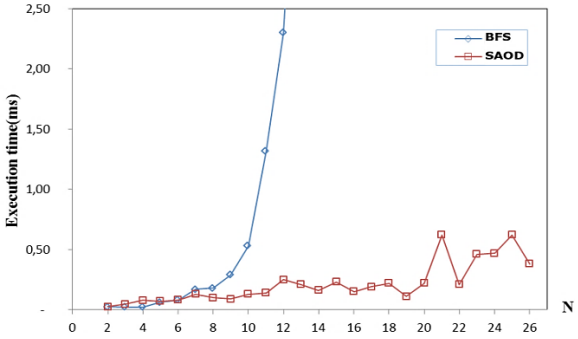

Figure 4. Execution time average for N between 2 and 20

4.3. Discusion

In view of the results presented before, our heuristic solutions SAOD give satisfactory results in terms of execution time. In addition, our solution gives an average tasks' processing time that is comparable and very close to the optimal solution with an almost linear execution time.

5. CONCLUSION

In this paper, we propose a simulated annealing based heuristic solution to solve a hard decision problem that jointly optimizes the processing time and computing resources for a smart mobile device within a mobile edge-computing node. The mobile device intends to optimally offload the content of a list of heavy tasks. Each task in its list is time-constrained with a proper deadline. The obtained results show the performance of the proposed simulated annealing based algorithm. By optimally adjusting the local and the remote computing frequencies, the proposed implementation shows the effectiveness of our solution. It brought a real processing time efficiency with almost a linear execution time that satisfies the decision time constraints in such edge systems. As a future work, we plan to generalize our study to the multi-user case while we introduce more relevant parameters, such as network state and wireless communication interference.

REFERENCES

[1] R. Chetan and R. Shahabadkar, "A comprehensive survey on exiting solution approaches towards security and privacy requirements of IoT," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 4, pp. 2319-2326, 2018.

[2] M. ETSI, "Mobile edge computing-introductory technical white paper," ETSI2014mobile, 2014. [Online]. Available: https://www.etsi.org/newsroom/news/

[3] 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.

[4] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," *in 2012 IEEE symposium on computers and communications (ISCC)*, pp. 000059-000066, 2012.

[5] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, pp. 1089-7801, 2014.

[6] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435-3447, 2016.

[7] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," *presented at the 2016 IEEE International Conference on Communications (ICC)*, 2016.
[8] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253-2266, 2014.
[9] V. Cardellini, *et al*., "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421-449, 2016.
[10] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974-983, 2014.
[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795-2808, 2016.
[12] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887-3901, 2016.
[13] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398-401, 2017.
[14] T. Francis and M. Madhiajagan, "A Comparison of Cloud Execution Mechanisms: Fog, Edge and Clone Cloud Computing," *Proceeding of the Electrical Engineering Computer Science and Informatics*, pp. 446-450, 2017.
[15] L. Pallavi, A. Jagan, and B. T. Rao, "ERMO² algorithm: an energy efficient mobility management in mobile cloud computing system for 5G heterogeneous networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 3, pp. 1957-1967, 2019.
[16] P. Prakash, K. Darshaun, P. Yaazhlene, M.V. Ganesh, and B. Vasudha, "Fog Computing: Issues, Challenges and Future Directions," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 6, pp. 3669-3673, 2017.
[17] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge Cloud Offloading Algorithms: Issues, Methods, and Perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1-23, Feb. 2019
[18] Y. Jararweh, M. Al-Ayyoub, M. Al-Quraan, A. T. Lo'ai, E. Benkhelifa, "Delay-aware power optimization model for mobile edge computing systems," *Personal and Ubiquitous Computing*, vol. 21, no. 6, pp. 1067-1077, 2017.
[19] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," *presented at the IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, 2017.
[20] H. Li, "Multi-task Offloading and Resource Allocation for Energy-Efficiency in Mobile Edge Computing," *International Journal of Computer Techniques*, vol. 5, no. 1, pp. 5-13, 2018.
[21] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," *presented at the Proceedings of the sixth conference on Computer systems*, 2011.
[22] 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.
[23] M. EL Ghmary, M. O. Cherkaoui Malki, Y. Hmimz, and T. Chanyour, "Energy and Computational Resources Optimization in a Mobile Edge Computing Node," in *2018 9th International Symposium on Signal, Image, Video and Communications (ISIVC)*, pp. 323-328, 2018.
[24] M. El Ghmary, T. Chanyour, Y. Hmimz, and M. O. Cherkaoui Malki, "Efficient multi-task offloading with energy and computational resources optimization in a mobile edge computing node," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 4908-4919, 2019.
[25] K. Zhang, *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE access*, vol. 4, pp. 5896-5907, 2016.
[26] Z. Fan, H. Shen, Y. Wu, and Y. Li, "Simulated-Annealing Load Balancing for Resource Allocation in Cloud Environments," *presented at the 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2013.
[27] L. Chen, J. Wu, X. Long, and Z. Zhang, "ENGINE: Cost Effective Offloading in Mobile Edge Computing with Fog-Cloud Cooperation," *arXiv preprint arXiv:1711.01683*, pp. 1-11, 2017.
[28] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing," *Future Generation Computer Systems*, vol. 64, pp. 1-14, 2016.
[29] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol 12, no. 5, pp. 726-738, 2018.