International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 6, December 2019, pp. 4908~4919
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i6.pp4908-4919

Efficient multi-task offloading with energy and computational resources optimization in a mobile edge computing node

Mohamed El Ghmary, Tarik Chanyour, Youssef Hmimz, Mohammed Ouçamah Cherkaoui Malki
Department of Computer Science, Sidi Mohamed Ben Abdellah University, Morocco

Article Info

Article history:

Received Jan 15, 2019
Revised Jun 2, 2019
Accepted Jun 26, 2019

Keywords:

Computation offloading
Energy optimization
Mobile edge computing
Simulated annealing

ABSTRACT

With the fifth-generation (5G) networks, Mobile edge computing (MEC) is a promising paradigm to provide near computing and storage capabilities to smart mobile devices. In addition, mobile devices are most of the time battery dependent and energy constrained while they are characterized by their limited processing and storage capacities. Accordingly, these devices must offload a part of their heavy tasks that require a lot of computation and are energy consuming. This choice remains the only option in some circumstances, especially when the battery drains off. Besides, the local CPU frequency allocated to processing has a huge impact on devices energy consumption. Additionally, when mobile devices handle many tasks, the decision of the part to offload becomes critical. Actually, we must consider the wireless network state, the available processing resources at both sides, and particularly the local available battery power. In this paper, we consider a single mobile device that is energy constrained and that retains a list of heavy offloadable tasks that are delay constrained. Therefore, we formulated the corresponding optimization problem, and proposed a Simulated Annealing based heuristic solution scheme. In order to evaluate our solution, we carried out a set of simulation experiments. Finally, the obtained results in terms of energy are very encouraging. Moreover, our solution performs the offloading decisions within an acceptable and feasible timeframes.

Corresponding Author:

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

1. INTRODUCTION

Mobile Edge Computing (MEC) concept [1-3], it has been recognized as the next generation computing infrastructure that is based on Mobile Cloud Computing paradigm [4-7]. 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 [8-10] some parts of their havy applications via wireless access to a resource-rich edge node, and then effectively reduces their power consumptions [11]. Moreover, to efficiently offload a greedy application while respecting deadlines, it is often decomposed into several independent offloadable tasks with a deadline constraint [12-14]. Many papers studied resource allocation within a MEC infrastructure to optimize the procecing time [15-18]. On the other hand, many state of the art works studied resource allocation within a MEC infrastructure to optimize the energy consumption [13, 19, 20]. In [21], the authors investigate a resource allocation policy to maximize the available processing capacity for MEC IoT networks with constrained power and unpredictable tasks.Unfortunatly, most of them consider users with a unique task only. However, current Smart Mobile Dvices (SMDs) can host several greedy applications that have to offload a part of their tasks to improve the quality of the experience or simply to avoid the waste of their available resources. Therefore,

the offloading decision should be generalized according to a multi-task scenario. This problem relies on the joint decision of tasks' offloading and the allocation of communication or computing resources.

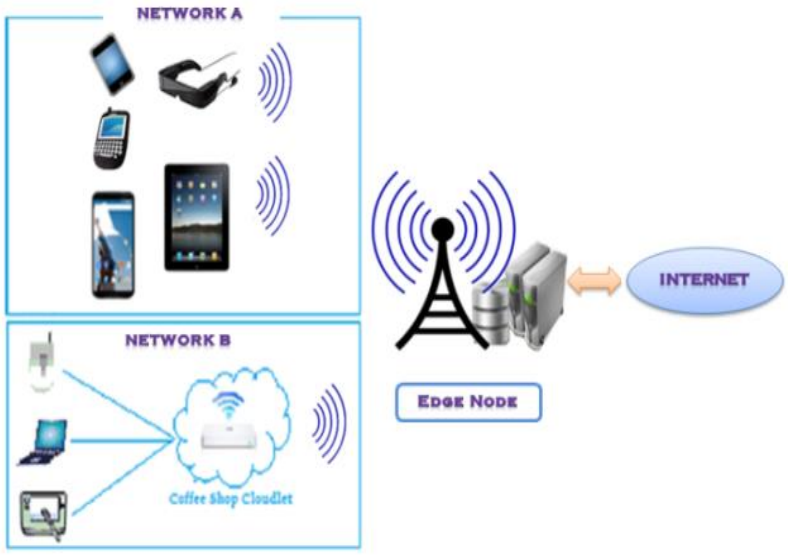

Figure 1. Mobile edge computing illustration

Recently, the authors of [13] 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 energy consumed 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 consumed energy via the offloading by considering the tasks' latency constraints and a threshold of available energy.

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 [22, 23]. 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.

Figure 2. System model illustration

For all these considerations, we derive a mathematical energy consumption 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 energy minimization problem.

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: $e_i^L = k_L . f_L^2 . \lambda_i$ [24]. Hence, the total energy consumption while executing all tasks that were decided to be locally executed in the SMD is given by

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

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 $\tau_i$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:

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

Similarly, energy consumption at the edge server while executing τ_i is given by: $e_i^S = k_S . f_S^2 . \lambda_i$ [8]. The execution energy for all the offloaded tasks is:

$$e^S = k_S . f_S^2 . \sum_{i=1}^{N} \lambda_i x_i \quad (5)$$

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 energy consumption for the SMD is composed of its local energy consumption, the communication energy as well as the execution energy at the EN, and it is given by $\mathbb{E}(\mathbb{X}, f_L, f_S) = e^L + e^C + e^S$. Then, according to Equations (2), (4) and (5) and if we note $\Lambda = \sum_{i=1}^{N} \lambda_i$, the total energy consumption can be formulated as:

$$\mathbb{E}(\mathbb{X}, f_L, f_S) = \left(k_S f_S^2 - k_L f_L^2 + \frac{p^I}{f_S}\right) \sum_{i=1}^{N} \lambda_i x_i + \frac{p^T}{r} \sum_{i=1}^{N} d_i x_i + k_L f_L^2 \Lambda \quad (6)$$

2.2. Problem formulation

In this section, we present our optimization problem formulation that aims to minimize the overall energy consumption in the local execution or 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:

$$\boldsymbol{\mathcal{P}1}: \min_{\{x, f_L, f_S\}} \left\{ \left(k_S f_S^2 - k_L f_L^2 + \frac{p^I}{f_S}\right) \sum_{i=1}^{N} \lambda_i x_i + \frac{p^T}{r} \sum_{i=1}^{N} d_i x_i + k_L f_L^2 \Lambda \right\}$$

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

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

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

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

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

$$(C_{1.6}) \quad e^L = k_L . f_L^2 . \sum_{i=1}^{N} \lambda_i (1 - x_i) \le E^{max}.$$

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 $(C_{1.6})$ can be reformulated as $f_L \leq \sqrt{\frac{E^{max}}{k_L(\Lambda_N - \Lambda_N^1)}}$. For ease of use, let note:

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

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

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

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

$$\boldsymbol{\mathcal{P}2}(\mathbb{X}): \ \min_{\{f_L, f_S\}} \left\{(\Lambda_N - \Lambda_N^1)k_L f_L^2 + \Lambda_N k_S f_S^2 + \Lambda_N \frac{p^I}{f_S} + D_N^1 \frac{p^T}{r}\right\}$$
$$\text{s.t.}(C_{2.1}) \qquad F_L^{min} \leq f_L \leq F_L^{max};$$
$$(C_{2.2}) \qquad f_L^- \leq f_L \ ;$$
$$(C_{2.3}) \qquad f_S^- \leq f_S \leq F_S \ ;$$
$$(C_{2.4}) \qquad k_L f_L^2 (\Lambda_N - \Lambda_N^1) \leq E^{max}.$$

Considering the continuous variables f_L and f_S, problem P2 is a continuous multi-variable optimization problem. The objective function $\mathbb{E}_{\mathbb{X}}(f_L, f_S) = (\Lambda_N - \Lambda_N^1)k_L f_L^2 + \Lambda_N^1 k_S f_S^2 + \Lambda_N^1 \frac{p^I}{f_S} + D_N^1 \frac{p^T}{r}$ can be decomposed into the following two independent functions $\mathbb{E}_1(f_L)$ and $\mathbb{E}_2(f_S)$ where $\mathbb{E}_1(f_L) = (\Lambda_N - \Lambda_N^1)k_L f_L^2$ and $\mathbb{E}_2(f_S) = \Lambda_N^1 k_S f_S^2 + \Lambda_N^1 \frac{p^I}{f_S} + D_N^1 \frac{p^T}{r}$. 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.

$$\boldsymbol{\mathcal{P}3.1}(\mathbb{X}): \ \min_{\{f_L\}} \{\mathbb{E}_1(f_L) = (\Lambda_N - \Lambda_N^1)k_L f_L^2\}$$
$$\text{s.t.}(C_{3.1.1}) \qquad F_L^{min} \leq f_L \leq F_L^{max};$$
$$(C_{3.1.2}) \qquad f_L^- \leq f_L \leq f_L^+.$$
$$\boldsymbol{\mathcal{P}3.2}(\mathbb{X}): \ \min_{\{f_S\}} \left\{\mathbb{E}_2(f_S) = \Lambda_1 k_S f_S^2 + \Lambda_N^1 \frac{p^I}{f_S} + D_N^1 \frac{p^T}{r}\right\}$$
$$\text{s.t.}(C_{3.2.1}) \qquad f_S^- \leq f_S \leq F_S.$$

3.2. Problems resolution

For the $\mathcal{P}3.1$ problem, the objective function $\mathbb{E}_1(f_L)$ 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 $(C_{3.1.1})$, we can derive the following function's optimum f_L^* given by:

$$f_L^* = \begin{cases} 0 & \text{if } \mathbb{X} = \mathbb{X}_1 \\ \emptyset & \text{if } f_L^- > F_L^{max} \text{or } f_L^+ < F_L^{min} \text{ or } f_L^- > f_L^+ \\ F_L^{min} & \text{if } f_L^- < F_L^{min} \\ f_L^- & \text{otherwise} \end{cases} \quad (12)$$

For the $\mathcal{P}3.2$ problem, the objective function $\mathbb{E}_2(f_S)$ is a continuous function according to its variable $f_S$with a first order derivate: $\frac{\partial \mathbb{E}_2(f_S)}{\partial f_S} = 2\Lambda_N^1 k_S f_S - \frac{\Lambda_N^1 p^I}{f_S^2}$; consequently, $\mathbb{E}_2(f_S)$ decreases on $\left]0, \sqrt[3]{\frac{p^I}{2k_S}}\right]$ and increases on $\left[\sqrt[3]{\frac{p^I}{2k_S}}, +\infty\right[$. Then, $\mathbb{E}_2(f_S)$ has an optimal minimum value at the point $\sqrt[3]{\frac{p^I}{2k_S}}$ without considering constraint $(C_{3.2.1})$. Therefore, with $(C_{3.2.1})$, we can derive the following function's optimum f_S^* given by:

$$f_S^* = \begin{cases} \emptyset & \text{if } \min_i \left\{t_i^{max} - \frac{D_i^1}{r}\right\} \le 0 \text{ or } f_S^- > F_S \\ F_S & \text{if } \frac{p^I}{2k_S} \ge {F_S}^3 \\ f_S^- & \text{if } \frac{p^I}{2k_S} \le (f_S^-)^3 \\ \sqrt[3]{\frac{p^I}{2k_S}} & \text{otherwise} \end{cases} \quad (13)$$

3.2.1. Processing frequencies determination

From the above results, with a given offloading decision vector $\mathbb{X}$, we present the next Algorithm 1 that gives the optimal allocated local frequency $f_L$as well as the remote edge server's processing frequency f_S.

3.2.2. The energy consumption determination

Similarly, given an offloading decision vector $\mathbb{X}$ the next algorithm 2 uses the first algorithm to determine the minimal energy consumption:

Algorithm 1: frequencies optimum for a given $\mathbb{X}$

Input: The offloading policy $\mathbb{X}$.
Output: $f_L$and f_S.
1: Determinate $\mathbb{X}_1$ according to (7);
2: if $\mathbb{X} = \mathbb{X}_1$**then**
3: $f_L = 0$;
4: **goto** 16;
5: end if
6: Calculate: f_L^-, f_L^+ according to (9) and (10) respectively;
7: if $f_L^- > F_L^{max}$**or** $f_L^+ < F_L^{min}$ **or** $f_L^- > f_L^+$**then**
8: **return ∅**;
9: else
10: **if** $f_L^- < F_L^{min}$**then**
11: $f_L = F_L^{min}$;
12: **else**
13: $f_L = f_L^-$;
14: **end if**
15: end if
16: if $\min_i \left\{t_i^{max} - \frac{D_i^1}{r}\right\} \le 0$ **then**
17: **return ∅**;
18: else
19: Calculate: f_S^- according to (11);
20: **if** $f_S^- > F_S$ **then**
21: **return ∅**;
22: **else if** $\frac{p^I}{2k_S} \ge {F_S}^3$ **then**
23: $f_S = F_S$;
24: **else if** $\frac{p^I}{2k_S} \le (f_S^-)^3$**then**
25: $f_S = f_S^-$;
26: **else**

27: $f_S = \sqrt[3]{\frac{p^I}{2k_S}}$;
28: **end if**
29: **end if**
30: **end if**
31: **end if**
32: **return** (f_L, f_S)**;**

Algorithm 2: Energy calculation
Input: The list τ of N sub-tasks, offloading policy $\mathbb{X}$.
Output:$\mathbb{E}(\mathbb{X}, f_L, f_S)$.
1: Call **Algorithm 1** to calculate (f_L, f_S) using $\mathbb{X}$;
2: if $f_L = \emptyset$ **or** $f_S = \emptyset$ **then**
3: **return** ∞;
4: else
5: Calculate $\mathbb{E}(\mathbb{X}, f_L, f_S)$according to (6);
6: **return** $\mathbb{E}(\mathbb{X}, f_L, f_S)$;
7: end if

3.3. Proposed solutions

Next, the problem relies on determining the optimal offloading decision vector $\mathbb{X}$ that gives the optimal energy consumption. 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 energy consumption as well as its completion time. Now, the next algorithm summarizes the Brute Force Search Solution.

Algorithm 3 : Brute Force Search Offloading
Input: The list τ of N sub-tasks;
Output: the offloading policy $\mathbb{X}^*$.
Initialize: minEnergy=∞**;**
1: for i=1 to $2^N - 1$ **do**
2: Use the N bits representation of integer i to build the policy $\mathbb{X}$**;**
3: Call **Algorithm 2** to get newEnergy using τ and $\mathbb{X}$;
4: **if newEnergy** $<$ **minEnergy then**
5: **minEnergy** $\leftarrow$ **newEnergy ;**
6: $\mathbb{X}^* \leftarrow \mathbb{X}$;
7: **end if**
8: end for
9: return $\mathbb{X}^*$ **;**

3.3.2. Simulated annealing offloading based on workload density threshold

For the second solution, we propose the use of a Simulated Annealing (SA) based method. The SA technique was adopted as a heuristic solution in the optimization field especially for hard problems. To improve a solution, it employs iterative random solution variation. Interested readers can refer to the following works [26] and [27] for more details about this issue. Some references dealing with the offloading in cloud environments [19, 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 energy is the one given by the smallest cycles' count. Besides, with almost the same cycles' count, the one that consumes less offloading energy 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 Workload Density based Simulated Annealing Offloading (WDSAO), we adopted the following general threshold probability:

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

where T_0 is the initial temperature constant. ΔE_i is the solutions' energy 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 energy performance are accepted in line with a certain probability threshold. Accordingly, Algorithm summarizes our heuristic solution.

Algorithm 4 takes as input: the sub-tasks' list τ, the initial temperature T_0, the cooling factor CF, the temperature treshold ε, and the workload density threshold ω_T.

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

3.3.3. Original simulated annealing offloading

For the third solution and for comparison purpose, we take the version of the solution proposed by [5] and denote it Original Simulated Annealing Offloading (OSAO). In this solution, the local execution probability increases with the increase of the computing density. This fact leads to offload tasks with big data size and workload and prevent tasks with low data size and high workload to take high offload priority.

Algorithm 4: workload density based Simulated Annealing Offloading

Input: The list τ of N sub-tasks, T_0, CF, ε, ω_T;
Output: the offloading policy $\mathbb{X}^*$.
Initialize: a random policy $\mathbb{X}$;

```
Call Algorithm 2 to calculate oldEnergy using τ and 𝕏;
minEnergy=∞;
while T0> ε do
  for each i in τ do
    if ω_i > ω_T then
      if task i not in 𝕏_1 then
        add i to 𝕏_1 ;
      end if
    else if ω_T − ω_i > (ω_T − ω_min) ∗ random(0,1) then
      if task i in 𝕏_1 then
        move i from 𝕏_1 to 𝕏_0 ;
      end if
    else if task i in 𝕏_0 then
        move i from 𝕏_0 to 𝕏_1 ;
      end if
    end if
  end if
  Update 𝕏 using the new 𝕏_1;
  Call Algorithm 2 to get newEnergy using τ and 𝕏;
  if newEnergy≠ ∞ then
    ΔE_i = newEnergy − oldEnergy
    if ΔE_i < 0 then
      oldEnergy=newEnergy;
      if newEnergy<minEnergy then
        minEnergy =newEnergy ;
        𝕏* = 𝕏 ;
      end if
    else
      Calculate p according to (13);
      if e^(−ΔE_i/T_0) > random(0,1) then
```

```
31:                         oldEnergy = newEnergy;
32:                     else
33:                         Put back i to its original set;
34:                     end if
35:                 end if
36:             end if
37:         end for
38:         T0= T0*CF
39: end wile
40: return X* ;
```

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. Additionally, they are run on a laptop equipped with a 2.4 GHz Intel Core i5 processor and 8 GB of RAM. 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 $F_L^{min} = 1MHz$ and $F_L^{max} = 60MHz$. The CPU frequency of the remote edge server node will be optimized under the value $F_S = 6GHz$. 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 methods, the following parameter values are adopted: factor = 0.5, ε= 0.3, T_0 = 200, Δt = 0.02(in OSAO), and CF=0.85.

4.2. Performance analysis

We present our results in terms of average decision time and average energy consumption. We start by studying the average energy's consumption throughput for each method. Thus, we carried an experiment where we vary the number of tasks parameter between 2 and 50 tasks.

4.2.1. The parameter ω_T

The Figure 3 shows a rapid decrease of the energy consumption using the WDSAO method for ω_T between 0.3 and 0.45 for N in {10,15,20,25,30}. Then, this energy increases from ω_T = 0.5 to ω_T = 0.75 for all values of N. In addition it slightly decreases after ω_T = 0.75 only for N = 10 and N = 30. As a result, we find that the best value of ω_T that minimizes the energy consumption for most of the values of N is ω_T = 0.5. Thereafter, we will set ω_T to the value 0.5.

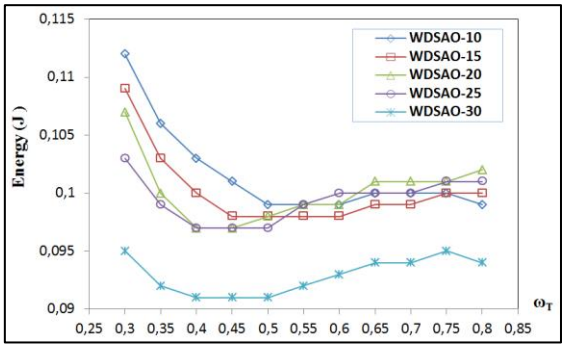

Figure 3. Average Energy consumption for ω_T between 0.25 and 0.85

4.2.2. The energy consumption

In terms of energy consumption, the experiment's results are depicted in the following two figures. Figure 4 represents the obtained results for the three methods where N is taken between 3 and 25. On the one hand, it shows a small distance between the results of the optimal BFS method and the OSAO method. This difference varies from 1.53% to 9.30%. On the other hand, the WDSAO results are almost the same as the optimal results. The difference varies from 0.00% to 2.88%.

Beyond the value N=25, and because of the considerable processing time of the BFS solution, we compared only the OSAO and the WDSAO methods. Figure 5 shows that the results of the WDSAO solution are better than those of OSAO for all N values. The results of the first represent a gain in energy consumption that varies between 5.55% and 8.33%.

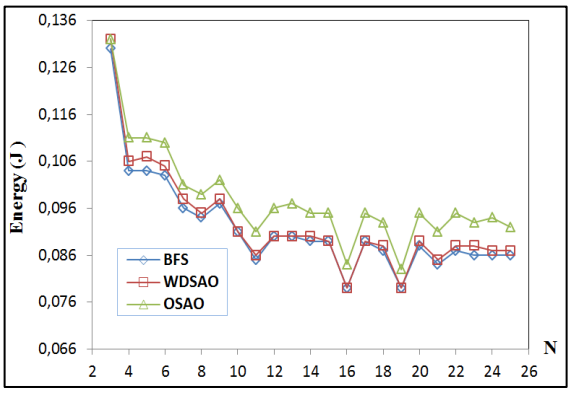

Figure 4. Average energy consumption for N between 3 and 25

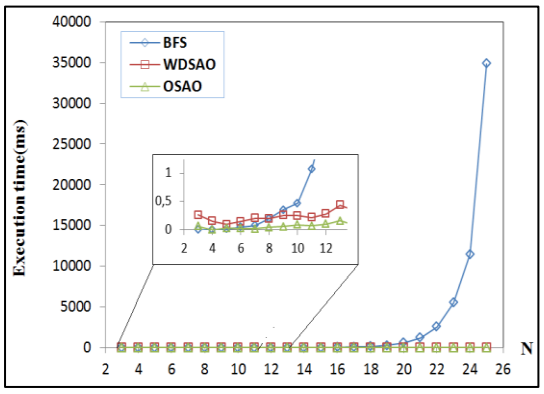

Figure 5. Average energy consumption for N between 26 and 50

4.2.3. The Average Execution Time

Now, we consider the average execution time in both Figure 6 and Figure 7. The first one illustrates the execution time comparison for all the three methods while N is between 3 and 25. It clearly shows the exponential variation of the BFS solution time with the N parameter. Additionally, The OSAO and WDSAO solutions give a stable execution time that reached for N=25 respectively 0.27ms and 0.91ms for both methods.

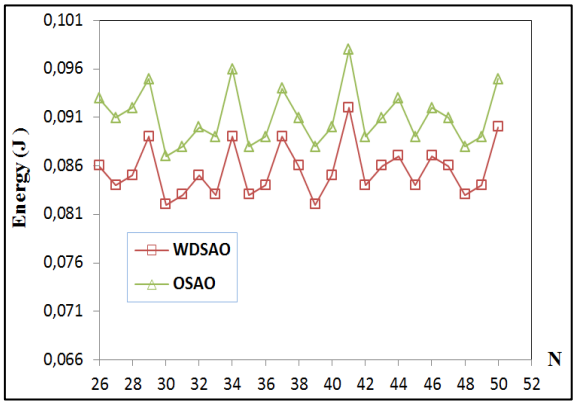

Figure 6. Executuion time average for N between 3 and 25

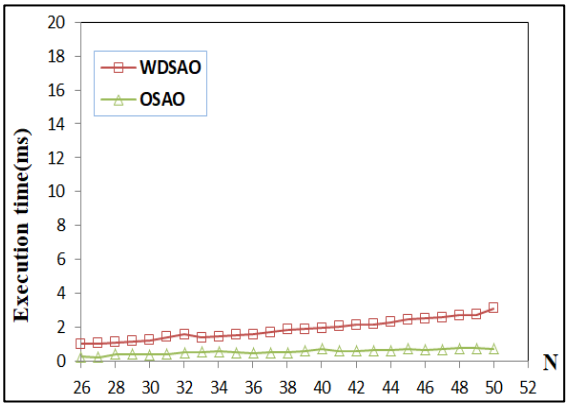

Figure 7. Executuion time average for N between 26 and 50

The second Figure 7 illustrates the execution time comparison for OSAO and WDSAO methods while N is between 26 and 50. The OSAO curve illustrates a stable running time that starts from 0.28 ms for N=26 and reaches 0.75 ms for N=50. On the other hand, the WDSAO curve illustrates a near linear running time that starts from 1.01 ms for N=26 and reaches 3.11 ms for N=50. Accordingly, the performances in terms of the execution time of the OSAO method are slightly higher than those of the WDSAO method. Nevertheless, both solutions' performances are always very very high and more acceptable compared to the FBS exact solution. This last reaches an execution time of 34862.35 ms for N=25 only, which is inappropriate for the context of this work.

4.3. Discussion

In view of the experiments' results, in the first experiment we study the effect of the ω_T parameter for our WDSAO solution. Thus, this study revealed that a value for the threshold ω_T=0.5 is beneficial for the energetic performance without any effect on the execution time. On the other hand, a second series of experiments revealed thet the OSAO and WDSAO heuristic solutions give satisfactory results in terms of execution time compared to the BFS exact solution which is with exponential time complexity. In addition, our WDSAO developed solution gives an energy consumption that is comparable and very close to the BFS exact solution with almost a linear execution time. Moreover, compared to the OSAO method, eventhough the execution time of our solution is slightly greater, its energetic performace are very close to the exact solution.

5. CONCLUSION

In this paper, we propose a simulated annealing based heuristic to solve a hard decision problem that jointly optimizes energy and computational 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 as much as possible where each task is time-constrained with a proper deadline t_i^{max}. 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 energy efficiency as well as near linear execution time that satisfies the decision's 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] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, pp. 398-401, 2017.

[2] 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*, vol. 8, pp. 4646-4653, 2018.

[3] H. Chang, *et al*., "Bringing the cloud to the edge," *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 346-351, 2014.

[4] L. Pallavi, *et al*., "ERMO2 algorithm: an energy efficient mobility management in mobile cloud computing system for 5G heterogeneous networks," *International Journal of Electrical and Computer Engineering*, vol. 9, pp. 1957-1967, 2019.

[5] H. Mora, *et al*., "Multilayer Architecture Model for Mobile Cloud Computing Paradigm," *Complexity*, vol. 2019, 2019.

[6] N. Fernando, *et al*., "Mobile cloud computing: A survey," *Future generation computer systems*, vol. 29, pp. 84-106, 2013.

[7] H. M. Mora, *et al*., "Flexible framework for real-time embedded systems based on mobile cloud computing paradigm," *Mobile information systems*, vol. 2015, 2015.

[8] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 1628-1656, 2017.

[9] P. Prakash, *et al*., "Fog Computing: Issues, Challenges and Future Directions," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 3669, 2017.

[10] J. Wang, *et al*., "Edge Cloud Offloading Algorithms: Issues, Methods, and Perspectives," *ACM Computing Surveys*, vol. 52, pp. 1-23, 2019.

[11] Y. Jararweh, *et al*., "Delay-aware power optimization model for mobile edge computing systems," *Personal and Ubiquitous Computing*, vol. 21, pp. 1067-1077, 2017.

[12] M. H. Chen, *et al*., "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.

[13] H. Li, "Multi-task Offloading and Resource Allocation for Energy-Efficiency in Mobile Edge Computing," vol. 5, pp. 5-13, 2018.

[14] J. Liu, *et al*., "Delay-optimal computation task scheduling for mobile-edge computing systems," *presented at the 2016 IEEE International Symposium on Information Theory (ISIT)*, 2016.

[15] Y. Wu, *et al*., "Delay-Minimization Nonorthogonal Multiple Access enabled Multi-User Mobile Edge Computation Offloading," *IEEE Journal of Selected Topics in Signal Processing*, 2019.

[16] Y. Wang, *et al*., "Cooperative Task Offloading in Three-Tier Mobile Computing Networks: An ADMM Framework," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2763-2776, 2019.

[17] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Communications Letters*, vol. 21, pp. 1481-1484, 2017.
[18] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, vol. 27, pp. 85-97, 2019.
[19] M. H. Chen, *et al*., "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," *presented at the IEEE International Conference on Communications (ICC)*, 2016.
[20] L. Huang, *et al*., "Multi-Server Multi-User Multi-Task Computation Offloading for Mobile Edge Computing Networks," *Sensors*, vol. 19, pp. 1446, 2019.
[21] M. Qin, *et al*., "Power-Constrained Edge Computing with Maximum Processing Capacity for IoT Networks," *IEEE Internet of Things Journal*, 2018.
[22] B. G. Chun, *et al*., "Clonecloud: elastic execution between mobile device and cloud," *presented at the Proceedings of the sixth conference on Computer systems*, 2011.
[23] Y. Mao, *et al*., "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 3590-3605, 2016.
[24] X. Chen, *et al*., "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 2795-2808, 2016.
[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, *et al*., "Simulated-annealing load balancing for resource allocation in cloud environments," *presented at the International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2013.
[27] L. Chen, *et al*., "ENGINE: Cost Effective Offloading in Mobile Edge Computing with Fog-Cloud Cooperation," arXiv preprint arXiv:1711.01683, pp. 1-11, 2017.
[28] K. Liu, *et al*., "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, *et al*., "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, 2018.