Offloading Decisions in a Mobile Edge Computing Node with Time and Energy Constraints

Mohamed EL GHMARY[1*], Tarik CHANYOUR[1], Youssef HMIMZ[1] and Mohammed Ouçamah CHERKAOUI MALKI[1]

[1]Sidi Mohamed Ben Abdellah University, LIIAN Labo, FSDM, Fez, Morocco.

Abstract: This article describes a simulated annealing based offloading decision with processing time, energy consumption and resource constraints in a Mobile Edge Computing Node. Edge computing mostly deals with mobile devices subject to constraints. Especially because of their limited processing capacity and the availability of their battery, these devices have to offload some of their heavy tasks, which require many calculations. We consider a single mobile device with a list of heavy tasks that can be offloadable. The formulated optimization problem takes into account both the dedicated energy capacity and the total execution time. We proposed a heuristic solution schema. To evaluate our solution, we performed a set of simulation experiments. The results obtained in terms of processing time and energy consumption are very encouraging.

Keywords: Mobile Edge Computing; Computation Offloading; Processing Time Optimization; Simulated Annealing.

1. Introduction

Recently , the development of the Internet of Things (IoT) has been largely supported by the parallel development of cloud computing capabilities. Cloud computing provides on-demand resilient computing power that can host and support critical IoT functions. However, the proliferation of connected objects poses the problem of the confidentiality of the data produced. This proliferation also threatens the performance of support networks. Due to restricted battery power, memory and computational capacity, mobile devices face challenges in executing delay-sensitive and resource-hungry mobile applications such as augmented reality and online gaming. Mobile Edge Computing (MEC) is foreseen as a remedy to alleviate this problem. In MEC, the mobile edge is enhanced with analysis and storage capabilities, possibly by a dense deployment of computational servers or by strengthening the already-deployed edge entities such as small cell base stations. Consequently, mobile devices are able to offload their computationally expensive tasks to the edge servers while requesting some specific quality of service. This process, referred to as computation offloading, is feasible due to the fact that edge servers are deployed in close proximity of mobile users, specifically in comparison to the remote cloud servers.

Mobile Edge Computing (MEC) is a new technology, providing an IT service environment and cloud computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers. The goal is to reduce latency, ensure network operation and delivery of highly efficient services, and improve the users experience. The environment of Mobile Edge Computing is characterized by low latency, proximity, high bandwidth, and real-time insight into radio network information and location awareness. MEC is a natural development of the evolution of mobile base stations and the convergence of computer and telecommunication networks. Based on a virtualized platform, MEC is recognized by the European 5G PPP (Public-Private Partnership for Infrastructure 5G) as one of the key emerging technologies for 5G networks [1]. As illustrated in Figure 1, the authors of [2-4] studied that MEC can increase the capabilities of mobile devices by offloading some of their tedious applications via wireless access to a resource-rich edge node, in order to effectively reduce their energy consumption [2]. In [5], the offloading decisions are adjusted to minimize the overall energy consumption. Similar to [6], to save energy consumption, an optimization problem which decides the offloading policy is studied. Often, a greedy application is broken down into several independent tasks with a time constraint in order to offload them efficiently and quickly [3, 7, 8]. Many papers studied resource allocation within a MEC infrastructure to optimize the processing time [9-12]. On the other hand, Many states of the artworks studied resource allocation within a MEC infrastructure to optimize the energy consumption [7, 13]. The current global trend of IoT evolves very quickly from user needs, the corresponding technologies and protocols, such as massive connectivity, energy constraints, scalability and reliability limitations and security, remain open research questions [14, 15]. In [16], the authors investigate a resource allocation policy to maximize the available processing capacity for MEC IoT networks with constrained power and unpredictable tasks. Unfortunately, most of them consider users with a unique task only. However, current Smart Mobile Devices (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.

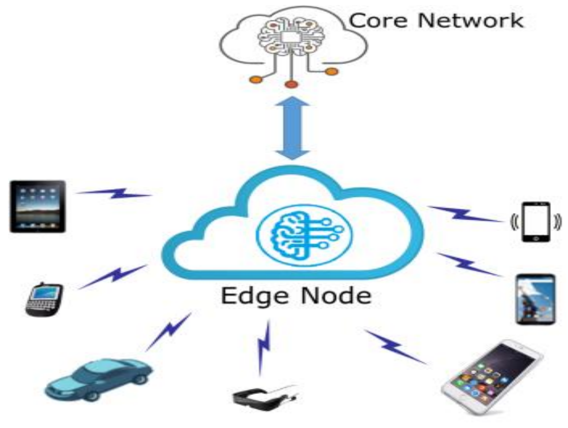

Figure 1. Mobile edge computing illustration

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

In this section, we present the adopted system model. First, we briefly present smart mobile devices parameters. Then, we describe in details our optimization 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 [17, 18]. 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.

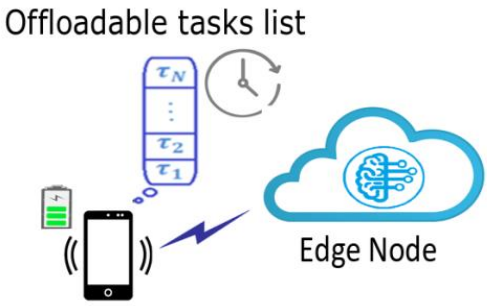

Figure 2. System model illustration

For all these considerations, we derive a mathematical 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.

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:

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

Additionally, the corresponding energy consumption is given by: $e_i^L = k_L . f_L^2 . \lambda_i$ [19]. Hence, the total energy consumption while executing all tasks that were decided to be locally executed in the SMD is given by :

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

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 [20]. 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:

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

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:

$$e^C = \frac{p^T \sum_{i=1}^{N} x_i d_i}{r} \tag{4}$$

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:

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

Then, according to equations **(1)** and **(3)**, the total execution time can be formulated as:

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

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:

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

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

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

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

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

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

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

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. Hence, we begin by the problem's decomposition in Section 3.1. In Section 3.2 we present the problems' resolution in order to get the optimal solutions. After that, we present the proposed solutions.

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:

$$\mathbb{X}_1 = \{ i \in \mathbb{X} \ / \ x_i = 1 \} \tag{7}$$

$$\mathbb{X}_0 = \{ i \in \mathbb{X} \ / \ x_i = 0 \} \tag{8}$$

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 $\frac{\Lambda_i - \Lambda_i^1}{t_i^{max}} \le f_L;\ \forall\, i \in [\![1;N]\!]$. Finally, it is equivalent to one constraint: $\max_i \left\{ \frac{\Lambda_i - \Lambda_i^1}{t_i^{max}} \right\} \le f_L$. Likewise, constraint $(C_{1.5})$ for an offloadable task means $\frac{D_i^1}{r} + \frac{\Lambda_i^1}{f_S} \le t_i^{max}$ $(\forall\, i \in [\![1;N]\!])$. 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 $\frac{\Lambda_i^1}{t_i^{max} - \frac{D_i^1}{r}} \le f_S;\ \forall\, i \in [\![1;N]\!]$. Finally, it is equivalent to one constraint: $\max_i \left\{ \frac{\Lambda_i^1}{t_i^{max} - \frac{D_i^1}{r}} \right\} \le f_S$.

Similarly, constraint $(C_{1.6})$ means $k_L . f_L^2 . (\Lambda_N - \Lambda_N^1) + \frac{p^T D_N^1}{r} \le E^{max}$. 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 $f_L \le \sqrt{\frac{E^{max} - \frac{p^T D_N^1}{r}}{k_L(\Lambda_N - \Lambda_N^1)}}$. For ease of use, let note:

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

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

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

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

$$\boldsymbol{\mathcal{P}2}(\mathbb{X}): \min_{\{f_L, f_S\}} \left\{ \frac{\Lambda_N - \Lambda_N^1}{f_L} + \frac{D_N^1}{r} + \frac{\Lambda_N^1}{f_S} \right\}$$

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

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

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

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

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.

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

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

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

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

s.t. $(C_{3.2.1})$ $f_S^- \leq f_S \leq F_S$.

3.2. Problems Resolution

The objective function $T_1(f_L)$ of the problem $\mathcal{P}3.1$ is a strictly increasing continuous function according to its variable f_L. On the other hand, the objective function $T_2(f_S)$ of the problem $\mathcal{P}3.2$, decreases strictly w.r.t. the variable f_S.

3.2.1. Processing Frequencies Determination

From an offloading decision vector $\mathbb{X}$, and taking into account the constraints obtained $(C_{3.1.1})$, $(C_{3.1.2})$ and $(C_{3.2.1})$, the following algorithm gives the optimal allocated local frequency f_L as well as the remote edge server's processing frequency f_S.

```
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 (8);
 2: if $\mathbb{X} = \mathbb{X}_1$ then
 3:     $f_L = 0$;
 4:     goto 12;
 5: end if
 6: Calculate: $f_L^-, f_L^+$ according to (9) and (10)
        respectively;
 7: if $E^{max} \leq \frac{p^T D_N^1}{r}$ or $f_L^- > F_L^{max}$ or $f_L^+ < F_L^{min}$ or
        $f_L^- > f_L^+$ then return $\emptyset$;
 8: else if $f_L^+ > F_L^{min}$ then $f_L = F_L^{max}$;
 9:     else $f_L = f_L^+$;
10:     end if
11: end if
12: if $\min_i \left\{t_i^{max} - \frac{D_i^1}{r}\right\} \leq 0$ then return $\emptyset$;
13: else
14:     Calculate: $f_S^-$ according to (11);
15:     if $f_S^- > F_S$ then return $\emptyset$;
16:     else $f_S = F_S$;
17:     end if
18: end if
19: return $(f_L, f_S)$;
```

3.2.2 The Processing Time Determination

To evaluate the processing time of a state, the following expression is adopted. It gives the minimal processing time by calling the first algorithm to calculate (f_L, f_S) if the problem is feasible. Otherwise, it ∞.

$$T(\mathbb{X}, f_L, f_S) = \begin{cases} \infty & \text{if } f_L = \emptyset \text{ or } f_S = \emptyset \\ T(\mathbb{X}, f_L, f_S) \text{ according to (12)} & \text{otherwise} \end{cases} \quad (12)$$

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 Simulated Annealing Offloading Solution

The following algorithm presents the pseudo-code of the simulated annealing based heuristic as described above.

```
Algorithm 3: Simulated Annealing Pseudo-code
Input: The list $\tau$ of N sub-tasks.
Output: the offloading policy $\mathbb{X}^*$.
Initialize: a random policy $\mathbb{X}$.
 1: Call Algorithm 2 to calculate $T$ using $\tau$ and $\mathbb{X}$;
 2: For k = 0 to kmax do
 3:     Temp ← temperature(k/kmax)
 4:     Pick a random neighbour, $\mathbb{X}_{new}$ ← neighbour($\mathbb{X}$)
 5:     Call Algorithm 2 to calculate $Te_{new}$ using $\tau$ and
          $\mathbb{X}_{new}$;
 6:     if $T_{new} \neq \infty$ then
 7:         if P($T$, $T_{new}$, Temp) ≥ random(0, 1) then
 8:             $\mathbb{X} \leftarrow \mathbb{X}_{new}$ ;
 9:             $T \leftarrow T_{new}$;
10:         End if
11:     end if
12: end for
13: $\mathbb{X}^* \leftarrow \mathbb{X}$ ;
```

With, ***random (0,1)*** is a function that allows to generate a random number in [0,1]. ***neighbor (***$\mathbb{X}$***)*** is a function to generate a decision vector state near the input $\mathbb{X}$. ***P(T, Tnew, Temp)*** is an acceptance probability function that depends on processing time ***T*** and T_{new} of the two states $\mathbb{X}$ and $\mathbb{X}_{new}$, and on a global Temp parameter called the temperature that varies over the iterations.

In our proposed first solution, which we denote Original Simulated Annealing Offloading Decision (SAOD), among the main parameters introduced in this solution, we present the computation density proportion variable for each task i given by:

$$\chi_i = \frac{\omega_i - \omega_{min}}{\omega_{max} - \omega_{min}} \quad (13)$$

Where $\omega_i = \frac{d_i}{\lambda_i}$ is the i task computation density.

In our proposed second solution, which we denote Modified Simulated Annealing Offloading Decision (MSAOD), instead of the threshold probability $p_0 = \frac{1}{(1+e^{-\Delta T/Temp_0})}$ used in SAOD. Since p_0 belong to $\left[0;\frac{1}{2}\right]$, we adopted the following probability:

$$p = \frac{3}{2} - p_0 \, . \quad (14)$$

Where p belong to $\left[\frac{1}{2};1\right]$, and $Temp_0$ is the initial temperature constant. ΔT_i is the solutions' processing time variation while changing the task i state.

Moreover, the ε parameter introduced in SAOD, which represents a tolerance parameter, is critic (as other parameters) for the proposed heuristic. Thus, in our final solution MSAOD we choose to regularly vary this parameter between ω_{min} and ω_{max}. Then we select the best solution.

4. Results and Discussion

In this section, we carried out a serie of experiments to evaluate the performance of our proposed solution. First, we present simulation setup parameters. Then, several performance analysis are detailed to prove the efficiency of our approach.

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 $F_L^{min} = 1MHz$ 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 transmission power is set to be $p^T = 0.1$ Watt. 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 = 0.75, ε= 0.3, $\varepsilon_{min} = 0.1$ and $\varepsilon_{max} = 0.4$, Temp₀ = 200, Δt = 0.02(in SAOD), 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 50 tasks.

4.2.1 Factor Parameter

The following figure shows a rapid decrease of the task processing time of the MSAOD method for a factor between 0.4 and 0.9 for N in {10,15,20,25}. Then, the processing time remains almost stable for all values of N.

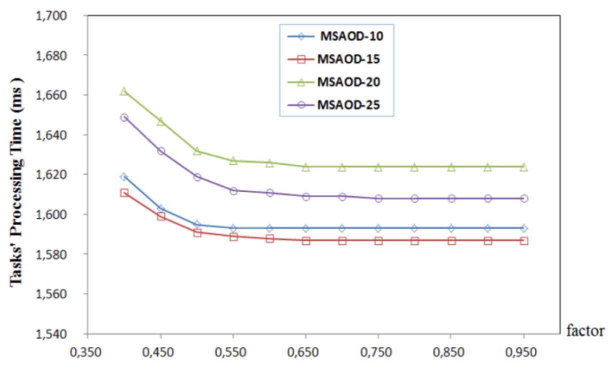

Figure 3. Tasks' Processing Time for factor between 0.4 and 0.95

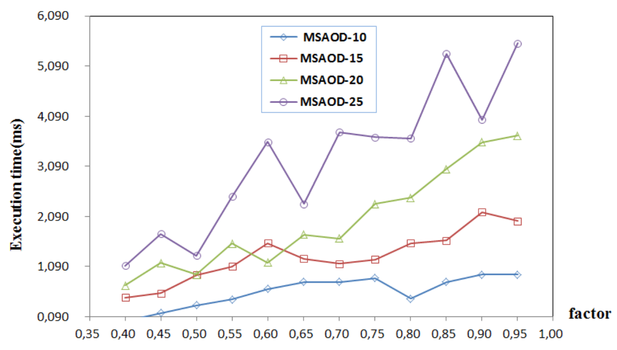

Figure 4. Execution Time Average for factor between 0.4 and 0.9

The figure 4 illustrates the running time for N in{10,15,20,25} w.r.t. the factor value. Therefore, we find that the best factor value that minimizes the task processing time for most N values is factor = 0.75. Subsequently, we will set the factor to 0.75.

4.2.2 The Processing Time

As the performance study of the proposed solution in [21] with three situations (in MEC context), Figure 5 and Figure 6 Show a comparison between three situations (with N between 2 and 50): 1) All tasks are offloaded to the Edge Node. 2) The offloading decision is done using our MSAOD solution. 3) All tasks are executed locally.

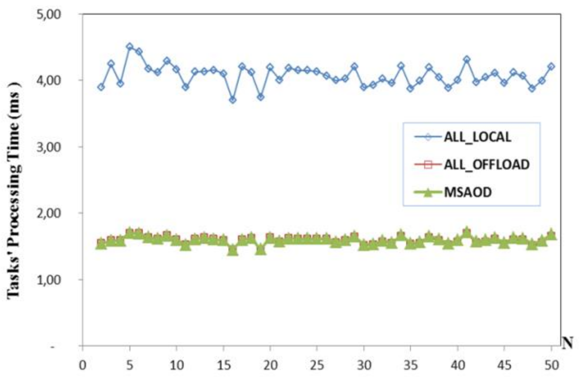

Figure 5. Tasks' Processing Time for N between 2 and 50.

From this figure, our proposed solution saves between 148% and 168% of processing time compared to the local execution of all tasks. However, when we compare our MSAOD solution to the offloading of all tasks, we find some negligible degradation that does not exceed 2%, it comes from the fact that, when offloading tasks we ignore the offloading constraints; as a result, we get the infeasible solutions. Therefore, we show that our proposed heuristic scheme efficiently manages the offloading decisions. To show the real difference between MSAOD solution and the offloading of all tasks, that is shown in Figure 5, we report their results with a zoom in Figure 6.

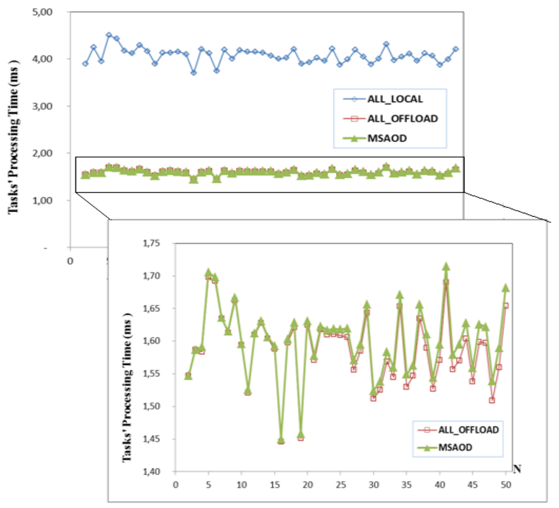

Figure 6. Tasks' Processing Time for N between 2 and 50.

Figure 7 shows that the results of the MSAOD solution are better than those of SAOD for all N values. The results of the first represent a gain in processing time that varies between 0% and 2%.

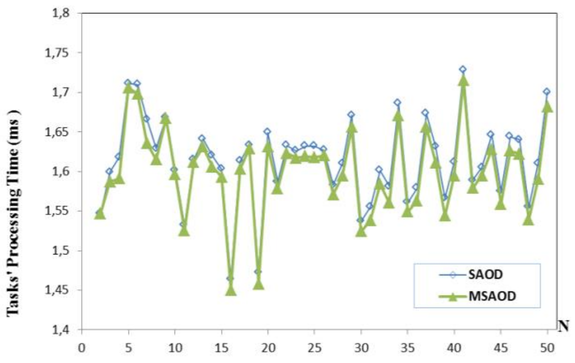

Figure 7. Tasks' Processing Time for N between 2 and 50.

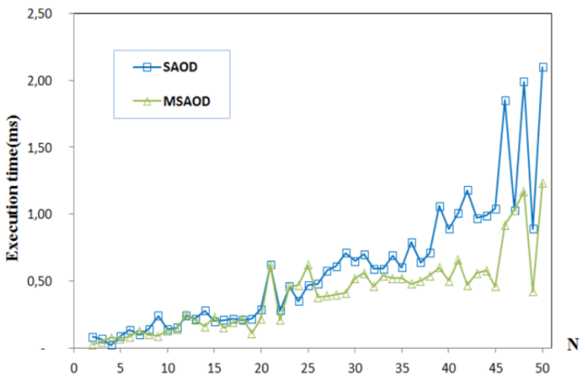

Figure 8. Execution Time Average for N between 2 and 50.

Figure 8 illustrates the execution time comparison for MSAOD and SAOD methods while N is set between 2 and 50. The MSAOD and SAOD solutions give a near linear execution time that reached for N=50 respectively 1.23ms and 2.10ms. Accordingly, the performance in terms of execution time of the SAOD method is slightly higher than the MSAOD method.

5. Conclusion

In this paper, we have considered a single smart mobile device (SMD) containing an offloadable multi-task list. We are devoted to enable the latency sensitive tasks to run on user equipments by implementing partial computation offloading. First, we have consided the single user offloading problem, where the MEC resources are considered to be unconstrained. Later, we extend it to the multi-task offloading problem, where the resource constraints are taken into consideration. As demonstrated by emulation experiment based on numerical simulation, the proposed simulated annealing algorithm in this article is capable of offloading tasks that require a lot of processing time and simultaneously improve the offloading rate to ultimately reduce the total processing time of tasks. The obtained results in terms of execution time of the proposed algorithms are very encouraging. In the course of future work, we will generalize our study to the multi-user case.

References

[1] D. Catteeuw and B. Manderick, "Heterogeneous populations of learning agents in the minority game," in International Workshop on Adaptive and Learning Agents, Springer, pp. 100-113, 2011.

[2] Y. Jararweh, M. Al-Ayyoub, M. Al-Quraan, A. T. Lo'ai, and E. Benkhelifa, "Delay-aware power optimization model for mobile edge computing systems," Personal and Ubiquitous Computing, vol. 21, no. 6, pp. 1067-1077, 2017.

[3] 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, pp.1-9,2017.

[4] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1275-1284, 2018.

[5] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," IEEE Transactions on Signal and Information Processing over Networks, vol. 1, no. 2, pp. 89-103, 2015.

[6] X. Xu et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," Journal of Network and Computer Applications, vol. 133, pp. 75-85, 2019.

[7] 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.

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

[9] Y. Wu, L. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-Minimization Nonorthogonal Multiple Access enabled Multi-User Mobile Edge Computation Offloading," IEEE Journal of Selected Topics in Signal Processing, pp. 392-407, 2019.

[10] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative Task Offloading in Three-Tier Mobile Computing Networks: An ADMM Framework," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2763-2776, 2019.

[11] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," IEEE Communications Letters, vol. 21, no. 7, pp. 1481-1484, 2017.

[12] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," IEEE/ACM Transactions on Networking, vol. 27, no. 1, pp. 85-97, 2019.

[13] 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), pp. 1-6, 2016.

[14] L. Li, Y. Xu, Z. Zhang, J. Yin, W. Chen, and Z. Han, "A prediction-based charging policy and interference mitigation approach in the wireless powered Internet of Things," IEEE Journal on Selected Areas in Communications, vol. 37, no. 2, pp. 439-451, 2018.

[15] I. Khan, "Performance analysis of 5G cooperative-NOMA for IoT-intermittent communication," International Journal of Communication Networks and Information Security, vol. 9, no. 3, pp. 314-322, 2017.

[16] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Power-Constrained Edge Computing with Maximum Processing Capacity for IoT Networks," IEEE Internet of Things Journal, pp.4330-4343, 2018.

[17] 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, pp. 301-314, 2011.

[18] 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.

[19] 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.

[20] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," IEEE access, vol. 4, pp. 5896-5907, 2016.

[21] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," IEEE Internet of Things Journal,pp. 4804-4814, 2018.