A Dynamic Cost Model to Minimize Energy Consumption and Processing Time for IoT Tasks in a Mobile Edge Computing Environment

João Luiz Grave Gross, Kassiano José Matteussi,
Julio C. S. dos Anjos, and Cláudio Fernando Resin Geyer

Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil, 91509–900
{jlggross,kjmatteussi,jcsanjos,geyer}@inf.ufrgs.br
https://www.inf.ufrgs.br/

Abstract. The rapid growth of devices, e.g., smartphones, wearables, tablets, and other sensors connected to the Internet, has been leading to a complex problem regarding how to optimize energy consumption for data-intensive processing. Most applications tend to offload task processing to remote servers, usually to Data Centers in the Cloud, geographically located away from the end-user and the IoT device, increasing communication latency and energy costs. In such a context, this work proposes a dynamic cost model to minimize energy consumption and task processing time for IoT scenarios in a Mobile Edge Computing environments. The solution presents a Time and Energy Minimization Scheduler (TEMS) that solves the cost model, validated through simulation. Results reveal a reduction in the system's energy consumption by up to 51.61% as well as 86.65% in task's completion time.

Keywords: Mobile Edge Computing · Internet Of Things · Cost Minimization Model · Energy Consumption · Scheduling Algorithm.

1 Introduction

Billions of smart devices can now connect to the Internet in the form of *Internet of Things* (IoT) due to advances in information technologies communication [2]. Moreover, according to an IDC report, up to 2025 there will be 41.6 billion IoT devices with the potential to generate 79.4 ZB of data [11].

IoT applications have evolved, especially those used in artificial intelligence, artificial vision, and object tracking, which require high computing power [5] [15]. They usually rely on task processing offload and data storage to remote *Cloud Computing* (CC) Data Centers to boost processing time and reduce battery energy consumption [21]. Unfortunately, those remote servers are geographically located away from the end-user and the IoT device, resulting in high latency due to delay and congestion over the communication channels [1]. Usage of centralized control (provider-centric) cannot deliver connectivity or even support computation closer to the edge of the network. It became inefficient for high distributed scenarios, and the spreading of CC infrastructure is a significant promise for IoT applications and services proliferation.

In such a scenario, Mobile Edge Computing (MEC) is a suitable alternative to CC, as it represents a network architecture designed to provide low latency and better QoS to end-users [10]. Also, MEC relies on top of high-speed mobile networks such as 5G to allow fast and stable connectivity for mobile devices and users. Thus, CC services can be deployed close to mobile devices, in the MEC layer, bringing processing and storage closer to cellular base stations [24].

Nevertheless, energy consumption remains an open issue on mobile device networks, such as MEC environments [16]. Most IoT sensors and actuators are small and run on batteries with limited energy capacity. Furthermore, IoT devices need to handle lots of data, which is also energy-consuming. Thus, reducing energy consumption in networks with IoT devices is a goal worth exploring.

State-of-the-art presents a set of studies that use MEC to offload tasks in order to offer local processing to IoT devices. Some works [25] [8] [18] [21] [6] [13] have measured the energy consumption for data transmissions or even DVFS (Dynamic Voltage and Frequency Scaling) techniques. In contrast, our approach enables a most detailed cost model, including not only energy and time consumed during processing but also the cost of data transmissions. CC is also considered as an option for processing when local resources are depleted, making the network more reliable in stress scenarios.

This work tackles these issues by proposing a dynamic cost model to minimize energy consumption and task processing for IoT scenarios in MEC environments. Also, we propose the TEMS scheduling algorithm that implements the dynamic cost minimization model. It calculates the cost associated with the allocation options in the system and chooses one that yields the lesser cost for the task's execution. Finally, to execute the TEMS algorithm and gather data about the environment and associated energy and time costs, a simulator has been developed for a MEC environment with IoT devices and CC resources[1].

The main contributions of this work are i) we evaluate tasks with different profiles such as critical tasks (with a deadline) and non-critical tasks (without a deadline) in a variety of case scenarios; ii) Our methodology covers a considerable number of energy and time metrics for task processing and data transmissions, including the accounting of CPU cores idle energy; iii) The model uses a DVFS technique aiming to optimize CPU nodes processing time and energy consumption; iv) Our model considers three possible processing sites, including processing in the IoT device itself, in a local MEC server and in a remote Data Center from CC; v) We develop and adapts experiments based on a well-defined simulation tool for scenarios with IoT devices, MEC servers, and CC Data Centers.

The remainder of this paper is organized as follows. Section 2 introduces the dynamic cost minimization model for the system with three different allocation policies, (1) local processing in the IoT device, (2) local processing in the MEC server and (3) CC remote processing. Section 3 introduces the heuristic TEMS scheduling algorithm designed to solve the system's cost minimization model. Section 4 details the implementation and shows the results of the experiments

[1] MEC Simulator available at https://github.com/jlggross/MEC-simulator.

using the TEMS scheduling algorithm. Section 5 presents some of the previous Related Work found in the literature. Finally, in Section 6, the conclusions.

2 Dynamic Cost Minimization Model

In this section, we introduce a detailed view of our dynamic cost minimization model.

2.1 Architecture and Task Processing Flow

Figure 1 presents the system's architecture design decoupled in three layers. Following a bottom-up approach:

- **IoT Layer (L1)**: IoT devices compose the first layer, which generate the application tasks. These devices run on batteries and have limited processing power.
- **MEC Layer (L2)**: The second layer has local MEC servers with a limited number of CPU cores and less processing power if compared to the CC Data Centers, but are located close to the end-users, providing smaller communication delays.
- **CC Layer (L3)**: The third layer constitutes the Data Centers from CC. This layer is far located from the end-users, and it is geographically distributed, with high processing power. Also, it adds high network latency due to the data transmission with more communication hops compared to other layers.

The model associates a cost for each layer in terms of energy consumed and elapsed time for task processing and data transmissions. The TEMS scheduling algorithm analyses the cost level for processing tasks in each layer, seeking the best cost among three task allocation policies, (1) local processing in the IoT device, (2) local processing in the MEC server and (3) remote processing in the CC Data Centers. The policy that provides the lowest cost is selected and the task is offloaded to the corresponding layer.

The rest of this section covers the cost models for each layer of our architecture. First, we introduce some ground premises about the architecture's network. After that, the cost models for local computing in the IoT device, local computing in the MEC server, and remote computing in the Cloud are shown. Finally, the individual costs are combined into a final equation that represents the total system cost.

2.2 Network Model

The system network has a finite set $D = \{1, 2, 3, ..., d\}$ of mobile IoT devices, $S = \{1, 2, 3, ..., s\}$ local MEC servers and $W = \{1, 2, 3, ..., w\}$ wireless communication channels. Each local device or MEC server can process zero or more tasks. The system has a total of $A = \{1, 2, 3, ..., a\}$ tasks. Each task A_i is represented by a tuple $A_i = (C_i, s_i, d_i, t_i)$, where $i \in A$.

For each task A_i, C_i represents the number of CPU cycles required for its complete execution. s_i and d_i represent, respectively, the source code and data

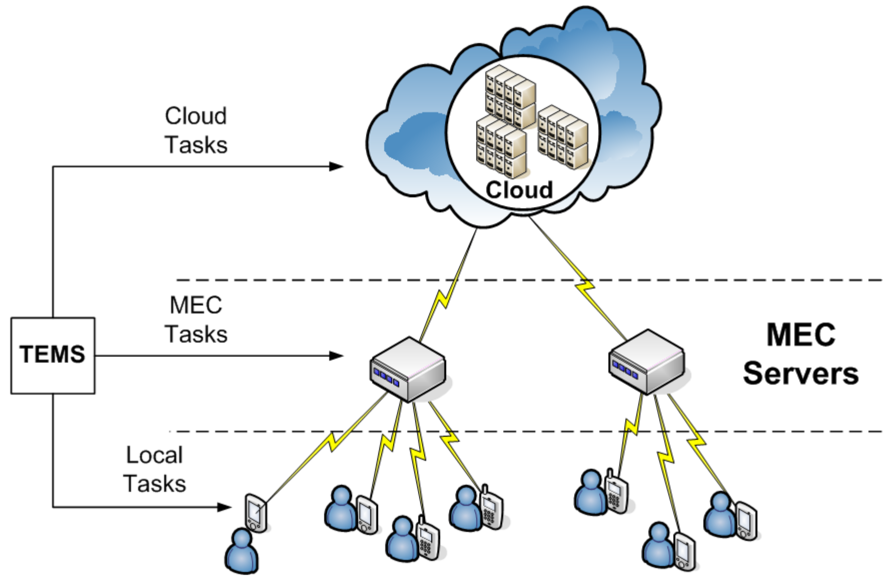

Fig. 1. System's architecture and task allocation policies.

entry sizes. t_i represents the task's deadline, which is the maximum time to complete it's execution. In such a case, tasks with deadline equal to zero are non-critical tasks.

Also, set H stores the communication channel associated to each task, thus $H = \{h_1, h_2, h_3, ..., h_a\}$. If a task i has a communication channel w associated to it, them $h_i = w$, otherwise $h_i = 0$, where $h_i \in W \cup\ 0$.

2.3 Local Computing in the IoT Device

Each mobile device $j \in D$ on the network has one or more CPU cores. Thus, the processing cores available on the network for the j device are given by $PL_j = \{pl_{j,1}, pl_{j,2}, pl_{j,3}, ..., pl_{j,n}\}$. Every core $pl_{j,k} \in PL_j$ has a value of 0 or 1, 0 if it is vacant and 1 if it is busy. For vacant CPU cores, *idle* energy is counted and for occupied CPU cores dynamic energy ($E_{i,local}$). Each core has an operating frequency ($f_{local,j,k}$), i.e., processing capacity, an effective commutative capacitance ($C_{local,j,k}$), dependent of the chip architecture [23], and a voltage supply ($V_{local,j,k}$). In addition, each task i has a total number of CPU cycles (CT_i) for its execution.

So it is possible to calculate the total execution time of task $i \in A$ by a CPU core $j \in PL$ using CT_i [19], as well as the dynamic energy consumed during the execution, which is $\propto CV^2f$ [14], both shown in Equations 1 and 2. In Equation 3 we compute the local dynamic energy consumed by the IoT device.

$$T_{i,local} = \frac{CT_i}{f_{local,j,k}} \tag{1}$$

$$P_{i,local} = C_{local,j,k} * V^2_{local,j,k} * f_{local,j,k} \tag{2}$$

$$E_{i,local} = P_{i,local} * T_{i,local} \tag{3}$$

Considering battery level and latency as model constraints, a device D_j must decide whether it is more appropriate to process the task, locally or remotely. As the battery level is a critical factor in the decision, the system will appreciate a policy that reduces energy consumption. The local cost of **one task *i*** can be expressed by:

$$Cost_{i,local} = u_{localT} * T_{i,local,total} + u_{localE} * E_{i,local} \tag{4}$$

In Equation 4 u_{localT} and $u_{localE} \in [0, 1]$, and $u_{localT} + u_{localE} = 1$. They are coefficients used to represent the weight of time and energy, respectively. As mentioned in [21] these two variables work as a trade-off between execution time and energy consumption, being used to prioritize minimizing one of the costs.

2.4 Local Computing in the MEC Server

Local MEC servers are expected to have multiple CPU cores. Thus, the CPU cores available on a local server S_j are given by $PS_j = \{ps_{j,1}, ps_{j,2}, ps_{j,3}, ..., ps_{j,n}\}$. Each core $ps_{j,k}$ has an operating frequency ($f_{mec,j,k}$), an effective commutative capacitance ($C_{mec,j,k}$), and a supply voltage ($V_{mec,j,k}$).

Communications that take place between the IoT device and the local server and use the same wireless channel cause mutual interference between each other, expressed as I_i. In this case, the data transfer rate $r_i(h_i)$ to offload task i to its corresponding channel h_i is attenuated according to *Shannon's formula* [23]. The data transfer rate and the mutual interference between wireless channels can be given by:

$$r_i(h_i) = W * \log_2 \left(1 + \frac{p_m * g_{j,m}}{N + I_i} \right) \tag{5}$$

$$I_i = \sum_{n \in A|\{i\}:h_n = h_i} p_{m'} * g_{j',m'} \tag{6}$$

For Equation 5, W is the channel bandwidth in *Hertz*, $g_{j,m}$ represents the channel gain between the mobile device m and a local server, represented by j. The variable p_m represents the transmission power of the mobile device m when offloading task i to the local server. N represents the power of the thermal noise of the wireless channel, h_n represents the wireless channel associated to task n and I_i is the mutual interference between transmissions over the same channel.

Unlike processing in the device, processing in the local server requires data (d_i) and code (s_i) from the application to be transmitted to the server and the generated results (d'_i) must be transmitted back to the origin. Thus, the time required for an IoT device $j \in D$ to transmit data and download results from the local server are given by:

$$T_{i,mec-up}(h_i) = \frac{s_i + d_i}{r_i(h_i)} \tag{7}$$

$$T_{i,mec-down}(h_i) = \frac{d_i'}{r_i(h_i)} \tag{8}$$

The total time required to complete the task execution in the local server considers $T_{i,mec-up}(h_i)$, $T_{i,mec-down}(h_i)$ and the task execution time $T_{i,mec}$, calculated the same way as for the IoT devices. The total time for a MEC server is given by:

$$T_{i,mec,total} = T_{i,mec-up}(h_i) + T_{i,mec} + T_{i,mec-down}(h_i) \tag{9}$$

The energy consumed for the data transmissions from the IoT device to the local server and from the local server to the IoT device, are calculated by the power consumed in the data transfers ($p_{wireless}$) multiplied by the elapsed time, which could be $T_{i,mec-up}(h_i)$ or $T_{i,mec-down}(h_i)$, depending on the information carried in the data transmission. Finally, the dynamic energy consumed is calculated the same way as for the IoT device, $P_{i,mec} * T_{i,mec}$, and the total dynamic energy consumption is given by:

$$E_{i,mec,total} = E_{i,mec-up}(h_i) + E_{i,mec} + E_{i,mec-down}(h_i) \tag{10}$$

The cost equation for the local server is expressed as follows:

$$Cost_{i,mec} = u_{mecT} * T_{i,mec,total} + u_{mecE} * E_{i,mec,total} \tag{11}$$

2.5 Remote Computing in the Cloud

CC is assumed to have unlimited resources, which is why cores are not distinguished. Its equations are very similar to those of the local MEC server but with some more components such as time spent and energy consumed during data transmission between MEC server and CC Data Centers, added to the transmissions between IoT devices and MEC servers. The total elapsed time and total energy consumption for CC are as follows:

$$\begin{aligned} T_{i,cloud,total} = {} & T_{i,mec-up}(h_i) + T_{i,cloud-up} + T_{i,cloud} + \\ & T_{i,cloud-down} + T_{i,mec-down}(h_i) \end{aligned} \tag{12}$$

$$\begin{aligned} E_{i,cloud,total} = {} & E_{i,mec-up}(h_i) + E_{i,cloud-up} + E_{i,cloud} + \\ & E_{i,cloud-down} + E_{i,mec-down}(h_i) \end{aligned} \tag{13}$$

Finally, the cost to run a single task i in the Cloud is given by:

$$Cost_{i,cloud} = u_{cloudT} * T_{i,cloud,total} + u_{cloudE} * E_{i,cloud,total} \tag{14}$$

The *idle* energy cost of CC is not considered, since the CPU offer is virtually infinite. Therefore it does not make sense to account for this cost, which would cause the system to have equally infinite cost.

2.6 System's Dynamic Cost Minimization Equation

The system minimization cost equation is developed as an Integer Linear Problem (ILP). For each of the system's scheduling policies, there is a specific cost equation. The cost of each task i is given by a minimization equation, and the total system cost by the sum of all task costs and idle energy costs, calculated for the hardware that is waiting for processing workload.

$$\begin{aligned} Cost_i = min(&\alpha * Cost_{i,local}, \\ &\beta * Cost_{i,mec}, \\ &\gamma * Cost_{i,cloud}) \end{aligned} \tag{15}$$

$$Cost_{system} = \sum_{i=1}^{A} Cost_i + \alpha * E_{local,idle} + \beta * E_{mec,idle} \tag{16}$$

In Equations 15 and 16, α, β and $\gamma \in [0, 1]$, and $\alpha + \beta + \gamma = 1$. These coefficients operate similarly to the energy and time coefficients used in the cost equation of each policy. They can be used to prioritize one policy to being chosen. For instance, if α has the lowest value among the three coefficients, then processing the task locally in the IoT device itself will be more appealing, because of the minimization function. The same goes for β and γ and their corresponding scheduling policies.

2.7 Constraints on Battery Level for IoT Devices

A healthy battery level is essential to the proper operation of IoT devices. If the battery level B_j of a battery from device j is below a *Lower Safety Limit* (LSL), task allocation on the device is disabled to preserve the functioning of the equipment with the remaining battery. If B_j reaches zero, all tasks generated by the device j are canceled. Therefore, all cost equations used to allocate tasks on the IoT device are subject to the following constraints: $B_j > E_{i,local}$, $B_j > E_{i,mec-up}(h)$ and $B_j > LSL$.

3 Time and Energy Minimization Scheduler (TEMS)

The TEMS heuristic scheduling algorithm was developed in order to execute the dynamic cost minimization model with reduced computational cost. It has complexity $O(n^2)$.

In Algorithm 1 are presented the steps of TEMS. Step 1 defines the data sets of the IoT devices, MEC servers, and communication channels. The battery levels of the IoT devices are collected, and the LSL is set. The algorithm collects the number of CPU nodes available in each IoT device and MEC server, their operating frequencies, and supply voltages. This process also occurs for CC Data Centers, but the number of CPUs is assumed to be infinite.

Algorithm 2 details step 2 from TEMS, which is the scheduler's task allocation decision-making process. Firstly tasks are classified between critical and

Algorithm 1: Time and Energy Minimization Scheduler (TEMS)

```
Result: Task mapping to the processing nodes
1 execute Step 1 - Collection of system information and initialization
2 repeat
3     execute Step 2 - Task allocation
4     execute Step 3 - Task conclusion monitor
5     execute Step 4 - New tasks and device battery level monitor
6 until user decides to keep running;
```

Algorithm 2: Step 2 – Task allocation

```
1  foreach task $A_i$ from ordered list do
2      foreach CPU $pl_{j,k} \in PL_j$ do
3          if $(B_j > LIS)$ and $(B_j > E_{estimated})$ and $(pl_{j,k} = 0)$ then
4              calculate IoT device execution time
5          end
6      end
7      foreach CPU $ps_{j,k} \in PS_j$ do
8          if $ps_{j,k} = 0$ then
9              calculate MEC server execution time and transmission times
10         end
11     end
12     calculate CC execution time + transmission times
13     offload task to the CPU core with minimal total time
14 end
15 foreach task $A_i$ from list of normal tasks do
16     foreach CPU $pl_{j,k} \in PL_j$ do
17         if $(B_j > LIS)$ and $(B_j > E_{estimated})$ and $(pl_{j,k} = 0)$ then
18             calculate energy consumption, execution time and cost for the IoT
                 device
19         end
20     end
21     foreach CPU $ps_{j,k} \in PS_j$ do
22         if $ps_{j,k} = 0$ then
23             calculate energy consumption for dynamic processing and data
                 transmission
24             calculate execution time and transmission times
25             calculate MEC server cost
26         end
27     end
28     calculate energy consumption for dynamic processing and data
         transmission, execution time and transmission times and final cost for CC
29     offload task to the CPU core with minimal cost
30 end
```

non-critical. The time and energy consumption for task processing on the different CPU cores of the network is calculated, as well as the time and energy consumption of the data transmissions for MEC servers and CC Data Centers.

Critical tasks are ordered by the deadline and allocated by the lowest total elapsed time. Then non-critical tasks are ordered by creation time and allocated by the minimum total cost. Critical tasks are the first to be scheduled due to the sensitivity of the deadline.

In step 3, tasks are monitored for their completion status, and when completed, the CPU core resources are released and made available for other allocations in step 2. Tasks that use CC resources don't need to release them since CC is supposed to have unlimited resources, absorbing any number of tasks. The battery level check is performed only for mobile IoT devices since the local servers have external power that allows them to operate uninterrupted. Task cancellation may occur if the elapsed time is higher then the deadline or if the IoT device runs out of battery.

Finally, in step 4, the battery level from each IoT device is collected, as well as newly created tasks. Execution continues as long as tasks are being created.

4 Evaluation

This section explains the simulation details and the different experimental scenarios used.

4.1 Simulated Hardware and Software Stack

The simulated environment was comprised by IoT devices type Arduino Mega 2560, with operating frequencies of 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz and corresponding supply voltages of 5 V, 4 V, 2.7 V, 2.3 V, and 1.8 V. The MEC servers were simulated on top of 5 Raspberry Pi 4 Model B boards, each board with a Quad-core Cortex-A72 1.5GHZ (ARM v8) 64-bit, summing a total of 20 CPU cores per server. The CPU cores have operating frequencies of 1,500 MHz, 1,000 MHz, 750 MHz, and 600 MHz and corresponding supply voltages of 1.2 V, 1 V, 0.825 V, and 0.8 V. The capacitance of the CPU cores is set to 1.8 nano Farads. For CC it was chosen Data Centers with *Intel Xeon Cascade Lake* processors of 2.8 GHz per CPU core, reaching up to 3.9 GHz with *Turbo Boost* on[2].

The network throughput was configured to achieve up to 1 Gbps speed and latencies to 5ms, for both 5G and fiber optics communications [9] [4]. Moreover, two vehicular applications were defined [12] for the experiments, as shown in Table 1.

4.2 Experiments and Results

The tested scenarios in our experiments used different configurations for parameters such as computational workload of each task, coefficients for energy consumption and elapsed time, size of data entry and results, task generation rate, critical task's deadline, level of batteries for IoT devices and use of DVFS to see how the TEMS algorithm would respond to task allocation and energy

[2] Technical information can be found in the *datasheets* of the electronic components.

Table 1. Characteristics of chosen applications.

Characteristics	**Application 1**	**Application 2**
Task generation rate (seconds)	10	0,1
Data entry (mega bytes)	36.3	4
Results (bytes)	1,250	625
Computational workload (millions of CPU cycles)	2,000	20
Critical Tasks (% from total tasks)	10	50
Deadline for critical tasks (mili seconds)	500	100

and time reduction. Some of the results are discussed below.

a. Analysis on task workload: This experiment evaluates how TEMS allocate a set of tasks, according to the workload described in Table 1 for Application 1. The scenario has 500 tasks distributed to 100 IoT devices in two different cases, a single MEC server (1a and 1b) and two MEC servers (2a and 2b). Figure 2 shows the results for the execution of Application 1.

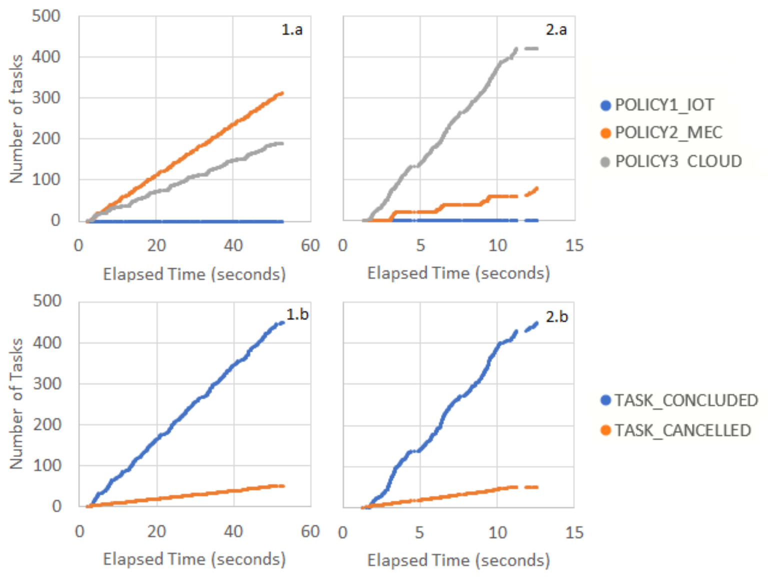

Fig. 2. Processing workload for Application 1 defined to $2,000 * 10^6$ CPU cycles.

The energy and time coefficients were set, respectively, as 4/5 and 1/5, that is, a high weight was given to the energy consumed so that it could be minimized. In Figure 2 from 1.a to 2.a the increase in the number of MEC servers made fewer tasks be to allocated in the CC layer. This positively impacts the total energy consumed. Compared to another scenario with 500 tasks and 100 IoT devices, but without the use of MEC servers, the reduction in energy consumption for case 1 was 42.51%, while for case 2 it reached 44.71%. Thus, the use of MEC servers helps the system to lower the total energy consumed.

With Application 2, that has lower workload compared to Application 1, the allocation profile changed. Most of allocations took place on the device itself, regardless of the number MEC servers. The cause to this phenomenon is due to the low processing workload of Application 2. The hardware of the IoT devices presents higher energy consumption per CPU cycle. However, it does not require data transmissions, which add energy cost and elapsed time to the system. Thus, for a small processing workload, IoT devices are the first allocation option.

b. Analysis of the impact on chosen energy consumption and elapsed time coefficients: This experiment measures the coefficients for energy consumption and elapsed time for Application 2 with a workload of $20 * 10^6$ CPU cycles. Each case has 500 tasks, 100 IoT devices, and one MEC server. The energy coefficients were set to 1/5, 2/5, 3/5, 4/5 and the time coefficients to 4/5, 3/5, 2/5 and 1/5.

Table 2 lists the minimum costs perceived by the system task scheduler for each case. Cases 2 and 3 had their lowest cost in the same allocation option. In Case 1, it occurred for the MEC server, with DVFS configured to 1,500 MHz and 1.2 V. MEC was the chosen option for these three cases because time had still a high coefficient associated to it, and executing the task in a MEC server made lower processing time possible. As for Case 4, with 4/5 for energy and 1/5 for time coefficients the allocation took place on the IoT device itself, with DVFS configured at 8 MHz and 4 V. Now, energy has a high-value coefficient, which makes the scheduler choose for the policy that provides the lowest energy cost, reducing total cost.

Table 2. Costs for Application 2 with a workload of $20 * 10^6$ CPU cycles, varying the cost coefficients for energy and time. E_{Total} in $W*s$ and T_{Total} in seconds.

Case	E_{Coeff}	T_{Coeff}	Cost	E_{Total}	T_{Total}	f (MHz)	T (V)	Policy
$C1$	1/5	4/5	0.01859	0.14550	0.03336	1,500	1.2	MEC
$C2$	2/5	3/5	0.02597	0.14276	0.03469	750	0.825	MEC
$C3$	3/5	2/5	0.03318	0.14276	0.03469	750	0.825	MEC
$C4$	4/5	1/5	0.03544	0.07040	0.25000	8	4.0	IoT

To reduce energy consumption, the best option is using 4/5 for the energy coefficient, prioritizing the minimization of energy consumption, which reduced up to 51.61% for normal tasks compared to the other cases. If the concern is on reducing task completion time, setting time coefficients to 4/5, 3/5, or 2/5 will prioritize the minimization of total elapsed time. One can achieve a reduction of up to 86.65% in the completion time for normal tasks compared to the most costly case. For Cases 1, 2, and 3, a considerable reduction in total elapsed-time was perceived because running the task in the IoT device for the evaluated application made the time component spike.

c. Analysis of the input data size: This experiment evaluates Application 1 with the workload set to $20 * 10^6$ CPU cycles, energy cost coefficient at 4/5, 500 tasks, 100 IoT devices and two MEC servers. Four different input sizes were used, 3.6 MB, 36 MB, 362 MB, and 3.6 GB.

As the size of data entry increases, the calculated costs progress in the MEC and CC allocation policies. The cost to execute in the IoT devices remains the same, as no data transmissions are carried out. When data entry scales, allocation policies that require data transmissions become very costly, and allocation on the device itself becomes increasingly advantageous. Therefore, it is crucial to scale applications so that data transfers over the network are not too large per task, avoiding high data transmission costs.

We designed two other cases, one with 5,000 tasks and 362 MB per task and another with 500 tasks and 3.6 GB per task, with roughly 1.8 TB in total each. The system energy consumption and the total elapsed time for the 500 tasks case were 59,160.92 W*s and 14,515.21 s. For the experiment with 5,000 tasks, the costs were 45,011.49 W*s and 10,305.94 s, that is, a decrease of 23.92% and 29%, respectively. Therefore, as explained above, in an environment with continuous task generation, a low deadline, and the need for low latencies, tasks should not be very data-intensive, as data transmissions add additional energy and time expenses.

d. Analysis of IoT device battery consumption: The evaluation of energy consumption indicates that shallow battery levels quickly achieve LSL and make IoT devices unavailable for processing. Very high computational workloads also impact the battery level negatively. Therefore, a battery with a healthy energy level and adequate task processing workloads, allows the allocation to be carried out on the IoT device, without making it unavailable due to lack of battery.

e. Analysis of deadline for critical tasks: This experiment analyses the task deadlines. Deadlines with too restrictive time limits made critical tasks to be canceled because the scheduler couldn't find any allocation policy to make completion successful. Therefore, the deadline must be properly configured for at least one of the allocation policies to have enough time to process and concluded the task correctly.

f. Analysis on the use of the DVFS technique: With DVFS activated, the total energy consumption decreased by 13.74%, while the total time increased by 28.32% in comparison with DVFS off. This demonstrates the effectiveness of the proposed model and the scheduling algorithm, in minimizing total energy consumption. Although the whole time may have been longer in the approach with DVFS, it is no problem because the tasks were completed within the time limit imposed by the deadline.

5 Related Work

The reduction of energy consumption and response latency to applications in IoT environments is an open issue since the creation of the first *Edge Computing* architectures [17]. The use of CC as the only option to offload tasks adds greater latencies to IoT applications. However it can be used as an alternative if the resources of the local network are exhausted. Few works use CC as an option to download tasks [16], [23]. Others, such as in [20], [7], [22], and [8], use the *Fog Computing* architecture to offer local processing to IoT devices, but without using CC [3]. The use of the MEC architecture occurs in [16], [25], [21] and [23]. In contrast, our model has a three-layer architecture altogether, which includes MEC and CC services in addition to local IoT computation. Thus, this approach enables a more detailed cost model, including not only energy and time consumed during processing but also the cost of data transmissions.

As for the variables used by cost minimization models, the energy consumption of task processing is unanimous, used by all works mentioned, however the energy consumption for data transmissions are restricted to [25], [8], [18] and [21]. The processing time of tasks is also used by the majority, except for [20] and [22], while the time spent on data transmissions is restricted to [16], [25], [8], [21] and [23]. Energy consumption when there is no processing, that is, with the equipment in idle state, is used only in [20] and [22]. Models that include the battery level of the IoT devices are only [25], [8] and [21].

All these variables are condensed in our model, including a monitor for battery levels from IoT devices and two types of tasks, critical and non-critical, the former having to deal with deadline constraints. Finally, we use the DVFS technique, allowing dynamic minimization of energy and time during task processing [6] [13]. These combined characteristics enable a more refined approach aiming to reduce both energy and time consumed for task allocation.

6 Conclusion

In an environment with accelerated generation of large volumes of data and mobile devices connected to the Internet with restricted QoS requirements and battery limitations, energy and time reduction are mostly needed. The TEMS scheduler could choose the best allocation options in the system, reducing both energy consumption and elapsed time. Experiments show that the adequate adjustment of the cost coefficients were essential for the final cost perceived by the scheduling algorithm. Adequate coefficients allowed the system's energy to be reduced by up to 51.61% or the total times to be reduced by up to 86.65%, ending critical tasks before deadline. The system has become more sustainable and the user experience has not been affected.

The use of MEC servers helped extend the battery life of the IoT devices and made task execution more agile. Also, using the DVFS technique brought interesting results, helping to reduce the overall energy consumption. Major contributions are the TEMS algorithm, the addition of data transmissions to the model, accounting for idle costs, calculating transmission rate interference, use of the DVFS technique, and the interaction with the CC layer to provide resources whenever the local network becomes saturated.

As future works, we can relate the improvement of the system cost model, with the insertion of new variables and new experiments to explore applications in new scenarios such as industry, healthcare, aviation, mining, among others.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. PROPESQ-UFRGS and by projects: "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#162551-0000 488-9) and SmartSent (#172551-0001 195-3) from FAPERGS, PNPD Capes, and CNPq Brazil, program PRONEX 122014.

References

1. Aijaz, A.: Towards 5g-enabled tactile internet: Radio resource allocation for haptic communications. In: 2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). pp. 145–150 (2016)
2. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys Tutorials **17**(4), 2347–2376 (2015)
3. Anjos, J.C.S., Matteussi, K.J., De Souza, P.R.R., da Silva Veith, A., Fedak, G., Barbosa, J.L.V., Geyer, C.R.: Enabling strategies for big data analytics in hybrid infrastructures. In: 2018 International Conference on High Performance Computing Simulation (HPCS). pp. 869–876 (July 2018). https://doi.org/10.1109/HPCS.2018.00140
4. Brogi, A., Forti, S., Ibrahim, A.: Deploying fog applications: How much does it cost, by the way? In: CLOSER (2018)
5. Chen, T.Y.H., Ravindranath, L., Deng, S., Bahl, P., Balakrishnan, H.: Glimpse: Continuous, real-time object recognition on mobile devices. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems. p. 155–168. SenSys '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2809695.2809711
6. Chen, Y.L., Chang, M.F., Yu, C.W., Chen, X.Z., Liang, W.Y.: Learning-directed dynamic voltage and frequency scaling scheme with adjustable performance for single-core and multi-core embedded and mobile systems. Sensors **18**(9), 3068 (Sep 2018). https://doi.org/10.3390/s18093068
7. Galache, J.A., Yonezawa, T., Gurgen, L., Pavia, D., Grella, M., Maeomichi, H.: Clout: Leveraging cloud computing techniques for improving management of massive iot data. In: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications. pp. 324–327 (2014)
8. Gedawy, H., Habak, K., Harras, K.A., Hamdi, M.: Awakening the cloud within: Energy-aware task scheduling on edge iot devices. In: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). pp. 191–196 (March 2018). https://doi.org/10.1109/PERCOMW.2018.8480266
9. Gupta, A., Jha, R.K.: A survey of 5g network: Architecture and emerging technologies. IEEE Access **3**, 1206–1232 (2015). https://doi.org/10.1109/ACCESS.2015.2461602
10. Haouari, F., Faraj, R., AlJa'am, J.M.: Fog computing potentials, applications, and challenges. In: 2018 International Conference on Computer and Applications (ICCA). pp. 399–406 (Aug 2018). https://doi.org/10.1109/COMAPP.2018.8460182
11. IDC: The growth in connected iot devices is expected to generate 79.4zb of data in 2025 (2019), available at: `https://www.idc.com/getdoc.jsp?containerId=prUS45213219`.

12. Jansson, J.: Collision Avoidance Theory with Application to Automotive Collision Mitigation. Ph.D. thesis (06 2005)
13. Jin, X., Goto, S.: Hilbert transform-based workload prediction and dynamic frequency scaling for power-efficient video encoding. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **31**(5), 649–661 (May 2012). https://doi.org/10.1109/TCAD.2011.2180383
14. Liu, Y., Yang, H., Dick, R.P., Wang, H., Shang, L.: Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In: 8th International Symposium on Quality Electronic Design (ISQED'07). pp. 204–209 (March 2007). https://doi.org/10.1109/ISQED.2007.158
15. Matteussi, K.J., Zanchetta, B.F., Bertoncello, G., Dos Santos, J.D.D., Dos Anjos, J.C.S., Geyer, C.F.R.: Analysis and performance evaluation of deep learning on big data. In: 2019 IEEE Symposium on Computers and Communications (ISCC). pp. 1–6 (June 2019). https://doi.org/10.1109/ISCC47284.2019.8969762
16. Sarangi, S.R., Goel, S., Singh, B.: Energy efficient scheduling in iot networks. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. p. 733–740. SAC '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3167132.3167213
17. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE Pervasive Computing **8**(4), 14–23 (Oct 2009). https://doi.org/10.1109/MPRV.2009.82
18. Skarlat, O., Schulte, S., Borkowski, M., Leitner, P.: Resource provisioning for iot services in the fog. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). pp. 32–39 (2016)
19. Tanenbaum, A.S., Austin, T.: Structured Computer Organization. Prentice Hall, 6th edn. (2012)
20. Wan, J., Chen, B., Wang, S., Xia, M., Li, D., Liu, C.: Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Transactions on Industrial Informatics **14**(10), 4548–4556 (Oct 2018). https://doi.org/10.1109/TII.2018.2818932
21. Wang, C., Dong, C., Qin, J., Yang, X., Wen, W.: Energy-efficient offloading policy for resource allocation in distributed mobile edge computing. In: 2018 IEEE Symposium on Computers and Communications (ISCC). pp. 00366–00372 (June 2018). https://doi.org/10.1109/ISCC.2018.8538612
22. Wu, H., Lee, C.: Energy efficient scheduling for heterogeneous fog computing architectures. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). vol. 01, pp. 555–560 (July 2018). https://doi.org/10.1109/COMPSAC.2018.00085
23. Yu, H., Wang, Q., Guo, S.: Energy-efficient task offloading and resource scheduling for mobile edge computing. In: 2018 IEEE International Conference on Networking, Architecture and Storage (NAS). pp. 1–4 (Oct 2018). https://doi.org/10.1109/NAS.2018.8515731
24. Yu, Y.: Mobile edge computing towards 5g: Vision, recent progress, and open challenges. China Communications **13**(Supplement2), 89–99 (N 2016). https://doi.org/10.1109/CC.2016.7833463
25. Zhang, G., Zhang, W., Cao, Y., Li, D., Wang, L.: Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. IEEE Transactions on Industrial Informatics **14**(10), 4642–4655 (Oct 2018). https://doi.org/10.1109/TII.2018.2843365