A cost efficient model for minimizing energy consumption and processing time for IoT tasks in a Mobile Edge Computing environment

João Luiz Grave Gross[1], **Cláudio Fernando Resin Geyer**[1]

[1]Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil, 91509–900

{jlggross, geyer}@inf.ufrgs.br

Abstract. *In a scenario with increasingly mobile devices connected to the Internet, data-intensive applications and energy consumption limited by battery capacity, we propose a cost minimization model for IoT devices in a Mobile Edge Computing (MEC) architecture with the main objective of reducing total energy consumption and total elapsed times from task creation to conclusion. The cost model is implemented using the TEMS (Time and Energy Minimization Scheduler) scheduling algorithm and validated with simulation. The results show that it is possible to reduce the energy consumed in the system by up to 51.61% and the total elapsed time by up to 86.65% in the simulated cases with the parameters and characteristics defined in each experiment.*

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 [Al-Fuqaha et al. 2015]. According to an IDC report, by 2025, there will be 41.6 billion IoT devices generating 79.4 ZB of data [IDC 2019]. *Mobile Edge Computing* (MEC) is a network architecture designed to provide low latency and better QoS [Haouari et al. 2018] to end-users. It is focused on mobile networks such as 5G, ideal to the connectivity of current mobile devices. Most applications today tend to offload task processing and data storage to remote servers, usually to *Data Centers* in the Cloud, geographically located away from the end-user and the IoT device, adding too much communication latency [Aijaz 2016]. MEC allows provisioning of *Cloud Computing* (CC) services close to mobile devices, bringing processing and storage closer to cellular base stations [Yu 2016].

However, energy consumption is still an open issue on mobile device networks, such as MEC environments [Sarangi et al. 2018]. Most IoT sensors and actuators are small and run on batteries, having huge energy limits. Given the fact that IoT devices handle many data, and require much energy to process them, reducing energy consumption in networks with IoT devices is a goal worth exploring. To solve this problem, a cost-minimization model was developed for IoT devices in a MEC environment that does communicate with the Cloud and has as its main objective minimize the total energy consumed by the system and the total elapsed time from task creation to conclusion. Also, we propose the TEMS scheduling algorithm that implements the cost minimization model, calculating the cost associated with the allocation options in the system and choosing one that yields the lesser cost for the task's execution. The experiments show that our approach is energy and time effective in different test case scenarios.

The remainder of this paper is organized as follows. Section 2 presents some of the previous related work found in the literature. Section 3 presents the 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) remote processing in the Cloud. Section 4 introduces the heuristic TEMS scheduling algorithm designed to solve the system cost minimization model. Section 5 details the implementation and shows the results of the experiments using the TEMS scheduling algorithm. Finally, Section 6 concludes the paper.

2. Related Work

The development of cost models that aim to reduce energy consumption and response latency to applications in IoT systems is a topic discussed since the creation of the first Edge Computing architectures [Satyanarayanan et al. 2009]. 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. There are few works that use CC as an option to download tasks [Sarangi et al. 2018] [Yu et al. 2018]. Others, such as in [Wan et al. 2018], [Wu and Lee 2018], and [Gedawy et al. 2018], use Fog Computing to offer local processing to IoT devices, but without using CC. The use of MEC occurs in [Sarangi et al. 2018], [Zhang et al. 2018], [Wang et al. 2018], and [Yu et al. 2018]. 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 the cost minimization model, the energy consumption of task processing is unanimous, used by all works mentioned. However, the energy consumption for data transmissions is restricted to [Zhang et al. 2018], [Gedawy et al. 2018], and [Wang et al. 2018]. The processing time of tasks is also used by the majority, with the exception of [Wan et al. 2018] and [Wu and Lee 2018], while the time spent on data transmissions is restricted to [Sarangi et al. 2018], [Zhang et al. 2018], [Gedawy et al. 2018], [Wang et al. 2018], and [Yu et al. 2018]. Energy consumption when there is no processing, that is, with the equipment in idle state, is used only in [Wan et al. 2018] and [Wu and Lee 2018]. Our model also includes the battery level of the IoT devices, used only in [Zhang et al. 2018], [Gedawy et al. 2018], and [Wang et al. 2018].

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 [Chen et al. 2018] [Jin and Goto 2012]. These combined characteristics enable a more refined approach aiming to reduce both energy and time consumed for task allocation.

3. System Cost Minimization Model

In this section we present the detailed proposed cost minimization model with three task allocation policies, (1) local processing in the IoT device, (2) local processing in the MEC server, and (3) remote processing in the Cloud.

3.1. 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 have 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 $A_i = (C_i, s_i, d_i, t_i)$, for $i \in A$.

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 size and data entry and t_i represents the tasks deadline, that is, the maximum time to complete the task. Tasks with deadline equal to zero are not critical tasks. Also, if a task i has a communication channel associated to it, then $h_i = w$, otherwise $h_i = 0$. Thus, $h_i \in W \cup 0$ and h_i is an element of set H, where $H = \{1, 2, 3, ..., a\}$.

3.2. Local Computing in the IoT Device

Each $j \in D$ mobile device on the network has one or more CPU cores. Thus, the processing cores available on 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 ($C_{local,j,k}$), depending on the chip architecture [Yu et al. 2018], 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 [Tanenbaum and Austin 2012], as well as the dynamic energy consumed during the execution which is $\propto CV^2f$ [Liu et al. 2007].

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

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

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

Considering battery level and latency as model constraints, a D_j device must decide whether it is more appropriate to perform 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 are used to represent the weight of time and energy, respectively. As mentioned in [Wang et al. 2018] these coefficients work as a trade-off between execution time and energy consumption, being used to prioritize minimizing one of the costs.

3.3. Local Computing in the MEC Server

Local MEC servers, as in IoT devices, 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 ($f_{server,j,k}$), an energy capacitance coefficient of the chip architecture ($C_{server,j,k}$) and a supply voltage ($V_{server,j,k}$).

Communications that take place between the IoT device and the local server and use the same wireless channel cause mutual interference between the each other, expressed as I_i. In this case, the data transfer rate $r_i(h_i)$ for offloading the task *i* in the corresponding channel h_i is attenuated according to *Shannon's formula* [Yu et al. 2018]:

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

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

For Equation 5, W is the channel bandwidth in *Hz*, $g_{j,m}$ represents the channel gain between the mobile device *m* and a server local, represented by *j*. The variable p_m represents the transmission power of the mobile device *m* when offloading the 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 transmission on 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 a $j \in D$ mobile device to transmit data and download results from the server is 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 $T_{i,mec-up}(h_i)$ and $T_{i,mec-up}(h_i)$ and the task execution time $T_{i,mec}$, calculated the same way as for the IoT devices. The total time 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 transmission of data 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 ($p_{wireless}$) times the elapsed time ($T_{i,mec-up}(h_i)$ or $T_{i,mec-down}(h_i)$). 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 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:

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

3.4. Remote Computing in the Cloud

The Cloud 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 with some more components such as time spent and energy consumed during data transmission between MEC server and the Cloud, added to those of the transmission from IoT devices to MEC servers. The total elapsed time and total energy consumption for the Cloud are as follows:

$$T_{i,cloud,total} = T_{i,mec-up}(h_i) + T_{i,mec-cloud-up} + T_{i,cloud} + T_{i,mec-cloud-down} + T_{i,mec-down}(h_i) \quad (12)$$

$$E_{i,cloud,total} = E_{i,mec-up}(h_i) + E_{i,mec-cloud-up} + E_{i,cloud} + E_{i,mec-cloud-down} + E_{i,mec-down}(h_i) \quad (13)$$

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

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

The *idle* state of the Cloud 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 if considering energy consumption for CPU cores in idle state.

3.5. System Minimization Cost 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 minimizing equation and the total system cost is given by the sum of all tasks costs and idle energy costs, calculated for the hardware that is waiting for processing load.

$$Cost_i = min(\alpha * Costi, local, \beta * Cost_{i,mec}, \gamma * Cost_{i,cloud}) \quad (15)$$

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

3.6. Constrains on Battery Level for IoT Devices

A healthy battery level is essential for the proper functioning of IoT devices. If the battery level $B_{i,local}$ of a battery from device i 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 $B_{i,local}$ reaches zero, all tasks generated by the device i are canceled. Therefore, all cost equations for allocating tasks on the IoT device are subject to the following constrains: $B_{i,local} > E_{i,local}$, $B_{i,local} > E_{i,mec-up}(h)$ and $B_{i,local} > LSL$.

4. Time and Energy Minimization Scheduler (TEMS)

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

In Algorithm 1 step 1 defines the data sets of the IoT devices, MEC servers and communication channels. The battery level of the IoT devices is collected and the LSL is set. The algorithm collects the number of CPUs available in each IoT device and in the MEC servers, their operating frequencies and supply voltages.

In step 2 tasks are classified between critical and normal tasks. The time and energy consumption for task processing on the different CPU cores of the network are

Algorithm 1: Time and Energy Minimization Scheduler (TEMS)

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

calculated, as well as the time and energy consumption of the data transmissions, for MEC servers and for the Cloud. Critical tasks are ordered by deadline, and allocated by lowest total time. Then normal tasks ordered by creation time and allocated by minimum total cost. Critical tasks are the first to be allocated 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 being executed in the Cloud don't need to release resources since the Cloud 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 are not mobile and 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's battery level runs out.

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

5. Implementation and Evaluation

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

5.1. Simulated hardware architecture and application parameters

The simulated network considered as IoT devices Arduino Mega 2560 boards, 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. MEC servers were composed 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 1500 MHz, 1000 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 nF. For the Cloud 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[1].

For data transmissions, it was established that both 5G and fiber optics communications could reach speeds of up to 1 Gbps and latency was set at 5ms [Gupta and Jha 2015] [Brogi et al. 2018]. Also two vehicular applications were defined [Jansson 2005] for the experiments as shown in Table 1.

5.2. Experiments and Results

The scenarios tested in the experiments varied parameters such as computational load of each task, coefficients for energy consumption and elapsed time, size of data entry and results, task generation rate, critical task deadline, level of batteries for IoT devices and

[1]Technical can be found in the electronic components *datasheets*.

Table 1. Characteristics of the chosen applications.

Characteristics	Application 1	Application 2
Task generation rate (seconds)	10	0,1
Data entry (mega *bytes* - MB)	36,3	4
Results (bytes)	1250	625
Computational load (millions of CPU cycles)	2000	20
Critical Tasks (%)	10	50
Deadline for critical tasks (ms)	500	100

use of DVFS to see how the TEMS algorithm would respond to task allocation and energy and time reduction. Some of the results are discussed below.

5.2.1. Varying the tasks computational load

For this experiment, a scenario with 500 tasks and 100 IoT devices was chosen so that each IoT device could create five tasks. Two different cases were designed, (1) with 1 MEC server and (2) with 2 MEC servers. In Figure 1 it is shown the results for the load of Application 1, that is specified in Table 1.

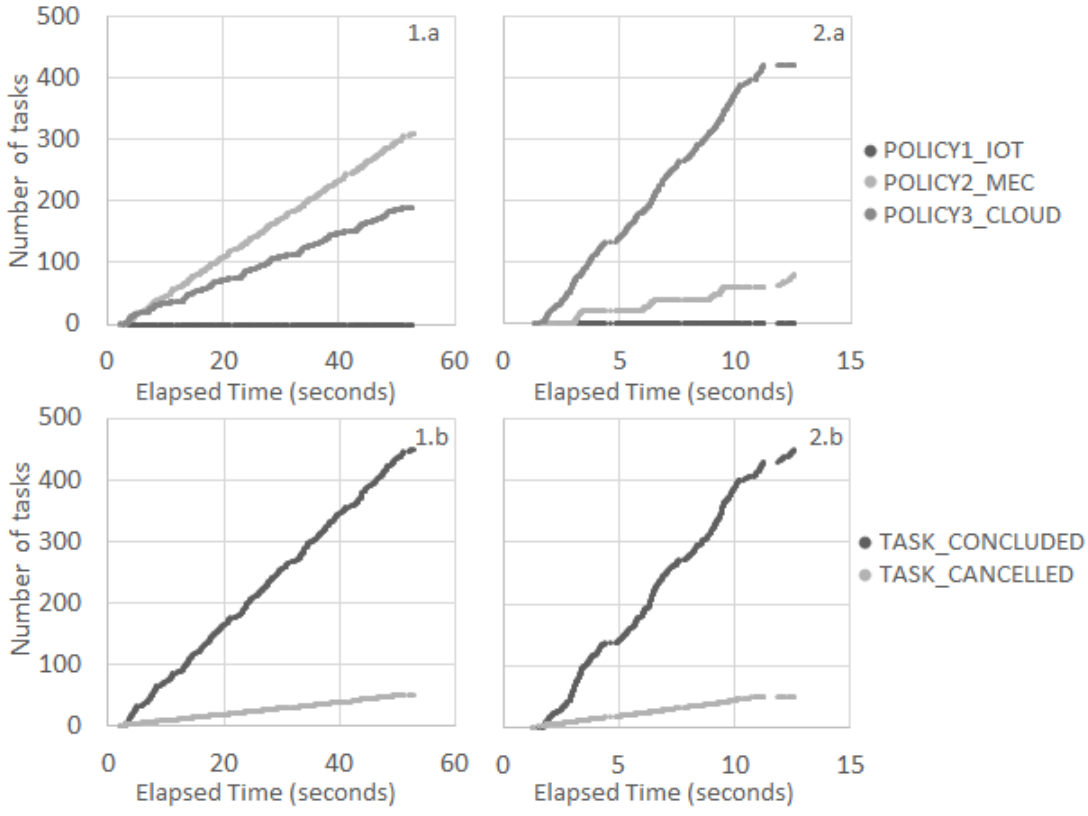

Figure 1. Processing load for Application 1 defined to $2.000 * 10^6$ CPU cycles.

The energy and time coefficients were set, respectively, at $4/5$ and $1/5$, that is, a high weight was given to the energy consumed, so that it could be minimized. In Figure 1 from 1.a to 2.a the increase in the number of MEC servers made fewer tasks to allocated in the Cloud. This positively impacts the total energy consumed. Compared to a 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 load compared to Application 1, the allocation profile changed. Most allocations took place on the device itself, regardless of the number MEC servers. The cause of this phenomenon is due to the low processing load 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. Therefore, for a low processing load, IoT devices are the first allocation option.

5.2.2. Varying coefficients for energy consumption and elapsed time

This experiment used Application 2 with $2*10^6$ CPU cycles, instead of $20*10^6$. Each case has 500 tasks, 100 IoT devices, and one MEC server. The coefficients used for energy were $1/5$, $2/5$, $3/5$, $4/5$, and the time coefficients were respectively $4/5$, $3/5$, $2/5$ and $1/5$. These pairs of coefficients were chosen to observe how prioritizing the minimization of energy or time could impact the results.

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

Case	Cost	E_{Total}	T_{Total}	f (*MHz*)	T (*V*)	Policy
$C_{E1.0}$	0,01859	0,14550	0,03336	1.500	1,200	MEC
$C_{E2.0}$	0,02597	0,14276	0,03469	750	0,825	MEC
$C_{E3.0}$	0,03318	0,14276	0,03469	750	0,825	MEC
$C_{E4.0}$	0,03544	0,07040	0,25000	8	4,000	IoT

Table 2 lists minimum costs perceived by the system task scheduler for each case. Cases 2 and 3 had its lowest cost in the same allocation option. In Case 1 the lowest cost occurred for the MEC server, with DVFS configured to 1500 MHz and 1.2 V. 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.

To reduce energy consumption the best option is $4/5$ and $1/5$ for energy and time coefficients, reducing energy consumption up to 51.61% for normal tasks compared to the other cases. If the focus is on reducing task completion time, setting energy coefficients to $1/5$, $2/5$ and $3/5$, with their corresponding time coefficients, one can achieve a reduction of up to 86.65% in the completion of normal tasks compared to the most costly case.

5.2.3. Other experiments and Findings

Varying the size of the application data entry had a huge impact on energy consumption and elapsed times. Big data entry consumes a lot of energy and take to long to transfer data. An approach with more tasks and less data per task had 23.92% and 29% cost decreases for energy and time in the simulated scenarios.

For critical task deadline it was observed that very low deadlines presented problems for the simulated applications, since the available allocation policies had difficulty in completing the critical tasks in an adequate time. This caused several tasks to be cancelled.

Experiments for battery level showed that very low battery levels quickly reach LSL and make IoT devices unavailable for processing. Very high computational loads also impacts battery level negatively. Therefore, a battery with a healthy energy level and adequate task processing loads, allows the allocation to be carried out on the IoT device, without making it unavailable due to lack of battery.

Finally, experiments with DVFS activated had total energy consumption decreased by 13.736%, while the total time increased by 28.32%, while compared to DVFS off. This demonstrates the effectiveness of the system model, as well as the scheduling algorithm, in minimizing total energy consumption. The total time may have been longer in the approach with DVFS, but it is not problematic because the application tasks were completed within the time limit imposed by the deadline.

6. Conclusion

In an environment with an 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 was 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 the 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 transmissions to the model, accounting for idle costs, calculating transmission rate interference, use of the DVFS technique, and also the interaction with CC to provide resources whenever the local network becomes saturated. Further analysis will be made for new cost variables to the model, such as financial and storage costs, which can help make more assertive task allocations.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

Aijaz, A. (2016). Towards 5g-enabled tactile internet: Radio resource allocation for haptic communications. In *2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 145–150.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Brogi, A., Forti, S., and Ibrahim, A. (2018). Deploying fog applications: How much does it cost, by the way? In *CLOSER*.

Chen, Y.-L., Chang, M.-F., Yu, C.-W., Chen, X.-Z., and Liang, W.-Y. (2018). 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.

Gedawy, H., Habak, K., Harras, K. A., and Hamdi, M. (2018). 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)*, pages 191–196.

Gupta, A. and Jha, R. K. (2015). A survey of 5g network: Architecture and emerging technologies. *IEEE Access*, 3:1206–1232.

Haouari, F., Faraj, R., and AlJa'am, J. M. (2018). Fog computing potentials, applications, and challenges. In *2018 International Conference on Computer and Applications (ICCA)*, pages 399–406.

IDC (2019). The growth in connected iot devices is expected to generate 79.4zb of data in 2025. Available at: `https://www.idc.com/getdoc.jsp?containerId=prUS45213219`.

Jansson, J. (2005). *Collision Avoidance Theory with Application to Automotive Collision Mitigation*. PhD thesis.

Jin, X. and Goto, S. (2012). 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.

Liu, Y., Yang, H., Dick, R. P., Wang, H., and Shang, L. (2007). Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In *8th International Symposium on Quality Electronic Design (ISQED'07)*, pages 204–209.

Sarangi, S. R., Goel, S., and Singh, B. (2018). Energy efficient scheduling in iot networks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 733–740, New York, NY, USA. Association for Computing Machinery.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

Tanenbaum, A. S. and Austin, T. (2012). *Structured Computer Organization*. Prentice Hall, 6th edition.

Wan, J., Chen, B., Wang, S., Xia, M., Li, D., and Liu, C. (2018). Fog computing for energy-aware load balancing and scheduling in smart factory. *IEEE Transactions on Industrial Informatics*, 14(10):4548–4556.

Wang, C., Dong, C., Qin, J., Yang, X., and Wen, W. (2018). Energy-efficient offloading policy for resource allocation in distributed mobile edge computing. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00366–00372.

Wu, H. and Lee, C. (2018). Energy efficient scheduling for heterogeneous fog computing architectures. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 555–560.

Yu, H., Wang, Q., and Guo, S. (2018). Energy-efficient task offloading and resource scheduling for mobile edge computing. In *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–4.

Yu, Y. (2016). Mobile edge computing towards 5g: Vision, recent progress, and open challenges. *China Communications*, 13(Supplement2):89–99.

Zhang, G., Zhang, W., Cao, Y., Li, D., and Wang, L. (2018). Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Transactions on Industrial Informatics*, 14(10):4642–4655.