Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 4-A, pp. 1091–1103
DOI : 10.47974/JIOS-1894

An MCDM and partial computations based deadline and energy-aware real-time workflow scheduling technique for fog integrated cloud environment

Rishika Mehta *
Department of Computer Science & Engineering
School of Engineering & Technology
The NorthCap University
Gurugram 122017
Haryana
India

Shilpa Mahajan †
Department of Computer Science & Engineering
School of Engineering & Technology
The NorthCap University
Gurugram 122017
Haryana
India

Kavita Khanna §
Department of Computer Science
Delhi Skill and Entrepreneurship University
Dwarka 110077
New Delhi
India

Abstract

The phenomenal rise of Internet of Things(IoT) has driven the ascend of fog computing as a distributed model to reduce delays in networks. Generally, fog devices are resource-restricted while IoT applications are becoming computationally demanding which require certain QoS to be accomplished within hard time limits. Therefore, it is preferable for IoT jobs to finish their processing within deadline limits by producing approximate results than producing accurate result late. The placement of IoT jobs on fog and cloud resources for

* *E-mail:* **rishikamehta10@gmail.com (Corresponding Author)**
† *E-mail:* **shilpa@ncuindia.edu**
§ *E-mail:* **Kavita.khanna@dseu.ac.in**

execution is a widely-recognized NP-hard problem. We study the placement of real-time IoT workflows in a fog and cloud infrastructure by applying approximate computations and TOPSIS. Our methodology aims to place complete as well as partial tasks in the available idle schedule holes in the schedules of fog as well as cloud resources. The proposed technique is verified through simulation experiments and is contrasted with state-of-the-art techniques on different performance measures. The experimental findings establish that the proposed technique is able to render better performance compared to its alternatives in terms of performance metrics like SLA violation ratio, response time and energy consumption at an insignificant loss of 0.15% of result precision for all experimental scenarios that are taken into consideration.

Subject Classification: *00A72.*

Keywords: *Internet of Things, Topsis, Real-time scheduling, Partial computations.*

1. Introduction

Since *Internet of Things* persists in incorporating a plenitude of sensors and end-devices, an unparalleled data amount and variety is produced at an astounding rate, increasingly demanding processing within stringent time limits [1-4]. IoT data are typically handled by real-time workflows comprising of inter-dependent tasks where one task's output is utilized as input by subsequent application tasks.

Therefore, a successor task cannot begin to execute until every predecessor has completed processing and the corresponding data and control dependencies are satisfied. Since IoT has grown exponentially, fog computing surfaced as a novel paradigm to complement cloud computing [5]. Furthermore, geographical proximity impacts end-to-end delay, therefore, the fog brings the compute resources to the network edge which reduces the transmission of massive amount of IoT data to the cloud [6-8]. Generally, only particular type of data—for instance those requiring perennial storage—are submitted to the cloud. Fog computational resources are virtualizable, just like cloud computing resources. Subsequently, a fog node can act as a virtual machine [16]. In contrast to the resources in a cloud environment, the quantity and processing power of the computational instances in a fog environment are usually constrained. Often IoT data needs to be processed under stringent time limits. In this type of real-time scenario, the timeliness of the results is as important as their precision [9-10].

In view of this, Lin et al. introduced partial computations method that allows a real-time task to yield intermediate and imperfect outcome within the deadline limit [11]. Every workflow task in this approach is assumed to be monotone, meaning that every task is split into the mandatory portion that produces a proximate outcome of the least feasible quality while an optional portion ameliorates the outcome of the mandatory portion [12]. For the task to yield an acceptable outcome, the mandatory portion of the task must be finished. The task's precision increases if the optional part is allowed to execute for an extended amount of time. The technique of schedule hole utilization has recently been implemented for a fog-cloud integrated environment. However, existing

research studies don't make use of available schedule holes to place partial tasks. In addition, no Multi Criteria Decision Making Technique has been used to reduce response time of the IoT jobs and simultaneously improve Energy Savings of the fog and cloud computational resources by incorporating partial computations.

A scheduling strategy that reduces an application's response time at the disbursement of higher energy consumption is not the best option for fog and cloud resources. This gets significantly more challenging when there are several conflicting objectives that must be met concurrently.

Specifically, it is challenging to guarantee adherence to deadlines by simultaneously reducing response time and increasing energy savings of the computational resources. Therefore, a multi-objective optimization method is required to establish the proper balance between several optimization objectives. To achieve this, we make use of the most commonly utilized MCDM technique Topsis in addition to approximate computations to schedule different workflow tasks in fog integrated cloud infrastructure.

Following is the organization for the rest of the paper. Section 2 discusses the relevant works on task scheduling in fog-cloud infrastructure. Section 3 describes the system model, workload model, approximate computations model and energy consumption model utilized in the work. Section 4 discusses the proposed workflow scheduling methodology. Performance assessment on the basis of quantitative measures is elaborated in section 5 while concluding statements are specified in Section 6.

2. Related Work

In heterogeneous environments, one of the most popular job scheduling techniques is the Heterogeneous Earliest Finish Time (HEFT) methodology, which was introduced by Topcuoglu et. al. [13]. The maximum priority task is chosen during the task selection stage which is subsequently allocated to the appropriate processor which can offer least finish time by exploiting idle time gaps in its schedule.

The Predict Earliest Finish Time (PEFT) method, an improved variant of HEFT, is presented by Arabnejad and Barbosa in [14]. They introduced a feature of look ahead built on an OCT. With regard to schedule length ratio, the presented methodology performs better than HEFT.

The task scheduling strategies in fog integrated cloud environments have been progressively gaining more attention [15-17]. Toor et al. [18] developed an energy-aware task placement technique in a fog-cloud scenario using DVFS. This methodology adjusted the speed of computing instances to deal with increased requests and high consumption of energy in the fog infrastructure. A number of simulation experiments were run to assess the efficacy of this policy. However, the workload did not consist of monotone nature of the workflows, therefore, no partial computations were used. Further workflow deadlines were also not taken into consideration.

Pham et al. [19] presented an application scheduling strategy which aimed at achieving a compromise between makespan and cost by utilizing idle schedule slots. Since IoT applications are dynamic, this static method is inappropriate for usage in real-time. Further, it simply took into account single scheduling workflow and also cost of communication is not taken into consideration during the scheduling process.

With the objective to offer adaptable implementation frameworks for mobile devices, Mora Mora et al. [20] presented a technique that made use of partial computations. However, the workload comprised standalone tasks and not jobs. Additionally, in this strategy, the tasks were not split into mandatory and optional portions instead few tasks were presumed to be mandatory and few to be optional.

Stavrinides and Kratza [21] presented deadline and cost aware workflow scheduling strategy for fog and cloud environment. They considered schedule gap utilization only when entire task could occupy a schedule gap. However, they did not consider monotone tasks due to which partial computations was not utilized in this strategy.

Conversely, the fog and cloud-aware strategy presented in this research can be used to dynamically schedule numerous real-time workflows, making use of potential schedule gaps even in situations where only a portion of the task can be executed. Additionally, the communication cost is taken into consideration while ranking the workflow tasks which is not considered in the existing works.

3. Problem Formulation

3.1 *System Model*

This work considers a three-tiered IoT-fog-cloud infrastructure. The IoT device tier comprises of devices and sensors that perform transmission of data to the fog tier. The fog tier comprises of h^{fog} physical hosts. There is a set $V^{fog} = \{v_1^{fog}, v_2^{fog}, \ldots, v_m^{fog}\}$ of m^{fog} VMs in the fog tier where every virtual machine is allotted a vCPU.

The cloud environment comprises of h^{cloud} physical hosts with heterogeneous processors. There is a set $V^{cloud} = \{v_1^{cloud}, v_2^{cloud}, \ldots, v_m^{cloud}\}$ of m^{cloud} virtual machines in the cloud tier where every virtual machine is allotted a vCPU. A fog or cloud virtual machine's virtual CPU (vCPU) is equivalent to the physical core of the pertinent device. The operational frequency f_k of a virtual machine vm_k is the operational frequency of its corresponding core. Task scheduling for the virtual machines (VMs) of both cloud and fog tiers is done by a central scheduler operating at a particular device within the fog tier.

3.2 *Workload Model*

A workflow is depicted as a Directed Acyclic Graph (DAG) G = (N, E), where N signifies the set of vertices of the graph whereas E signifies the set of directed edges connecting the vertices. Every vertex depicts a task n_p of the workflow while the data to be conveyed from task n_p to n_q is depicted by a

directed edge e_{pq}. For the remainder of the paper, the words DAG, workflow and job will be used interchangeably. Workflow component tasks are considered to be non-preemptible since doing so could negatively impact the performance [22-23]. Every workflow task has a weight w_p which indicates its computational volume and is defined as the count of clock cycles needed to complete the respective task's instructions. On a virtual machine (VM) vm_q, the computational cost of task n_p is computed as follows:

$$Comp(n_p, vm_q) = w_p / f_q \quad (1)$$

The edge e_{pq} connecting two tasks n_p and n_q is assigned a weight z_{pq} that signifies its communication volume, i.e., amount of data required to be transmitted from the task n_p to n_q. The edge e_{pq} incurs a communication cost when data is moved from task n_p (placed on vm_i) to task n_q (placed on vm_j). It is defined as follows:

$$Comm((n_p, vm_i), (n_q, vm_j)) = z_{pq}/l_{ij} \quad (2)$$

where l_{ij} is the communication link's data transmission rate between the virtual machines vm_i and vm_j. If tasks n_p and n_q are scheduled on the same virtual machine (VM), then edge e_{pq}'s communication cost is deemed insignificant.

In a workflow, a path's length is determined by adding up the computational and communication expenses of each task and edge respectively lying on the path. The critical path length (CPL) in a graph is the lengthiest path. Every real-time job has a predetermined deadline D by which all of the tasks that make up the workflow must be completed. It is formulated as:

$$D = A + RD \quad (3)$$

where, A is the workflow arrival time and RD signifies its relative deadline.

3.3 *Approximate Computations Model*

The computational volume w_i of every workflow task n_i is constituted by a mandatory portion mp_i, and an optional portion op_i:

$$w_i = mp_i + op_i \quad (4)$$

where, $0 \leq mp_i \leq w_i$.

A task is considered to be complete when at least its mandatory part has finished its execution. The outcome of an incompletely executed task n_i are proximate and hence the task produces output error, which is computed as follows:

$$OE_i = \frac{\delta_i}{op_i} \quad (5)$$

where, δ_i is the repudiated portion of the task's optional portion op_i.

The task n_i's input error is computed by adding up the output errors of its predecessors:

$$IE_i = \sum_{n_j \varepsilon P_i} OE_j \quad (6)$$

where, P_i depicts the set of predecessors of the task n_i.

The execution time of a task is affected if there is an input error. Particularly, there is an extension in its mandatory part which is given by:

$$mpe_i = mp_i . IE_i \quad (7)$$

Where, IE_i and mp_i represent the input error and the mandatory portion of task n_i, respectively. There is an input error limit IEL_i for each component task n_i, above which the error in its input cannot be managed. The IEL_i takes values in the range $[1,P_i]$.

3.4 *Energy Consumption Model*

The estimated energy consumption of a fog virtual machine vm_j^{fog} during execution of a task n_i is given by:

$$\text{EECActive}_j^{\text{fog}} = \sum_{n_i \varepsilon G:G \varepsilon EW}(alloc_{ij}^{fog} \times \text{Comp}(n_i, vm_j^{fog}) \times \text{ECActive}_j^{\text{fog}}) \quad (8)$$

where, $alloc_{ij}^{fog} = 1$, if task is placed on fog VM while $\text{Comp}(n_i, vm_j^{fog})$ is the execution cost of task n_i on vm_j^{fog}.

The estimated energy consumption of a cloud VM vm_j^{cloud} while executing a task n_i is given by:

$$\text{EECActive}_k^{\text{cloud}} = \sum_{n_i \varepsilon G:G \varepsilon EW}(alloc_{ik}^{cloud} \times \text{Comp}(n_i, vm_k^{cloud}) \times \text{ECActive}_k^{\text{cloud}}) \quad (9)$$

where, $alloc_{ij}^{cloud} = 1$, if task is placed on cloud VM while $\text{Comp}(n_i, vm_j^{cloud})$ is the execution cost of task n_i on vm_j^{cloud}. It is vital to calculate the energy consumption because of the unoccupied schedule gaps that may not be utilized to place any workflow task.

$$EECIdle_j^{fog} = [\sum_{n_i \varepsilon G:G \varepsilon EW}^{v_l \varepsilon V^{fc}}(alloc_{il}^{fc} \times \text{Comp}(n_i, vm_l^{fc})) - \sum_{n_i \varepsilon G:G \varepsilon EW}(alloc_{ij}^{fog} \times \text{Comp}(n_i, vm_j^{fog}))] \times ECIdle_j^{fog} \quad (10)$$

$$EECIdle_k^{cloud} = [\sum_{n_i \varepsilon G:G \varepsilon EW}^{v_l \varepsilon V^{fc}}(alloc_{il}^{fc} \times \text{Comp}(n_i, vm_l^{fc})) - \sum_{n_i \varepsilon G:G \varepsilon EW}(alloc_{ik}^{cloud} \times \text{Comp}(n_i, vm_k^{cloud}))] \times ECIdle_k^{cloud} \quad (11)$$

where, V^{fc} represents the set of all fog and cloud VMs and EW depicts an ensemble of completed workflows.

The overall energy consumption of a fog VM vm_j^{fog} is given as:

$$EEC_j^{fog} = \text{EECActive}_j^{\text{fog}} + EECIdle_j^{fog} \quad (12)$$

The overall energy consumption of a cloud VM vm_j^{cloud} is given as:

$$EEC_k^{cloud} = \text{EECActive}_k^{\text{cloud}} + EECIdle_k^{cloud} \quad (13)$$

Thus, the overall energy consumption of all VMs is given by:

$$TEC = \sum_{v_j \varepsilon V^{fog}} EEC_j^{fog} + \sum_{v_k \varepsilon V^{cloud}} EEC_k^{cloud} \quad (14)$$

4. Scheduling Methodology

4.1 *Task Selection Stage*

Each task is ranked on the basis of its workflow's deadline. The task with the lowest deadline gets the uppermost rank. As a tie-breaking rule, the task having largest mean computational cost gets prioritized.

4.2 *Processor Selection Stage*

When the scheduler selects a task, then its expected completion time and expected energy consumption on a VM is computed. Every VM in the cloud and fog layer is taken into account. The Expected Energy Consumption of a component task n_i on vm_j is computed according to equations (8) and (9) depending on fog or cloud vm.

a) The ECT of task n_i on vm vm_j is calculated as:

$$ECT(n_i, vm_j) = max\{t_{avail\_data}(n_i, vm_j), t_{avail\_vm}(n_i, vm_j)\} + Comp(n_i, vm_j) \quad (15)$$

Here, the term $t_{avail\_data}(n_i, vm_j)$ denotes the time when task n_i's input will be received by vm_j and $t_{avail\_vm}(n_i, vm_j)$ depicts the availability time of vm_j so as to initiate the execution of the task n_i. To compute the value of $t_{avail\_vm}(n_i, vm_j)$, we check if an idle slot exists in the vm_j's schedule. It is computed as:

$$\text{idle\_slot} = t_{avail\_data}(n_i, vm_j) - t_{cur\_time} \quad (16)$$

b) Here, we check if entire task can fit in the identified idle slot

$$\text{idle\_slot} \geq w_i/f_j \quad (17)$$

c) In the event when entire task cannot fit in, then we determine the maximum feasible repudiated portion δ_{max_i} of the optional portion of the ready task in order to prevent exceeding of input error limit for every child task. This is provided by:

$$\delta_{max_i} = op_i \,.\, min\,\{1, \underset{n_j \varepsilon P_i}{min}(IEL_j - IE_j)\} \quad (18)$$

where P_i is the set of successors of the task n_i.

Consequently, we select the minimum ECT value and the corresponding Energy Consumption ($EECActive_j^{fc}$) for that VM. The altenative matrix (AM) is produced using the Expected Completion Time, Expected Energy Consumption, Waiting Time, and Resource Utilization metrics. Next, this alternate matrix (AM) is subjected to Topsis. The steps involved in Topsis are:

1. Normalize matrix AM to obtain values am'_{ij} of the matrix $A\,M'$
2. Every normalized column of $A\,M'$ is multiplied by its corresponding weight to yield Weighted normalized matrix BM.

$$bm_{ij} = wt_j \times am'_{ij} \quad (19)$$

Since four criteria are take into consideration, $wt_j = 0.25$.

3. From weighted normalized matrix BM', Positive and Negative Ideal Solutions are obtained.
4. Compute Eucledian distance of each VM from Positive and Negative Ideal Solutions.

$$S_i^+ = \sqrt{\sum_{j=1}^{k}\left(bm_j^+ - bm_{ij}\right)^2} \tag{20}$$

$$S_i^- = \sqrt{\sum_{j=1}^{k}\left(bm_j^- - bm_{ij}\right)^2} \tag{21}$$

5. Compute rank of each VM using the following equation

$$R_i = \frac{s_i^-}{s_i^+ + s_i^-} \tag{22}$$

The task is placed on the VM with the maximum rank.

5. Performance Evaluation

The proposed work is compared with a baseline technique that does not employ approximate computations and a state-of-the-art strategy [21]. The state-of-the-art policy, baseline policy and the proposed work are observed under difference values of Result Precision Threshold to showcase the efficiency of the proposed work. The simulation parameters employed in this work are shown below in Table 1.

Table 1
Simulation parameters

IoT Layer	
Average rate of data transmission of IoT-fog network	25 Mbps
Average rate of data transmission of IoT-cloud network	20 Mbps
Fog Layer	
Count of fog VMs	32
Operational frequency of Fog VM vCPU	2.5 – 2.9 GHz
Mean rate of data transfer in Fog network	1 Gbps
Fog VM Energy Consumption Rate during idle time	25-40
Fog VM Energy Consumption Rate during task execution	105-130
Cloud Layer	
Count of cloud VMs	32
Operational frequency of Cloud VM vCPU	3 – 3.8 GHz
Mean rate of data transfer in Cloud network	10 Gbps
Fog VM Energy Consumption Rate during idle time	75-100
Fog VM Energy Consumption Rate during task execution	150-175
Workload Characteristics	
Number of tasks in a workflow	20-40
Average data size of entry task	1 GB
Average computational volume of component task	8.93×10^{11} clock cycles
Average communication volume of edge	44.74 GB
Result precision threshold	0.7

Fig. 1 shows the SLA Violation Ratio of the proposed policy, baseline policy and state-of the art policy for different number of workflows. It is evident that for all workload values, the proposed Topsis and Approximate Computations based strategy, SawTopsis_AC, outperformed the other two strategies, yielding the least violation of SLA. This is attained at a negligible 0.15% decrease in result precision as shown in Table 2.

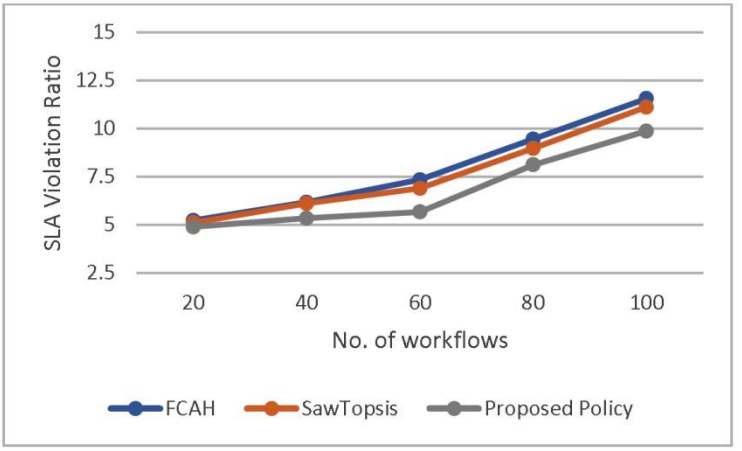

Figure 1
SLA Violation Ratio

The simulation findings in Fig.2 and Fig.3 reveal that, with regard to Response Time and Energy Consumption, the proposed technique outperforms both baseline as well as state-of-the-art policies.

Table2
Average Result Precision

No.of workflows	FCAH	SawTopsis	Proposed Policy
20	1	1	0.999427
40	1	1	0.999677
60	1	1	0.998211
80	1	1	0.998231
100	1	1	0.996616

This is achieved due to the consideration of reduction in waiting time and resource utilization improvement along with the optimization of response time and energy consumption in the proposed scheduling technique. Further, the proposed technique takes advantage of the monotone nature of the workflow tasks to utilize schedule holes where partial tasks can be placed and therefore, produces optimal schedules which are able to reduce response time and

simultaneously minimize Energy Consumption of the fog and cloud computational resources.

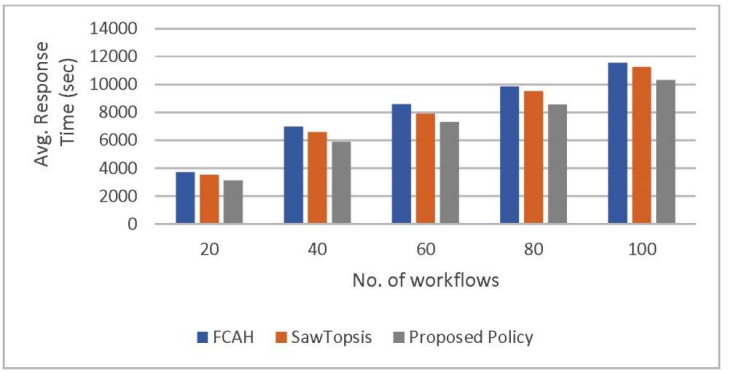

Figure 2
Average Response Time

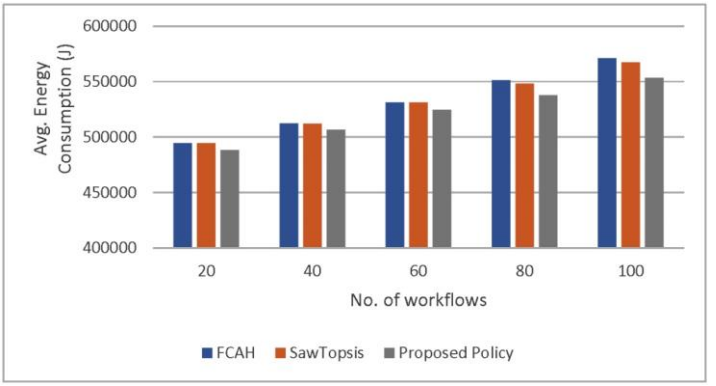

Figure 3
Average Energy Consumption

6. Conclusion

In this work, we leveraged approximate computations for scheduling several real-time IoT jobs in a heterogeneous fog integrated cloud computing infrastructure. The impact of approximations on timeliness for real-time tasks has not been thoroughly investigated in a fog-cloud environment. In lieu of this, we proposed a multi-objective optimization strategy that tried to increase energy savings while concurrently decreasing workflow response time and the number of SLA violations by compromising result quality at a negligible rate of 0.15%. The simulation findings affirmed that the proposed Topsis and Approximate Computations based workflow scheduling policy performed better than baseline as well as state-of-the-art policies with regard to response time, SLA violations and Energy Consumption.

References

[1] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer,* vol. 49, no. 8, pp. 112–116 (2016).

[2] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Comput.,* vol. 21, no. 2, pp. 44–53 (2017).

[3] G. L. Stavrinides and H. D. Karatza, "The impact of data locality on the performance of a SaaS cloud with real-time data-intensive applications," in Proc. 21st IEEE/ACM Int. Symp. Distributed Simulation and Real-Time Applications (DS-RT), pp. 1–8 (2017).

[4] R. Mehta, J. Sahni, and K. Khanna, "Internet of Things: Vision, applications and challenges," *Procedia Comput. Sci.*, vol. 132, pp. 1263–1269 (2018).

[5] R. Mehta, J. Sahni, and K. Khanna, "Task scheduling for improved response time of latency sensitive applications in fog integrated cloud environment," *Multimedia Tools Appl.*, vol. 82, no. 21, pp. 32305–32328 (2023).

[6] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.,* vol. 4, no. 2, pp. 26–35 (2017).

[7] Cisco Systems, Inc., "Fog computing and the Internet of Things: Extend the cloud to where the things are," *Tech. Rep*. C11-734435-00 (2015).

[8] Y. Jararweh, T. H. Bataineh, M. Y. Alzain, A. Y. Tahat, and M. Kharbutli, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in Proc. 23rd Int. Conf. Telecommunications (ICT), 2016, pp (1–5).

[9] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, vol. 24, Springer (2011).

[10] G. Wainer and M. Moallemi, "Designing real-time systems using imprecise discrete-event system specifications," *Softw. Pract. Exp.,* vol. 50, no. 8, pp. 1327–1344 (2020).

[11] K. J. Lin, S. Natarajan, and J. W. S. Liu, "Imprecise results: Utilizing partial computations in real-time systems," *NASA Tech. Rep. NAS* 1.26:180561 (1987).

[12] S. Yao, B. Sheng, D. Liu, W. Jiang, and Q. Li, "Scheduling real-time deep learning services as imprecise computations," in Proc. 2020 IEEE 26th Int. Conf. Embedded and Real-Time Comput. Syst. and Appl. (RTCSA), pp. 1–10 (2020).

[13] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.,* vol. 13, no. 3, pp. 260–274 (2002).

[14] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.,* vol. 25, no. 3, pp. 682–694 (2014).

[15] Y. Chen, Service-Oriented Computing and System Integration: Software, IoT, Big Data, and AI as Services, 6th ed., Kendall Hunt Publishing (2018).

[16] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.,* vol. 5, no. 4, pp. 3246–3257 (2018).

[17] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in Proc. 2017 IFIP/*IEEE Symp. Integrated Network and Service Management* (IM), pp. 1222–1228 (2017).

[18] A. Toor, M. T. Alam, S. M. Rizvi, A. Almogren, and I. Ahmad, "Energy and performance aware fog computing: A case of DVFS and green renewable energy," *Future Gener. Comput. Syst.,* vol. 101, pp. 1112–1121 (2019).

[19] X. Q. Pham, N. D. Hoang, M. H. Nguyen, M. S. Hossain, and C. S. Hong, "A cost- and performance-effective approach for task schedul-

ing based on collaboration between cloud and fog computing," *Int. J. Distrib. Sens. Netw.,* vol. 13, no. 11, pp. 1–16 (2017).

[20] H. Mora Mora, D. Gil, J. F. Colom López, and M. T. Signes Pont, "Flexible framework for real-time embedded systems based on mobile cloud computing paradigm," *Mobile Inf. Syst.,* vol. 2015, Art. no. 652462, pp. 1–14.

[21] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimedia Tools Appl.,* vol. 78, pp. 24639–24655 (2019).

[22] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, 3rd ed., Springer (2011).

[23] G. L. Stavrinides and H. D. Karatza, "The effect of workload computational demand variability on the performance of a SaaS cloud with a multi-tier SLA," in Proc. IEEE 5th Int. Conf. Future Internet of Things and Cloud (FiCloud), pp. 10–17 (2017).

Received October, 2024