A multi-objective optimization technique for scheduling real-time IoT Applications in Fog Computing Using Approximate Computations and TOPSIS

Rishika Mehta[1], Jyoti Sahni[2], Shilpa Mahajan[3], Kavita Khanna[4]

[1,3]The NorthCap University, Gurugram, India
[2]Victoria University of Wellington, New Zealand
[4]Delhi Skill and Entrepreneurship University, New Delhi, India
[1]rishikamehta10@gmail.com, [2]jyoti.sahni@ecs.vuw.ac.nz, [3]shilpa@ncuindia.edu, [4]kavita.khanna@dseu.ac.in

Abstract: In the last decade, fog computing arose as a distributed computing paradigm to handle latency-sensitive real-time IoT applications in an effective way. By utilizing fog resources, improved performance such as timely service provisioning, optimal energy usage, decreased network load, etc. can be achieved. Fog resources usually have finite computational capacities. Conversely, Internet of Things (IoT) applications are getting complicated in addition to being computationally intensive, necessitating a specific degree of QoS in stringent time restrictions. In real-time, many times it is preferable for an IoT application to complete its execution by its deadline by generating an imprecise outcome instead of yielding a delayed accurate output. We study the placement of real-time IoT applications in a heterogeneous fog infrastructure by applying approximate computations. In this technique, we considered that if a constituent task yields an inaccurate outcome, the error may not only be limited to its closest predecessor tasks but may also proliferate to the succeeding workflow tasks, thereby, affecting the overall result of the workflow. We simultaneously study the impact of error proliferation on energy consumption of fog resources. The proposed workflow orchestrating model is compared to a baseline technique and a state-of-the-art policy, where the effects of partial computations are studied under varying values of proliferation probability of input error and result precision threshold. The simulation findings reveal that the proposed technique outperformed both policies in terms of the number of deadline misses, energy savings, schedule hole utilization, and overall result precision.

Keywords: IoT workflow, Error proliferation, Topsis, Real-time scheduling, Partial computations

1. Introduction

The Internet of Things (IoT) envisions to make human life more intelligent and automated [1]. An expeditious increase in the count of IoT devices is not unanticipated in the near future. The initial IoT phases have seen an immense upsurge in IoT data production which is further likely to increase exponentially as a vast number of everyday objects, including, mobile devices, actuators, and sensors are linked to the Internet, producing at an astounding rate an unparalleled amount and variety of data. Cloud resources may be inadequate to efficiently process such large volumes of data within strict time limits [2-3].
This generates the need to utilize local computing resources to meet IoT requirements. Essentially, fog computing has emerged to bridge this void [4]. Scheduling IoT applications on fog resources enhances performance metrics, including timely service provisioning, decreased network load, and optimized energy consumption. Fog computing supplements cloud computing by providing an extra computing infrastructure layer between Internet of Things devices and cloud resources. Such an interconnected infrastructure is termed a fog-integrated cloud [5]. IoT data often requires processing within strict time constraints. The accuracy and utility of the results in such a real-time scenario rely not only on their result quality but also on timeliness [6-7]. Generally, real-time complex applications comprising various constituent tasks with antecedence and data constraints, process the IoT data. In other words, every job forms a workflow, wherein the outcome of a task is utilized by subsequent tasks of the job [8]. Only when all its predecessors have been completed can a child task begin to be executed. Usually, every job is associated with a fixed deadline by which all of its tasks must be finished. A delayed result would be futile otherwise. As a result, it is preferable for an application to give an inexact outcome by its deadline as opposed to an accurate delayed outcome. It is to be noted that the terminologies workflow, job, and application are used interchangeably.
Considering this, Lin et al. presented an approximate computations strategy, where, a workflow is permitted to produce in-between and inexact results of lower yet acceptable quality by its deadline [9]. In this technique, each constituent task is presumed to be monotone which means that every task is divided into a mandatory portion that generates an inexact result of the lowest permissible quality, and an optional portion that improves the outcome of the mandatory portion [29]. For a task to yield a satisfactory result, the task's mandatory portion must be finished. If a task's optional portion is permitted to be processed for a prolonged period, the task's result precision gets further raised (e.g. video streaming and statistical guesstimation) [10]. When a parent task yields imprecise results, it is imperative to analyse the input error proliferation impact from parent to child tasks because of the data interdependence among the constituent tasks of the workflow.
Majority of the research studies consider that error in parent's output is always handled by immediate child tasks [15,18]. Only a few research studies consider the case where error would proliferate to subsequent tasks of the workflow [4]. Subsequently, the impact of approximate computations on computation time of the workflow tasks and energy consumption of fog computational resources need to be examined simultaneously. Furthermore, no

Multi Criteria Decision Making (MCDM) technique is utilized in the literature for selecting computational resources when workflow tasks are monotone in nature. This applies not only to fog environments but also to other distributed computing environments. Consequently, the effect of error proliferation among the tasks of the workflow in addition to its effect on the computation time of every impacted task and energy consumption of each computational resource should be examined while taking the limited computation capabilities of fog resources into consideration.

Towards this end, we propose the placement of various real-time jobs in fog computing by utilizing approximate computations. The primary contributions of our proposed policy are as follows:

a) The workflows are first ranked based on Least Deadline First technique. The constituent tasks are further prioritized on the basis of higher value of Expected Processing Time which considers both computation and communication expenses.
b) The computing instances are ordered based on Expected Finish Time, Energy Consumption, as well as Computational Expense by utilizing Multi Criteria Decision Making technique - Topsis.
c) The best fit idle slot utilization technique is considered to place constituent tasks in the available schedule holes thereby considering the effect of error proliferation on the computation time of every influenced task and energy consumption of each computational resource.
d) The proposed technique is contrasted with a Topsis-based baseline strategy and a state-of-the-art strategy. Experimental outcomes establish that the proposed technique yields considerably enhanced results in comparison to both policies.

The paper is arranged into six sections: Section 2 discusses some of the latest research works on approximate computations. Section 3 describes the fog computing infrastructure, the workload model, the approximate computations and error proliferation model as well as the energy consumption model. The presented placement methodology is presented in Section 4. Section 5 discusses the utilized assessment parameters and examines the simulation outcomes. The paper is summarized and concluded in Section 6, providing insights for further research.

2. Related work

A number of works on approximate computations-based workflow scheduling in non-fog environments have been proposed in the literature which are as follows:

Feng and Liu [10] devised a real-time placement methodology that makes use of imprecise computations. The authors demonstrated how input error influenced the static placement of a linear deadline-bound job by scheduling it on a single processor. The mandatory as well as optional portions of the predecessor task were prolonged to address and fix the error generated by the parent tasks. That is, the scenario in which an input error could be passed on to the subsequent child tasks of the workflow was overlooked.

A heuristic for placing real-time jobs with imprecise computations on multiple processor systems was presented by Ravindran et al. [11]. The authors proposed a simplistic approach to compute collective output error of exit tasks with the goal of increasing the result quality while accounting for limited computation capability and energy consumption of fog resources. The authors utilized imprecise computations technique and input error but error proliferation and its influence on the execution time of every constituent task was not taken into consideration.

Esmaili et al. [12] introduced a heuristic approach for scheduling time-sensitive tasks by utilizing imprecise computations on multiple processors. Even with limited energy resources, the proposed approach was able to find optimal schedules. This proposed strategy was compared with a Mixed Integer Linear Program where in a few instances, the proposed strategy achieved the same QoS values as those yielded by the latter approach. While both strategies considered the implications of input error when the results produced by the parent task were imprecise, the idea behind both was that the input error would invariably be offset by the extension of every impacted predecessor task's mandatory component. Therefore, both strategies failed to consider how error from the entry task might proliferate to the exit task.

Stavrinides and Kratza [13-18] presented the placement of multiple real-time jobs with partial computations. These works considered the impact of input error on immediate child tasks such that it gets corrected at the immediate child level. Therefore there was no consideration of error proliferation in any of these proposed solutions.

Numerous research efforts have addressed the issue of workflow scheduling in fog environment. However, we found that only a few scheduling policies utilized partial computations in the fog environment.

Cao et al. [19] introduced a QoS optimization technique for fog environment that took into account the reusable nature of IoT devices. These IoT devices were considered to be powered by hybrid energy sources that included grid electricity and renewable generations. By utilizing approximate computations, the authors designed a two-phased task placement strategy. Application-level and component-level energy provisioning were carried out one after the other at the Internet of Things layer to create a local placement solution. Through renewable-adaptive computation offloading, a local-remote placement technique solution was later derived at the fog layer. However, this strategy considered the workload to be a bag-of-tasks and the impact of error proliferation was not considered.

Mora Mora et al. [20] presented an approximate computations based task placement strategy for fog-cloud environment. This technique considered IoT data as bag-of tasks instead of workflows. In addition, this technique did not consider monotone tasks where the computational volume of the task is divided into mandatory and optional constituents. The authors assumed few tasks to be mandatory and rest to be optional. Subsequently, input error proliferation impact was not taken into consideration.

Mo and Kritikakou [21] introduced a mathematical framework that integrates partial computations for energy-effective placement of deadline-bound jobs in a cyber-physical ecosystem in fog environment. Due to the problem being complex, it was initially expressed as an MINLP (Mixed Integer Non-Linear Programming) problem which was further linearized to MILP (Mixed Integer Linear Programming) problem. This work considered workload as workflow jobs but it did not consider initial IoT data required for entry tasks of the workflow. Additionally, the error proliferation impact was also not taken into consideration.

Specifically, we found only one study related to partial computations that considers the impact of end-to-end error proliferation in workflow processing in fog computing environment.

Stavrinides and Karatza [4] proposed a heuristic based deadline aware policy that utilized partial computations and depicted the impact of error proliferation in a workflow. This work showed the impact on the computation expense of every impacted constituent task in the workflow but overlooked the impact of error proliferation on the energy consumption of the fog computing resources.

In this policy [4], the tasks are prioritized according to the lower value of the deadline i.e. task with a lower deadline is given higher priority. In case when the deadlines of the tasks match then the task with a higher computational volume is given higher priority i.e. the communication expense of the task is not considered for the ranking of tasks. Expected Processing Time Calculation is an important metric that is generally considered for the ranking of tasks [28]. It needs to be calculated based on computation as well as communication expenses. Next, this methodology aims at reducing the Finish Time (or response time) of workflows without considering the computation expense of fog computing instances while allocating the task to a computing instance. The scheduling technique should take into account the computation expense of the processors since the fog nodes have limited processing capacities. Another vital factor in fog infrastructure is energy consumption in addition to response time which is not considered in [4]. Also, this policy uses a first-fit idle slot utilization strategy which may not be an ideal choice to utilize available schedule holes. Additionally, this policy is a single objective optimization problem that aims only at minimization of the response time of the workflow. All these identified drawbacks are the motivation for this work.

Therefore, we propose a Topsis-based Partial Computation (TPC) technique which is a multi-objective optimization technique that takes into consideration the optimization of energy consumption as well as response time and simultaneously aims at improving schedule hole utilization and overall result precision of the workflows by eliminating the drawbacks recognized in the state-of-the-art work [4]. In our presented methodology, all the workflow jobs for execution are acquiesced to an intermediary fog node which acts as a fog scheduler. It obtains the status of all the available fog computational resources. The proposed application scheduling strategy assigns the acquiesced applications to appropriate fog resources in such a manner that finds the right balance between the response time of real-time applications and the energy consumption of fog computing instances. Further, the presented policy employs the best-fit idle slot utilization strategy in place of the first-fit to improve the overall system performance.

3. The system and workload model

This section describes the system model which is a fog-IoT environment, workload model and energy consumption model employed in this proposed work.

3.1 System Model

The proposed work considers a fog-enabled IoT system which is a two-layer architecture: Fog nodes at the topmost layer and end-user devices at the lowermost layer. The IoT-fog ecosystem is shown in Fig. 1. The bottom or terminal layer is constituted by IoT devices, e.g., sensors, smart vehicles, smartphones, home appliances, and wearable devices [22, 27]. These devices communicate with upper-level fog resources for application processing by sending requests and data.

In this study, IoT devices are considered as data generation sources that lack the capability to execute the produced data. The fog computing layer is typically deployed close to IoT devices. It is constituted by nodes residing close to the network edge e.g. access points, base stations, routers, and switches with restricted processing, transmission, and transient storage capability [23].

Specifically, the fog environment consists of a set $R = \{r_1, r_2, \ldots, r_{|R|}\}$ of $|R|$ physical hosts with heterogeneous processors. A pool of v_i VMs is provided by every physical host r_i. Jointly, the fog layer provides a set M

$=\{v_1^{r_1},\ldots,v_{|M|}^{r_{|R|}}\}$ of $|M| = \sum_{r_i \varepsilon R} v_i$ fog VMs where every fog virtual machine is assigned a vCPU. Every virtual CPU (vCPU) and therefore, every virtual machine (VM) has a queue of allocated tasks that must be completed.

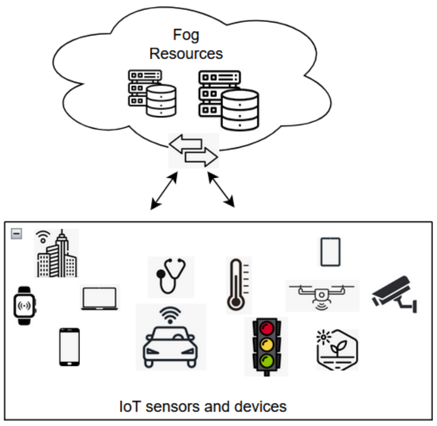

Fig.1 IoT-Fog ecosystem

To get the required services, users can connect to the fog nodes. The fog resources are anticipated to lie one or two hops distant from the users to satisfy stringent latency requirements [24].

The rate of data transmission (t^{IoT}) amongst the Internet of Things tier and fog tier follows uniform distribution in the following interval:

$$t^{IoT} \sim U[\tau^{IoT}.(1 - T^{IoT}/2), \tau^{IoT}.(1 + T^{IoT}/2)] \qquad (1)$$

where τ^{IoT} and T^{IoT} depict the mean data transmission rate amongst the Internet of Things tier and fog tier, and heterogeneity grade of bandwidth of the network connecting both layers respectively. The virtual machines in the fog tier are linked via a wireless network connecting both layers over the Internet.

The rate of data transmission (t_{ij}^{fog}) between two fog virtual machines (VMs) v_i^{fog} and v_j^{fog} follows uniform distribution in the following interval:

$$t_{ij}^{fog} \sim U[\tau^{fog}.(1 - T^{fog}/2), \tau^{fog}.(1 + T^{fog}/2)] \qquad (2)$$

where τ^{fog} and T^{fog} depict the mean data transmission rate among the fog nodes and heterogeneity grade of the virtual fog network respectively. It should be noted that the variable names use superscripts to distinguish between the variables associated with each layer. The fog resource queueing model (adopted from [4]) is illustrated Fig. 2.

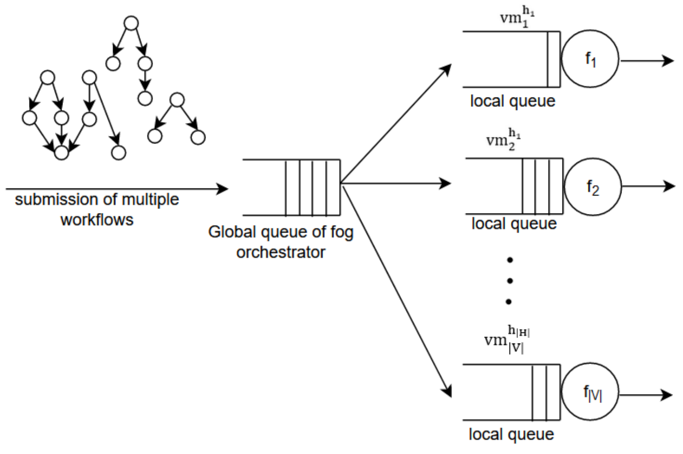

Fig. 2 Fog resource queuing model

Task scheduling on the fog layer virtual machines (VMs) is done by a central scheduler known as fog broker operating on a specific computing node in the fog layer. It collects user requests, controls resources on fog nodes, and generates optimal schedules for input workflows.

3.2 Workload Model

This section describes the computational and communication characteristics of the workflow as well as the approximate computations and error proliferation model used in this work.

3.2.1 Computational and Communication Characteristics

The data produced by IoT layer devices are transmitted to the second layer where real-time workflows process it. A sample diagram of the workflow is shown in Fig. 3.

A directed acyclic graph (DAG) $G = (N, E)$, where N denotes the set of graph nodes and E denotes the set of directed edges connecting the nodes, is used to describe each *workflow*. Every node signifies a workflow task n_i, while, a directed edge e_{ij} connecting these workflow tasks n_i and n_j signifies the data that needs to be conveyed from a predecessor task n_i to a successor task n_j. A workflow's constituent tasks are assumed to be non-preemptible since preemption might result in a decrease in performance for real-time tasks [6].

Every task n_i is associated with a weight w_i that represents its computational volume i.e. the count of clock cycles needed to process the task's instructions. It follows an exponential distribution around the mean $\bar{k}$. The computational expense of the task n_i on a VM vm_v is determined by:

$$Comp(n_i, vm_v) = w_i/f_v \tag{3}$$

where f_v is the frequency at which VM vm_v operates. Every edge e_{ij} from a task n_i to task n_j is associated with a weight w_{ij} that depicts its communication volume, i.e. the amount of data (in GigaBytes) needed to be transmitted from task n_i to n_j.

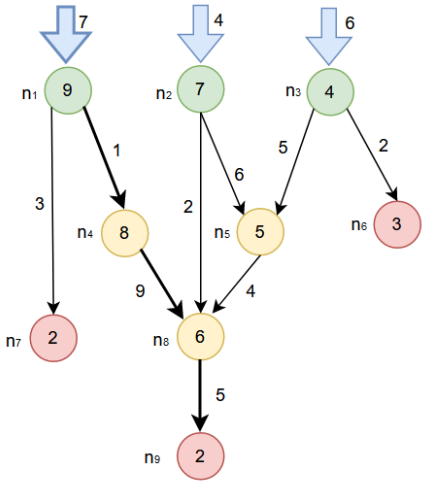

Fig. 3 Computational and Communication Characteristics of Workflow

It follows an exponential distribution around the mean $\bar{q}$. Data transfer from task n_i (placed on VM vm_p) to task n_j (placed on VM vm_q), incurs an edge communication expense which is defined as:

$$Comm((n_i, vm_p), (n_j, vm_q)) = w_{ij}/(t_{pq}^{fog}) \tag{4}$$

where t_{pq}^{fog} is the data transmission rate of the virtual connection amongst the virtual machines (VMs) vm_p and vm_q.

Every job's entry task needs input data generated by the Internet of Things layer. An entry task n_i's input data size d_i has mean $\bar{d}$. The communication expense experienced by input data transmission from the Internet of Things layer devices to a task n_i placed on a fog virtual machine vm_v is given by:

$$Comm(n_i, vm_v) = d_i/t^{IoT} \tag{5}$$

where t^{IoT} is the data transfer rate between the IoT and fog layer.

The path length in the graph is determined by adding up the computational expense of all the tasks and the communication expense of all the edges on that path which includes the cost of transferring initial input data to the entry task. The critical path length (CPL) refers to the maximum path length in a graph. Every real-time job has a fixed deadline, denoted as D, by which the execution of all of its constituent tasks must be finished. It is defined by:

$$D = AT + RL \tag{6}$$

where AT denotes the workflow's arrival time and RL is its relative deadline limit, which follows uniform distribution in the interval [CPL, 2CPL]. In the time-sensitive setting being studied, the deadline of every workflow must be adhered to, or else its results would go futile. Thus, in such a scenario, the job is otherwise, regarded as lost.

3.2.2 Approximate Computations and Error Proliferation Model

This section describes the approximate computations and error proliferation model considered in this work. Every workflow submitted to the fog orchestrator is bounded by a deadline which must be satisfied for the results to be meaningful. A job is regarded as lost if all of its constituent tasks do not get completed by its deadline. To address this issue, approximate computations can be employed which trades off result precision for timeliness i.e. this method allows a job to return results that are not precise but yet acceptable in quality in case when its deadline cannot be satisfied. This technique assumes workflow's constituent tasks to be monotone i.e. computational volume w_i of every constituent task n_i comprises of two parts: a mandatory portion mp_i and an optional portion op_i.

$$w_i = mp_i + op_i \tag{7}$$

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

When the respective mandatory portion of a task has completed its execution, then the constituent task is deemed complete. Depending on the central scheduler's decision, the task can either finish the execution of the entire optional portion, partial optional portion or it may omit its entire optional portion. The outcome of a partly executed task n_i is inaccurate and thus, the task generates an output error. Since the predecessor task's outcome is utilized as input by its successors, therefore, input error is present in the input data of the successors. Additionally, there's a possibility that the child task's incoming error is passed on to its outcome if it is unable to rectify the error through additional computations, and as a result, its input error is passed on to its output which is governed by the task's input error proliferation factor.

Therefore, the output error of a task n_i is computed as:

$$OE_i = \frac{\delta_i}{op_i} + \varphi_i \times IE_i \tag{8}$$

where δ_i is the unexecuted portion of the task's optional portion op_i, whereas φ_i is the task's input error proliferation factor (IEPF). The value of φ_i is dictated by p (i.e. proliferation probability of input error) which takes values in the range [0,1]. For a task n_i, the input error is taken to be equivalent to the mean output error of its predecessors as shown below:

$$IE_i = \sum_{n_j \epsilon F_i} \frac{OE_i}{|F_i|} \tag{9}$$

where F_i is the set of the predecessors n_j corresponding to successor n_i. There is a direct impact of a task's input error on its execution time. To tackle the error impact and generate a satisfactory end outcome, additional instructions and therefore clock cycles (computations) are needed. As a result, the task's mandatory portion is extended. In particular, the mandatory portion extension of a task n_i because of its input error is given by:

$$mpe_i = mp_i \times IE_i \tag{10}$$

The task's mandatory portion extension is summed with its primary mandatory portion. It is crucial to note that the task's optional portion is not impacted by its incoming error.

The result precision with respect to task n_i is given by:

$$RP_i = RPT + (1 - RPT)(1 - OE_i) \tag{11}$$

where RPT depicts the result precision threshold, below which a task's results are unacceptable.

The result precision of a task lies in the range [RPT, 1], i.e.:

$$RPT \leq RP_i \leq 1 \tag{12}$$

The result precision of a workflow is determined by taking average of the result precision of its exit tasks, i.e.:

$$\mathrm{RP} = \sum_{n_i \in N_{exit}} \frac{RP_i}{|N_{exit}|} \tag{13}$$

where N_{exit} denotes the set of workflow exit tasks.

From equations (8) and (11) it follows that if an input error proliferates into a task's output, the outcome of the task will remain inaccurate even after the task's entire extended computational volume has been executed. Hence, the discrepancy in the input of a task because of its incoming error cannot always be recompensed by carrying out supplementary computations. Therefore, it is not always possible to make up for a task's input error discrepancy by doing more computations.

When there is no proliferation of input error into the task's output then result precision threshold is determined by dividing the task's mandatory portion by its computational volume.

$$RPT = {}^{mp_i}/_{w_i} \tag{14}$$

As per this imprecise computations and error proliferation model, whenever the result of a constituent task of a workflow is imprecise, the resulting error can spread beyond just the directly connected child tasks ultimately impacting the overall outcome of the workflow.

The error proliferation among the workflow tasks is quantified by the input error proliferation index which is calculated as shown below:

$$IEPI = \frac{|E^{err}| + |N_{exit}^{err}|}{|E| + |N_{exit}|} \tag{15}$$

where E^{err} depicts set of edges that proliferate input error from a predecessor to a successor, N_{exit}^{err} depicts the set of exit tasks which yield outcome comprising proliferated input error, E denotes the set of edges, while, N_{exit} denotes the set of exit workflow tasks.
Subsequently, in the imprecise computations scenario, upon reaching the deadline, the job is assumed to be completed if all the remaining constituent tasks are exit tasks and all of them have finished the execution of their respective mandatory constituent tasks. Despite the job result being imprecise, the result quality is still acceptable.

3.3 Energy Consumption Model

In this section, we compute the energy consumed by all the VMs during both busy and idle times. The expected energy consumption due to the tasks scheduled on VM vm_j is given as follows:

$$\mathrm{EEC}_j^{\mathrm{Busy}} = \sum_{n_i \varepsilon G:G\varepsilon E} (\mathrm{alloc}_{ij} \times \mathrm{Comp}(n_i, vm_j) \times \mathrm{ECRate}_j^{\mathrm{Busy}}) \qquad (16)$$

where, $alloc_{ij} = 1$, if task is scheduled on VM vm_j (and 0 otherwise) and $Comp(n_i, vm_j)$ is the computation expense of task n_i on vm_j. It is important to compute the energy consumption due to idle time slots that may not be utilized for the execution of any task.

$$EEC_j^{idle} = [\sum_{n_i \varepsilon G:G\varepsilon E}^{v_k \varepsilon V} (alloc_{ik} \times \mathrm{Comp}(n_i, vm_k)) - \sum_{n_i \varepsilon G:G\varepsilon E} (alloc_{ij} \times \mathrm{Comp}(n_i, vm_j))] \times ECRate_j^{Idle} \qquad (17)$$

where, $\sum_{n_i \in G:G\varepsilon E}^{v_k \in V} (alloc_{ik} \times Comp(n_i, vm_k))$ gives the total computation expense of all tasks n_i of the workflow G such that G ε E, where, E depicts the ensemble of completed workflows.

Therefore, the total energy consumption of a VM vm_j is given as:

$$EEC_j = EEC_j^{Busy} + EEC_j^{Idle} \qquad (18)$$

Thus, the total energy consumption of all VMs is given by the following equation:

$$TEC = \sum_{v_j \varepsilon V} EEC_j \qquad (19)$$

4. The Proposed Model

To schedule workflow tasks submitted at the fog orchestrator, a dynamic two-state strategy is implemented which entails a Task Prioritization state and a Computational Resource Selection state. To determine the sequence of execution of the workflow tasks, the algorithm first computes their ranks. Subsequently, a suitable fog VM is chosen for each task based on the most commonly employed MCDM technique - TOPSIS.
The proposed policy ranks the workflows based on the Least Deadline First strategy followed by the calculation of the Expected Processing Time of workflow tasks to decide their scheduling order. The execution sequence of tasks has a substantial effect on the performance of the scheduling technique. Next, the proposed policy utilizes the most commonly employed MCDM technique TOPSIS in order to determine the processor on which the task will be scheduled for execution. The proposed policy considers Expected Finish Time, Computational Resource Computation Expense, Energy Consumption as well as best fit policy for occupancy of available idle slots (which may appear in the processor's schedule) while selecting the appropriate processor for task scheduling in order to meet the QoS criteria.

4.1. Task Prioritization

The task prioritization phase involves assigning priorities to the constituent tasks in order to determine their sequence of execution, mostly to optimize scheduling during the computational resource selection phase. To achieve this, the workflows in the fog scheduler's global waiting queue are assigned ranks based on their end-to-end deadline D. The workflow with the least deadline is assigned the topmost priority. Workflows are therefore ranked using the Least Deadline First (EDF) criteria. Then, we assign priorities to the workflow's constituent tasks on the basis of their Expected Processing Times (EPT), which is the critical metric that signifies the time by which the task is expected to be completed.
The Expected Processing Time of a constituent task n_i is computed as follows:

$$EPT(n_i) = \max_{n_p \in F_i} [EPT(n_p) + ECE(n_i) + DTE(n_p, n_i)] \qquad (20)$$

where, $ECE(n_i)$ is the Average Expected Computation Expense of task n_i and is equal to w_i/f_{avg}
and $DTE(n_p, n_i)$ is the time taken to convey data from predecessor task n_p to n_i. Its value is equal to w_{pi}/t_{avg}^{fog}.
If n_i is an entry task (i.e. $n_i \varepsilon N_{entry}$) then its Expected Processing Time is equal to the Data Transfer Expense from the IoT layer which is computed as follows:

$$EPT(n_i) = ECE(n_i) + DTE(n_i) \tag{21}$$

where, $DTE(n_i)$ is the time taken to convey initial IoT data to the fog layer and is equal to d_i/t^{IoT}.

4.2 Computational Resource Selection

The task n_i with the maximum rank is chosen for execution from the global queue that can only start to execute when all the predecessors of n_i have finished execution and the target execution node has acquired the necessary input data of n_i. MCDM approaches prove to be very useful when VM instances need to be ordered based on multiple metrics. Several Multi Criteria Decision Making techniques such as ANP (Analytic Network Process), AHP (Analytical Hierarchy Process), TOPSIS, etc. can be utilized to associate ranking with the alternatives (i.e. VMs). TOPSIS is beneficial over other MCDM techniques [25] since TOPSIS generates hypothetical best as well as worst results. It then ranks the available alternatives based on their proximity to the best and the worst results. Additionally, TOPSIS is simplistic, easily understandable, and the most commonly used MCDM technique [28]. Of paramount importance is the low computational complexity of TOPSIS, which facilitates the designing of placement policy with lower computational complexity required for the fog computing environment.
To rank the various available VMs, multiple metrics such as Expected Finish Time, Processor Computation expense and Energy Consumption are taken into consideration. To look for a suitable idle slot within the schedule of the same VM, the difference between available slot size and effective slot occupancy is considered.

The fog scheduler selects a ready task (n_i) from the global queue as per the task ranking and assigns the task to the virtual machine (VM) that can reduce the finish time as well as increase the Energy Savings of the computational instances.
The steps performed for suitable VM selection are as follows:

1. The Estimated Energy Consumption due to the allocation of task n_i on VM vm_k is computed as follows:
$$EEC_{ik} = Comp(n_i, vm_k) \times ECRate_k^{Busy} \tag{22}$$
The Expected Finish Time of task n_i on vm_k is calculated as follows:

$$EFT(n_i, vm_k) = max\{t_{data\_avail}(n_i, vm_k), t_{vm\_avail}(n_i, vm_k)\} + Comp(n_i, vm_k) \tag{23}$$

where, the term $t_{data\_avail}(n_i, vm_k)$ signifies the time at which the ready task's input data will reach vm_k. If n_i belongs to a set of workflow entry tasks then the term $t_{data\_avail}(n_i, vm_k)$ depicts its preliminary input which needs to be transmitted to the fog tier from the IoT tier. In rest of the scenarios, the term $t_{data\_avail}(n_i, vm_k)$ depicts the data produced by predecessors of n_i. Considering the state of the local queue of vm_k at the moment, the term $t_{vm\_avail}(n_i, vm_k)$ gives an estimate of the time when vm_k will be available to process the task n_i. Considering the state of the vm_k local queue at the moment

To ascertain the value of the term $t_{vm\_avail}(n_i, vm_k)$, following steps are performed:

1. Based on the priority of task n_i, it is first scheduled at the potential position in the local queue of vm_k.
2. Next, we determine whether a schedule hole exists in the processor vm_k's schedule to check if task n_i can be scheduled before its potential position given that task n_i's required input data is already available at vm_k and task n_h scheduled at the front of vm_k's local queue is still waiting for its requisite input data from other processing nodes or the IoT data sources. It is to be noted that a schedule hole appears in the processor's schedule when vm_k is idle i.e. no task is being executed at that moment.
$$sh = t_{data\_avail}(n_h, vm_k) - t_{present} \tag{24}$$
3. If a schedule hole exists then we attempt to place the ready task n_i in the schedule hole by doing the following:
   3.1 First, we find out the schedule holes where the complete task can fit i.e.
   $$sh \geq w_i / f_k \tag{25}$$
   where, w_i depicts the task n_i's computational volume and f_k shows the vm_k's operating frequency. In case multiple schedule holes exist, then n_i is inserted in the schedule hole where the difference between available schedule hole size and task n_i's computational expense is the least.
   3.2 In case complete task n_i cannot be accommodated in the identified schedule holes then, we try to place a segment of the ready task n_i by using partial computations. Particularly, we assess whether the minimum plausible fraction of the task n_i can fit into the schedule hole i.e.
   $$sh \geq (w_i - \delta_i^{max}) / f_k \tag{26}$$
   Subsequently, we determine whether the aggregate average extra computation time imposed due to the mandatory portion extension of task n_i's immediate successor tasks is equal to or lesser than the

saved execution time from the unexecuted optional portion of ready task n_i which is computed as shown below:

$$\delta_i^{max}/f_k \geq \sum_{n_j \epsilon C_i} \sum_{vm_l \epsilon V} \frac{(mp_j \times IE\prime_j)/f_l}{|V|} \tag{27}$$

In addition, we also check whether the aggregate average extra energy consumption levied due to the execution of the mandatory portion extension of task n_i's child tasks is equal to or lesser than the energy consumption from the unexecuted optional portion of ready task n_i as shown below:

$$(\delta_i^{max}/f_k) * ECRate_k^{Busy} \geq \sum_{n_j \epsilon C_i} \sum_{vm_l \epsilon V} \frac{(mp_j \times IE_j')/f_l}{|V|} * ECRate_l^{Busy} \tag{28}$$

where C_i depicts the child tasks' set n_j corresponding to the parent task n_i.

IE_j' denotes the probable input error corresponding to the child task n_j which is computed as:

$$IE_j' = IE_j + \frac{\delta_i^{max}/op_i + \varphi_i \times IE_i}{|F_j|} \tag{29}$$

where IE_j signifies child task n_j's current input error while F_j represents the set of n_j's predecessor tasks. Only a portion of the ready task n_i can be introduced in the schedule hole if conditions (26), (27), and (28) are met. In this scenario, the portion of the task that would be processed corresponds to its computational volume w_i' that would fit in the schedule hole i.e.

$$w_i' = sh \times f_k \tag{30}$$

Among the identified schedule holes, the one with the maximum capacity (schedule hole size) is chosen to schedule the ready task n_i so that its maximum portion can be executed in an attempt to increase the result precision of task n_i and reduce its output error.

After getting the Expected Finish Time and Expected Energy Consumption values for all VMs, alternative matrix (A) is obtained with Expected Finish Time, Expected Energy Consumption, and Processor Computation expense as VM dependent metrics. Then, Topsis is applied on this alternative matrix (A). The steps followed for Topsis are described below:

Step 1: The matrix A is normalized using Eq. (31) where a_{ij}' is the element of the normalized matrix A'.

$$a_{ij}' = \frac{a_{ij}}{\sqrt{\sum_{i=1}^{n} a_{ij}^2}} \tag{31}$$

Step 2: To represent the significance, every metric is assigned a weight. Every column of normalized matrix A' is multiplied with its corresponding weight to get Weighted normalized matrix B.

$$b_{ij} = w_j \times a_{ij}' \tag{32}$$

where, i={1, 2, ..., n}, $\sum_{j=1}^{k} w_j = 1$, w_j represents jth VM dependent metric's weight.

In the proposed work, every metric is allocated a fixed equal weight in the fog environment as all the VMs need to be examined based on the same standard. Since three metrics are taken into consideration, therefore, $w_1 = w_2 = w_3 = 0.33$.

Step 3: Using the formulas given in Eqs. (33) and (34), Positive Ideal Solution (PIS^{+}), and Negative Ideal Solution (NIS^{-}), are obtained from the matrix.

$$PIS^{+} = [b_1^{+}, \dots, b_k^{+}] \tag{33}$$

$$NIS^{+} = [b_1^{-}, \dots, b_k^{-}] \tag{34}$$

This step attempts to determine the best a the worst value of every metric. For each metric in B, PIS^{+} depicts the vector corresponding to the best value while NIS^{-} depicts the vector corresponding to the worst value.

Step 4: Calculate the Euclidean distance of every VM from the PIS and NIS i.e. S_i^{+} and S_i^{-} respectively.

$$S_i^{+} = \sqrt{\sum_{j=1}^{k} \left(b_j^{+} - b_{ij}\right)^2} \tag{35}$$

$$S_i^{-} = \sqrt{\sum_{j=1}^{k} \left(b_j^{-} - b_{ij}\right)^2} \tag{36}$$

Step 5: For each VM, determine its relative closeness, or rating of vm (RV_i) from the ideal solution using S_i^{+} and S_i^{-}.

$$RV_i = \frac{S_i^{-}}{S_i^{+} + S_i^{-}} \tag{37}$$

The set of VMs can now be ranked in descending order based on the rating of each VM i.e. RV_i and the task is assigned to the fog VM having maximum value of RV_i.

5. The Performance Study

The proposed work is contrasted with a baseline-Topsis based multi-objective optimization strategy (i.e. BMO) which utilizes schedule holes according to best fit policy only when complete task can occupy a schedule hole and a state-of-the-art work [4] for the placement of workflows by utilizing approximate computations and Topsis. The shortcomings of [4] have been recognized and mentioned in Section 1. This section demonstrates the efficacy of the presented strategy in relation to Baseline policy and the state-of-the-art technique [4].
For comparison, the presented scheduling technique, Baseline technique and the state-of-the-art technique [4] are observed under varying values of RPT (result precision thresholds) and p (i.e. proliferation probabilities of input error). In particular, we examined the efficacy of the presented strategy over Baseline policy and state-of-the-art work [4] for each potential pair of the following values of these two attributes, RPT ={0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9} and p={0, 0.35, 0.7,1}.

5.1 Performance Metrics
The following metrics are used to assess the efficiency of the proposed approach:

1. *SLA Violation Ratio :* It is calculated as the ratio of the count of jobs that were unable to accomplish their execution within their deadline - and therefore, were discarded - to the count of total jobs that reached at the fog scheduler.
2. *WA-RP:* It is the weighted average of the result precision of completed jobs where the result precision of every completed job was weighted by the count of its exit tasks.
3. *AV-RT:* It is the average response time of the completed jobs.
4. *AP-TSG:* It is the average percentage of constituent tasks of completed jobs that were assigned to the schedule holes for their execution by the fog scheduler.
5. *AV-IEPI:* It is the average input error proliferation index of completed jobs.
6. *P-PCG:* It is the percentage of completed jobs which on reaching the deadline had at least one partly completed exit task, therefore, generated imprecise yet acceptable quality results.
7. *TEC:* It gives the total amount of energy consumed by the fog resources in executing the completed jobs during the observed simulation duration.

5.2 Simulation Experiments
In order to assess the performance of the proposed policy, iFogSim [26] is utilized to simulate the Fog environment. For our simulation experiment, we considered |N|=10 physical nodes in the fog environment offering |V|= 64 heterogeneous virtual machines in total. The values of workload and other simulation parameters used in the proposed model are shown in Table 1.

5.3 Performance Evaluation
In this section, the proposed policy Topsis based Partial Computations technique (i.e. TPC), is compared to a Topsis based baseline multi-objective optimization policy (i.e. BMO) and a state-of-the-art Heuristic based Partial Computations technique (i.e. HPC) [4]. In Fig. 4, TPC, BMO and HPC are compared under various RPT (Result Precision Threshold) and p (error proliferation) values on the basis of the SLA Violation Ratio parameter.

Table 1 Simulation input parameters

IoT Layer	
Mean data transfer rate of IoT-fog network	τ^{IoT} = 50 Mbps
Heterogeneity grade of IoT-fog network	T^{IoT}=0.5
Fog Layer	
Count of physical nodes	|N|=10
Count of fog VMs	|V|=64
Operational frequency of fog VM vCPU	f ={2.5 – 3.8} GHz
Mean data transfer rate in fog network	τ^{fog} = 1 Gbps
Heterogeneity degree of fog resources	T^{fog} = 0.5
Fog VM Energy Consumption Rate during idle time	$ECRate^{Idle}$ = [25-40]
Fog VM Energy Consumption Rate during task execution	$ECRate^{Busy}$ = [105-130]
Workload Characteristics	

Number of DAGs	[40-100]
Minimum count of tasks in a workflow	W_{min}=10
Maximum count of tasks in a workflow	W_{max}=32
Mean input data size for entry task	$\bar{d}$ =1 GB
Communication to Computation Ratio	2
Mean computational volume of constituent task	$\bar{k} = 8.93\times10^{11}$ clock cycles
Mean communication volume of edge	$\bar{q}$ = 44.74 GB
Result precision threshold	RPT ={0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
Proliferation probability of Input error	p ={ 0, 0.35, 0.7, 1}

It is evident from Fig. 4 that for all values of p and RPT, the proposed approach TPC outperformed the baseline policy (BMO) and the state-of-the-art technique (HPC). There was a significant decline of 9.93% and 36.5% compared to HPC and BMO respectively in the SLA Violation ratio as depicted in Table 2. Since, BMO did not utilize approximate computations, therefore, it could utilize schedule holes only when entire task could occupy a schedule hole. In comparison to HPC, this decrease was more significant, particularly, for low result precision values because the proposed methodology incorporated the best fit policy due to which at lower RPTs it could place more tasks in the schedule holes than state of the art policy HPC as evident from Fig.5. Hence, scarcer workflows lost their deadline in TPC as compared to HPC and BMO.

The variation in schedule hole utilization between the two scheduling methods TPC and HPC became less noticeable for higher result precision thresholds. Overall, still, TPC approach made use of more schedule holes than HPC. This is shown in Fig. 5. Further, according to the simulation findings shown in Fig. 4, increase in the p values resulted in a greater number of jobs failing to meet their deadline. However, the proposed technique, TPC, met more deadlines as compared to HPC and BMO, even in the instances when input error proliferation remained constant throughout the constituent tasks of the job (i.e. p=1). This demonstrates that TPC was less susceptible to the proliferation of input errors across the workflow tasks.

The proposed policy (TPC) could place a lesser number of tasks in schedule holes for larger p values, as can be seen in fig. 5. This is because there was a greater likelihood of an increase in the average additional processing time and average additional energy consumption due to the proliferation of input error which is why fewer constituent tasks satisfied the requirements in (26), (27) and (28) during the selection of the potential VM.

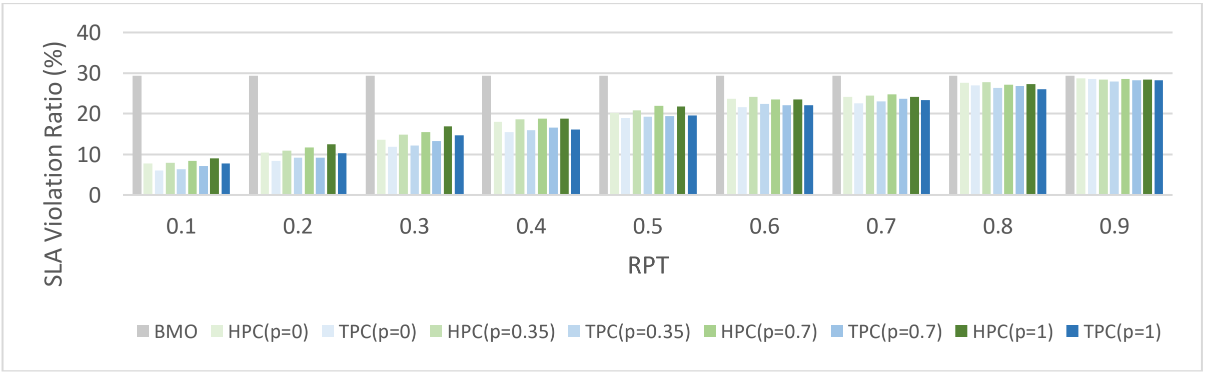

Fig. 4 SLA Violation ratio percentage under different values of input error proliferation probability (p) and Result Precision threshold (RPT)

Nevertheless, TPC used more schedule gaps than both HPC and BMO scheduling methodologies even for the highest p value (i.e. p=1). This was because it took advantage of the best fit idle slot utilization which let the fog orchestrator to insert more partial tasks in the holes as opposed to HPC's first fit policy whereas BMO could not take place partial tasks in the available schedule holes due to which it could utilize lesser number of schedule holes than TPC.

Table 2: SLA Violation Ratio Percentage Decrease

RPT	TPC vs HPC					TPC vs BMO				
	p=0	p=0.35	p=0.7	p=1	Average	p=0	p=0.35	p=0.7	p=1	Average
0.1	22.79	20.05	150	14.05	18.02	79.05	77.97	75.19	73.21	76.36
0.2	19.69	16.94	21.08	17.11	18.70	71.09	68.31	68.00	64.14	67.89
0.3	13.05	17.25	14.86	12.97	14.54	59.03	57.51	54.03	48.96	54.88
0.4	13.32	14.19	11.51	14.38	13.35	45.97	44.34	42.29	43.92	44.13

0.5	6.27	7.81	11.06	10.09	8.81	34.05	33.15	32.35	31.86	32.85
0.6	8.66	6.89	6.06	6.11	6.93	24.88	21.99	23.45	23.14	23.37
0.7	6.55	5.32	4.21	2.98	4.77	21.61	19.67	17.76	18.66	19.42
0.8	2.35	5.22	1.25	4.30	3.28	6.32	8.48	6.85	9.42	7.77
0.9	1.04	1.58	0.70	0.70	1.01	1.04	2.74	1.63	2.08	1.87
Overall Avg.					9.93	Overall Avg.				36.50

Figure 6 shows a comparison of all orchestration techniques in terms of the weighted average result precision parameter. For lower RPTs and higher p values, the result precision of the accomplished workflows was lower (although over the mandatory level) for both HPC and TPC. However, the proposed policy could yield a better weighted average result precision due to the use of best fit strategy. However as RPT increased, the difference between the two policies became less noticeable. This was caused by the fact that the optional portion of the tasks got smaller at higher RPT values and as a result, the percentage of the optional portion that the proposed policy could discard was also less.

It can also be observed from the simulation findings in Figs. 4 and 6 that TPC traded-off lesser value of result precision for timeliness in comparison to HPC. In contrast to the decline in missed deadlines, the decline in job result precision was of less significance for TPC than baseline policy BMO and state of the art technique HPC.

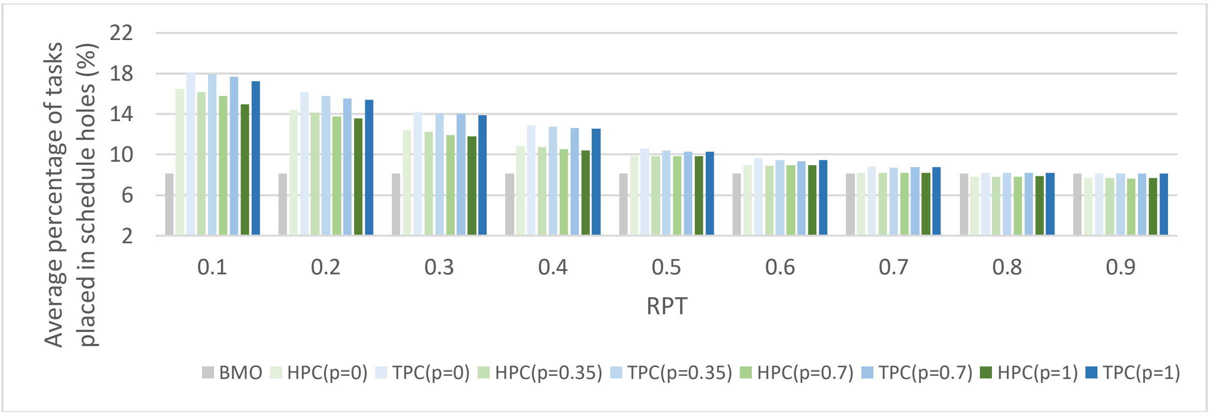

Fig.5 Average percentage of tasks occupying schedule holes vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

The performance of the scheduling methods TPC, HPC and BMO concerning the average response time is displayed in fig. 7. Due to better utilization of available schedule holes in TPC, there was a significant difference between response time values among the two policies TPC and HPC for lower RPTs. However, for moderate to higher RPT values, the execution time of the workflows and therefore, the response time of the workflows got increased. This was because lesser tasks could be positioned in schedule holes when the RPT values were raised (Fig.5). It can also be observed that TPC strategy resulted in a slight higher value of response time than HPC policy for higher values of result precision thresholds as the former incorporates optimization of Energy consumption along with response time minimization while the latter only aims at reduction of response time.

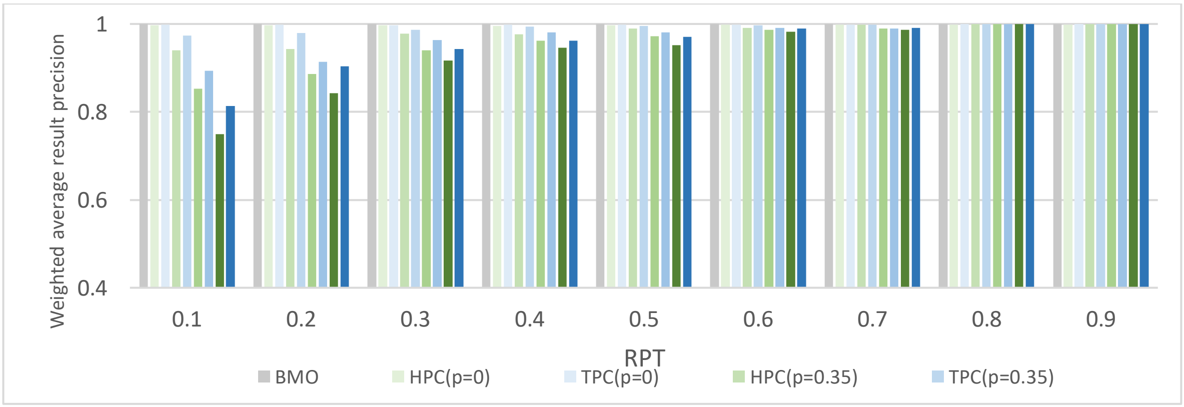

Fig.6 Weighted average result precision vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

Moreover, in a real-workload scenario, meeting deadline is utmost important. This does not always imply decreasing the response time of the workload [6]. Therefore, in contrast to the state-of-the-art policy, the presented technique's reduction in the SLA violation ratio is much more significant than slight higher value of response time of the workflows since the proposed policy also increases Energy Savings significantly. Specifically, TPC increased Energy Savings by 12.54% and 29.33 % in comparison to HPC and BMO respectively as shown in Fig 8. For lower result precision thresholds, energy savings in TPC are higher compared to HPC and BMO. This is attributed to the efficient utilization of schedule holes in TPC. Even for moderate to high values of RPT, when the proportion of computational volume that could be discarded got lesser, TPC was able to save significantly higher amount of energy than HPC and BMO.

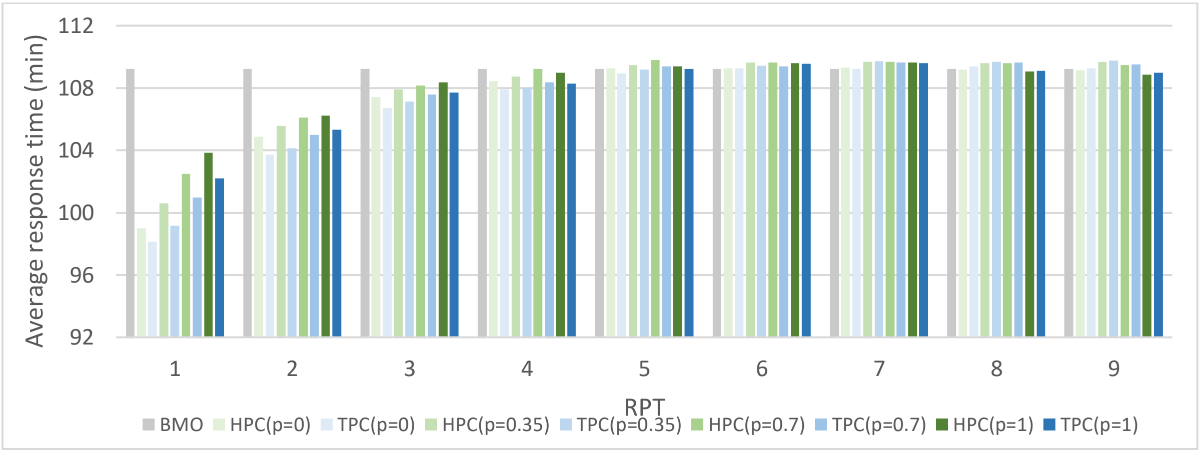

Fig. 7 Average response time vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

The percentage of partly completed workflows is displayed in Fig. 9. A workflow is said to be partly completed when its exit tasks cannot accomplish their entire execution prior to the workflow's deadline. For lower RPT values, because of the better utilization of schedule holes by the in-between workflow tasks in the proposed policy, there was a higher likelihood that their exit tasks would be finished ahead of their deadline. Due to this, when the RPT value was lower, the number of partly executed workflows was lesser for the presented technique. In contrast, when the RPT value was medium to high, the percentage of partly completed workflows reduced. This was because, for higher RPT values, fewer intermediate tasks could be inserted into schedule holes (Fig.5) due to which the response time of the workflow jobs increased (Fig.8). When a workflow job reached its deadline, the probability of its remaining unexecuted tasks being exclusively exit tasks was very low.

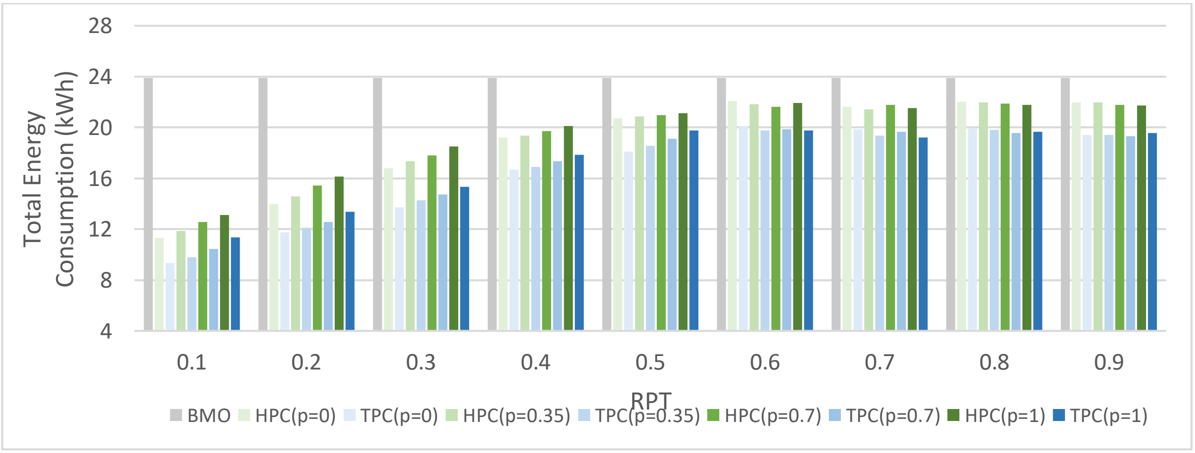

Fig. 8 Total Energy Consumption vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

Even if all the remaining workflow tasks belong to the set of exit tasks, it was less probable to satisfy upper result precision standards set by upper RPT values. As a result, not many workflows were partly completed and therefore exceeded their deadline. However, the number of partly completed workflows was higher in the case of TPC than HPC which also confirms that the SLA Violation ratio was lower in TPC (Fig. 4).

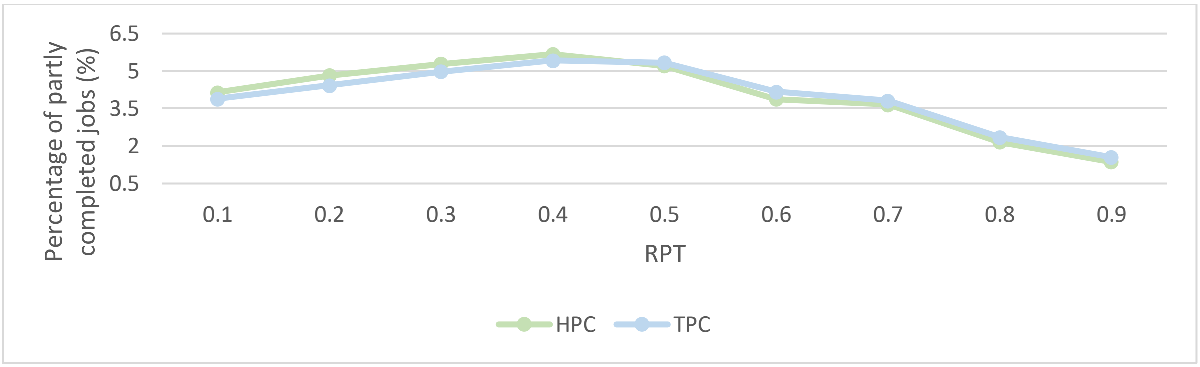

Fig. 9 Percentage of partly completed jobs vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

Fig. 10 shows the comparison of both scheduling strategies with respect to the weighted average input error proliferation index. The value of IEPI in the case of TPC is higher because it utilized more schedule holes due to the utilization of best fit strategy. However, with an increase in RPT values, lesser tasks could be positioned in schedule holes which resulted in a decrease in the input error proliferation index for both HPC and TPC. Therefore, the difference between IEPI values of TPC and HPC gets lesser with an increase in RPT values.

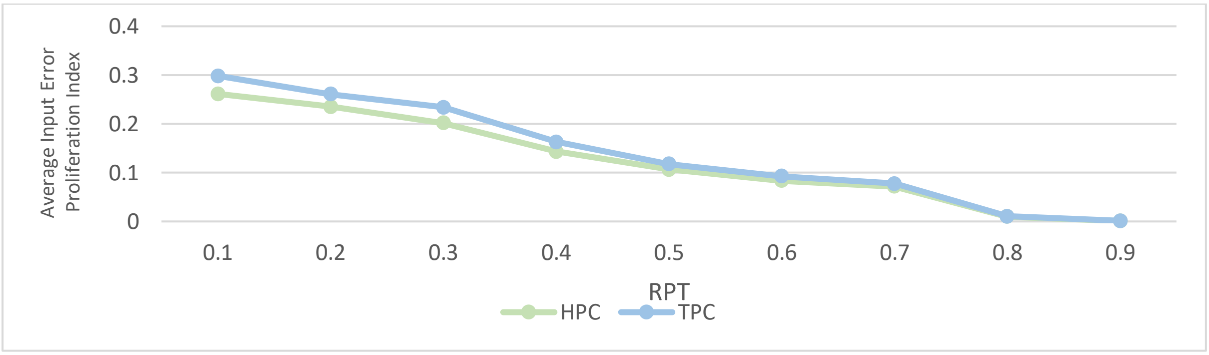

Fig. 10 Average input error proliferation index vs. different values of input error proliferation probability (p) and Result Precision threshold (RPT)

6. **Conclusion**:

In this work, we examined the orchestration of several real-time IoT jobs in a heterogeneous fog computing infrastructure by leveraging approximate computations. The impact of approximate computations has not been much explored to achieve timeliness for real-time jobs in a fog computing environment. None of the research works in fog computing studied the effect of input error proliferation on processing time and energy usage of computational resources simultaneously. Towards this end, we proposed a multi-objective optimization approach that aims at reducing workflow response time and the number of deadline misses while simultaneously increasing the energy savings. The proposed policy employed the most commonly used MCDM technique Topsis to schedule the workflow tasks on suitable fog computational resources. To rank the VMs, different vm dependent metrics such as Expected Finish Time, Processor Computation expense, and Energy Consumption are considered. The best fit idle slot utilization technique is employed to make effective use of the available schedule holes. The simulation results confirmed that the proposed policy outpaced the baseline as well as state-of-the-art policy regarding the SLA violation ratio (i.e. deadline miss ratio) and overall result precision. In addition, the proposed policy significantly increased energy savings for a relatively insignificant rise in response time. Explicitly, the proposed policy provided an average SLA Violation ratio decrease of 9.93%, and 36.5% while simultaneously increasing Energy Savings by 12.54% and 29.33% compared to the state-of-the-art policy and baseline policy respectively. Moreover, the experimental findings demonstrate that the proposed technique was more resistant to the impact of error proliferation athwart the workflow tasks.

In future, we aim at integrating proposed technique into a four-tier environment where mist as well as cloud resources will be available to process varied types of workloads. In addition, error proliferation impact on workflow processing cost can also be observed.

7. References

1. Soori, Mohsen, Behrooz Arezoo, and Roza Dastres. "Internet of things for smart factories in industry 4.0, a review." Internet of Things and Cyber-Physical Systems 3 (2023): 192-204.
2. Nabavi, S., Wen, L., Gill, S. S., & Xu, M. (2023). Seagull optimization algorithm based multi-objective VM placement in edge-cloud data centers. Internet of Things and Cyber-Physical Systems, 3, 28-36.
3. Elazhary, H. (2019). Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. Journal of network and computer applications, 128, 105-140.
4. Stavrinides, G. L., & Karatza, H. D. (2021). Orchestrating real-time IoT workflows in a fog computing environment utilizing partial computations with end-to-end error proliferation. Cluster Computing, 24(4), 3629-3650.
5. Wang, N., Varghese, B., Matthaiou, M., & Nikolopoulos, D. S. (2017). ENORM: A framework for edge node resource management. IEEE transactions on services computing, 13(6), 1086-1099.
6. Buttazzo, G. C. (2011). Hard real-time computing systems: predictable scheduling algorithms and applications (Vol. 24). Springer Science & Business Media.
7. Wainer, G., & Moallemi, M. (2020). Designing real-time systems using imprecise discrete-event system specifications. Software: Practice and Experience, 50(8), 1327-1344.
8. Chen, Y., & Tsai, W. T. (2014). Service-oriented computing and web software integration: from principles to development. Kendall/Hunt Publishing Co.
9. Lin, K. J., Natarajan, S., & Liu, J. W. S. (1987). Imprecise results: Utilizing partial computations in real-time systems (No. NAS 1.26: 180561).
10. Feng, W. C., & Liu, J. S. (1997). Algorithms for scheduling real-time tasks with input error and end-to-end deadlines. IEEE Transactions on Software Engineering, 23(2), 93-106.
11. Ravindran, R. C., Krishna, C. M., Koren, I., & Koren, Z. (2014). Scheduling imprecise task graphs for real-time applications. International Journal of Embedded Systems, 6(1), 73-85.
12. Esmaili, A., Nazemi, M., & Pedram, M. (2019). Energy-aware scheduling of task graphs with imprecise computations and end-to-end deadlines. ACM Transactions on Design Automation of Electronic Systems (TODAES), 25(1), 1-21.
13. Stavrinides, G. L., & Karatza, H. D. (2010). Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise computations. Journal of Systems and Software, 83(6), 1004-1014.
14. Stavrinides, G. L., & Karatza, H. D. (2011). The impact of input error on the scheduling of task graphs with imprecise computations in heterogeneous distributed real-time systems. In Analytical and Stochastic Modeling Techniques and Applications: 18th International Conference, ASMTA 2011, Venice, Italy, June 20-22, 2011. Proceedings 18 (pp. 273-287). Springer Berlin Heidelberg.
15. Stavrinides, G. L., & Karatza, H. D. (2012). Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes. Future Generation Computer Systems, 28(7), 977-988.
16. Stavrinides, G. L., & Karatza, H. D. (2015, August). A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds. In 2015 3rd International Conference on Future Internet of Things and Cloud (pp. 231-239). IEEE.
17. Stavrinides, G. L., & Karatza, H. D. (2018, August). Energy-aware scheduling of real-time workflow applications in clouds utilizing DVFS and approximate computations. In 2018 IEEE 6th international conference on future internet of things and cloud (FiCloud) (pp. 33-40). IEEE.
18. Stavrinides, G. L., & Karatza, H. D. (2019). An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. Future Generation Computer Systems, 96, 216-226.
19. Cao, K., Zhou, J., Xu, G., Wei, T., & Hu, S. (2019). Exploring renewable-adaptive computation offloading for hierarchical QoS optimization in fog computing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(10), 2095-2108.
20. Mora Mora, H., Gil, D., Colom López, J. F., & Signes Pont, M. T. (2015). Flexible framework for real-time embedded systems based on mobile cloud computing paradigm. Mobile information systems, 2015.
21. Mo, L., & Kritikakou, A. (2019). Mapping imprecise computation tasks on cyber-physical systems. Peer-to-Peer Networking and Applications, 12(6), 1726-1740.
22. Bittencourt L, Immich R, Sakellariou R, Fonseca N, Madeira E, CuradoM, Villas L, da Silva L, Lee C, Rana O (2018) The internet of things, fog and cloud continuum: Integration and challenges. Internet of Things 3:134–155

23. Mahmud, R., Ramamohanarao, K., & Buyya, R. (2018). Latency-aware application module management for fog computing environments. ACM Transactions on Internet Technology (TOIT), 19(1), 1-21.
24. Mehta, R., Sahni, J., & Khanna, K. (2023). Task scheduling for improved response time of latency sensitive applications in fog integrated cloud environment. Multimedia Tools and Applications, 82(21), 32305-32328.
25. BehzadianM, Otaghsara SK, YazdaniM, Ignatius J (2012) A state of the-art survey of TOPSIS applications. Expert Syst Appl 39(17): 13051–13069
26. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience, 47(9), 1275-1296.
27. Mehta, R., Sahni, J., & Khanna, K. (2018). Internet of things: Vision, applications and challenges. Procedia computer science, 132, 1263-1269.
28. Ijaz, S., Munir, E. U., Ahmad, S. G., Rafique, M. M., & Rana, O. F. (2021). Energy-makespan optimization of workflow scheduling in fog–cloud computing. Computing, 103, 2033-2059.
29. Yao, S., Hao, Y., Zhao, Y., Shao, H., Liu, D., Liu, S., ... & Abdelzaher, T. (2020, August). Scheduling real-time deep learning services as imprecise computations. In 2020 IEEE 26th international conference on embedded and real-time computing systems and applications (RTCSA) (pp. 1-10). IEEE.