

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository: <https://orca.cardiff.ac.uk/id/eprint/140021/>

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Ijaz, Samia, Munir, Ehsan Ullah, Ahmad, Saima Gulzar, Rafique, M. Mustafa and Rana, Omer F. 2021. Energy-makespan optimization of workflow scheduling in fog-cloud computing. Computing 103 , pp. 2033-2059. 10.1007/s00607-021-00930-0

Publishers page: <http://dx.doi.org/10.1007/s00607-021-00930-0>

Please note:

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See <http://orca.cf.ac.uk/policies.html> for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.



Energy-Makespan Optimization of Workflow Scheduling in Fog-Cloud Computing

Samia Ijaz · Ehsan Ullah Munir · Saima
Gulzar Ahmad · M. Mustafa Rafique ·
Omer F. Rana

the date of receipt and acceptance should be inserted later

Abstract The rapid evolution of smart services and Internet of Things (IoT) devices accessing cloud data centers can lead to network congestion and increased latency. Fog computing, focusing on ubiquitously connected heterogeneous devices, addresses latency and privacy requirements of workflows executing at the network edge. However, allocating resources in this paradigm is challenging due to the complex and strict Quality of Service constraints. Moreover, simultaneously optimizing conflicting objectives, e.g., energy consumption and workflow makespan increases the complexity of the scheduling process. We investigate workflow scheduling in fog-cloud environments to provide an energy-efficient task schedule within *acceptable* application completion times. We introduce a scheduling algorithm, *Energy Makespan Multi-Objective Optimization (EM-MOO)*, that works in two phases. First, it models the problem as a multi-objective optimization problem and computes a trade-off between conflicting objectives while allocating fog and cloud resources, and schedules latency-sensitive tasks (with lower computational requirements) to fog resources and computationally complex tasks (with low latency requirements) on cloud resources. We adapt the *Deadline-Aware stepwise Frequency Scaling (DAFS)* approach to further reduce energy consumption by utilizing unused time slots between two already scheduled tasks on a single node. Our evaluation using synthesized and real-world applications shows that our approach reduces energy consumption, up to 50%, as compared to existing approaches with minimal impact on completion times.

Samia Ijaz, Ehsan Ullah Munir, Saima Gulzar Ahmad
COMSATS University Islamabad, Wah Campus, Pakistan
E-mail: {samia_s; ehsan; saimagulzarahmad}@ciitwah.edu.pk

M. Mustafa Rafique
Department of Computer Science, Rochester Institute of Technology, United States
E-mail: mrafique@cs.rit.edu

Omer F. Rana
School of Computer Science & Informatics, Cardiff University, United Kingdom
E-mail: ranaof@cardiff.ac.uk

Keywords Cloud Computing · Fog Computing · Workflow Scheduling · Makespan · Energy Consumption · DVFS

1 Introduction

The use and adoption of heterogeneous Internet of Things (IoT) devices has revolutionized Information and Communication Technology (ICT) [?]. IoT have extended internet connectivity beyond conventional devices such as tablets and phones to a wide range of smart devices such as TV, wearable appliances, vehicles and surveillance cameras [?]. These smart connected devices are integrated into many applications, e.g. intelligent traffic control systems, health monitoring units and security systems. The IoT market is growing rapidly and billions of devices are expected to be connected to the Internet to provide valuable services to the users [?]. The deluge of data being generated needs fast processing and analysis to extract valuable information.

Over the last decade, the storage, analysis, and computations associated with data-intensive applications are performed on resources hosted at centralised cloud data centers [?]. However, due to this centralisation, cloud computing is experiencing challenges to meet the requirements of latency-sensitive IoT applications that require real-time response times [?]. As cloud data centers are further away from data generation sources, by the time the data arrives at the cloud center for analysis and processing, it might already be too late to extract useful information from the received data. Moreover, with the predicted increase in the number of connected IoT devices and large physical distances between these devices and cloud data centers, the conventional architecture will be unable to handle communication requirements for transmitting large data volumes over public networks to a centralised location, due to traffic congestion and communication costs [?]. Therefore, a computing paradigm suitable for handling the variety, velocity, and volume of the data generated by IoT devices is needed.

Fog computing architectures provide cloud-like services near the edge of the network [?, ?]. The network edge is transformed into a distributed computing paradigm by connecting edge devices to provide storage and computation services to run IoT applications [?, ?]. Fog nodes [?] can be deployed near an IoT data source, such as near bus terminals, on University premises or at shopping malls, to offer services for time-critical applications. The computationally-intensive application tasks are offloaded to the cloud data centers for processing and long-term storage while lightweight and time-sensitive tasks can be processed locally on edge devices [?].

In practice, IoT applications are generally composed of modules that execute tasks (e.g. microservices), where each module carries out part of the overall processing (either independently or with dependency/data flow constraints between tasks) [?]. Workflow is one of the most commonly used models in these applications and can be represented as a directed acyclic graph (DAG) [?]. An application workflow can be executed across a number of dif-

ferent types of fog-cloud systems that make up the execution environment for such workflows.

Workflow scheduling algorithms must take dependency constraints and resource properties into account, aiming to improve effective resource utilization (i.e. make most effective use of the available resources), whilst minimising the overall completion time (or makespan) of the application. It is a well-known NP-complete problem [?] and should be optimized using approximate solutions in near polynomial time [?].

Apart from makespan, energy consumption is another crucial parameter in a fog-cloud environment. The energy consumption of cloud data centers has increased considerably during the last decade subsequently resulting in soaring economic costs of energy apart from operational expenses and environmental impacts. Moreover, the resource constrained nature of fog nodes makes energy a serious challenge since they usually run on batteries or have access to limited (renewable) energy, or are deployed in areas with limited and unreliable energy sources [?]. Therefore, green cloud computing has received significant attention in both academia and industry and reducing energy consumption in emerging fog-cloud infrastructure has become a major issue [?].

A scheduling algorithm that can minimize the makespan of an application but consume considerable energy is not an optimal choice for use on fog resources. This becomes more challenging when multiple conflicting objectives must be satisfied simultaneously. For instance, it is challenging to reduce makespan while also reducing the energy consumption required to complete application processing. Therefore, a bi-objective optimization approach is required for finding the right compromise between these optimization objectives.

The scheduling problem has been widely explored for cloud environments, either as a unique objective or a multi-objective optimization problem, however, it is not well-studied for fog-cloud infrastructures. In this paper, we first formulate the problem as a multi-objective optimization model that considers both makespan minimization and reduction in energy consumption. Since both the objectives are conflicting in nature, we apply an adaptive weighted bi-objective cost function. The value of the weight indicates which criteria (makespan or energy) is considered to be more important by a user. The overall aim is to obtain the right compromise between application completion time and energy consumed during workflow execution.

A common power management practice is the *Dynamic Voltage and Frequency Scaling (DVFS)* technique [?]. DVFS permits the dynamic adjustment of frequency and voltage of a processor. The voltage regulators in modern multi-core processors enable each core to operate at a different frequency and voltage level [?]. We apply a frequency scaling technique to achieve further reduction in energy consumption by utilizing DVFS on the underlying processors while satisfying application completion time constraints. This is achieved by availing possible gaps (empty slots) in a schedule between tasks. We perform experimental evaluation and compare our approach with existing work on synthesized and real-world application workflows, e.g. Cybershake [?] and Montage [?]. The results show that our approach generates efficient sched-

ules with optimized energy consumption. Specifically, we make the following contributions in this paper.

The rest of this paper is organized as follows. We summarize related work in Section 2. Section 3 introduces the fog-cloud architecture that we consider in this paper. In Section 4, we formally define the problem addressed in this paper and present its proposed solution in Section 5. We present performance evaluation of our solution in Section 6. Finally, we conclude the paper in Section 7.

2 Related Work

There have been several studies that address the workflow scheduling problem in heterogeneous computing systems. The NP-hard nature of the problem demands more heuristic approaches to approximate optimal solutions.

Heterogeneous Earliest Finish Time (HEFT) is the most commonly adopted scheduling algorithm [?]. HEFT operates in two phases; the task prioritization phase and the processor selection phase. First phase assigns priorities to the tasks on the basis of their upward ranks while the second phase chooses suitable processor for the task execution considering the minimal task completion time. Another well-known algorithm in this category is Predict Earliest Finish Time (PEFT) [?]. In PEFT, the Optimistic Cost Table (OCT) is computed to assign priorities to tasks and it also helps in determining the most appropriate processor for task execution in the scheduling phase. Both HEFT and PEFT are single-objective optimization approaches that take into account only makespan minimization while EM-MOO is a multi-objective optimization approach that considers energy consumption along with makespan.

In [?], a makespan minimization algorithm Minimal Optimistic Processing Time (MOPT) is introduced that modifies the prioritization phase by computing Optimistic Processing Times (OPT) of tasks on all processing nodes and then ranks are assigned based on average OPT values. The node selection phase improves the entry task duplication feature by allowing duplication only in case this helps in minimizing the completion time of successor tasks. Again, MOPT is a single-objective optimization technique as compared to our proposed work in this article. A hybrid meta-heuristic approach combining the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) is suggested in [?] for minimizing makespan in a multi-processor cloud environment. The bottom level (b-level) of a task is used to assign priorities. The b-level is maximum execution time a task takes to traverse all levels of the graph. Then, ACO is applied to identify a suitable path that is further improved by applying GA.

One of the few studies that consider task scheduling in fog computing as a DAG scheduling problem is presented in [?]. It introduces Cost-Makespan aware Scheduling (CMaS) algorithm to satisfy the user's QoS requirements of makespan and cost optimization and proposes a utility function to determine the tradeoff between both these objectives. The obtained schedule is improved by the task reassignment phase.

Task Scheduling in Fog Computing (TSFC) algorithm is based on the classification mining algorithm [?]. The association rules obtained from the I-Apriori algorithm are combined with the task completion times without taking bandwidth into account between machines. Task scheduling in fog computing supported software-defined embedded systems (FC-SDES) [?] minimizes the makespan. It proposes a low-complexity 3-phase algorithm that incorporates task scheduling, resource management, and I/O request balancing.

The computational requirements of users have increased significantly as a result of cloud computing and the emergence of fog paradigm, therefore energy consumption has become a challenging objective for optimization [?, ?]. This is an active research area and efforts are being made towards developing workflow scheduling algorithms that consider energy consumption factors. Next, we summarize energy optimization algorithms used in cloud and fog environments.

DVFS-enabled Energy Efficient Workflow Task Scheduling (DEWTS) algorithm is a state of the art algorithm in the cloud environment that applies Dynamic Voltage and Frequency Scaling (DVFS) technique to optimize energy usage during unused time slots in the scheduling process [?]. The core idea is to turn-off the less utilized machines and reassign their tasks to the turned-on machines. Next, the algorithm utilizes idle time slots of the machines under lower frequency and voltage to allocate tasks and obtain reduced power consumption.

The energy-efficient scheduling problem in a mobile cloud environment is investigated in [?]. This approach, Energy Makespan in Mobile Cloud Computing (EM-MCC), starts by allocating tasks to machines using the least-delay scheduling approach followed by task reassignment phase that migrates tasks among the local cores or cloud nodes to reduce energy usage. The DVFS technique is then applied for further energy reduction of mobile devices. EM-MCC is one of the algorithms used in the evaluation process of our proposed work. One of the limitations of EM-MCC is that its task migration phase increases the computational complexity of the algorithm.

In [?], an energy-aware processor merging (EPM) algorithm in a heterogeneous parallel and distributed framework is presented. EPM turns off the most effective machine in terms of energy saving. To overcome the high computational complexity of EPM, it introduces a less complex quick EPM (QEPM) algorithm that simply turns off machines with minimum energy consumption. However, QEPM consumes more energy than EPM. In [?], authors handle multiple conflicting objective problems in cloud environments by introducing a pareto-based multi-objective hybrid approach, hybrid Particle Swarm Optimization algorithm (HPSO). The algorithm tries to optimize energy, budget and schedule length of the application by generating a set of Pareto optimal solutions. DVFS approach is used to optimize energy consumption.

An energy-efficient scheduling algorithm for placing tasks on nodes based on their remaining CPU time and energy state is proposed in [?]. Improved Round Robin (IRR) and DVFS approaches are followed to achieve energy-efficient scheduling. The strategy aims to place delay-sensitive tasks to fog

Table 1: Summary of related work.

Approach	Objective	Limitation
CPU time, round robin policy [?]	Makespan	Suitable only in the fog computing environment
List scheduling [?]	Makespan	High energy consumption
Task migration/DVFS [?]	Makespan, energy	High computational complexity due to task migration
Processor merging [?]	Energy	High computational complexity
Heuristic [?]	Makespan, cost	Small dataset used for experimentation
Hybrid meta-heuristic [?]	Makespan	High computational complexity
Convex optimization [?]	Time, energy	Not suitable for distributed fog environment
Heuristic [?]	Makespan	Does not consider dependent task scheduling

devices effectively. Execution time and power optimization for task allocation in fog-cloud environments is studied in [?]. The problem is split into three separate domains; fog, cloud, and WAN. Existing optimization approaches are applied to provide solutions to each sub-problem. The study reveals that fog computing boosts performance as latency and bandwidth are minimized. However, more WAN communications increase overall execution delays and workload on fog nodes increases their energy consumption.

We present a summary of the common scheduling algorithms in Table 1. Although the problem has widely been studied for the cloud environment, there is much space for the NP-hard scheduling problem to be explored in the fog-cloud paradigm. Moreover, existing work mostly focuses on single objective optimization that reduces performance especially in environment where complex applications generated from real-time IoT devices need to be executed. To overcome limitations in the existing studies, we have proposed a simplified and less complex solution to the workflow scheduling problem for finding a tradeoff between energy consumption and makespan. This approach has not been explored much in the fog-cloud context in the related studies. Moreover, our algorithm utilizes DVFS on the underlying processors in order to fill empty scheduling slots and thus, achieves reduced energy consumption with adaptive deadline adjustment.

3 Fog Enabled Cloud System Architecture

The fog-cloud system architecture, shown in Fig. 1, typically has three layers [?]. The terminal layer or bottom layer consists of IoT devices, e.g., home appliances, sensors, wearable devices, smartphones and smart vehicles [?]. These devices send requests and data to the higher-level layers for application processing. In this paper, we consider IoT devices that are sources of data generation and do not have capabilities to process the generated data.

The fog computing layer is the intermediate layer between IoT devices and cloud data centers and is generally deployed near IoT devices. It is formed by a large number of nodes residing near the edge of the network, e.g., routers,

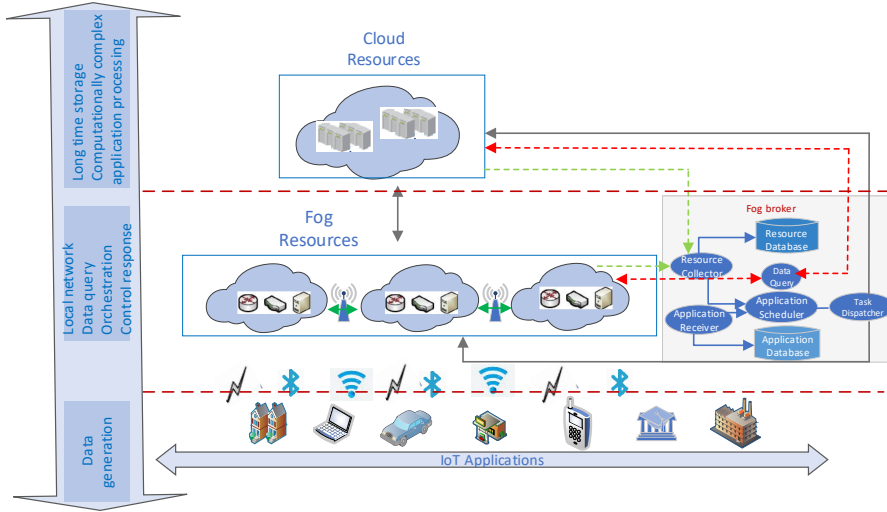


Fig. 1: Fog-cloud architecture.

access points, switches, base stations with limited computation, transmission, and temporary storage nodes [?]. Users can access the fog nodes to obtain the required services. The fog layer is connected with the cloud data centers to satisfy complex computing and storage requirements. The nodes at this layer are organized in a hierarchical fashion. The lower level layer is expected to be one or two hops away from the end-users to provide services with low, i.e., few milliseconds, latency requirements [?].

The uppermost cloud layer of the fog-cloud platform comprises of multiple high-performance cloud servers and provides powerful computation and permanent storage services for IoT applications [?]. This layer provides computing services for compute-intensive workloads, and efficient and reliable storage facilities for persistent data. It is desirable to only offload compute-intensive tasks to the cloud layer as it is further away from data sources and incurs high communication delays to transfer the input and output data.

There is a dedicated fog node, known as fog broker, in the fog layer that plays the role of centralized management and task scheduler [?]. It is responsible for collecting user requests, managing resources on fog and cloud nodes and generating most suitable schedules for the input workflows.

4 Problem Definition

In this section, we provide a formal definition of the problem tackled in this paper. Workflow scheduling in the fog-cloud system is defined as the problem of allocating tasks of the workflow to the machines of the target system in a way that an optimal schedule is achieved with a tradeoff between energy consumption and makespan minimization. For ease of understanding, Table 2 provides a description of notation used in this paper.

Table 2: Major notations used in this paper.

Symbol	Notation
DAG	Directed Acyclic Graph
V	Set of all the tasks
E	Set of all edges showing dependencies in workflow
$e_{i,j}$	Edge/dependency between v_i and v_j
N	Set of all the processing nodes
v_i	The i^{th} task in the workflow
n_j	The j^{th} node in the system
$pred(v_i)$	The predecessor task of v_i
$succ(v_i)$	The successor task of v_i
$W(v_i, n_j)$	Computational weight of v_i on n_j
$CC(v_i)$	Average data transfer time associated with v_i
$MST(v_i, n_j)$	Minimum Start Time of v_i on n_j
$MCT(v_i, n_j)$	Minimum Completion Time of v_i on n_j
$AST(v_i, n_j)$	Actual Start Time of v_i on n_j
$ACT(v_i, n_j)$	Actual Completion Time of v_i on n_j
M	The makespan of the workflow
$ec(v_i, n_j)$	Energy consumed while executing v_i on n_j
f_{n_j}	Actual operating frequency of n_j
EC	Total energy consumption for the workflow
$OPT(v_i, n_j)$	Optimistic Processing Time of v_i on n_j
α	Weight/tradeoff factor for the cost function
D_{max}	Maximum deadline limit

4.1 Application Model

An application workflow is defined as a set of interdependent tasks and is modeled as a Directed Acyclic Graph, $DAG = (V, E)$ whose vertices are the tasks and edges describe data dependencies between them. Let $V = \{v_1, v_2, \dots, v_t\}$ be the set of t tasks in the workflow and E is the set of edges. An edge $e_{i,j} \in E$ defines the data flow dependency from task v_i to v_j with a precedence constraint that the processing of v_i must be complete before v_j could be initialized as former is a predecessor task. The set of all direct predecessors and successors of v_i are represented as $pred(v_i)$ and $succ(v_i)$ respectively. A task v_i without any predecessor is known as entry task v_{entry} , and a task without any successor is called an exit task, v_{exit} . In this model, we consider only one entry task and one exit task. Therefore, if a DAG has more entry/exit tasks, a dummy task with zero computation and communication costs is added. This does not have any impact on the schedule.

The size of each task is measured in million instructions and each task appears only once in a schedule. Moreover, we assume Non-Preemptive Execution (NPE) of tasks in our model. A task uses two types of inputs: (i) from a predecessor task; (ii) from other data sources on a cloud/fog platform. However, only one of these may be present in some instances.

4.2 System Model

The target fog-cloud system has heterogeneous computational nodes: fog and cloud nodes. Former have lower computational capabilities and are located

closer to data generation source while latter have higher computational capabilities but are far from data source.

We define the target system $S = (N, L)$ as a directed graph where $N = \{n_1, n_2, \dots, n_p\}$ is representing the set of all processing nodes and $l_{i,j} \in L$ denotes the link that connects n_i and n_j . Each node in N may be a cloud (N_c) or a fog node (N_f). Therefore,

$$N = N_c \cup N_f \quad (1)$$

such that, $w(N_f) < w(N_c) \forall N$, i.e., as described earlier, computation capability $w(N_f)$ of fog nodes is assumed to be less than that of cloud nodes $w(N_c)$ due to physical constraints on fog devices. Also, higher stability of LAN as compared to WAN results in better bandwidth between fog nodes than that of cloud nodes. The computational capacity of a node is measured in million instructions per second (MIPS). Let, $x_{i,j}$ represents the task allocation matrix. If v_i is allocated to n_j , then $x_{i,j} = 1$, else $x_{i,j} = 0$.

4.3 Makespan Model

The makespan represents overall completion time of the application. Minimizing makespan for workflow applications is a crucial problem in the fog-cloud environment to achieve efficient schedules. Suppose a task v_i is scheduled on a node n_j . Let $MST(v_i, n_j)$ and $MCT(v_i, n_j)$ be the Minimum Start Time and Minimum Completion Time for v_i on n_j respectively and n_j is available for the execution of v_i . If v_i is the entry task then,

$$MST(v_{entry}, n_j) = 0 \quad (2)$$

For remaining tasks in the graph, MST and MCT are computed recursively, starting from the top, as described in Eq. 3 and Eq. 4 respectively.

$$MST(v_i, n_j) = \max_{v_p \in pred(v_i)} \{ACT(v_p) + CC(v_i)\} \quad (3)$$

$$MCT(v_i, n_j) = MST(v_i, n_j) + W(v_i, n_j) \quad (4)$$

where the parameter $CC(v_i)$ represents average transfer time when all the input data has been transferred to the selected node n_j for execution of v_i . The cost will be zero in case both the parent and child tasks are assigned to the same node. The term $W(v_i, n_j)$ represents the computation time of v_i on n_j . After a task is scheduled on a node, the MST and MCT become Actual Start Time (AST) and Actual Completion Time (ACT) of the task respectively. Eventually, the makespan (M) of the application is equal to the actual completion time of the last task, v_{exit} , of the workflow i.e.

$$M(DAG) = ACT(v_{exit}) \quad (5)$$

4.4 Energy Model

In this study, we adopt the classic power model to analyze power and energy consumption [?]:

$$P = P_{static} + P_{dynamic} \quad (6)$$

where, P_{static} is the power consumption when the system does not execute any workload, i.e. it is power used when the system is turned on. Dynamic power dissipation $P_{dynamic}$, is the dominant and expensive component of energy and is defined as:

$$P_{dynamic} = C.V_{dd}^2.f \quad (7)$$

where C is the capacitance load, V_{dd} is the supply voltage and f is the frequency. Since $f \propto V_{dd}^\beta$ for $(0 < \beta < 1)$, or $V_{dd} \propto f^{1/\beta}$ i.e. the frequency-dependent dynamic power is,

$$P_{dynamic} \propto f^\gamma \text{ where } \gamma = 1 + 2/\beta \geq 3.$$

Therefore, in our study, we represent power consumption as:

$$P = P_{static} + C.f^\gamma \quad (8)$$

The maximum operating frequency of the node n_j is $f_{n_j}^{max}$ and there are L scaling factors ($a_{n,1} < a_{n,2} \dots < a_{n,L} = 1$). To acquire our objective of minimizing energy consumption, we will execute the tasks at the lowest frequency whilst ensuring that the completion time deadline is met. Therefore, the actual operating frequency on which n_j might operate can be computed as:

$$f_{n_j} = a_{n_j,l}.f_{n_j}^{max} \quad (9)$$

The power consumed by n_j on that particular frequency would be:

$$P(n_j) = P_{static} + a_{n_j}.f_{n_j}^{\gamma_j} \quad (10)$$

Therefore the energy consumption for processing v_i on n_j is computed as:

$$ec(v_i, n_j) = P(n_j).W(v_i, n_j) \quad (11)$$

where $W(v_i, n_j)$ is the computation time of v_i on n_j . The total energy consumption by n_j for executing all tasks v assigned to it is computed as:

$$ec_{total}(n_j) = \sum_{v \in V} ec(v, n_j) \quad (12)$$

The energy consumption function for running the complete workflow application is obtained by:

$$EC(DAG) = \sum_{n \in N} ec_{total}(n) \quad (13)$$

where n can be a cloud or a fog node.

4.5 Multi-objective Optimization Model

When the processing nodes operate at higher frequencies, the performance of the system increases in terms of reduced application completion time but at the cost of higher energy consumption. The two objectives, energy consumption and makespan are therefore conflicting in nature and the energy-makespan tradeoff is faced for appropriate node selection to run the application tasks. The joint minimization of both objectives is a multi-objective optimization problem (MOO). Our objective is to introduce a workflow scheduling solution that achieves a tradeoff for this MOO problem.

We address the scheduling problem in two phases to achieve the desired objectives. First, we formulate a MOO model for the optimization of makespan ($M(DAG)$) and energy consumption ($EC(DAG)$) for a workflow application $DAG(V, E)$ with a set V of tasks with aforementioned constraints and a set N as:

$$\text{Minimize } \begin{cases} M(DAG) \\ EC(DAG) \end{cases}$$

subject to the following constraints,

$$\begin{aligned} x_{i,j} &\in \{0, 1\}, \forall v_i \in V \& \forall n_j \in N \\ \sum_{n=1}^P x_{i,j} &= 1, \forall v_i \in V \end{aligned}$$

The constraints specify that a task can only be allotted to a single node.

The ever increasing demand for cloud computing and the power-constrained nature of fog devices makes energy consumption a challenging factor. Therefore, in the second phase of our work, we employ a step-wise frequency scaling technique for further reduction in energy consumption. To achieve this, we have put a deadline constraint on the overall execution time of the application produced in the previous phase and then the schedule gaps among the tasks are utilized. This helps in achieving further reduction in energy consumption. Mathematically,

$$\text{Minimize}(EC(DAG))$$

subject to,

$$M(DAG) < D$$

where, D is the deadline for the entire workflow that is computed based on the value of makespan obtained in the first phase.

5 Energy-Makespan based Multi-Objective Optimization (EM-MOO)

We design a novel workflow scheduling algorithm EM-MOO to minimize energy consumption and makespan and find a tradeoff between these conflicting objectives. Fig. 2 shows the proposed methodology and the interactions between different components. Initially, the algorithm assigns ranks to the tasks to determine their scheduling order. Next, an appropriate processing node is selected for each task based on the bi-objective cost function.

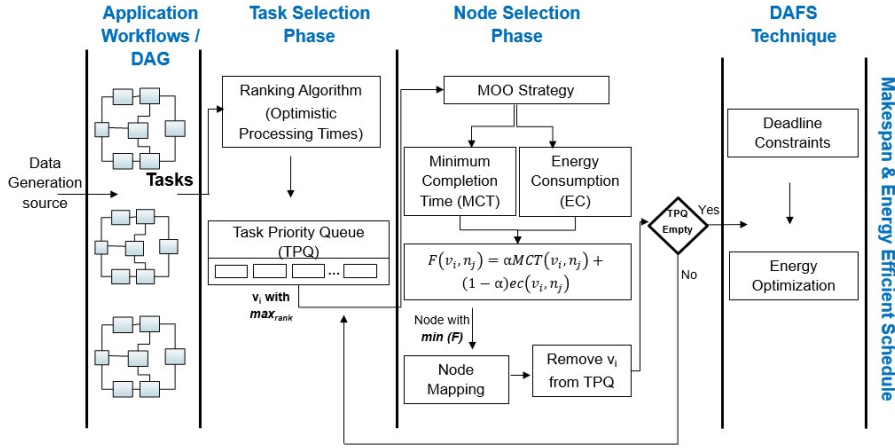


Fig. 2: Proposed methodology: EM-MOO algorithm.

5.1 Task Selection Phase

The task selection phase assigns weights to tasks to determine their execution order, primarily to achieve efficient schedules during the node selection phase. We assign weights to the tasks based on their Optimistic Processing Times (OPT), which is the key parameter that defines the overall completion time of the workflow application [?]. OPTs are computed before tasks are mapped to any node and are therefore called optimistic. To compute OPT for a task v_i on a node n_j , $OPT(v_i, n_j)$, first the Optimistic Minimum Start Time (OMST) of v_i on n_j is determined using the following equation.

$$OMST(v_i, n_j) = \max_{v_p \in pred(v_i)} \left[\min_{n_w \in N} \{OMST(v_p, n_w) + W(v_p, n_w) + CC(v_i)\} \right] \quad (14)$$

The inner function of Eq. 14 calculates the OPTs for the predecessors of v_i . Then, the $OPT(v_i, n_j)$ is computed as:

$$OPT(v_i, n_j) = OMST(v_i, n_j) + W(v_i, n_j) \quad (15)$$

where $W(v_i, n_j)$ is the computation cost of v_i on n_j . Once the OPT values for all tasks have been computed, the rank assignment is calculated using the following equation.

$$rank_{opt}(v_i) = \frac{\sum_{j=1}^p OPT(v_i, n_j)}{p} \quad (16)$$

Finally, a Task Priority Queue (TPQ) is maintained with the tasks arranged in the increasing order of $rank_{opt}$ and the task with the least rank value has the highest priority. Thus, priority is given to smaller tasks that result in reducing the waiting time for execution of the other tasks.

5.2 Node Selection Phase

The highest priority task v_i from TPQ is picked for execution that can only begin when all the predecessor tasks of v_i have completed execution and the input data of v_i has reached the target execution node. The MST and MCT for v_i are computed on all $n_j \in N$ using Eq. 3 and Eq. 4 respectively. Besides the minimum completion time, the proposed algorithm also incorporates the energy consumption for selecting an appropriate node by using Eq. 11.

The proposed solution provides the joint optimization of minimum completion time and energy usage. However, time and energy are different metrics with different units. Therefore, to make these metrics comparable, we apply normalization to both the objectives so they are normalised to lie between 0 and 100. The normalized value for $MCT(v_i, n_j)$ is obtained as:

$$MCT_n(v_i, n_j) = 100 \times \frac{MCT(v_i, n_j) - \min_{p \in N} MCT(v_i, p)}{\max_{p \in N} MCT(v_i, p) - \min_{p \in N} MCT(v_i, p)} \quad (17)$$

and the energy consumption is normalized as:

$$ec_n(v_i, n_j) = 100 \times \frac{ec(v_i, n_j) - \min_{p \in N} ec(v_i, p)}{\max_{p \in N} ec(v_i, p) - \min_{p \in N} ec(v_i, p)} \quad (18)$$

Using Eq. 17 and Eq. 18 we can define a weighted bi-objective function that computes tradeoff between the two objectives.

$$F(v_i, n_j) = \alpha MCT_n(v_i, n_j) + (1 - \alpha) ec_n(v_i, n_j) \quad (19)$$

where α ($0 \leq \alpha \leq 1$) is the weight/tradeoff factor. We use the terms weight and tradeoff interchangeably in this paper. Note that for $\alpha = 0$ or 1 , the function becomes a single objective optimization problem with energy or makespan minimization respectively. Task v_i is assigned to the node n_j that provides the minimum value of the cost function. After v_i is scheduled on n_j , $MST(v_i, n_j)$ and $MCT(v_i, n_j)$ become the Actual Start Time, ($AST(v_i)$), and the Actual Completion Time, ($ACT(v_i)$), of v_i respectively and the makespan of the entire workflow is equal to the $ACT(v_{exit})$ as defined in Eq. 5. Similarly, total energy consumption for workflow scheduling is obtained by Eq. 13. The algorithm to determine the schedule and mapping of tasks to nodes is presented in Algorithm 1.

5.3 Deadline Aware stepwise Frequency Scaling Algorithm (DAFS)

The scheduling algorithm attempts to find minimum completion time and energy consumption during task execution and depends on the value of tradeoff factor α . In that phase, we assume the nodes are operating at a maximum frequency that results in higher energy consumption. To further reduce energy consumption, we apply a stepwise frequency scaling approach. However,

Algorithm 1: Energy-Makespan Multi-objective Optimization (EM-MOO) algorithm.

Input: Application graph, $DAG = (V, E)$, A set N of processing nodes, $N = \bigcup_{j=1}^p \{n_j\}$.

Output: A schedule S with optimal mapping v tasks on n nodes operating at maximum frequency.

```

begin
  for all tasks  $v_i \in V$  do
    for all nodes  $n_j \in N$  do
      Compute  $OMST(v_i, n_j)$  and  $OPT(v_i, n_j)$  using Eq. 14 & Eq. 15
      respectively.
      Generate a Task Priority Queue (TPQ) by sorting the tasks based on
      the increasing mean OPT value.
    while the unscheduled task queue is non-empty do
      Select the highest priority task  $v_i$  from the list.
      Compute MST, MCT, ec for  $v_i$  on each  $n_j \in N$ , using Eq. 3, Eq. 4 and
      Eq. 11 respectively.
      Normalize MCT & ec to  $MCT_n$  &  $ec_n$  using Eq. 17 and Eq. 18 respectively.
      Compute cost function  $F(v_i, n_j)$  from the Eq. 19.
      Schedule task  $v_i$  to the node  $n_j$  that minimizes  $F(v_i, n_j)$  for  $v_i$ .

```

decreasing the operating frequency of a node can result in increased task completion times and makespan for the entire workflow. For real-world application workflows, the output schedule must meet the application performance requirements and the completion time specified by the user at the time of job submission. In our algorithm, this deadline is dependent on the value of weight parameter α . We classify maximum deadline limit D_{max} into two categories based on α , loose deadline and tight deadline. In the case of former, we restrict the deadline not to be more than 25% of the original makespan obtained in the previous phase while the latter case does not allow to increase more than 10% of the previous makespan. For $\alpha \geq 0.5$, the user is inclined more towards makespan. Therefore, we observe a tight deadline limit, i.e.,

$$D_{max} = 1.1 \times T_{total} \quad (20)$$

For $\alpha \leq 0.5$, we set deadline limit in an adaptive way such that for $\alpha = 0$, it may not increase more than 25% of the initial makespan (T_{total}). The deadline is adaptively chosen as,

$$D_{max} = I \times T_{total} \quad (21)$$

where

$$I = \begin{cases} 1.15 & \text{for } \alpha = 0.5 \text{ \& } 0.4 \\ 1.2 & \text{for } \alpha = 0.3 \text{ \& } 0.2 \\ 1.25 & \text{for } \alpha = 0.1 \text{ \& } 0.0 \end{cases}$$

Our proposed approach does not change the task to node mapping produced in the node selection phase and thus avoids task rescheduling during

Algorithm 2: Deadline Aware stepwise Frequency Scaling Algorithm.

Input: An application schedule S for $DAG = (V, E)$, generated by the EM-MOO algorithm

Output: A schedule S' with modified execution frequency of v tasks on same processing nodes p as in EM-MOO.

```

begin
  for each task  $v_i$  in application workflow do
     $l=1$ 
    while  $l < L$  do
      Compute the modified completion time  $CT(v_i, n_j)$  of  $v_i$  when
        executed on  $l$ -th frequency on the same processing node  $n_j$ .
      if  $\exists$  another task  $v_k$  on  $n_j$  then
         $deadline_{lim1} = AST_k$ 
      else
         $deadline_{lim1} = D_{max}$   $\triangleright$  Using Eq. 18. In this case,  $v_i$  is last task on
           $n_j$ 
      if  $v_i \notin exit$  task then
         $deadline_{lim2} = \min_{v_s \in succ(v_i)} AST(v_s)$ 
      else
         $deadline_{lim2} = D_{max}$ 
      if  $CT(v_i, n_j) \leq deadline_{lim1} \ \& \ deadline_{lim2}$  then
        allocate  $l$ -th frequency to  $v_i$ .
        update  $ACT(v_i, n_j)$ .
    
```

the DAFS phase. The precedence constraints and communication delays involved before task execution can generate slack time between two consecutive tasks' execution on a single node. Our algorithm uses these slack times during the stepwise frequency scaling phase. For a task v_i scheduled on n_j , we compute slack time, $SlackTime$, of v_i using Eq. 22. $SlackTime_{v_i}$ is the maximum value that can be added to the completion time of v_i without affecting the start time of the next scheduled task v_k on n_j , and the start time of the successor tasks, $v_s \in succ(v_i)$, of v_i .

$$SlackTime_{v_i} = \min \left[AST(v_k), \min_{v_s \in succ(v_i)} \{AST(v_s)\} \right] \quad (22)$$

and the slack time for the exit task, i.e., $SlackTime_{v_{exit}}$, is computed as,

$$SlackTime_{v_{exit}} = D_{max} - ACT(v_{exit}) \quad (23)$$

We begin with the lowest frequency scaling factor and try v_i execution on each frequency level (with total L levels) iteratively until we acquire a frequency level from which further reduction will result in delayed task completion times and higher makespan. The pseudo-code of the algorithm is given in Algorithm 2.

6 Performance Analysis and Evaluation

We develop the simulation platform to evaluate the proposed algorithm in MATLAB environment. The UML Diagram of the implementation setup is

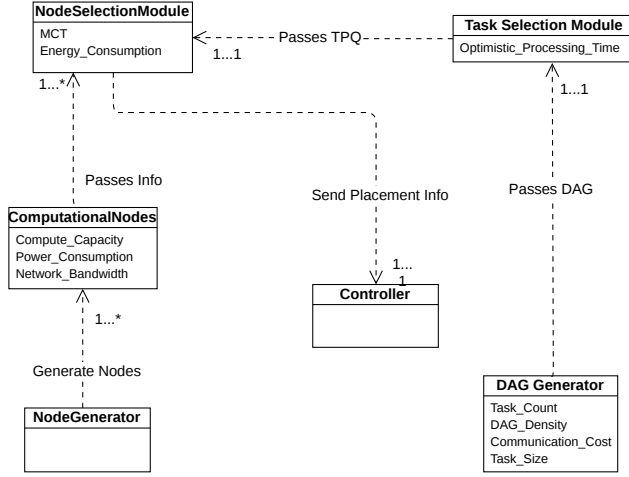


Fig. 3: UML Diagram of EM-MOO Implementation.

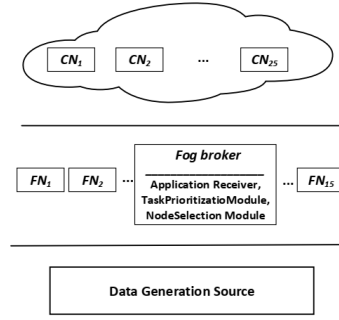


Fig. 4: Experimental Setup.

shown in Fig. 3 while Fig. 4 displays the experimental setup. We consider 40 processing nodes, out of which 15 are fog nodes (FN) and 25 are cloud nodes (CN) [?]. The processing capability of fog nodes is assumed to be lesser than that of cloud nodes and bandwidth in fog environment is higher than the cloud environment. We evaluate our work in two environments. The first environment considers synthesized workflows comprising of [100–500] tasks while for every evaluation, task size is randomly generated from [5000–50000] MI tasks. The density /shape parameter determines height of the graph. The processing capacity of the fog nodes is [10–500] MIPS and that of the cloud nodes is [500–1500] MIPS. For fog environment, the bandwidth is fixed at 1024 Mbps while the bandwidth for cloud environment is set as [10, 100, 512, 1024]. The

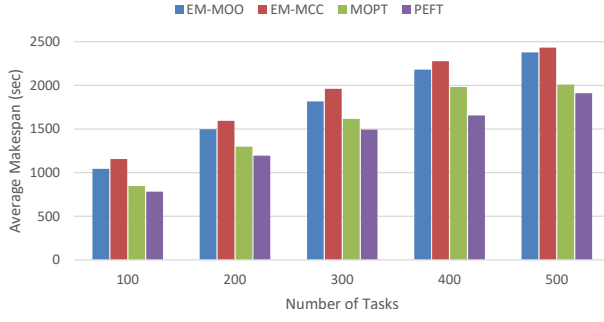


Fig. 5: Synthesized workflows: Average makespan as a function of the number of tasks.

second set is carried out on two types of benchmark workflows, i.e, Cyber-shake [?] and Montage [?], provided by the Pegasus workflow management system. To evaluate our work, we implemented EM-MCC [?], PEFT [?] and MOPT [?] algorithms. EM-MCC is multi-objective that optimizes energy as well as makespan, however, PEFT and MOPT are single objective makespan optimization approaches.

6.1 Performance Metrics

Three comparative metrics are used to evaluate our proposed work from various perspectives. The first metric used is makespan, which is the primary optimization objective and is widely used for evaluating workflow scheduling algorithms. It gives completion time of the entire workflow application. Energy consumption is another important metric that we consider in our multi-objective optimization approach. We conduct 1000 iterations for each set of experiments and compare average makespan and average energy consumption of EM-MOO with comparative algorithms. Moreover, to demonstrate that our work can acquire better tradeoff between energy and makespan than comparative approaches, we compute a third metric energy-makespan tradeoff (EMT) for all the algorithms as:

$$EMT(A_i) = \frac{\min_{A_k \in Alg} [EC(A_k)]}{EC(A_i)} \times \frac{\min_{A_k \in Alg} [M(A_k)]}{M(A_i)} \quad (24)$$

where $Alg = \{A_1, A_2, \dots, A_k\}$ is the set of all the algorithms. The maximum achievable EMT for any algorithm is 1, which is possible only if the algorithm produces the best results for both makespan and energy as compared to other algorithms. Otherwise, the higher the EMT for an algorithm, the better tradeoff it has achieved between both metrics.

6.2 Evaluation using Synthesized Workflows

Fig. 5 shows the average makespan comparison of the algorithms as a function of the workflow size that ranges from 100 to 500 tasks. PEFT and MOPT

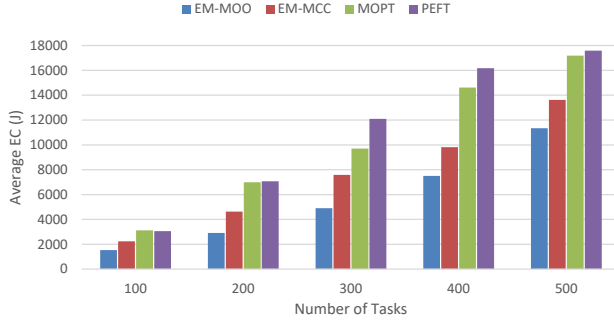


Fig. 6: Synthesized workflows: Average energy consumption as a function of the number of tasks.

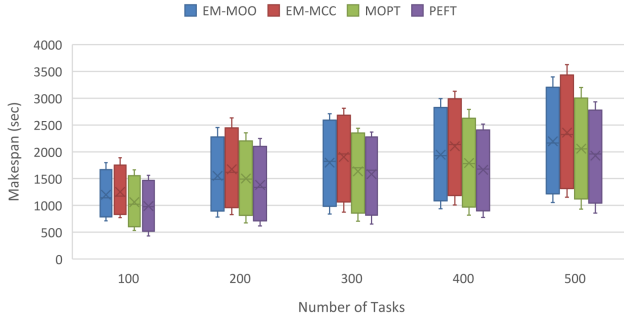


Fig. 7: Synthesized workflows: Boxplot of makespan as a function of the number of tasks.

produce reduced makespan as compared to EM-MOO and EM-MCC because the former algorithms consider only makespan optimization while the latter also incorporates energy consumption. EM-MOO produces better makespan as compared to EM-MCC as it puts strict deadline limits during step-wise frequency scaling phase by using α while the latter decreases energy consumption at the cost of increased makespan.

Fig. 6 illustrates that although PEFT and MOPT provide better performance in terms of makespan as compared to EM-MOO, the energy consumption by both algorithms is very high. On the other hand, our approach contributes towards achieving a balance between the optimization of makespan and energy consumption. The results confirm that the EM-MOO algorithm makes effective use of schedule gaps (slack). By putting a limit on the maximum deadline of the application with respect to the value of α , the improvements achieved in terms of energy reduction for workflows with 100 tasks are 17.12% as compared to EM-MCC and almost 50% as compared to both MOPT and PEFT. Similar improvements can be observed for other workflow sizes. Although both EM-MOO and EM-MCC do not change processing nodes during the frequency scaling phase, EM-MOO makes better resource utilization by considering the energy consumption parameter during the node selection phase apart from the completion time.

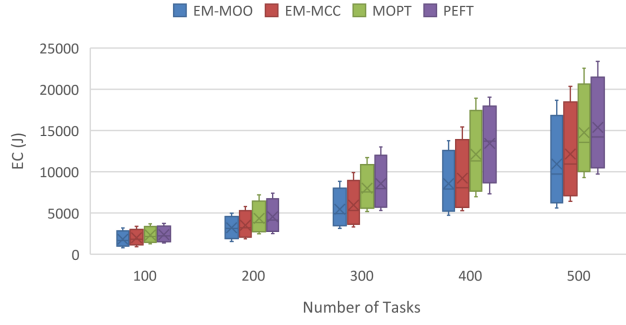


Fig. 8: Synthesized workflows: Boxplot of energy consumption as a function of the number of tasks.

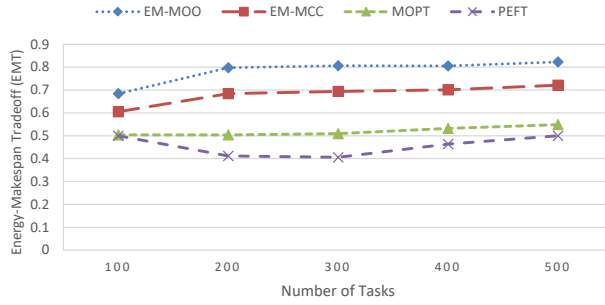


Fig. 9: Synthesized workflows: Energy-Makespan tradeoff.

A statistical analysis of the outputs for both metrics, i.e., makespan and energy consumption, are shown in Fig. 7 and Fig. 8 respectively. The plots show minimum, first quartile (25%), median, third quartile (75%) and maximum values from all the iteration results produced by the algorithms. The outcomes show the better performance of EM-MOO over other algorithms as discussed above.

Fig. 9 provides the energy-makespan tradeoff comparison of EM-MOO with comparative algorithms. The proposed approach, EM-MOO, achieves the highest tradeoff for all workflow sizes as compared to its competitors. For workflow size of 500 tasks, EM-MOO achieves 12%, 33%, and 39% improvement as compared to EM-MCC, MOPT, and PEFT respectively. The cost function and adaptive deadline-aware stepwise frequency scaling phase help our approach in achieving a better tradeoff between the conflicting objectives.

We also examine the behavior of EM-MOO by varying the number of cloud nodes and observing the impact on makespan and energy consumption for a fixed number of workflow tasks. We use 40 processing nodes in fog and cloud environment and change the computing environment by increasing the number of cloud nodes from 5 nodes to 25 nodes. The result of this experiment is shown in Fig. 10. We observe that increasing the number of cloud nodes provides better processing capability and leads to reduced makespan. The significant

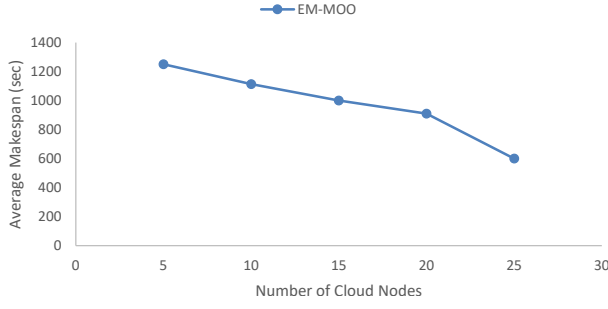


Fig. 10: Synthesized workflows: Impact of the number of cloud nodes on makespan.

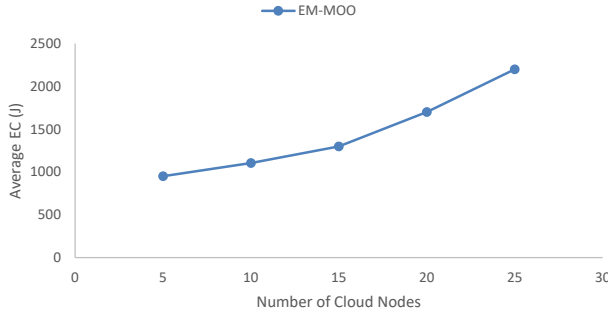


Fig. 11: Synthesized workflows: Impact of the number of cloud nodes on energy consumption.

Table 3: Energy consumption-Makespan per weight factor value.

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
M	3403.8	3072	2625.3	2247.1	2247.1	1849.7	1849.7	1713.2	1713.2	728.53	542.73
EC	30560	35614	41920	50056	50056	54951	54951	63143	63143	86027	88339

impact due to an increase in cloud nodes on energy consumption is displayed in Fig. 11.

Table 3 shows makespan and energy consumption for all values of the weight factor for a synthesized DAG with 100 tasks. Note that the refinement in makespan or energy reduction is linked with the value of the weight factor. Thus, depending upon the end-user requirements, the weight factor can be adjusted within the range ($0 \leq \alpha \leq 1$) such that one of the objectives is achieved by realizing an acceptable compromise for the other objective. Fig. 12a and Fig. 12b show the graphical representation of the data.

6.3 Evaluation using Real World Application Workflows

In the second set of experiments, we use real-world workflow applications Cybershake [?] and Montage [?], which are widely used for evaluating scheduling algorithms, from the Pegasus workflow management system. Cybershake is used by the Southern California Earthquake Center (SCEC) to identify the

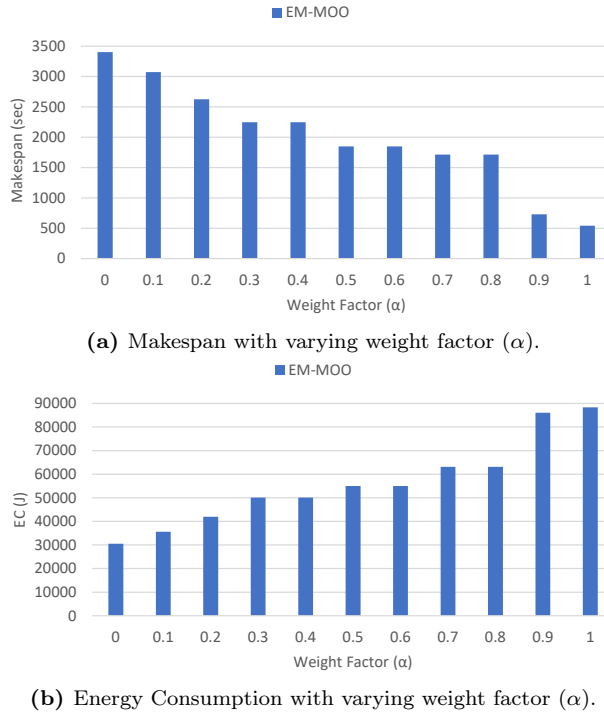


Fig. 12: Makespan and Energy consumption as a function of weight factor (α).

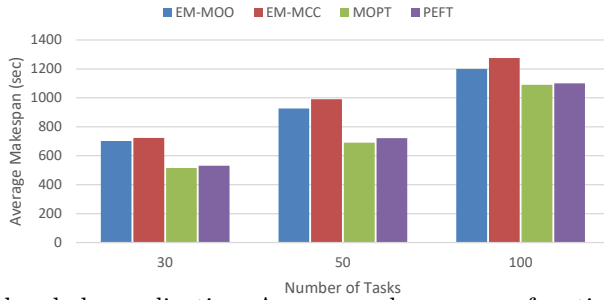


Fig. 13: Cybershake application: Average makespan as a function of the number of tasks.

earthquake within the region. It is relatively simpler workflow, but it can handle large datasets. Montage application workflow is employed in space industry to create mosaics of the sky by stitching together the input images. The size of workflow depends on the number of input images used to create the mosaic.

The analysis for Cybershake application is carried out on workflow sizes of 30, 50 and 100 tasks. The average makespan comparison is shown in Fig. 13 that indicates better EM-MOO performance as compared to EM-MCC but higher makespan as compared to MOPT and PEFT due to their single objec-

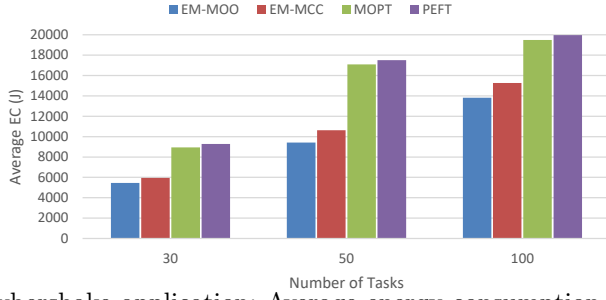


Fig. 14: Cybershake application: Average energy consumption as a function of the number of tasks.

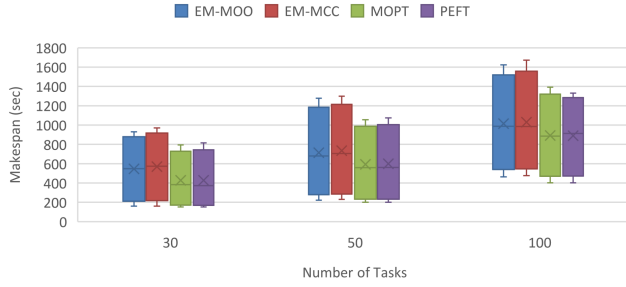


Fig. 15: Cybershake application: Boxplot of makespan as a function of the number of tasks.

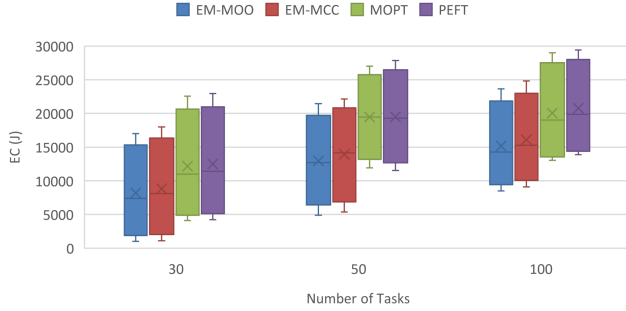


Fig. 16: Cybershake application: Boxplot of energy consumption as a function of the number of tasks.

tive optimization approach. The comparison of the average energy consumption can be analyzed in Fig. 14. The performance of EM-MOO is better than other popular alternatives that we have used in this paper. By restricting the deadline according to the value of α , the reductions in energy consumption are 9%, 29% and 30% as compared to those of EM-MOO, MOPT, and PEFT respectively for workflows with 100 tasks. Similar improvement trends are observed for workflows with 30 and 50 tasks. Fig. 15 and Fig. 16 provide the performance analysis of makespan and energy consumption, respectively, for

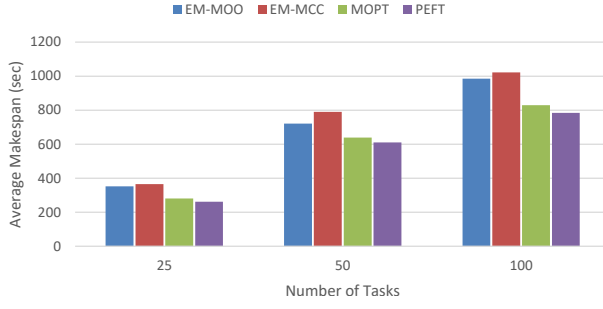


Fig. 17: Montage application: Average makespan as a function of the number of tasks.

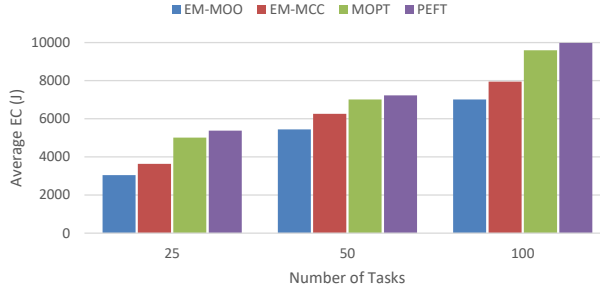


Fig. 18: Montage application: Average energy consumption as a function of the number of tasks

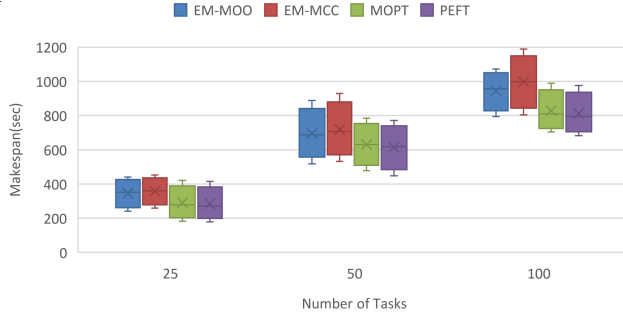


Fig. 19: Montage application: Boxplot of makespan as a function of the number of tasks.

varying number of tasks, and show that EM-MOO consumes less energy as compared to other alternatives while providing an acceptable makespan.

Similar experiments are conducted for Montage application, using workflows with 25, 50 and 100 tasks. Again, we note from Fig. 17 that the average makespan obtained by EM-MOO is better than EM-MCC as we have applied tight deadline constraints during the frequency scaling phase resulting in realizing a better tradeoff between energy consumption and makespan objectives. Since MOPT and PEFT are single objective makespan optimization algorithms, they generate lower makespans at the cost of high energy consump-

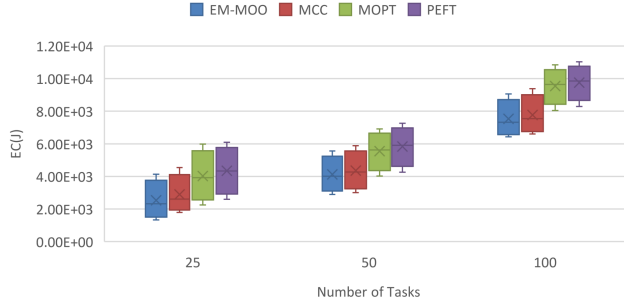


Fig. 20: Montage application: Boxplot of energy consumption as a function of the number of tasks.

tion, as shown in Fig. 18. The EM-MOO achieves a minimum improvement of 11.75% in terms of energy consumption for Montage workflow with 100 tasks compared to EM-MCC, to a maximum improvement of 43.33% for 25 tasks as compared to the PEFT approach. The average makespan for a similar number of nodes is improved by 3.54% from EM-MCC and decreased from PEFT by 20%. This is because our multi-objective approach is intended towards achieving a better tradeoff by putting a limit on the workflow completion time. Fig. 19 and Fig. 20 provide boxplot analysis of the algorithms using Montage application for makespan and energy consumption respectively. Since PEFT and MOPT are single objective optimization approaches, they produce a lower makespan compared to EM-MOO but at the cost of high energy consumption. Overall, the EM-MOO algorithm achieves minimum energy consumption within acceptable makespan as compared to other algorithms.

These results show that our approach achieves a good tradeoff between conflicting objectives and is a viable option for energy-efficient time-sensitive applications.

We analyze the computational complexity of the proposed algorithm in terms of workflow size t , the number of processing nodes p , and edge count e . The task selection phase takes $O(t \times p)$ time to compute the optimistic processing times of the tasks, $O(t)$ time for rank assignment and $O(t \log t)$ time for task sorting. The node selection phase requires $O(t \times p)$ time. The complexity therefore, becomes $O(t^2 \times p) + O(t \log t) + O(p)$. For the worst-case scenario with dense graphs, e becomes t^2 and complexity increases to $O(t^2 \times p)$. The stepwise frequency scaling phase has $O(t \times l)$ complexity where l is the number of frequency scaling levels. The overall computational complexity of the algorithm becomes $O(t^2 \times p)$ which is comparable to PEFT and MOPT but less than the EM-MCC algorithm that has higher complexity of $O(t^3 \times p)$ due to task migration phase. Thus, the proposed EM-MOO achieves a trade-off between energy and makespan with significantly less complexity.

7 Conclusion

With the rapid increase in latency-sensitive applications and with the evolution of power-constrained Internet of Things (IoT) devices, the workflow scheduling problem remains a crucial yet open challenge. The fog-cloud architecture provides a promising platform for the efficient processing of emerging applications because computing resources and services are distributed in fog nodes and lie near the edge of the network making them a viable alternative for data processing. The processing time of workflow applications and energy consumption of the cloud data centers and power-constrained IoT devices during the process are key challenges in the integrated IoT and cloud computing environments. In this paper, we propose a scheduling algorithm, energy-makespan multi-objective optimization (EM-MOO), to find a tradeoff between these conflicting objectives of reducing energy consumption and makespan. Initially, a weighted bi-objective cost function is introduced for selecting a processing node that minimizes task completion time and energy consumption based on a user-defined weighting factor. Next, we perform a further reduction in energy consumption by applying deadline constrained frequency scaling. The proposed work ensures completion of the workflow applications within the proposed deadline whilst also reducing energy consumption. The evaluation of our proposed approach using synthetic and real-world applications show that our proposed approach achieves better tradeoff between energy consumption and makespan as compared to the popular alternatives.

In future, we plan to explore meta-heuristics such as differential evolution and particle swarm optimization to solve the multi-objective optimization problem in fog-cloud paradigms.