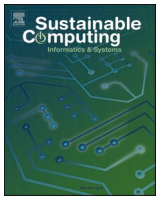
Sustainable Computing: Informatics and Systems 37 (2023) 100834

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom



IKH-EFT: An improved method of workflow scheduling using the krill herd algorithm in the fog-cloud environment

Navid Khaledian [a], Keyhan Khamforoosh [*,a], Sadoon Azizi [b], Vafa Maihami [a]

[a] Department of Computer Engineering, Sanandaj Branch, Islamic Azad University, Sanandaj, Iran
[b] Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Iran

ARTICLE INFO

Keywords:
Workflow scheduling
Cloud-fog computing
DVFS
Makespan
Krill Herd algorithm

ABSTRACT

Given the increase diversity of smart devices and objectives of the application management such as energy consumption, makespan users expect their requests to be responded to in an appropriate computation environment as properly as possible. In this paper, a method of workflow scheduling based on the fog-cloud architecture has been designed given the high processing capability of the cloud and the close communication between the user and the fog computing node, which reduces delay in response. We also seek to minimize consumption and reduce energy use and monetary cost in order to maximize customer satisfaction with proper scheduling. Given the large number of variables that are used in workflow scheduling and the optimization of contradictory objectives, the problem is NP-hard, and the multi-objective metaheuristic krill herd algorithm is used to solve it. The initial population is generated in a smart fashion to allow fast convergence of the algorithm. For allocation of tasks to the available fog-cloud resources, the EFT (earliest finish time) technique is used, and resource voltage and frequency are assumed to be dynamic to reduce energy use. A comprehensive simulation has been made for assessment of the proposed method in different scenarios with various values of CCR. The simulation results indicate that makespan exhibits improvements by 9.9, 8.7% and 6.7% on average compared with respect to the methods of IHEFT, HEFT and IWO-CA, respectively. Moreover, the monetary cost of the method and energy use have simultaneously decreased in the fog-cloud environment.

1. Introduction

It is unavoidable to use the Internet of Things (IoT) in life today. Given the ever-increasing development of IoT in the fields of health and industry, a very large number of applications have been produced in these areas [1,2]. Since users tend to use cellphones or tablet PCs to employ these applications, there are a large number of smart devices and applications, leading to the generation of big data and consequent complex analyses [3]. The complex analysis and massive processing of data causes small processing devices such as smart phones to encounter limitations in processing and resource allocation. Therefore, these devices have to outsource their processes, *i.e.* to leave the tasks of resource allocation and data processing to more powerful resources [4]. Currently, organizations need a dynamic information technology infrastructure to be able to have this increase in data and their analysis carried out within the framework of cloud computing. In a cloud data center, there are large computing resources that make it possible to run, store, and analyze different applications using virtual machines or containers [5]. However, the billions of data generated in sensors as big data cannot be transferred or processed in the cloud, since transfer bandwidth is limited, and IoT devices are far away from cloud data centers. Transfer of a huge amount of data and requests to the cloud may result in underapplication of network resources, extreme transfer delay, excessive processing costs, and network congestion. Moreover, some applications need to be run very rapidly or in real time [6]. All this has caused cloud computing to encounter challenges. In a smart health application, for instance, which monitors a patient's conditions, belated responses may jeopardize human life. This becomes even more important when a large amount of data generated by IoT devices is considered [7]. Therefore, an appropriate computing scheme is required to manage the diversity, speed, and amount of data generated by IoT devices. Fog computing is recommended to eliminate the limitations of the cloud [8]. The architecture of fog computing provides services similar to those of the cloud near the network edge. Fog computing is performed by a large

* Corresponding author.
E-mail addresses: Navid.khaledian@iausdj.ac.ir (N. Khaledian), k.khamforoosh@iausdj.ac.ir (K. Khamforoosh), s.azizi@uok.ac.ir (S. Azizi), Maihami@iausdj.ac.ir (V. Maihami).

https://doi.org/10.1016/j.suscom.2022.100834
Received 29 July 2022; Received in revised form 12 November 2022; Accepted 21 November 2022
Available online 26 November 2022

number of nodes located in a layer close to the network edge. These nodes include routers, switches, access points, and workstations [9,10]. Fog computing can establish network communication among the devices within a short time [11]. For example, users can access fog nodes to receive services at universities, bus stations, and shopping centers that are time-sensitive. The fog architecture reduces bandwidth and required financial costs with respect to where the data need to be sent to a data center or a cloud for processing [12,13]. The fog layer gets connected to the cloud data centers for complex computaion and storage. In fact, fog computing supplements cloud computing, and their integration forms an inspiring computing environment known as the fog-cloud. The scheduling problem has been investigated widely for cloud environments, both as a unique objective and as a multi-objective optimization problem. Despite the large number of studies recently conducted in the field, the issue is known as one of the hottest areas of research due to its various applications. In practice, IoT applications execute requests or tasks, which appear as dependent or independent. Independent tasks are parallel applications that are considered independently, and are applied in problems such as data mining [14]. However, this type of scheduling is not applicable to artificial intelligence and big data applications that involve complex analyses, and cannot be regarded as pairs of independent applications. Therefore, workflow scheduling is used in such cases [15].

A workflow considers applications as a series of tasks that are interdependent, and need to be run in a certain order and with specific priorities, where a weighted, directed graph known as the Directed Acyclic Graph (DAG) is used for that purpose [16]. Given the above discussion, the problem of workflow scheduling is an important one in distributed systems, which are composed of large numbers of heterogeneous computing resources[17]. The purpose of a workflow scheduling problem is to execute a number of parallel tasks on several fog-cloud resources in order to minimize parameters such as makespan, energy consumption, monetary cost, and delay and use the resources efficiently. These problems have been considered in a large number of works [18–23].

Energy consumption is a significant issue in fog-cloud computing, which raises financial costs and environmental problems. Energy depends on a large number of parameters such as throughput, processing time, and system power on and off times[24]. The computing nodes in a fog use batteries or restricted resources for energy consumption, reduction of which in the fog-cloud architecture has thus been turned into an important issue [25]. Energy consumption can be improved by better workflow scheduling, *i.e.* allocation of tasks to appropriate virtual machine (VMs) and specification of the order in which they are executed in each resource[26]. Therefore, reduction of energy consumption along with a proper (short) execution time provides an optimal choice for use in a resource. A new technology has recently been introduced as Dynamic Voltage and Frequency Scaling (DVFS) to reduce energy consumption [27]. The major function of DVFS is to alter the voltage and frequency of the available VMs dynamically. In fact, it seeks to select voltage and frequency for the VMs that provide less energy consumption while an application is being run. Resources will consume less energy if run over less voltage and frequency, a technique recently appealing to and widely used by researchers in workflow scheduling[22,24,25]. There aren't many DVFS-based articles for cloud-fog computing systems in the literature. And it usually applies to a distinct fog or cloud environment. As a result, there is a need for greater study on energy-efficient scheduling in cloud-fog computing settings.

Besides makespan and energy consumption, monetary cost is an important issue in application scheduling and running[19,26,27], since a short makespan and low energy consumption interfere with high financial costs, and can reduce the efficiency and utility of an algorithm. The cost of application of a processer to execute tasks is a significant parameter, which needs to be reduced based on task execution time and the monetary cost for each unit of task performance. Therefore, it is very important to calculate financial cost and reduce it while ensuring low energy consumption and earlier workflow termination. Recent efforts have not taken into account simultaneously decreasing these three conflicting factors in the combined cloud-fog environment. There are $n \times p^n$ different possibilities for scheduling a DAG with n nodes on a system with p VMs. This value is valid for totally independent tasks. If graph interdependency is considered, the number of possibilities for scheduling will be approximately n! This is regarded as an NP-hard problem as it cannot be solved within polynomial time. Different algorithms and methods have been presented in recent years for optimization of these problems [28,29]. This becomes more challenging when several contradictory objectives need to be met at the same time. The overall objective is to obtain an appropriate tradeoff between the application completion time and the energy consumed and monetary cost incurred during the workflow. Our investigations demonstrate that the best optimal points can be specified through solution of the problem using meta-heuristic algorithms as compared to other methods. Meta-heuristic solution methods make it possible to present near-optimal solutions within reasonable times [30]. Meta-heuristics exhibit many advantages particularly in the area of dependent tasks, which facilitates the processing of a workflow. The Krill herd method, one of the meta-heuristic algorithms, is popular because to its success in resolving multi-objective issues. Due to shrimp behavior, it supports both global and local optimality well, making it one of the newest and fastest meta-heuristic solution approaches. This program uses a constructive approach to produce better outcomes because our challenge has multiple criteria[31]. Using a novel way of searching in depth and two methods of searching at the level of the DAG, this article produces intelligent primary population and intelligent solutions, in contrast to other algorithms. Better outcomes and faster convergence are the effects of this. After that, we use the EFT technique to more effectively and economically assign tasks to available resources. Finally, the proposed method uses the DVFS technique to consume less energy. The suggested method is effective, as shown by several tests based on various CCR (communication-to-computation cost ratio), the volume of tasks and virtual machines, and comparisons with strong algorithms. The innovations of the paper include the following.

1. Three different operators are used in the proposed method for discovery of new solutions. Moreover, the krill herd algorithm uses three operators: movement toward food, movement toward the krill population, and stochastic movement. Large Code Distance and Short Code Distance strategy, which schedule the tasks available at each level of the DAG, are implemented using movement toward food and movement toward the krill population, respectively. Deep search (Deep Code Distance strategy) schedules tasks that are available at different levels but are independent of each other. This operator replaces stochastic search in the improved krill herd algorithm.

2. The dependency between tasks in a workflow is represented by a DAG. An acceptable solution is one that considers these dependencies and imposes less makespan, cost, and energy on the system. We have generated the initial solutions in a smart fashion in order to allow the algorithm to converge faster. Moreover, efficient searches in depth and at the surface are used to map the task sequence to a krill and provide the new krill generation in a smart fashion.

3. The objective function is defined based on the three criteria of energy, makespan, and monetary cost in the fog-cloud environment and is normalized so that all the three parameters can be minimized at the same time. Furthermore, DVFS is used to reduce energy use.

The remainder of the paper is organized as follows. In Section 2, a review of the literature is provided. Section 3 presents the proposed algorithm, referred to as IKH-EFT, a combination of Improved Krill Herd and Earliest Finish Time. Section 4 involves the results and an evaluation of IKH-EFT. A conclusion is made in Section 5. Also, Table 1 shows the abbreviations in this article.

Table 1
Abbreviations.

EFT	Earliest Finish Time
DVFS	Dynamic Voltage and Frequency Scaling
CCR	Communication-to-Computation cost Ratio
IOT	Internet of Things
DAG	Directed Acyclic Graph
VM	Virtual Machine
DCD	Deep Code Distance strategy
LCD	Large Code Distance strategy
SCD	Short Code Distance strategy
IWO-CA	Invasive Weed Optimization and Culture Algorithm
HEFT	Heterogenous Earliest Finish Time
IKH	Improved Krill Herd algorithm

2. Related works

Extensive research has been conducted in recent years on the issue introduced in the previous section, with each study optimizing a different parameter. An overview of the literature demonstrates that a single constraint or a maximum of two paradigms have been addressed in workflow scheduling rather than a larger number of constraints. Given the tradeoff nature of these problems, a single parameter fails to provide favorable results in some cases. It is also very important to select these parameters based on their degrees of significance for efficient task scheduling and proper solution of the algorithms. Moreover, some papers have considered tasks independently, which makes scheduling easy and free of the complexity involved in a workflow. To solve the problem of workflow scheduling, authors have used heuristic, meta-heuristic [32], machine learning[33], and hybrid algorithms [30]. Among these, exact algorithms are capable of finding optimal solutions accurately, but are inadequately efficient in the case of NP-hard problems, where it is impossible to find the solutions within a reasonable time. A large number of heuristic algorithms have been proposed in many papers that seek to solve the problems within reasonable times.

Chai [34] introduced an improved swarm intelligence algorithm in task queues, which modified the core scheduler for each task, and enhanced the configuration of the task scheduling strategy. The algorithm minimized mean makespan through optimal allocation of tasks at each node. Sharma and Garg [35] allocated the best computing resources to the received requests using neural networks through identification and reception of the environmental conditions such as the numbers of servers and virtual machines. This algorithm performed better than ones such as genetic and linear regression in terms of criteria such as energy consumption, makespan, and resource productivity. Cost was not considered in this paper as an objective, where the runtime of the algorithm was longer than those of heuristic and meta-heuristic ones. A hybrid algorithm named GSAGA is presented by the authors in [36] as a solution to the Task Scheduling Problem (TSP) in cloud computing. The Genetic Algorithm (GA), despite having a strong ability to search the issue space, performs poorly in terms of stability and local search. By combining the generic search capabilities of the GA with the Gravitational Search Algorithm, authors demonstrated that it is possible to develop a stable algorithm (GSA). Using a multi-objective genetic algorithm (MOGA), process scheduling issues are optimized in[37]. This study presented an initialization scheduling sequence technique, where each task's data size is taken into account when initializing its VM instance, to improve the search efficiency. The makespan and the energy consumption, which are two optimization goals in this work, can be properly traded off based on the original scheduling sequence. Potential trade-offs between the makespan and the price of virtual machine usage were discussed by the authors in [20]. They suggested a HEFT-ACO technique to minimize them, based on the ant colony algorithm (ACO) and the HEFT, but they did not take energy into account. Chhabra et al. [38] used the search features of two powerful meta-heuristic algorithms, namely Cuckoo Search (CS) and Particle Swarm Optimization (PSO), for the problem of task scheduling to overcome slow convergence and lack of diversity in the initial population. The algorithm was presented to reduce energy consumption and makespan in the cloud computing environment.

In [39], Jamil et al. proposed a novel multi-objective algorithm for delay-sensitive task scheduling in the fog computing environment, which optimized energy consumption, and minimized delay and network use. In [40], Zhang et al. presented an approach to multi-objective task scheduling in cloud computing, which was referred to as Enhanced Heterogeneous Earliest Finish Time based on rules (EHEFT-R) and used for optimization of performance, service quality, and energy consumption. Zhou et al. [41] presented a task scheduling algorithm, referred to as MGGS, by combining the modified genetic algorithm and the greedy strategy, where the greedy strategy was applied to the successors generated by the modified genetic algorithm and caused faster convergence. The algorithm obtained one optimal solution in smaller numbers of iterations. It considered load balancing, service quality maximization, and minimization of makespan, average response time, and monetary cost.

In [42] they proposed a method, named HDECO used in workflow scheduling in the cloud environment. They introduced an intelligent threshold detector to reduce the number of switch-on and off physical machines in the cloud datacenter. By classifying input tasks in workflows, they manage to optimize energy and cost respectively. The authors of [43] suggest combining heuristic and metaheuristic strategies for work scheduling. HEFT is a heuristic method, and fireworks algorithm (FWA) is a metaheuristic algorithm. The approach of bi-objective optimization is proposed to reduce the cost and makespan factors. The authors of [44] presented an improved algorithm, referred to as PGWO,[1] based on critical paths for workflow scheduling, which improved makespan, cost, and resource use. The authors of [45] proposed a heuristic method observing task priority and using the method of task duplication for the problem of dependent task scheduling in a heterogeneous cloud computing system, and assessed the criteria of speed, efficiency, and makespan. They obtained better results than those of algorithms such as earliest finish time, while they disregarded energy.

Moreover, a large number of meta-heuristic algorithms have been presented that are capable of finding proper (near-optimal) solutions within short times. Many of them have also been improved in recently-conducted works, and their drawbacks concerning the solution of this problem have been eliminated. In [21], the authors presented a hybrid meta-heuristic algorithm, referred to as HDPSO,[2] by combining the DPSO algorithm and the Hill Climbing technique for workflow scheduling and provision of global optima to avoid entrapment in local ones, which was focused on makespan, and obtained proper results. In [46], Hosseinioun et al. reduced energy consumption using the method of dynamic frequency and voltage scaling and low-voltage resource activity, and combined Invasive Weed Optimization (IWO) and Culture Algorithm (CA) to provide a valid sequence of tasks in order to reduce makespan. Abualigah and Diabat [47] proposed the antlion optimization algorithm (ALO) for the cloud computing environment by utilizing differential evolution as a local search technique to improve operation capacity and avoid entrapment in local optima in order to solve the task scheduling problem to optimize makespan and make maximal use of the resources. In [48], the authors presented an advanced antlion optimization algorithm that was combined with the PSO algorithm to optimize a workflow schedule particular to the cloud. Moreover, they adopted a security approach, referred to as Data Encryption Standard (DES), to encrypt cloud information while scheduling. They focused on cost, load, and makespan. In [49] they unveiled the DMFO-DE, an opposition-based hybrid discrete optimization method. To do this, the Differential Evolution (DE) algorithm is paired with a discrete and Opposition-Based Learning (OBL) version of the Moth-Flame Optimization (MFO)

[1] Pareto-based grey wolf optimizer
[2] Hybrid Discrete Particle Swarm Optimization

approach to increase convergence speed and avoid local optima issues. Natesan and Chokkalingam [50] presented a novel hybrid algorithm, referred to as the whale genetic optimization algorithm, by combining the Whale Optimization Algorithm (WOA) and the genetic algorithm to optimize cost and reduce makespan. The results of comparison to the standard genetic and whale algorithms demonstrated the better efficiency of the proposed algorithm. Nebojsa et. al. performed simulations for a workflow scheduling problem with two objectives cost and makespan and proposed an improved firefly algorithm in [51] that is tailored for overcoming workflow scheduling difficulties in a cloud-edge environment. The unique hybrid MBA approach is developed by Manikandan et al. in [52] for resolving multi-objective task scheduling issues in cloud computing environments. The multi-objective behavior reduces the makespan in the hybrid WOA based MBA algorithm by maximizing resource utilization. To ensure task distribution on fog and cloud nodes under time limitations, they suggest the multi-objective simulated annealing (MOSA) approach. Authors in [53], To optimize total deadline violation time for jobs and energy consumption, a system model is presented in the article for the job scheduling problem in fog-cloud computing. For efficient solution of the job scheduling problem in the fog-cloud environment, two nature-inspired optimization techniques are proposed besides grey wolf and grasshopper optimization algorithms. An architecture is considered in [54] that is composed of multiple fog computing providers, and a Hidden Markov Model (HMM) is presented to predict whether each fog computing provider is available with respect to the tasks offloaded from the fogs to the cloud computing, workflows with missed deadlines, number of requests incoming to each fog, and similar factors. The unsupervised Baum-Welch algorithm is used to train the model, along with the Viterbi algorithm, used to compute the probability that each fog is available. A fog computing provider is then selected using the probability that the fog provider is available, on which IoT workflows are scheduled.

Since the process of running a learning algorithm is time-consuming, it is not appropriate for scheduling delay-sensitive tasks and IoT device requests. Heuristic algorithms are usually difficult to design, and fail to obtain accurate solutions for optimal points in NP-hard problems. Meta-heuristic algorithms can partially resolve the above problems, if they are not trapped in local optima, and their problem of premature convergence is eliminated. In fact, a meta-heuristic algorithm is a type of approximate optimization algorithm, which may get out of local optima to obtain other optimum points due to the large space available for search in the solution of problems. Various algorithms of this type have been developed in recent decades. The comparison made between the available methods demonstrates that meta-heuristics are one of the most efficient methods of solving the scheduling problem. In this paper, the proposed meta-heuristic algorithm (improved krill herd involving movement in three directions for krill) is used for workflow scheduling, since DAG search enables movement in depth and at the surface in practice, providing population diversity that is missing in other papers. Moreover, the initial population is generated in a smart fashion to eliminate the drawbacks, the EFT technique is implemented for task allocation, and the three contradictory objectives are optimized in the fog-cloud environment, whereas population has been generated stochastically in other papers, and no investigation has been made of optimization of the three objectives of the workflows in the above environment.

3. Proposed method

Different architectures are used for workflow scheduling in fog-cloud computing. Given the significance of the problem, this section first addresses the architecture and the model used for representation of the relevant schedule. Then, the IKH-EFT algorithm is presented in detail.

3.1. Three-tier architecture

Fig. 1 shows the architecture of the IKH-EFT method, involving three layers: the terminal layer, the fog layer, and the cloud layer [55,56]. The terminal layer includes smart devices (IoT devices such as home appliances, sensors, smartphones, and smart automobiles). These devices submit requests, and send data to the higher-level layers to be processed by the applications [6]. The applications are considered as series of interdependent tasks that need to be executed in a particular order and with a specific priority. The fog layer is the middle one, located between the IoT devices and cloud data centers, close to the terminal layer devices. There is also a particular node, referred to as fog broker, in this layer that functions as the central manager and task scheduler. This node is in charge of user request collection and resource management over fog-cloud nodes. It also provides the best scheduling for workflows. The highest layer in this architecture is the cloud layer, which contains powerful servers for large amounts of complex processing and storage, and exhibits high efficiency. This layer provides computing services for intensive computing and workload and efficient, reliable data storage facilities. Given the long distances between the layer servers and data resources, there may be delays in processing, so delay-insensitive tasks with high complexity and no deadlines had better be performed in this layer [7].

This paper aims to allocate tasks to machines in order to minimize energy, makespan, and monetary cost in the fog-cloud architecture. The parameters used in the paper are listed in Table 2. The fog-cloud system used in this paper involves heterogeneous computing nodes. The fog nodes exhibit less computing capability, and are closer to data resources (where data are generated). The cloud nodes have greater computing capability, but are farther away from data generation resources. In this problem, the set of VM nodes, $N = \{n1, n2, \ldots np\}$, involves all nodes that are divided between the fog and cloud sets from the beginning, as represented by Eq. (1) [25]. Each node in N may be one in the cloud Nc or in the fog Nf.

$$N = \mathrm{Nc} \cup \mathrm{Nf}, Q(\mathrm{Nf}) < Q(\mathrm{Nc}) \forall N. \tag{1}$$

The lower computing capability of fog nodes $Q(\mathrm{Nf})$ than that of cloud nodes $Q(\mathrm{Nc})$ results from the physical limitation of fog devices [25]. As shown in Fig. 2, the relationships between tasks are represented using a directed acyclic graph $G = (V, E)$, which is weighted as well as directed [57]. Fig. 2 shows a simple example of a DAG, representing ten tasks and the workflows between them. The graph indicates that t_1 is an input task and t_{10} is an output task. Moreover, transfer cost (the time required for transfer) will be zero if t_1 and t_2 are executed on the same host, and it will be 15 if they are executed on separate hosts.

Each node is a member of $v = \{t1, t2, \ldots tn\}$, and indicates a task, and the node weight, shown as w, is a nonnegative number that specifies execution time or computation cost. w is an $m \times n$ computation cost matrix, where m is the number of tasks, and n is the number of system VMs. $w_{i,j}$ is the computation cost of task t_i on VM p_j, measured in milliseconds. The graph also involves a set of edges, namely E, indicating the prerequisite relationships between tasks. Each edge $e_{i,j} \in E$ is of nonnegative weight $c_{i,j}$, which denotes the amount of data transferred from task t_i to task t_j, *i.e.* the cost of communication between the two tasks. This cost is there when the tasks are executed on different VMs, and communication cost will be zero for tasks executed on the same VM. Prioritization is the most important step in a scheduling algorithm, as efficiency depends mainly on this step. In this paper, prioritization is calculated based on the cost, energy consumption, and makespan parameters, and is used in evaluations as a measure for algorithm efficiency.

3.1.1. Makespan

Makespan is an important factor in workflow scheduling, and is minimized in applications. The shorter this time, the higher the effi-

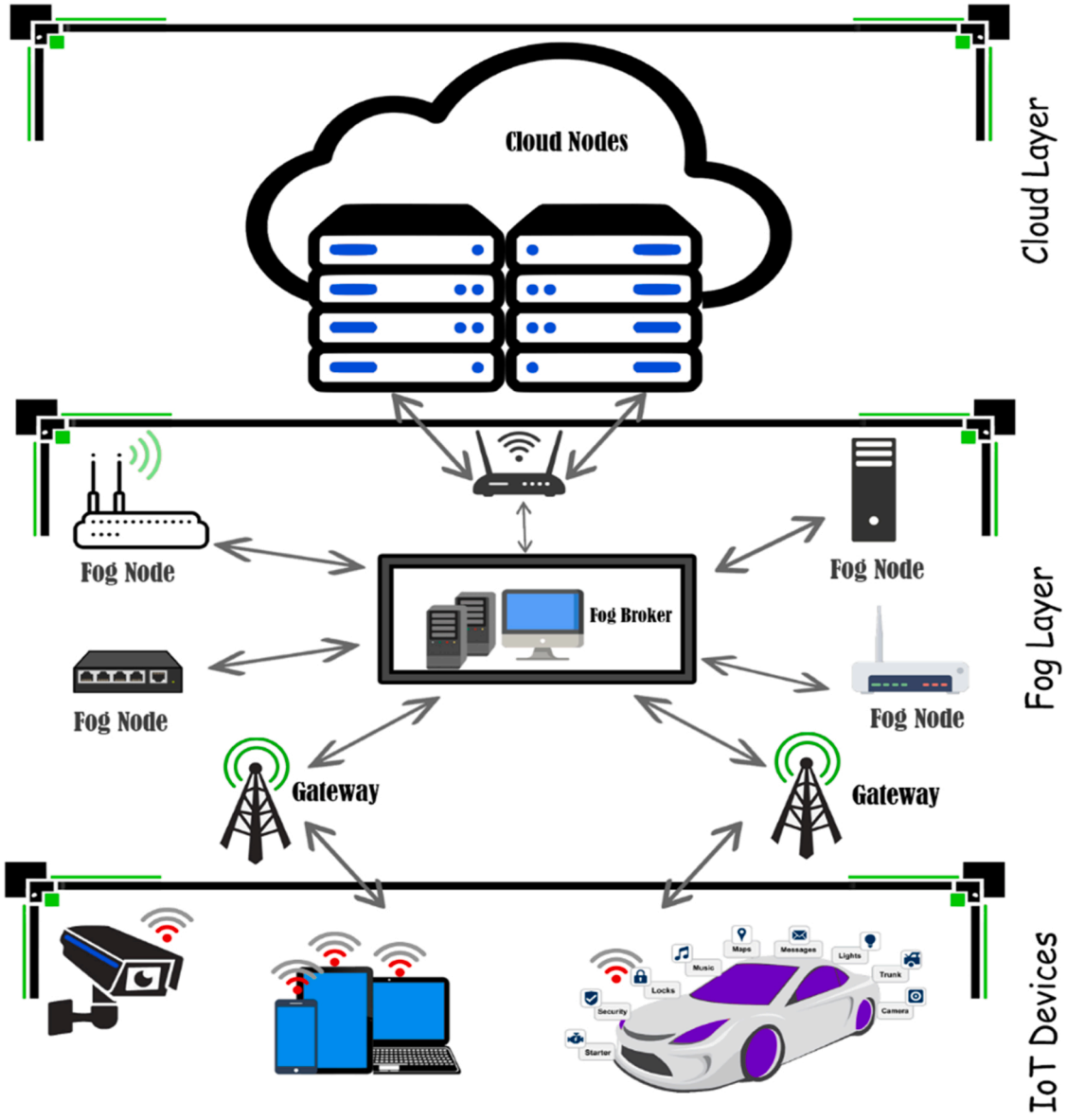


Fig. 1. Fog – Cloud architecture.

ciency of the system. To obtain the value of this measure, the earliest start time (EST) of task t_i on VM p_j needs to be calculated. Eq. (2) is used for that purpose for the input task with no predecessors and Eq. (3) for other tasks [58].

$$EST\left(t_{enter}, p_j\right) = 0 \tag{2}$$

$$EST\left(t_i, p_j\right) = \max_{t_j \in pred(t_i)} \left(AFT\left(t_j\right) + C\left(t_j, t_i\right)\right) \tag{3}$$

where t_{enter} is the input task with no predecessors, p_j is the j^{th} VM, t_i is the i^{th} task in DAG, $pred(t_i)$ is the set of essential predecessors of task t_i, and $C\left(t_j, t_i\right)$ is the closeness of communication between tasks t_i and t_j. Moreover, the actual start time of task t_i on VM p_j is calculated by Eq. (4) [58].

$$AST\left(t_i, p_j\right) = \max\left(EST\left(t_i, p_j\right), Avail\left(p_{kj}\right)\right) \tag{4}$$

where $Avail\left(p_j\right)$ is the time when VM p_j is still and ready to calculate a task from the queue. Eq. (5) [58] shows the earliest finish time of task t_i on VM p_j.

$$EFT\left(t_i, p_j\right) = t_{ij} + AST\left(t_i, p_{kj}\right) \tag{5}$$

The actual finish time of task t_i is a function that finds VM p_j for this task from among the VMs to reduce execution time, and is obtained using Eq. (6) [58]:

$$AFT\left(t_i, p_j\right) = \min_{1 \le l \le P} EFT\left(t_i, p_j\right) \tag{6}$$

3.1.2. Monetary cost

The execution cost of a specific task is calculated through multiplication of the total time spent for its execution at the selected node by the price of the node. Table 3 involves an example of monetary cost for each unit of task execution in cents. For instance, 2 cents is calculated in VM 1 for each millisecond (unit of execution), 3 cents in VM 2, and 4 cents in VM 3. Then, the monetary cost of the relevant task is obtained given its EFT value and cost of execution on the VM. Communication cost is not considered directly in the calculations, since it is considered indirectly in the calculation of EFT. Eq. (7) shows the monetary costs incurred in the fog and in the cloud. The total cost of the workflow is the sum of the execution costs of all the tasks [20].

Table 2
Parameters used in the paper.

t_i	i^{th} task in DAG
t_{ij}	Computation Cost of task t_i on VM p_j
p_j	j^{th} VM in the system
Q_N	Node computing capability
E	Set of edges
V	Set of tasks
N	Set of VMs
F_j	Frequency level for node j
$Dll(t_i, p_j)$	Monetary execution cost of task t_i on VM p_j
t_{entry}	Input task with no predecessors
t_{exit}	Output task with no successors
$Succ(t_i)$	Set of essential successors of task t_i
$Pred(t_i)$	Set of essential predecessors of task t_i
$W_{i,j}$	Computation cost of task t_i on VM p_j
$C(t_i, t_j)$	Communication between tasks t_i and t_j
$EST(t_i, p_j)$	Earliest start time of task t_i on VM p_j
$EFT(t_i, p_j)$	Earliest finish time of task t_i on VM p_j
$AST(t_i, p_j)$	Actual start time of task t_i on VM p_j
$AFT(t_k)$	Actual finish time of task t_i on best VM p_j
LFT(ti)	Last finish time of task t_i
$avail\{k\}$	Earliest time VM p_j is available for execution

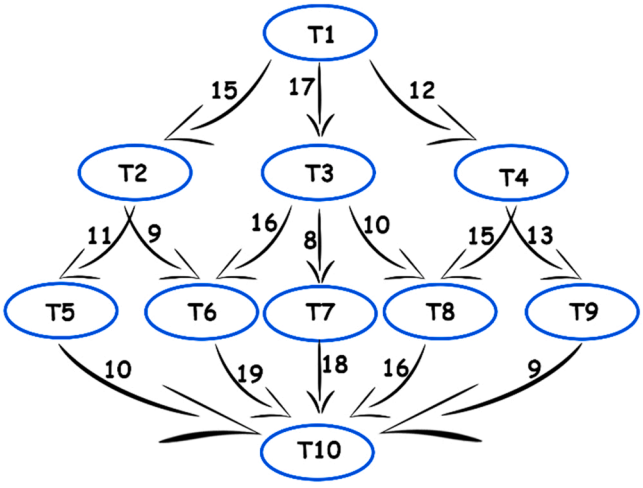


Fig. 2. Example DAG with ten nodes and communication cost.

Table 3
Computation and monetary costs.

Task	VM3		VM2		VM1	
	Cent	Computation	Cent	Computation	Cent	Computation
t_1	32	8	39	13	20	10
t_2	28	7	24	8	18	9
t_3	48	12	39	13	22	11
t_4	60	15	33	11	16	8
t_5	44	11	27	9	24	12
t_6	36	9	21	7	30	15
t_7	24	6	15	5	16	8
t_8	64	16	45	15	14	7
t_9	40	10	24	14	32	16
t_{10}	52	13	18	6	18	9

$$Cost = \left\{ \begin{matrix} \sum (EFT(t_i, p_k) * Dll(t_i, p_k)) * c11 \le l \le P, c1forNc \\ + \\ \sum (EFT(t_i, p_k) * Dll(t_i, p_k)) * c21 \le l \le P, c2forNf \end{matrix} \right\} \quad (7)$$

$Dll(t_i, p_k)$ is the monetary cost of execution of task t_i on VM p_k, c_1 is the constant cost of transition to the cloud, and c_2 is the constant cost of transition to the fog.

3.1.3. DVFS

The DVFS approach often lowers the CPU's operational frequency and voltage to lessen the energy consumption of processors during task execution. All computing systems can use the DVFS approach. However, decreasing the CPU frequency slows things down, thus the workflow deadline can unintentionally be missed. Therefore, the primary responsibility of DVFS-based scheduling frameworks is to establish the minimal operational frequency for the VMs that execute the workflow and adhere to the workflow deadline[49]. Different allocations of tasks to VMs create time gaps between the tasks once completed. If VM $p_k$works at a specific frequency, it can reduce speed at the gaps since it is idle there to consume less energy. Eq. (8) shows how a gap is calculated [59].

$$Gap = \max \sum_{j \in p} \cdot \sum_{i \in t} ESTi - EFTi - 1 \quad (8)$$

Reduced CPU processing speed (frequency) is not problematic as long as it lies within the limits of the gap and does not interfere with the subsequent task. We seek to reduce energy consumption using voltage, frequency, and the DVFS technique. The main function of DVFS is to alter the voltage and frequency of the available VMs dynamically. In fact, DVFS seeks to select appropriate voltage and frequency for the VMs as an application is run [60].

In this study, we use energy consumption, which includes static and dynamic use. Since dynamic energy consumption is more time-consuming and expensive, static energy consumption is typically disregarded in our scheduling strategy [49].

$$EN = EN_{static} + EN_{dynamic} \quad (9)$$

When the system is turned on EN_{static} represents the amount of power used while $EN_{dynamic}$ compute as follows [49]:

$$EN_{dynamic} = Z.V_{js}^2.f \quad (10)$$

Where Z is the dynamic power consumption constant, which depends on the device capacity. Additionally, $V_{jS}^2$is represents the voltage of the s^{th} level, the j^{th} VM and f represents the j^{th} VM frequency. The following equation can be used to calculate the energy consumption of the VMs [49]:

$$EN_{busy} = \sum_{i=1}^{n} Z \times V_{js}^2 \times f_{js} \times ET_{ij} = \sum_{j=1}^{n} EN_{dynamic} \times ET_{ij} \quad (11)$$

Where V_{js} denotes that the i^{th} is task is used on the j^{th} VM, with the s^{th} voltage level and ET_{ij} represents the execution time of the i^{th} task on the j^{th} VM. Additionally, f_{js} is the of processor's j^{th} VM frequency at the s^{th} voltage level. Additionally, to save the maximum energy, the voltage of VMs must be adjusted to the lowest setting while they are idle. The following definition applies to all available CPUs' idle energy consumption [49]:

$$EN_{idle} = \sum_{j=1}^{p} Z \times v_{jmin}^2 \times f_{jmin} \times IT_j = \sum_{j=1}^{p} EN_{ide} \times IT_{ij} \quad (12)$$

Where IT_j is regarded as the j^{th} VM's idle time $v_{jmin}$and f_{jmin} are its the minimum voltage and frequency of the j^{th} VM. These equations allow us to calculate the total energy consumption needed for DAG scheduling in a cloud-fog computing environment as follows:

$$EN_{total} = EN_{busy} + EN_{idle} \quad (13)$$

The method used to calculate the energy consumption function for the full workflow application is:

$$EN_{consumtion}DAG = \sum_{n} EN_{total}(n) \tag{14}$$

where n can be a fog or a cloud VM.

Table 4 shows an example of the voltage and frequency values for the computing resources. Resources consume less energy when run over lower voltage and frequency values [46].

3.1.4. Objective function

The purpose of our optimization problem is to minimize makespan, energy consumption, and monetary cost at the same time. In this paper, the problem of multi-objective optimization turns into a problem of single-objective optimization through application of the impact factor to each of the parameters with a weighted sum approach. Therefore, the objective function of the optimization problem can be expressed as Eq. (15) and Constraints (16) and (17).

$$\text{Min}(z) = ą \times \text{makespan} + \beta \times \text{cost} + \Omega \times \text{energy} \tag{15}$$

$$0 \leq ą, \beta, \Omega \leq 1 \tag{16}$$

$$ą + \beta + \Omega > 0, ą = 0.5, \beta = 0.25, \Omega = 0.25 \tag{17}$$

Constraint (16) states that each of the impact factors must be a number between zero and one, and Constraint (17) states that the sum of all the three coefficients is larger than zero. The relevant values are set given the importance of makespan in the problem and the tradeoff between energy and monetary cost. Finally, the values concerning the objectives need to be normalized, since they are not of the same type. For that purpose, we use the method of lower bound for each variable.

3.2. Proposed algorithm

The IKH-EFT algorithm involves two different phases: smart generation of the initial population considering the dependencies and adoption of a VM equipped with the DVFS technology in accordance with the objective function. A DAG, as in Fig. 2, is generated stochastically with values for execution cost and communication cost. VMs are also generated with specific frequencies and voltages, as in Table 4, and a monetary cost, as in Table 3. The given DAG is traversed top down, and the tasks are sorted independently at each level, so that they can be executed in parallel. Population is stochastically initialized to make it possible to solve the problem meta-heuristically using the improved krill herd algorithm for generation of different layouts. The EFT technique is used in accordance with the objective function for selection of a VM and its allocation to the task. The objective function aims to reduce three parameters at the same time: energy, makespan, and monetary cost.

i. Generation of the initial population

A solution is composed of n nodes, and must be valid given the precedence constraints. In the IKH-EFT method, the population involves 300 valid solutions that are generated in a smart fashion. Generation of the initial population is the first step in the krill herd algorithm, which considerably affects the algorithm performance. In a DAG-based workflow, the order of tasks in the population solutions must meet the precedence constraint. To spread tasks through one solution, the method of height distribution [46] and in-depth search is used.

First, minimum height $L(t_i)$ is calculated for each task using Eq. (18). Then, maximum height $H(t_i)$ in DAG is obtained by Eq. (19).

$$L(t_i) = \left\{ \begin{array}{c} 0 pred(t_i) = \varnothing \\ 1 + \max\limits_{t_j \in pred(t_i)} \left(L\left(t_j\right)\right) pred(t_i) \neq \varnothing \end{array} \right\} \tag{18}$$

$$H(t_i) = \left\{ \begin{array}{c} L(t_{exit}) succ(t_i) = \varnothing \\ \min\limits_{t_j \in succ(t_i)} \left(H\left(t_j\right)\right) - 1 succ(t_i) \neq \varnothing \end{array} \right\} \tag{19}$$

The results of height distribution for the DAG in Fig. 3 are shown in Table 5. The initial population is generated in our algorithm based on the height values, where the height of the task selected in a solution must not be greater than those of the predecessors or less than those of the successors.Table 6.

i. Krill movement

In the krill herd algorithm, three behaviors cause the creatures to move toward optimal regions: attraction, foraging, and stochastic movements. Every krill can have three types of movement in the sea, as follows. Induced movement N_i. In this movement, each krill is influenced by the neighboring krill, and makes movements according to information obtained from the neighbors. Foraging activity F_i. To find food, each krill makes movements toward food sources. Stochastic diffusion D_i. Each krill can exhibit stochastic behavior in its movements. The krill objective function has been stated as in Eq. (20).

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{20}$$

For a successor to differ from its predecessor in order to realize global optimization, the Strategy for Large Code Distance (LCD) is adopted to spread successors over a discrete solution space, where an action node is stochastically selected from a solution. Then, the nearest two tasks that ensure the precedence constraints are found among the predecessors and successors of the node. Therefore, the movement can be made within the interval between specific tasks. Moreover, an array of labels is constructed to record the states of task movement. Solution modification is terminated when a specific code distance is obtained, or all tasks are selected. Alternatively, short code distances are used for global

Table 4
Relative frequency and voltage [46].

Level	VM 3		VM 2		VM 1	
	Relative frequency	Voltage	Relative frequency	Voltage	Relative frequency	Voltage
0	1	85/1	1	6/1	1	90/1
1	85/.	70/1	85/.	4/1	9/.	75/1
2	65/.	45/1	60/.	15/1	8/.	60/1
3	45/.	20/1	50/.	9/.	7/.	45/1
4	30/.	95/.	40/.	8/.	6/.	30/1
5	–	–	–	–	5/.	15/1
6	–	–	–	–	4/.	00/1

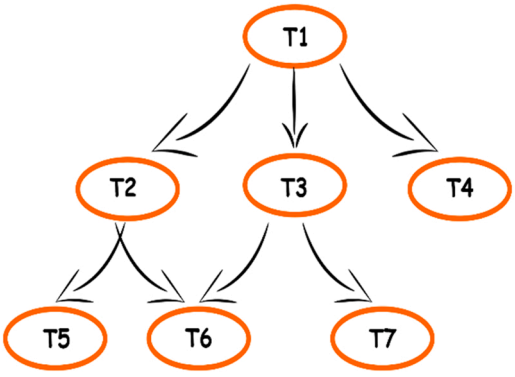


Fig. 3. Example of in-depth search.

Table 5
Height distribution.

Tasks	1	2	3	4	5	6	7	8	9	10
$L(t_i)$	0	1	1	1	1	1	2	2	2	3
$H(t_i)$	0	1	1	1	1	1	2	2	2	3

optimization [46]. In *SFSCD,*[3] tasks at the same height are of equal priority. Therefore, this strategy is capable of moving tasks at the same level with no violation of the precedence constraints. Besides these two, we present a novel method in this paper for provision of diversity in smart generation of the initial population. The proposed method involves a type of in-depth search, which we refer to as the Strategy for Deep Code Distance *(DCD)*.

3.2.1. Strategy for deep code distance

This strategy aims to provide diversity in the initial population according to in-depth search, and seeks to obtain sensitive paths from important nodes. The Dynamic Frequency Scaling (DFS) search method functions in a smart fashion here. Maintaining dependency, the strategy moves in depth between tasks beginning from the predecessor and continuing until the last successor that has not visited its predecessor. Processing is resumed from the beginning after completion. In equal conditions, the subsequent nodes are selected stochastically. As can be observed in Fig. 3, the search for t_3 continues in the Strategy for Large Code Distance until the node depth is completed. In the Strategy for Deep Code Distance, however, we can proceed in the depth of a node when all its predecessors have been visited. For instance, node t_6 is visited in the Strategy for Large Code Distance before node t_2, as in-depth search for t_3 is allowed, whereas t_6 follows t_2 in the Strategy for Deep Code Distance, since the successors of t_3 are visited when all its predecessors, including t_2, have been visited.

i. Task-to-VM mapping

Tasks must be allocated to the available VMs after the step of population initialization in the system. In IKH-EFT, the krill herd algorithm is used with the objective function in Eq. (15), where the optimal value is found through simultaneous reduction of three parameters: energy, makespan, and monetary cost, and the obtained optimal tasks are mapped to VMs using the EFT technique. The krill herd algorithm is used for global optima, and EFT is used for local optima. It should be noted that we do not apply crossover and mutation, which are there in the krill herd algorithm, given the different layouts generated *via* DCD, LCD, and SCD and the dependency between tasks in terms of precedence.

4. Evaluation

In this section, the performance of IKH-EFT is compared to that of IWO-CA [46], HEFT [58], and IHEFT [61] for investigation of its superiority. The reason for using these algorithms is to compare their

Table 7
Dataset properties.

Number of Tasks	[10, 25, 50, 100, 200, 400]
Number of VM	[3,5,7,10]
Number of Edges	1.5 * Number of Tasks
	5
Minimum Computation Cost	15
	Maximum Computation Cost
	10
Minimum Communication Cost	30
	Maximum Communication Cost
	1 Cent
Minimum Monetary Cost	10 Cent
	Maximum Monetary Cost
	[0.2, 1, 5, 10]
CCR	Fat
	Shape

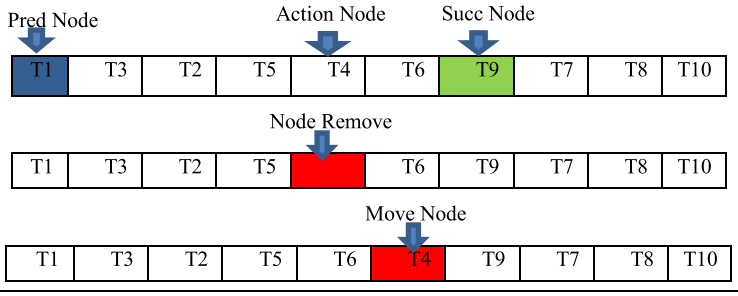
Table 6
Example for the Strategies for Long and Short Code Distance.

SCD

Node

10	9	8	7	6	5	4	3	2	1	Tasks
3	2	2	2	1	1	1	1	1	0	$L(t_i)$
3	2	2	2	1	1	1	1	1	0	$H(t_i)$

LCD



[3] Strategy for Short Code Distance

strength in the field of makespan reduction and energy consumption with DVFS technique. All the above methods are implemented over the same dataset, in equal conditions, with a Core i7 VM and 8 gigabytes of memory, and in MATLAB 2020 on Windows 10.

4.1. Dataset

The stochastic graph generator is used to generate a wide range of graphs. In this method, a DAG is stochastically generated based on the numbers of task and VMs and given the order in which the tasks are executed, providing different combinations. The properties of the generated graph can be observed in Table 7. Since stochastic graphs and real-world datasets are different in shape, and there is no balanced relationship between computation and communication costs, performance is evaluated given the value of CCR, a parameter that stands for the communication-to-computation cost ratio. The CCR value concerning each diagram is obtained through the ratio of total edge weight to total node weight. Graph type is computation-intensive if graph CCR is less than one, communication-intensive if it is greater than one, and moderate if it is equal to one. In this paper, graphs are generated and evaluated for different values of CCR. As can be observed in Table 7, the total number of tasks in the graph assumes the 10, 25, 50, 100, 200, and 400 values. The number of edges is 1.5 times greater than that of tasks, and there are one or two edges from the node at level s to that at level $s+1$. Moreover, links are generated between vertices with a uniform distribution. Three, five, seven, and ten virtual machines are considered in this paper.

4.2. Parameters used for evaluation and analysis of efficiency

Accurate evaluation requires an application of appropriate parameters. IKH-EFT is compared to the other methods in terms of different measures, including makespan, implementation cost, energy consumption, efficiency, speedup and SLR. For investigation of the performance of IKH-EFT in different conditions, stochastic datasets with different values are generated, each considered as input to the compared algorithms in one of three different scenarios, suggested in this paper for further investigation. Different experiments are conducted with the same numbers of tasks and virtual machines and different CCR values. For further experimentation, another parameter is changed each time, while the other two are kept constant. The number of iterations is 100, and the number of implementations for each scenario is 10. The mean values of the measures are finally obtained.

4.2.1. Makespan

Makespan is regarded as one of the most frequent parameters used for comparison in the scheduling problem, and is defined as the total length of the schedule. As stated in Section 3.2, makespan is obtained from Eq. (21).

$$makespan = Min\{\max(AFT(t_i))\} = \min\{AFT(t_{exit})\} \tag{21}$$

The values of each of the above parameters were detailed in Section 3.2. In Fig. 4, the value of makespan is obtained for different numbers of tasks. Since this is the most important measure, based on which the others are obtained, the method aims to calculate the minimum value of this one first. As can be observed, IKH-EFT exhibits better results (lower values) than the other methods in all the cases. This is because a heuristic algorithm cannot search a large space efficiently. The IWO-CA is the second-best method after ours. An important point to be noted is that IWO-CA generates diverse but low-quality solutions in some cases, since it generates initial populations only at the graph surface, and is rather focused on energy consumption. In the subsequent experiments, shown in Figs. 5 and 6, makespan is obtained based on the number of virtual machines and the value of CCR, respectively. The great power of IKH-EFT is demonstrated by its superiority in different scenarios, which is maintained as the value of CCR and the number of virtual machines increase.

The issue of energy consumption involves the entire energy consumed by computing resources that execute all workflow schedule tasks. The next experiment concerns the energy consumed for different algorithms, which are conducted in three different scenarios, as for makespan. The DVFS technique is used to reduce energy consumption. Figs. 7, 8, and 9 show the energy consumed by IKH-EFT and the compared methods. It is observed in Fig. 7 with an accurate calculation of gaps that it exhibits considerable superiority as the number of tasks increases. IKH-EFT efficiently utilizes the gaps between tasks, which reduces energy consumption with respect to that in the other methods. As can be observed for less than 100 tasks, the proposed method and IWO-CA exhibit superiority over the other methods in terms of energy consumption. All the methods perform similarly even for 25 tasks, since very small gaps take shape between tasks. Similarly, IKH-EFT is considerably superior in the reduction of energy consumption for five and seven virtual machines, as observed in Fig. 7. In Fig. 9, where energy consumption is calculated with respect to CCR, IKH-EFT performs almost the same as IWO-CA and slightly differently from HEFT and IHEFT for CCR = 0.2 and CCR = 1. Given the small number of tasks for these values, very small gaps take shape, which can hardly be used in IKH-EFT, hence the close values.

The next experiment aims to obtain and compare the monetary cost of implementation of an entire workflow in different scenarios. User satisfaction can be achieved with low values for the monetary cost of running the application along with the other measures. IKH-EFT is compared to IWO-CA, HEFT, and IHEFT in Figs. 10, 11, and 12. Fig. 10 shows the superiority of IHEFT with 10 tasks, where IKH-EFT is the second best. The HEFT algorithm exhibits better performance than the other methods with 25 tasks. With larger numbers of tasks, however, the

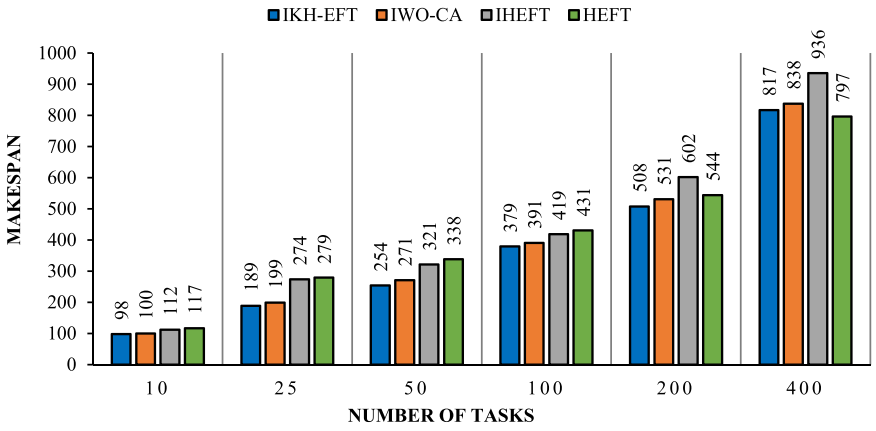


Fig. 4. Aerage makespan for the different task set.

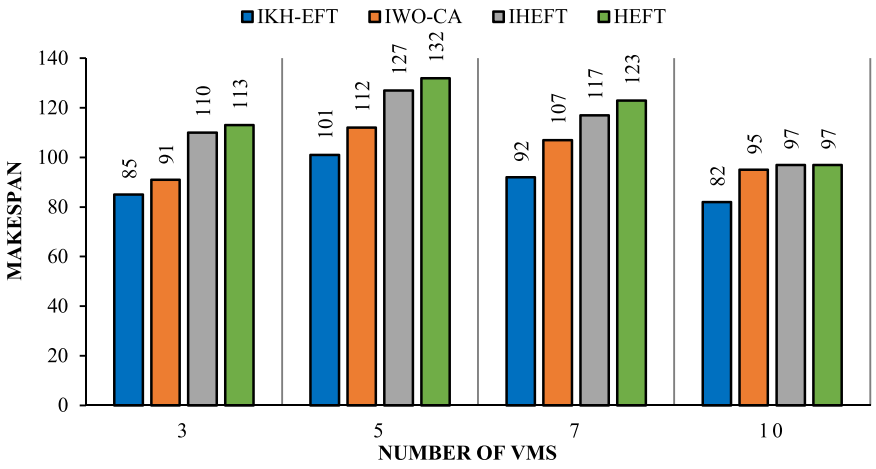


Fig. 5. Average makespan for the different VM set.

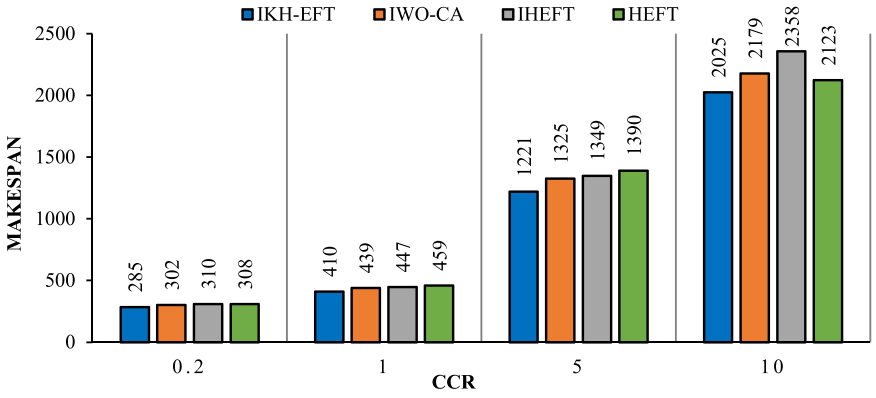


Fig. 6. Average makespan for the different value of CCR.

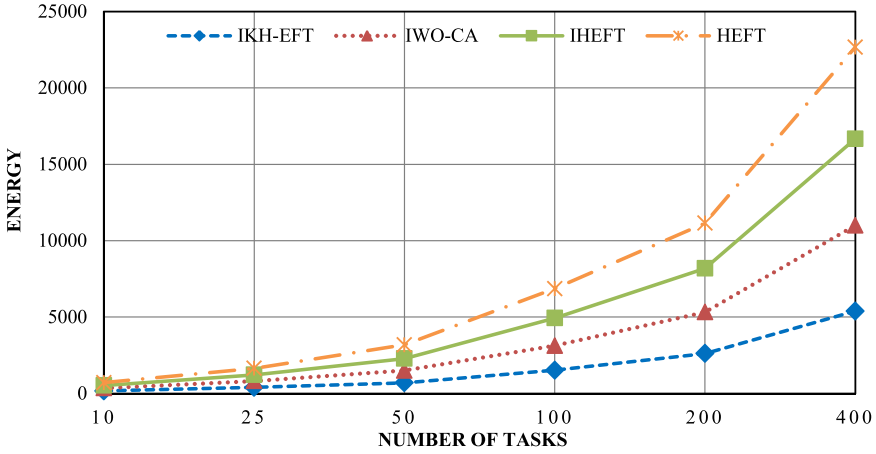


Fig. 7. Evaluation of energy consumption for the different task set.

proposed method outperforms the others. When DVFS is used to reduce energy consumption, greater use is made of virtual machines, since smaller gaps take shape between tasks, which establishes a tradeoff between energy consumption and monetary value. The same impact factor is considered in this paper for the two measures to provide a balance between them. Along the same lines, IKH-EFT and IHEFT exhibit considerable superiority over the other methods with five and seven virtual machines, as observed in Fig. 11. Fig. 12 shows that the proposed method performs better than all the others for CCR = 0.2, and there is less difference in monetary value between them for the other values of CCR.

4.2.2. Schedule length ratio (SLR)

Schedule length ratio (SLR) is a major measure used for evaluation of the performance of the scheduling algorithm over the graph. This measure is obtained based on the makespan and critical path parameters. The critical path is the longest path from the input node to the output node of the given DAG in the fastest VM. Since different large

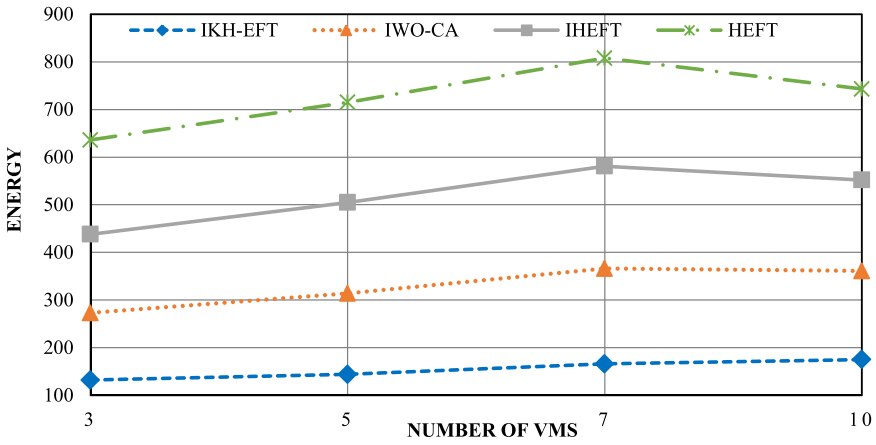


Fig. 8. Evaluation of energy consumption for the different VM set.

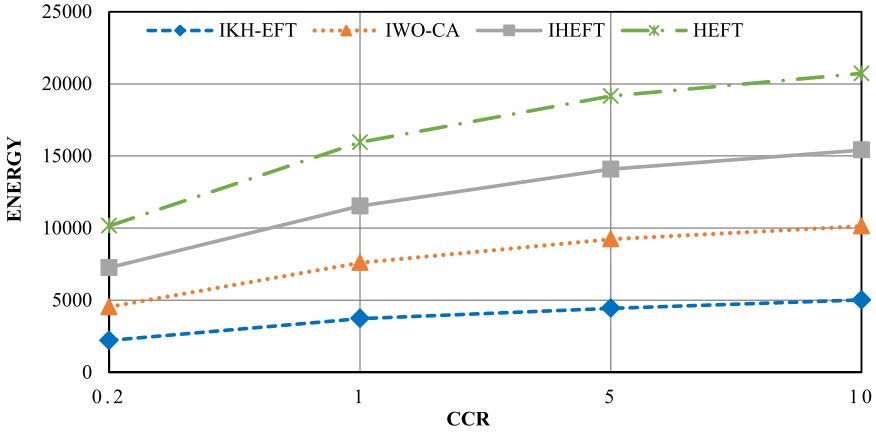


Fig. 9. Evaluation of energy consumption for the different value of CCR.

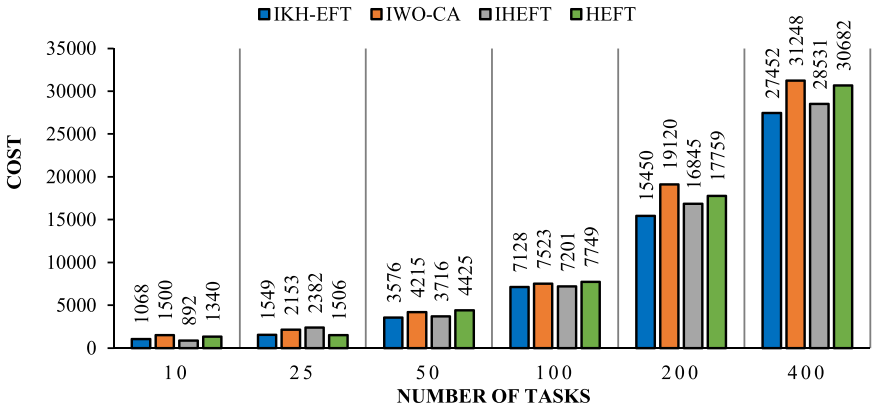


Fig. 10. Evaluation of cost consumption for the different task set.

graphs are used, the value of makespan needs to be normalized to a lower bound. If the scheduler runs the nodes on the critical path using the fastest virtual machine, the value of the makespan parameter cannot be lower than the critical path length. Therefore, the length ratio of any schedule is more than one. Since the denominator is the lower bound, SLR cannot be less than one. A scheduling algorithm with the lowest SLR is the best. If the value is close to one, the scheduler can be said to function highly properly. In other words, a lower SLR is favorable. The SLR of an algorithm over a graph is defined by the following equation.

$$SLR = \frac{Makespan}{\sum_{n_i \in CP\min} Min_{pj \in Q}\{W_{i,j}\}} \tag{22}$$

In this equation, the denominator is the sum of the minimum computation costs of the tasks on the critical path. For an unscheduled DAG, the critical path will be based on the minimum computation costs, represented as CP_{MIN}, if the computation cost of each node n_i is set to the minimum value. The average SLRs from the task graphs are used in several experiments in the paper. The mean SLRs in Fig. 13 are obtained based on the numbers of different tasks on t virtual machines and the

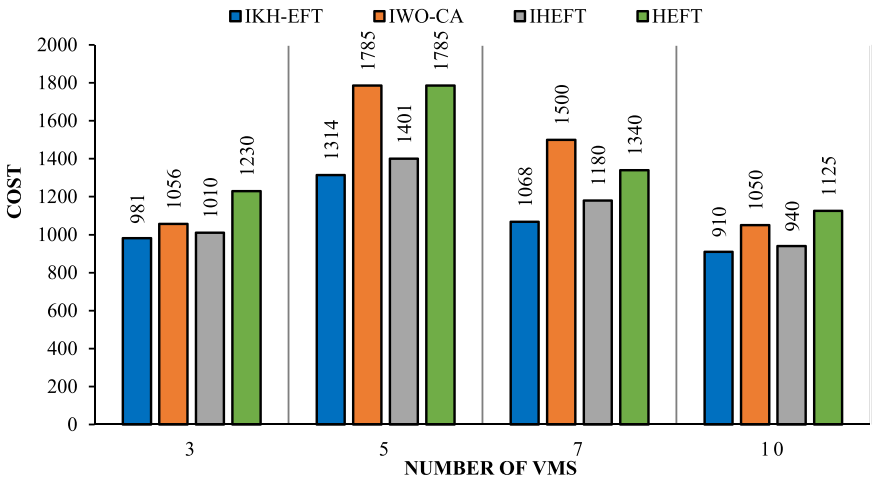


Fig. 11. Evaluation of cost consumption for the different VM set.

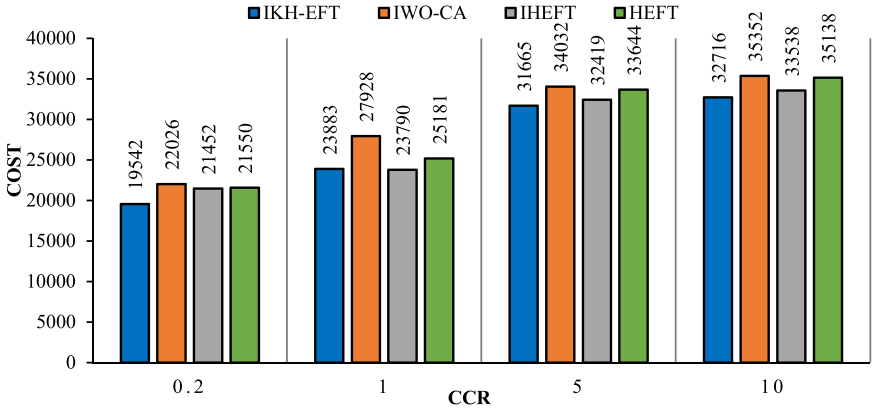


Fig. 12. Evaluation of cost consumption for the different CCR value.

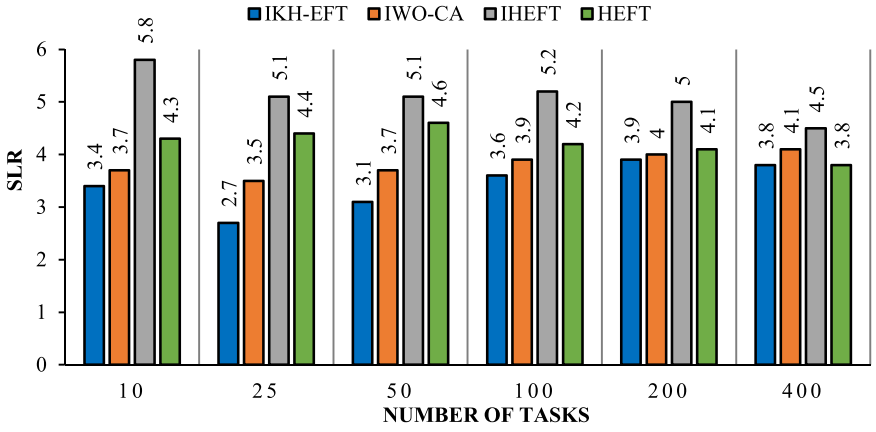


Fig. 13. Average schedule lengths ratio for the different task set.

constant value of CCR. With fewer than 100 tasks, IKH-EFT exhibits considerable superiority over the other methods. With 200 tasks, there is very little difference. With 400, IKH-EFT and HEFT assume the same SLR. Different numbers of virtual machines along with constant values for CCR and the number of tasks are investigated in Fig. 14. As can be observed, IKH-EFT exhibits lower SLRs than the other methods in all the scenarios, and the difference is still there as the number of virtual machines increases. The IHEFT algorithm assumes the highest SLRs in this experiment. A similar experiment is conducted based on CCR, as shown in Fig. 15, where IKH-EFT, IWO-CA, and HEFT obtain close results. With an examination of the experiments as different scenarios, it can be stated that IKH-EFT is superior to the IWO-CA, HEFT, and IHEFT algorithms in most cases. This results from its lower makespan and better performance in the surface and in-depth search for the critical path.Fig. 16.

4.2.3. *Speedup*

The increase in speed, known as speedup, for a specific graph is obtained through division of sequential runtime to parallel runtime

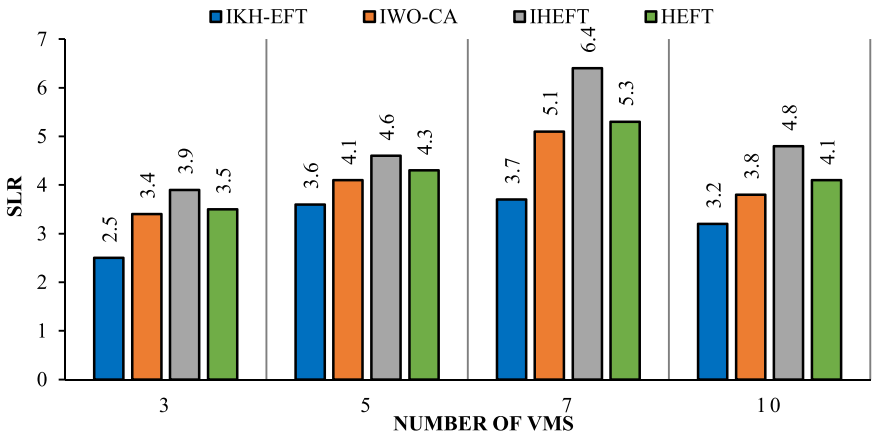


Fig. 14. Average schedule lengths ratio for the different VM set.

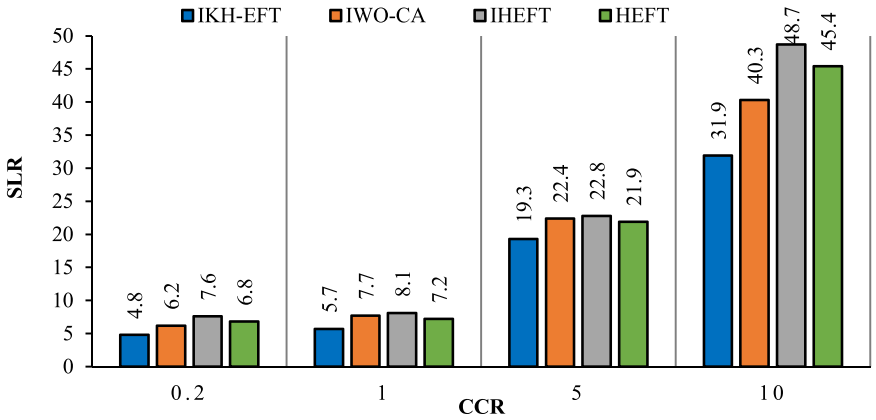


Fig. 15. Average schedule lengths ratio for the different CCR value.

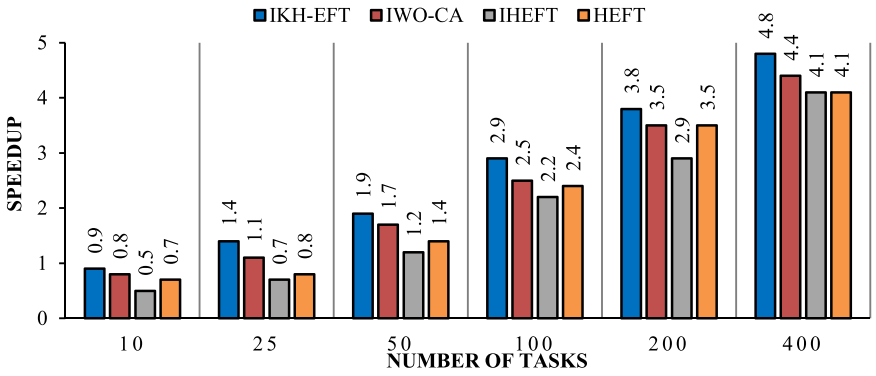


Fig. 16. Speedup comparison of the four algorithms for the different task set.

(makespan). Sequential runtime is the time required for allocation of all tasks to a single VM, which minimizes the sum of computation costs. The probable runtime in the above conditions is calculated in Eq. (23).

$$Speedup = \frac{\min_{pj \in Q}\{\sum_{ni \in N}\{W_{i,j}\}\}}{makespan} \tag{23}$$

If the sum of computation costs is maximized, speed will increase further, but the scheduling algorithms will eventually end with the same ranking. Figs. 17, 18, and 19 show speedup for the compared algorithms in different scenarios. According to Fig. 17, the speedup in IKH-EFT gets farther apart from the values in the other methods as the number of tasks increases. The same finding is observed with changes in the number of virtual machines and the CCR value. IKH-EFT outperforms the other methods in most cases.

4.2.4. Efficiency

The effective scheduling approach improves linearly in proportion to the number of virtual machines used. In other words, runtime will be divided by two if the scheduler uses one virtual machine instead of two. The efficiency parameter involves the ratio of speedup to the number of virtual machines. Eq. (24) is used to obtain the value of this parameter.

$$Efficiency = \frac{Speedup}{number\ of\ used\ VM} \times 100\% \tag{24}$$

As can be observed in Figs. 19, 20, and 21, IKH-EFT improves

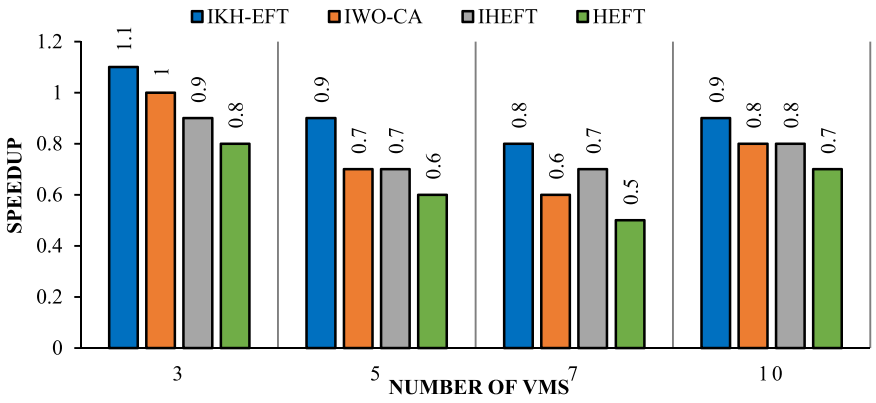


Fig. 17. Speedup comparison of the four algorithms for the different VM set.

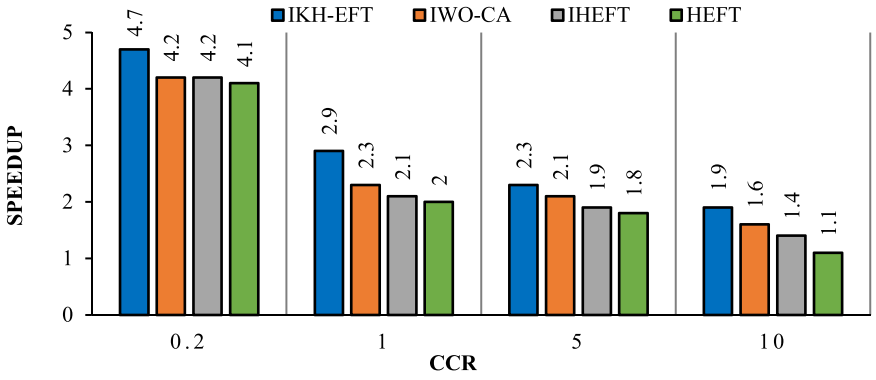


Fig. 18. Speedup comparison of the four algorithms for the different CCR value.

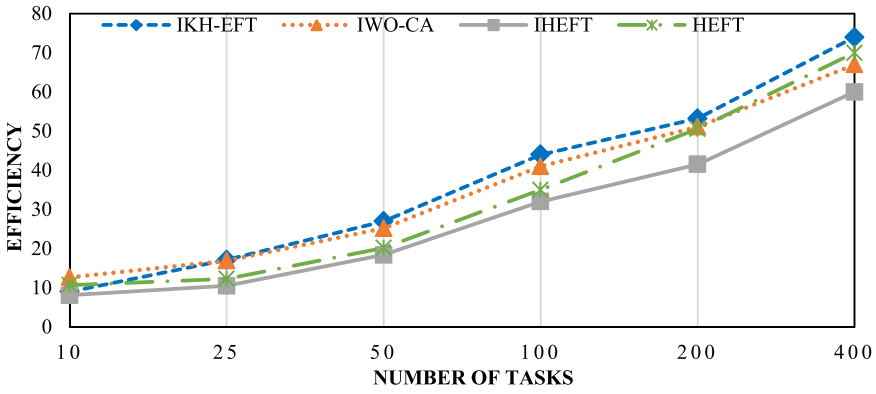


Fig. 19. Comparison of *efficiency* metric for the different task set.

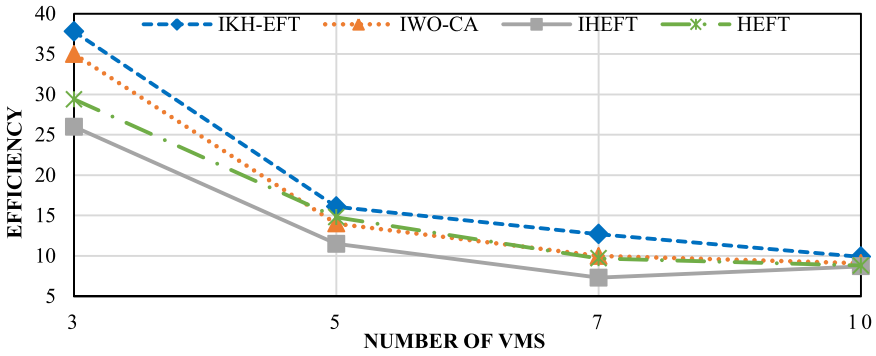


Fig. 20. Comparison of *efficiency* metric for the different VM set.

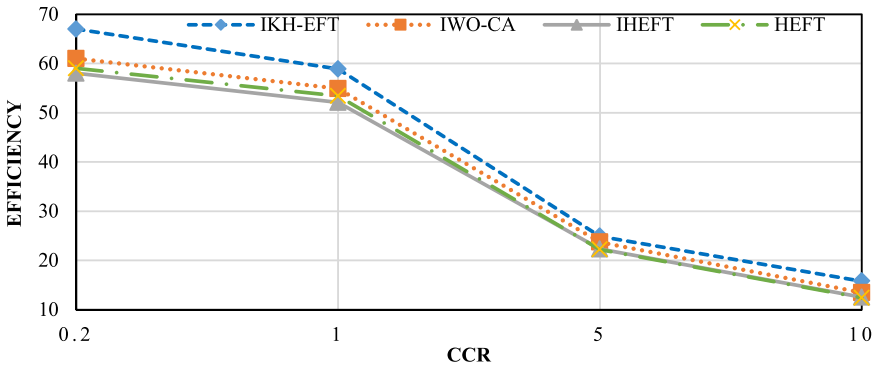


Fig. 21. Comparison of *efficiency* metric for the different CCR value.

efficiency with respect to that in the other algorithms over all the datasets, which demonstrates that it convergences faster than them, as a result of the great power of the krill herd algorithm in terms of global and local optima. The superiority is clear in Fig. 19 as the number of tasks increases and in Fig. 21 as CCR equals 0.2.

5. Conclusion

In the implementation of a workflow to provide services to users and organizations, it is of great significance to reduce makespan in order to perform tasks faster, to reduce monetary cost in order to execute tasks or run applications economically, and to reduce energy consumption in order to observe environmental issues including low node battery life. Given the interdependency of tasks and NP-hard nature of the scheduling problem, simultaneous reduction of the parameters examined in this paper in the fog-cloud environment has not been investigated in previous research. A workflow scheduling algorithm, referred to as IKH-EFT, is proposed here using the improved krill herd algorithm and the EFT technique in the fog-cloud environment for scheduling different DAGs. The algorithm first generates different workflows, and then provides lists of interdependent task execution observing precedence and priority in a smart rather than stochastic fashion based on the krill herd algorithm and movement at the surface and in the depth of the graph. It then isolates the superior lists using the EFT technique (*i.e.* objective function) that involve the lowest makespan and monetary cost. Next, the algorithm reduces energy consumption where gaps exist using the DVFS technique. Use of the krill herd algorithm has obtained proper results in NP-hard problems with large numbers of measures, and near-optimal solutions can be obtained in both global and local scales through combination with the EFT technique. Based on the conducted experiments, the three parameters of makespan, monetary cost, and energy consumption are simultaneously reduced in the fog-cloud environment in IKH-EFT using complex graphs with different values for CCR, number of tasks, and number of virtual machines. The challenge of using task deadline as a measure can provide research topic prospects. Moreover, solutions adopted in machine learning methods and their combination with meta-heuristic methods in multicloud environment can be considered in future studies.

Funding

The authors declare that they have no funding.

CRediT authorship contribution statement

Conceptualization: Navid Khaledian, Keyhan Khamforoosh, Sadoon Azizi. **Data curation:** Navid Khaledian, Keyhan Khamforoosh, Vafa Maihami. **Formal analysis:** Navid Khaledian, Vafa Maihami **Methodology:** Keyhan Khamforoosh, Sadoon Azizi. **Project administration:** Keyhan Khamforoosh. **Resources:** Keyhan Khamforoosh, Sadoon Azizi **Validation: Navid Khaledian**, Keyhan Khamforoosh, Sadoon Azizi **Visualization: Navid Khaledian**, Vafa Maihami, Sadoon Azizi **Writing – original draft:** Navid Khaledian, Sadoon Azizi. **Writing – review & editing:** Vafa Maihami, Sadoon Azizi.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

Data will be made available on request.

Acknowledgements

Not applicable.

Declaration

none.

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Authors' contribution

The specific contributions made by each author is as follows: NK: Conceptualization, methodology, implementation, writing-original draft, KK: writing—review & Editing, formal analysis. SA & VM: validation, review & editing, all authors read and approved the final manuscript.

References

[1] Z.N. Aghdam, A.M. Rahmani, M. Hosseinzadeh, The role of the internet of things in healthcare: future trends and challenges, Comput. Methods Prog. Biomed. 199 (2021), 105903.
[2] Wa Kassab, K.A. Darabkh, A–Z survey of internet of things: architectures, protocols, applications, recent advances, future directions and recommendations, J. Netw. Comput. Appl. 163 (2020), 102663.
[3] Luo, Q., et al., Resource scheduling in edge computing: A survey. IEEE Communications Surveys & Tutorials, 2021.
[4] A. Islam, et al., A survey on task offloading in multi-access edge computing, J. Syst. Archit. 118 (2021), 102225.
[5] A. Belgacem, et al., Efficient dynamic resource allocation method for cloud computing environment. Clust. Comput. 23 (4) (2020) 2871–2889.

[6] M. Laroui, et al., Edge and fog computing for IoT: A survey on current research activities & future directions, Comput. Commun. 180 (2021) 210–231.
[7] J.C. Guevara, N.L. da Fonseca, Task scheduling in cloud-fog computing systems, Peer to Peer Netw. Appl. 14 (2) (2021) 962–977.
[8] Bonomi, F., et al. Fog computing and its role in the internet of things. in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012.
[9] N. Kaur, A. Kumar, R. Kumar, A systematic review on task scheduling in Fog computing: taxonomy, tools, challenges, and future directions, Concurr. Comput.: Pract. Exp. 33 (21) (2021), e6432.
[10] S. Azizi, et al., Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: a semi-greedy approach, J. Netw. Comput. Appl. (2022), 103333.
[11] P. Hosseinioun, et al., aTask scheduling approaches in fog computing: a survey, Trans. Emerg. Telecommun. Technol. (2020), e3792.
[12] W.Z. Khan, et al., Edge computing: a survey, Future Gener. Comput. Syst. 97 (2019) 219–235.
[13] S. Javanmardi, et al., FUPE: A security driven task scheduling approach for SDN-based IoT–Fog networks, J. Inf. Secur. Appl. 60 (2021), 102853.
[14] D. Tychalas, H. Karatza, A scheduling algorithm for a fog computing system with bag-of-tasks jobs: simulation and performance evaluation, Simul. Model. Pract. Theory 98 (2020), 101982.
[15] M. Hussain, et al., Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers, Future Gener. Comput. Syst. (2022).
[16] L. Versluis, A. Iosup, A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies, Future Gener. Comput. Syst. 123 (2021) 156–177.
[17] Z. Ahmad, et al., Scientific workflows management and scheduling in cloud computing: taxonomy, prospects, and challenges, IEEE Access 9 (2021) 53491–53508.
[18] M. Abdel-Basset, et al., Energy-aware metaheuristic algorithm for industrial-Internet-of-Things task scheduling problems in fog computing applications, IEEE Inter. Things J. 8 (16) (2020) 12638–12649.
[19] M. Tanha, M. Hosseini Shirvani, A.M. Rahmani, A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments, Neural Comput. Appl. 33 (24) (2021) 16951–16984.
[20] A. Belgacem, K. Beghdad-Bey, Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost, Clust. Comput. 25 (1) (2022) 579–595.
[21] M.H. Shirvani, A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems, Eng. Appl. Artif. Intell. 90 (2020), 103501.
[22] N. Arora, R.K. Banyal, A particle grey wolf hybrid algorithm for workflow scheduling in cloud computing, Wirel. Pers. Commun. 122 (4) (2022) 3313–3345.
[23] A. Taghinezhad-Niar, S. Pashazadeh, J. Taheri, QoS-aware online scheduling of multiple workflows under task execution time uncertainty in clouds, Clust. Comput. (2022) 1–18.
[24] R. Ghafari, F.H. Kabutarkhani, N. Mansouri, Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review, Clust. Comput. (2022) 1–59.
[25] S. Ijaz, et al., Energy-makespan optimization of workflow scheduling in fog–cloud computing, Computing 103 (9) (2021) 2033–2059.
[26] A. Choudhary, et al., Energy-aware scientific workflow scheduling in cloud environment. Clust. Comput. (2022) 1–30.
[27] A. Taghinezhad-Niar, S. Pashazadeh, J. Taheri, Energy-efficient workflow scheduling with budget-deadline constraints for cloud, Computing 104 (3) (2022) 601–625.
[28] Hoseiny, F., et al. PGA: A Priority-aware Genetic Algorithm for Task Scheduling in Heterogeneous Fog-Cloud Computing. in IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2021. IEEE.
[29] P. Hosseinioun, et al., aTask scheduling approaches in fog computing: a survey, Trans. Emerg. Telecommun. Technol. 33 (3) (2022), e3792.
[30] E.H. Houssein, et al., Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends, Swarm Evolut. Comput. 62 (2021), 100841.
[31] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
[32] C. Guerrero, I. Lera, C. Juiz, Genetic-based optimization in fog computing: current trends and research opportunities, Swarm Evolut. Comput. (2022), 101094.
[33] F. Cheng, et al., Cost-aware job scheduling for cloud instances using deep reinforcement learning, Clust. Comput. 25 (1) (2022) 619–631.
[34] X. Chai, Task scheduling based on swarm intelligence algorithms in high performance computing environment, J. Ambient Intell. Humaniz. Comput. (2020) 1–9.
[35] M. Sharma, R. Garg, An artificial neural network based approach for energy efficient task scheduling in cloud data centers, Sustain. Comput. Inform. Syst. 26 (2020), 100373.
[36] P. Pirozmand, et al., GSAGA: A hybrid algorithm for task scheduling in cloud infrastructure, J. Supercomput. (2022) 1–27.
[37] X. Xia, et al., Multi-objective workflow scheduling based on genetic algorithm in cloud environment, Inf. Sci. (2022).
[38] A. Chhabra, G. Singh, K.S. Kahlon, QoS-Aware energy-efficient task scheduling on HPC cloud infrastructures using swarm-intelligence meta-heuristics, CMC Comput. Mater. Contin. 64 (2) (2020) 813–834.
[39] B. Jamil, et al., A job scheduling algorithm for delay and performance optimization in fog computing, Concurr. Comput. Pract. Exp. 32 (7) (2020), e5581.
[40] H. Zhang, Y. Wu, Z. Sun, EHEFT-R: multi-objective task scheduling scheme in cloud computing, Complex Intell. Syst. (2021) 1–8.
[41] Z. Zhou, et al., An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, Neural Comput. Appl. 32 (6) (2020) 1531–1541.
[42] A.G. Delavar, R. Akraminejad, S. Mozafari, HDECO: a method for decreasing energy and cost by using virtual machine migration by considering hybrid parameters, Comput. Commun. 195 (2022) 49–60.
[43] A.M. Yadav, K.N. Tripathi, S.C. Sharma, A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm, J. Supercomput. 78 (3) (2022) 4236–4260.
[44] S. Doostali, S.M. Babamir, M. Eini, CP-PGWO: multi-objective workflow scheduling for cloud computing using critical path, Clust. Comput. 24 (4) (2021) 3607–3627.
[45] NoorianTalouki, R., M.H. Shirvani, H. Motameni, A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms. Journal of King Saud University-Computer and Information Sciences, 2021.
[46] P. Hosseinioun, et al., A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm, J. Parallel Distrib. Comput. 143 (2020) 88–96.
[47] L. Abualigah, A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, Clust. Comput. 24 (1) (2021) 205–223.
[48] J. Kakkottakath Valappil Thekkepuryil, D.P. Suseelan, P.M. Keerikkattil, An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment, Clust. Comput. 24 (3) (2021) 2367–2384.
[49] O.H. Ahmed, et al., Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing, Appl. Soft Comput. 112 (2021), 107744.
[50] G. Natesan, A. Chokkalingam, Multi-objective task scheduling using hybrid whale genetic optimization algorithm in heterogeneous computing environment, Wirel. Pers. Commun. 110 (4) (2020) 1887–1913.
[51] N. Bacanin, et al., Modified firefly algorithm for workflow scheduling in cloud-edge environment, Neural Comput. Appl. 34 (11) (2022) 9043–9068.
[52] N. Manikandan, N. Gobalakrishnan, K. Pradeep, Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment, Comput. Commun. 187 (2022) 35–44.
[53] S. Dabiri, S. Azizi, A. Abdollahpouri, Optimizing deadline violation time and energy consumption of IoT jobs in fog–cloud computing, Neural Comput. Appl. (2022) 1–17.
[54] D. Javaheri, et al., An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing, Sustain. Comput. Inform. Syst. 36 (2022), 100787.
[55] L. Peng, A.R. Dhaini, P.-H. Ho, Toward integrated Cloud–Fog networks for efficient IoT provisioning: key challenges and solutions, Future Gener. Comput. Syst. 88 (2018) 606–613.
[56] Nazari, A., et al., An Intelligent SDN-Based Clustering Approach for Optimizing IoT Power Consumption in Smart Homes. Wireless Communications and Mobile Computing, 2022. 2022.
[57] L.F. Bittencourt, et al., Scheduling in distributed systems: a cloud computing perspective. Comput. Sci. Rev. 30 (2018) 31–54.
[58] H. Topcuoglu, S. Hariri, M.-Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[59] Z. Tang, et al., An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment, J. Grid Comput. 14 (1) (2016) 55–74.
[60] C. Jiang, et al., Energy aware edge computing: a survey, Comput. Commun. 151 (2020) 556–580.
[61] S. AlEbrahim, I. Ahmad, Task scheduling for heterogeneous computing systems, J. Supercomput. 73 (6) (2017) 2313–2338.