The Journal of Supercomputing (2024) 80:22315–22361
https://doi.org/10.1007/s11227-024-06315-2

MOTORS: multi-objective task offloading and resource scheduling algorithm for heterogeneous fog-cloud computing scenario

Prashant Shukla[1] · **Sudhakar Pandey**[1]

Accepted: 17 June 2024 / Published online: 23 June 2024
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract
Along with the rising popularity of pay-as-you-go cloud services, many businesses and communities are deploying their business or scientific workflow applications on cloud-based computing platforms. The primary responsibility of cloud service providers is to reduce the monetary cost and execution time of Infrastructure as a Service (IaaS) cloud services. The majority of current solutions for cost and makespan reduction were developed for conventional cloud platforms and are incompatible with heterogeneous computing systems (HCS) having service-based resource management approaches and pricing models. Fog-cloud infrastructures (FCI) have emerged as desirable target areas for workflow automation across several fields of application. In heterogeneous FCI, the execution of workflows involving tasks having different properties might influence the performance in terms of resource usage. The primary goal of this research is to efficiently offload the computational task and optimally schedule the workflow in such diverse computing environment. In this article, we present a novel strategy for building an environment that includes techniques for offloading and scheduling while balancing competing demands from the user and the resource providers. In order to address the issue of uncertainty, our approach incorporates a fuzzy dominance-based task clustering and offloading technique. To construct a suitable execution sequence of tasks that helps to limit the precedence relationship, by preserving dependency constraints among the tasks, a novel algorithm for tasks segmentation is employed. To simplify the problem of the complexity, a hybrid-heuristics based on Harmony Search Algorithm (HSA) and Genetic Algorithm (GA) for resource scheduling algorithm is used. The multi-objective optimization using three competing objectives is taken into consideration for investigation in heterogeneous FCI. The fitness function derived includes minimization of makespan and cost along with maximization of resource utilization. We performed experimental research using five workflow datasets in order to investigate and verify the efficacy of our proposed technique. We contrasted our proposed strategy with the primary, closely comparable strategies. Extensive testing using

Extended author information available on the last page of the article

scientific workflows confirms the effectiveness of our offloading approach. Our solution provided a substantially better cost-makespan tradeoffs, while achieving significantly less energy consumption and can execute marginally quicker than the existing algorithms.

Keywords Fog-cloud computing · Task offloading · Resource scheduling · MOTORS · FDTCO · HORSA

1 Introduction

The Internet of Things (IoT) is a dynamic, worldwide network architecture that enables linked physical things (such as gadgets, appliances, heavy equipment, and automobiles) to function independently and autonomously. These gadgets have become an integral part of our daily life. Demands are emerging for mobile devices (MDs) to execute complex computational programs. These devices are designed to be capable of sending and receiving data via the Internet, monitoring their surroundings, and taking actions depending on information exchange. They do this by using information and telecommunications technology. In contrast to the local computing environment, these devices often develop data-intensive activities that demand for more high-performance computing capabilities. Compared to desktop counterparts, local computing power is not enough to do complicated operations. MDs have constrained computing and battery life due to their physical size limitations. The bottleneck of MDs is thus the energy constraint. Offloading computation [1, 2] is a viable technique for overcoming the bottleneck mentioned above. It allows resource-restricted MDs to offload tasks to more resourceful servers for their most energy-consuming activities. The MD may offload computations to the cloud server (CS). The CS then performs tasks and delivers the results to the MDs. However, recent research has shown that offloading computations to the CS are not always a viable option since some applications that are sensitive to time needs to be entertained within their delay tolerance [3, 4]. Cloud computing (CC) is inappropriate, especially for network edge devices (EDs). EDs enable low-latency computation offloading, which helps with latency-sensitive operations, thanks to the availability of nearby resources using fog computing (FC).

FC is a contemporary concept of CC at the edge of the network. It has less processing power than CC. With the limitations of the delay tolerance and high transmission rate, we resolve the task offloading issue by reducing energy usage and overall cost [5]. Task offloading involves the transfer of tasks from MDs to either Fog Nodes (FN) or CS. The FN or CS then process the task and deliver the result to the MDs [6]. The significant distance between the CS and the ED results in higher energy consumption and expense, which has a direct impact on real-time applications. In recent years, there has been a partial migration of storage capabilities and computing from the computer science field to the field of functional networks. FC relocates storage and computing resources in close proximity to MDs [7]. FC enables the proximity of MDs to the data generation, computational capability, and data storage, eliminating the need for dependence on a distant central site. The primary factor for this is that real-time data is not affected by latency issues, which can impact the performance of an application [7, 8]. So, in a CC/FC architecture, effective workflow automation is crucial to model the significance of tasks, orchestrate their communication, and ensure their progress. The most difficult problem at hand is to optimize the workflow scheduling in Fog-Cloud Infrastructure (FCI), for which we have identified an acceptable solution. The primary benefits of FCI include: (1) FCI surpasses the issue of restricted computing capability in user equipment when compared to local computing. (2) Offloading duties directly to the CS can result in significant latency due to the considerable distance between the MDs and the CS [9, 10].

1.1 Motivation

The resource-limited end users can transfer the bunch of tasks to the nearest FN in their area in a typical FC system. However, the FN typically searches CS (vertical cooperation) and/or the other FNs because of the computational and storage resource constraints inside a FN (horizontal cooperation). In both offloading situations, more delay and energy use might be imposed. The constrained bandwidth of the uplink leads to a further delay in the completion of the task for the vertical cooperation in which the FN tries to download task details to the remote CS. Likewise, in horizontal cooperation with adjacent FNs, the lack of appropriate processing and storage capabilities in FNs is a concern although transmission delay is lower than in the FCI. The essential yet unresolved problem is thus to pick the offloading point i.e., the nearby FN or remote CS, and to share the task while ensuring the end-user time limit varies with the network traffic [11]. CC contains multiple computing resources that aid in enhancing and facilitating the execution of workflow. Collaboration between the CC and FC environments also enhances the implementation of workflows that may involve tasks with varied characteristics. Certain activities require high levels of computational power, while others necessitate rapid response times. FCI is defined by decentralized resources with limited capacity that are located at the periphery of the network. These resources, including FN and CS, aim to accomplish workflow tasks. Therefore, tasks that require low latency can be divided among the FNs, while tasks that require a lot of computational power can be transferred to the CS. Resource scheduling solutions can optimize the response and execution times of a workflow in an efficient and effective manner. The resources provided by FN are equivalent to those offered by the CS, hence utilizing FCI collaboratively will yield some more advantages [12, 13].

Resource scheduling problem are also well-known for being non-deterministic polynomial NP-hard [14]. There is now considerable research in the field of workflow technology, which tends to automate workflows and provide tools for their administration. These workflows combine many automated actions to accomplish a specified goal while being subject to severe limitations. The efficient execution of the complete workflow is made possible by scheduling these tasks within the bounds of the available IT resources. This helps in achieving the goals stated and in adhering to the limitations. Workflow has been used to automate a variety of sectors throughout time, including astronomy, biology, business, and more recently the IoT industry. The variety of fields covered results into various application kinds being automated by the same workflow. IoT activities aim to be completed with a quick reaction time, whereas scientific workloads need advanced processing capabilities.

1.2 Contributions

In this article, we have proposed an efficient offloading algorithm based on fuzzy logic to address the above-mentioned problem, taking into account input file size, output file size and length of task. In this paper, we also suggest a hybrid optimization-based resource scheduling method for heterogeneous computing systems based on Harmony Search (HS) and Genetic Algorithm (GA). The suggested work takes into account several competing goals, including resource usage, makespan and cost minimization. By preservation of the dependency constraints among the tasks, a novel representation of agents is shown. To construct a suitable execution sequence of activities that helps to limit the precedence relationship, recursive algorithms are used. Fitness function is derived using numerous objectives that are taken into consideration. Extensive simulations on various sets of scientific, Fast Fourier Transformation (FFT), and synthetic workflow applications data sets are used to examine and validate the performances. Our work's contributions can be summarized as follows:

- Firstly, we propose an efficient task offloading algorithm namely fuzzy dominance-based task clustering and offloading (FDTCO) for workflow applications in FCI, in which we first calculate task length, input and output file size. Then, after clustering the task based on these three task attributes, the proposed fuzzy inference system (FIS) is used to decide the optimal offloading layer.
- Secondly a novel hybrid optimization-based resource scheduling algorithm (HORSA) has been presented that makes use of numerous objectives that are thought to be in conflict. To provide a proper execution sequence of tasks while maintaining dependency restrictions, a novel algorithm has been proposed.
- A novel agent representation that always abides by precedence relationships has also been presented. It always makes sure to offer a viable solution to the resource scheduling problem. The agent's validity is ensured while the agent values are modified throughout the evolution. Additionally, a multi-objective fitness function derivation has been provided using several competing goals.
- Extensive simulation results on various data sets for scientific process applications have been used to validate the proposed study. Numerical results demonstrate that the proposed technique has optimal makespan, execution cost, energy consumption and resource utilization than other existing techniques.

The rest of the paper is arranged accordingly. The system model is presented in Sect. 2. The issue formula and solutions are described in Sect. 3. Section 4 has numerical results. The document is concluded in Sect. 5 Table 1 demonstrates the abbreviations used in this article.

Table 1 Abbreviations used

Abbreviation	Description
IaaS	Infrastructure as a service
HCS	Heterogeneous computing systems
FCI	Fog-cloud infrastructures
HSA	Harmony search algorithm
GA	Genetic algorithm
MEC	Mobile edge computing
MD	Mobile devices
ED	Edge devices
TL	Terminal layer
TA	Terminal agent
CC	Cloud computing
CS	Cloud server
CL	Cloud layer
CA	Cloud agent
FC	Fog computing
FN	Fog node
FL	Fog layer
FA	Fog agent
FDTCO	Fuzzy dominance-based task clustering and offloading
HORSA	Hybrid optimization-based resource scheduling algorithm
FFT	Fast Fourier transformation
FIS	Fuzzy inference system
MPSO	Modified particle swarm optimization
VM	Virtual machine
CSP	Cloud service provider
DAG	Directed acyclic graph
RUV	Resources utilization vector
RUF	Resource utilization factor
RUM	Resource utilization magnitude
CI	Computation intensity
CP	Computing power
MIPS	Million instructions per second
MC	Memory capacity
QoS	Quality of service
ORS	Optimal resource schedule
MOTORS	Multi-objective task offloading and resource scheduling
TSA	Topological sorting algorithm
HM	Harmony memory
MI	Maximum number of iterations
NI	Number of iterations
PS	Population size
CR	Crossover rate

Table 1 (continued)

Abbreviation	Description
MR	Mutation rate
HMS	Harmony memory size
HMCR	Harmony memory considering rate
PAR	Pitch adjusting rate
BW	Displacement bandwidth
MS	Makespan
EN	Energy consumption
ART	Algorithm running time
CO	Cost of execution
ACO	Ant colony optimization
HPSOGWO	Hybrid particle swarm optimization-gray wolf optimization
MAA	Multi-objective based artificial algae algorithm

2 Related work

Extensive research is currently being conducted in the topic of resource scheduling, with the aim of automating workflows and developing tools for their management. These workflows integrate multiple automated processes to achieve a certain objective while operating under strict constraints. The optimal implementation of the entire workflow is facilitated by scheduling these processes within the limitations of the accessible computing resources. This facilitates the attainment of the stated objectives and ensures compliance with the constraints [15, 16]. Several efforts were recently dedicated to the offloading of computation in FC [17–19] and mobile edge computing (MEC) [20, 21]. In the paper, the authors studied primarily the combined optimization of decision factors and the allocation of computation and communication resources [22]. Dual connection for end-users was anticipated during offloading [19]. In particular, they assume that one link is connected to the FN and the other is connected to the greater computational units at the base station. For example, Chen et al. have suggested an optimal solution to process the tasks in the FN or remote CS through a system model made for a single user with one task [23], a single user with multiple tasks at [24] and multiple users with multiple tasks at [11], under a different nomenclature called computer access points. Nonetheless, [11, 23, 24] regarded most of these efforts to be the situation of a single FN. When the FN cannot finish the tasks within delays and energy constraints, it just discharges the tasks into the CS. A multi-FN situation and the CS are taken into account in [25] to ensure minimal latency.

The optimal solution would be computation offloading scenario where a FN could compute the fully offloaded task data in the FCI [26, 27]. However, either horizontal or vertical cooperation may be necessary for the FN. Some work is being done on collaborating horizontally with nearby FNs [28–30] and on vertical collaboration with the remote CS [11, 23, 24]. Here are a few examples. Wang et al. [5] recently proposed a situation, where the allocation of computing resources is evaluated both horizontally and vertically to reduce the time required for the completion of the task process. However, in their research, the waiting time due to the queue in the FN was not taken into account. In addition, for average delay reduction and the allocated computing and communication resources, multi-user and varied delay demands for every user play a significant role. In [28] and [29], the latency of transmission across FNs was taken into account but the multi-user situation was not treated explicitly. To improve the selection on computation offloading, the multi-user scenario has also been examined in [16, 31]. However, the waiting time [32] for task handling at the FN, which receives the tasks both from end-users and other FNs, is still crucial to take into account. None of this research, however, [5, 16, 29] examined the delay in queuing at the FN, especially in cooperation with the horizontal FN.

Task scheduling or resource management in FCI has also been the subject of several research. [33] proposes a task scheduling method for use in a FCI while taking into account the deadline, task priority level, and resource availability at fog layer (FL). The suggested approach uses resource allocation and load balancing to reduce the task execution time and cost. The Time–Cost aware Scheduling method was suggested by the authors in [34] to optimize a trade-off between task execution time and operational costs. The modified particle swarm optimization (MPSO) technique is employed in [35] for load balancing and task scheduling. To meet the latency and deadline requirements of tasks in a smart manufacturing framework, an intelligent scheduling method of the computer resources is presented in [36]. By recommending three parallel methods for task offloading, task buffering, and resource allocation while taking into consideration the estimated task execution time, laxity, and transmission latency to the CS, the authors in [37] increased task completion ratio and throughput. Ref. [38] describes a trade-off between the task's processing requirements and time limitations.

The authors suggested genetic algorithm-based scheduling in the FCI. But neither resource demands of the tasks nor virtual machines (VMs) and the resources that were accessible in the FL were taken into account. The allocation of IoT application modules to the FN and CS by taking into account the necessary resources constraints is done in [39], and it has increased resource utilization in FCI. In [40], the authors used evolutionary algorithms that allocate tasks to the processing nodes to optimize energy usage and service quality. The aforementioned studies fall short in their treatment of factors that might impact task processing time, such as communication overhead between FNs, resource availability at the processing node and communication overhead between FL and cloud layer (CL). In [41], the resources needed by the tasks, their due dates, and the strain on the processing nodes in addition to the CS's transmission time is taken into account. Additionally, the proposed approach lowers communication overhead because it runs at the central fog agent.

The literature extensively discusses the restricted resource management problem's optimization [15–41]. The majority of these studies seek to arrange scientific workflows without considering the latency sensitivity of IoT workflow activities. In addition, every resource management solution has been centralized designed in such a distributed setting. The workflow issue has been addressed using a variety of approaches, including fuzzy logic and meta-heuristic methodologies, which have shown success in workflow optimization and so are effective. In order to improve task offloading and workflow scheduling in FCI, we want to provide a novel method in this study. To address the problem of task offloading and thus assist in decision-making, we proposed the FDTCO method. To make up for the shortcomings of traditional theory based on ideas of approximation reasoning, fuzzy logic is used to explore the representation of imprecise information, approximate reasoning, and the modelling of hazy conceptions of natural language. This sort of logic depends on a set of rules and fuzzy inference techniques to decrease uncertainty and arrive at a solution that is appropriate in a short period of time. In our study, we have also utilized the hybridization of optimization strategies, depending on multi-objective fitness function, to achieve successful scheduling based on proposed HORSA technique. When it comes to FCI systems where rigor is not ideal, such as when CS and FN are executing workflow scheduling, FIS based offloading has another benefit that allows enormous flexibility.

3 System model and problem formulation

The core challenge of resource management is to allocate workflow tasks to computer resources in a manner that satisfies all imposed limitations and maximizes established objectives. The issue raised and the suggested model for the approach are presented in this section.

3.1 Workflow model

This issue has been dealt with in FCI on two different levels, namely the workflow user and the cloud service provider (CSP). There are restrictions and goals placed on both of these elements. A workflow is represented as a Directed Acyclic Graph (DAG) [42] from the user's perspective and is indicated by G (T, A), where T is the collection of n tasks that make up process G and are stated in the form $T = (T_1,\ldots, T_n)$. Each T_i that belongs to a T is in charge of the instruction lines throughout the workflow's execution. A is collection of edges, which denotes the restriction on task priority. Each edge A_{ij} illustrates how tasks T_i and T_j are related to one another, with T_i serving as the parent task and T_j as the child task. As a result, T_j begins to run after its parent task T_i has finished running. The output data from task T_i that will be passed on to task T_j is represented by the variable A_{ij}. In our work, parent tasks that have no predecessors are regarded as being ready for execution.

3.2 System model

As shown in Fig. 1, the architecture of FCI is organized around three major levels. The term "terminal layer" (TL) refers to the end-user layer. It is separated into a number of nearby sections and reflects the end-user environment. A FN is connected to numerous IoT devices that supply each region. The processing of tasks created by the TL might take place at the local FN, a nearby FN, or a CS. The task's characteristics, prerequisites, and the FN's capabilities all play a role in

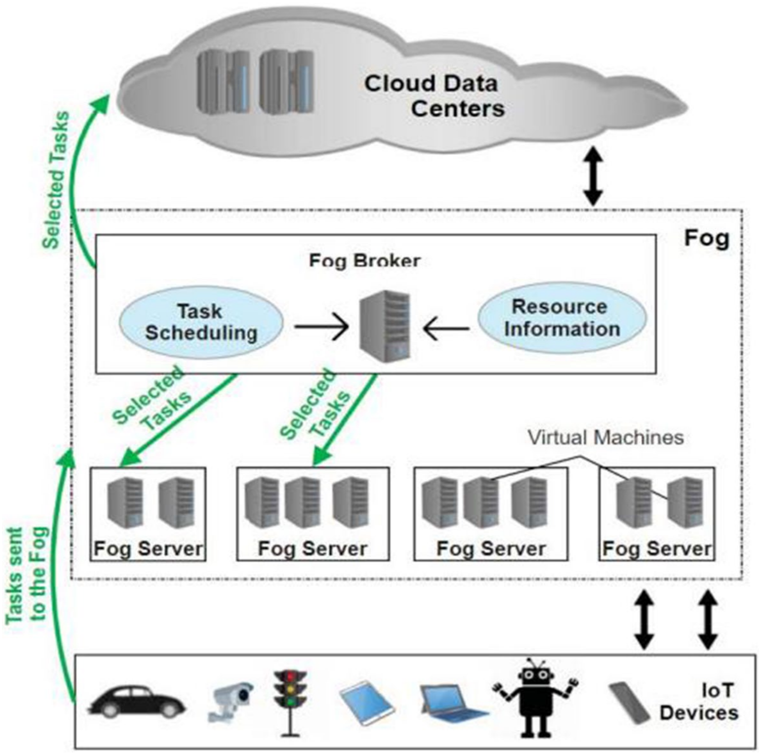


Fig. 1 Fog-cloud computing scenario [43]

this. The FL consists of a collection of FNs, designated as $N^{\mathrm{Fog}} = \{N_1^{\mathrm{Fog}} \ldots N_m^{\mathrm{Fog}}\}$, each of which is identified by a d-dimensional Vector of Resources Utilization (*RUV*), a geographic area (G_i^{Fog}), and a utilization threshold (UT_i).

$$\mathrm{RUV} = \{RU^1_{vm_1^{\mathrm{Fog}}}, \ldots \ldots, RU^d_{vm_d^{\mathrm{Fog}}}\}$$

The VMs that are stored on each FN are represented by the equation $VM^{\mathrm{Fog}} = (vm_1^{\mathrm{Fog}}, \ldots, vm_d^{\mathrm{Fog}})$, where each vm_i^{Fog} is located on an N^{Fog} defined by following elements:

- The number of instructions that can be processed in a second is referred to as computing power (*CP*).
- Bandwidth (BW) denotes a network-connected VM's capacity to communicate with other VMs and send data. Megabits per second (Mb/s) is the BW unit of measurement that we use in our work.
- Rate is an amount paid per minute.
- MC stands for memory capacity.

- a resource utilization amount $RU^{i}_{vm_i^{\mathrm{Fog}}}$ that represents how much of a VM is being used at any one time and is determined by the aggregate of the sizes of the tasks that were allotted, as specified by Eq. (1).

$$RU^{i}_{vm_i^{\mathrm{Fog}}} = \sum_{j=1}^{c} Z_j \tag{1}$$

where Z_j represents the length of Task T_j and c is the number of tasks that have been allotted in vm_i^{Fog}. The CL is intended to be the top layer in contact with the FL. It contains a collection of servers as well as a number of data centers $DC = (DC_1,\ldots, DC_e)$. Each server in DC_i eliminates a group of VMs. $VM_i^{\mathrm{Cloud}} = (vm_1^{\mathrm{Cloud}}\ldots vm_d^{\mathrm{Cloud}})$, where each $vm_i^{\mathrm{Cloud}} \in DC$ is defined by a computation speed expressed by the number of instructions executed per second, a bandwidth indicating the features of a network linked to the other VMs and a cost per time unit.

3.3 Quality of service (QoS) metrics model

- Makespan:

The maximum time for a task to be completed within a workflow defines the Makespan of the workflow G. The Makespan measure is provided by Equation (2) [44]:

$$\text{Makespan} = \max\{TF(T_i)\} \tag{2}$$

where T_i denotes ith task in the workflow and $TF(T_i)$ denotes the execution finish time of task T_i.

- Cost:

The cost measure shows the overall expense of executing a workflow. Equation (3) [44] provides the cost:

$$\text{Cost} = \sum_{i=0}^{n}\sum_{j=0}^{m}(TF(T_i) * U_j) + \sum_{i=1}^{n}\sum_{j=1}^{m}(Cw_{ij} * CTR_{ij}) \tag{3}$$

where the task's execution completion time is represented as $TF(T_i)$. The pricing unit of a VM_j that processes the task T_i is U_j. The connection weight between T_i and T_j is represented by Cw_{ij}. The cost of transferring information between the machine where T_i is mapped and another machine where T_j has been affected is denoted by the symbol CTR_{ij}.

- Energy Consumption:

The energy consumption cost is taken from [45], which contains active energy components denoted by E_{active} and idle energy components denoted E_{idle}. The E_{active} is related to the energy used while performing a task, whereas the E_{idle}, is referred to the energy consumed by idle resources. The term "active energy" can be determined using

$$E_{\text{active}} = \sum_{i=1}^{n} \alpha f_i v_i^2 \left\{ TF(T_i) - TS(T_i) \right\} \tag{4}$$

where α is a constant, f_i represents the frequency and v_i represent the supply voltage for the resource on which task i is being performed. $TS\left(T_i\right)$ and $TF\left(T_i\right)$ represents the starting and finishing time of execution for task T_i, respectively. When idle, the resource enters a sleep state with a low power supply and less relative frequency. As a result, [45] is used to calculate the energy consumed over this period:

$$E_{\text{idle}} = \sum_{j=1}^{m} \sum \text{idle}_{jk} \in \text{IDLE}_{jk} \alpha f_{\min_i} v_{\min_i}^2 L_{jk} \tag{5}$$

where IDLE_{jk} is a set of all idle slots of resource j. f_{mini} and v_{mini} represent the lowest supply voltage and frequency of resource j, respectively. L_{jk} is the amount of idle time for $idle_{jk}$. During the execution of tasks in the workflow, the overall energy consumed by the FCI is

$$E = E_{\text{active}} + E_{\text{idle}} \tag{6}$$

- Resource Utilization Factor:

Resource Utilization Factor (RUF) is measured by assigning the most workload to each VM while staying within the resource utilization threshold, as well as by reducing cost and power use. In order to determine how well resources are being used, we use the RUF equation as described in Eq. (7) [46]:

$$\text{RUF} = \frac{Ts}{VMs} \tag{7}$$

The scheduling strategy that employs the least number of VMs for scheduling workflows and increases the RUF value is the one which optimally leverages the VMs, leading to greater energy efficiency of the system.

3.4 Problem formulation

The issue to be resolved is how to most effectively offload and schedule a specific workflow in a heterogeneous FCI while taking the user's restrictions into account. The task scheduling issue is then resolved by establishing the sequence in which workflow tasks should be completed as well as the VMs to which they should be assigned. Some goals and restrictions must be met by the developed strategy. In this study, we want to maximize the computational resource usage while minimizing the makespan and cost factors. Additionally, we aim to accommodate the user's financial and time restrictions. Consequently, the issue can be stated as follows: how to design a workflow scheduling strategy that distributes tasks to VMs in order to minimize makespan, cost but maximize computing resource utilization while performing tasks within a restricted budget and time frame. We then express the issue formally as a multi-objective optimization problem which minimizes objective parameters to be within enforced limitations.

$$Multi-ObjectiveOptimization\ \left\{minimize(x \in X)\, F(X) = (f_1(x), f_2(x), \ldots, f_n(x))\right\} \tag{8}$$

where n is the number of objective functions, which is always greater than or equal to 2, and $x = (x1, x2, \ldots, xr) \in X, F(X)$ is a vector of decision-making variables. X is also the set of possible solutions.

4 Proposed approach for resource management

In this section, we outline our proposed resource management approach for the FCI. The issue is seen as a complicated one that must be resolved in the best manner possible while taking into consideration a number of factors such as objectives and restrictions. Here, we suggest the use of a Multi-Agent System (MAS) model to provide a distributed resource management strategy. The MAS model that we suggest consists of three different kinds of cognitive agents, having one agent or a group of agents responsible for each layer of the system architecture, as shown in Fig. 2. These agents work together to achieve a certain goal.

- Terminal Agent:

A terminal agent (TA) is in charge of carrying out the workflow by actively seeking the appropriate resources while abiding by the applicable QoS metrics and the established limits. The main function is for the TA to divide the workflow into three distinct sections, executing the light-weighted small tasks on its own, offloading delay-sensitive heavy tasks to Fog Agents (FA) as well as computationally demanding huge tasks to the Cloud Agent (CA).

- Fog Agents $\{FA_1 \ldots FA_m\}$:

In an IoT area, each Fog Agents (FA) oversees a FN. Its main job is to react to a received scheduling request with an offer that maximizes its interest, meets the limitations, and improves the enforced QoS metrics. When an offloading request is submitted; the Fog Broker (FB) is introduced as the central agent, assuring cooperation with nearby FAs and CA to develop an appropriate scheduling scheme with the least amount of waiting time. The next step is for the FB to scheduling delay-sensitive tasks among FAs and offloading computationally demanding tasks to the CA.

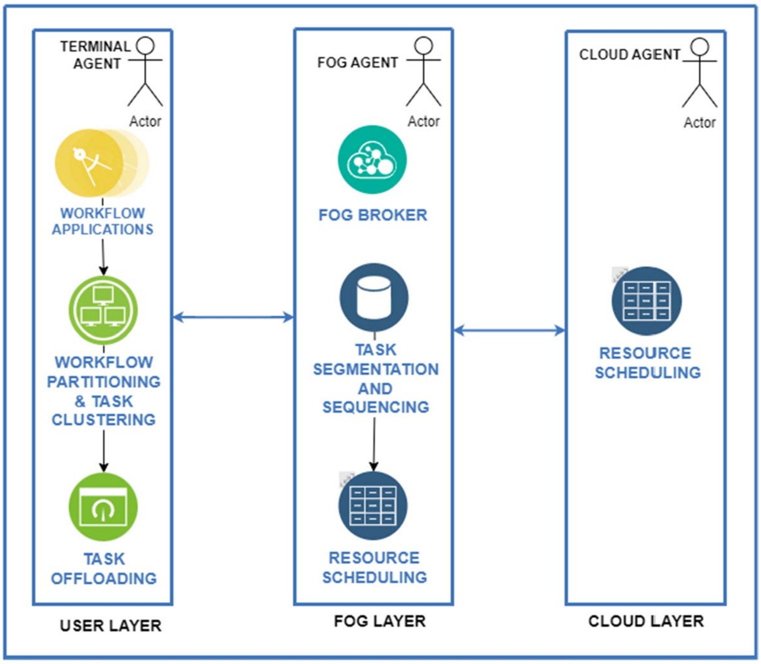


Fig. 2 Layered architecture of the proposed system model

- Cloud Agents $\{CA_1, \ldots, CA_e\}$:

In the CS context, each Cloud Agent (CA) is a resource provider. The function of CA_i is to deliver an acceptable number of resources with the least processing time and minimal expense possible with the CS's limitless resources [47]. If necessary, this agent can communicate with the FB. When it gets a request from a FB, it creates a scheduling solution with its resources. As a result, the CA is in standby mode awaiting a request. The FB tries to maximize the distribution of tasks across the FL while handling an IoT workflow.

4.1 Proposed algorithms

In this subsection, we describe the methods used for the task offloading and resource scheduling presented in this paper. The TA of workflow application must first be provided with the workflow to start the process of workflow partitioning, task clustering and sequencing and task offloading. A fuzzy clustering-based task offloading request as shown in Eq. (9) is created by the TAs of workflow application after partitioning workflow into different clusters of tasks based on task length, input file size and output file size. We also develop task segmentation and sequencing techniques before the scheduling algorithm as the major objective is to minimize the overhead of the entire scheduling procedure. Algorithm 1 depicts the various stages of our strategy, which aims to organize the processing of workflow while meeting the goals of both the user and the provider.

$$Req = \{Tid, PC, Z\} \tag{9}$$

where *Tid* is a set of individual identifiers for every task in the workflow. *PC* represents how the tasks are related to one another and define the scheduling sequence that needs to be followed. Z indicates the length of task as defined by the number of instructions.

Algorithm 1: Resource Management Algorithm

```
Input: Workflow W, FN and CS resources.
Output: Optimal Makespan, Cost and RUF.
1 Begin
2 /* TA is assigned a workflow to execute
3     Workflow Partitioning into Tasks
4     Fuzzy Dominance based Task Clustering using Algorithm-2
5 /* TA offloads tasks assigned to nearest FA/FB and CA
6     Task Offloading using Fuzzy Task-Layer Assignment Rules
7     Offload ($TA$, $F$A ($Req$ = {Tid, $P$C, Z}))
8     Offload ($TA$, CA ($Req$ = {Tid, $P$C, Z}))
9 /* FB on receiving the offloaded tasks from TA
10    Task Segmentation based on EC
11    Task Sequencing an using TSA
12    Resource Scheduling using Algorithm-3
13 /* FB offloads tasks with type=0 to CA with precedence
14    Offload (FB, CA ($Req$ = {Tid, $P$C, Z}))
15 /* CA on receiving the offloaded tasks from FB
16    Resource Scheduling using Algorithm-3
17 /* Resource are allocated to tasks using optimal schedule at all layers
18    Allocate (ORS (Ti, Rj))
19    Execute (Ti, Rj)
20 /* Evaluate QoS parameters of optimal resource schedule(ORS).
21    Evaluate (ORS ($Makespan$, Cost, RUF))
22 End
```

4.1.1 Fuzzy dominance-based task clustering and offloading

Figure 3 provides an illustration of our proposed offloading approach. Workflows are grouped using a fuzzy workflow partitioning strategy in accordance with task specifications. These task groups are then clustered into L clusters and assigned to the appropriate tiers using fuzzy task-layer assignment rule, which is created, evolved, and optimized.

Fuzzy logic integrates quantitative and qualitative data derived from human specialists. Task clustering is the initial and crucial phase in the fuzzy logic-based workflow partitioning method, which tries to divide the workflows into

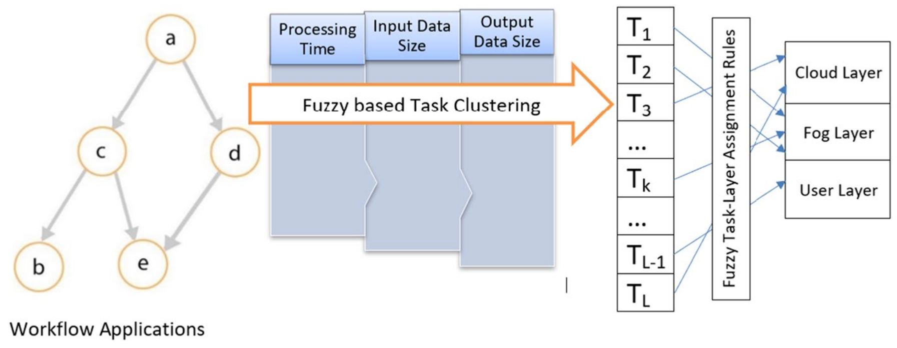


Fig. 3 Proposed fuzzy dominance-based task clustering and offloading (FDTCO)

tasks before realizing the current status of the VMs. The fuzzy nearness calculation approach employed in [48] is also used here to classify tasks into several clusters based on the three characteristics of the tasks.

As the computational workload directly influences the computing resource requirement, and the communication overhead influences the latency between different tiers. Three features of tasks: task length, input and output file size, are used to cluster the tasks. Equations (10) and (11) are used to determine the input and output file sizes for task k, respectively:

$$F^{\sim}_{\text{in}}(k) = \sum j\, D^{\sim}(j, k),\ \forall j \in Parents(k) \tag{10}$$

$$F^{\sim}_{\text{out}}(k) = \sum j\, D^{\sim}(k, j),\ \forall j \in Children(k) \tag{11}$$

where the sets of parent and child tasks for task k are indicated by *Parents(k)* and *Children(k)*, respectively. Apart from relationship among tasks in the workflow, each task can be identified by their well-known characteristics with three possible fuzzy values:

- Task Length: $Z^{\sim} \in \{$Short, Moderate, Long$\}$;
- Input File size: $F^{\sim}_{in} \in \{$Small, Medium, Large$\}$;
- Output File size: $F^{\sim}_{out} \in \{$Small, Medium, Large$\}$;

Fuzzy dominance is used for sorting and labelling the tasks in every dimension. Algorithm 2 is presented with the thorough pseudo-code. Figure 4a demonstrates how maximum nearness value is calculated and Fig 4b illustrates the computation of minimum distance between clusters. In accordance with every dimension as described in Algorithm-2, tasks are arranged in decreasing order of $l^{\sim}$ and the lowest, the mean, and highest values for every dimension are determined by the centroids of related clusters. In this manner, three levels are assigned to each task.

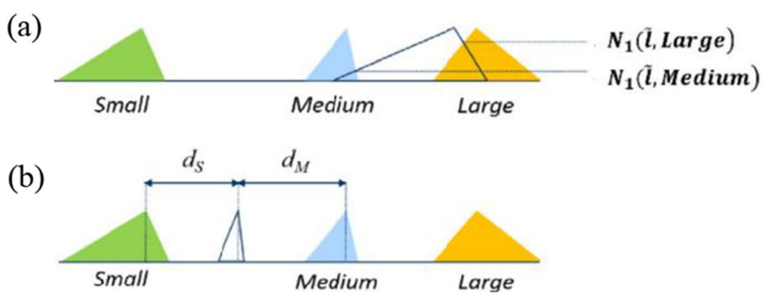


Fig. 4 **a** Maximum nearness value (upper) **b** Minimum distance (below)

Algorithm 2: Fuzzy Dominance-based Task Clustering

```
Begin
    For $r$ in {Task Length, Input file size, Output file size}
        $\tilde{l} = \tilde{Z} / \tilde{F}_{in} / \tilde{F}_{out}$ and sort the tasks in accordance to the descending order of $\tilde{l}$;
        $Large =$ max$(\tilde{l})$; $Medium =$ avg$(\tilde{l})$; $Small =$ min$(\tilde{l})$;
        For each task $k$
            $N_{1L}(k) = N_1(\tilde{l}, Large)$; $N_{1M}(k) = N_1(\tilde{l}, Medium)$; $N_{1S}(k) = N_1(\tilde{l}, Small)$.
            If $\nexists N_{1X}(k) > 0, \forall x$ (As shown in Fig. 4 (b))
                $D_S(k) = dis(Small, \tilde{l})$; $d_M(k) = dis(Small, \tilde{l})$; $d_L(k) = dis(Small, \tilde{l})$.
                $Level_r(k) =$ argmin$(d_S(k), d_M(k), d_L(k))$;
            Else
                $Level_r(k) =$ argmax$(N_{1S}(k), N_{1M}(k), N_{1L}(k))$;
        End
    End
End
```

After workflow partitioning, all tasks are tagged and grouped into $3 \times 3 \times 3 = 27$ clusters. The next step is to understand and identify the "cluster-layer" allocation rule, which is shown in Fig. 5. Each task cluster has three options: the TL, the FL and the CL. The solution space for layer assignment is 3^{27}, which is too big for the brute force technique. Therefore, this rule is created and optimized through threshold tuning. After rigorous experimentation with huge numbers of permutations and combinations of possible scenarios, we finally devised the FIS rules for task-tier allocation. Following the determination of the task-layer assignment rule, the tasks will be distributed to the VMs of the respective layer in accordance with the proposed scheduling technique.

4.1.2 Task segmentation algorithm

The segmentation algorithm is executed as soon as a set of offloaded workflow tasks are received from the TA. This algorithm was created to handle a variety of operations that may be carried out on the FL. These tasks are distinguished from those demanding advanced computing abilities over the limits of the FN by the demands which fall under the capability of the available resources. This phase

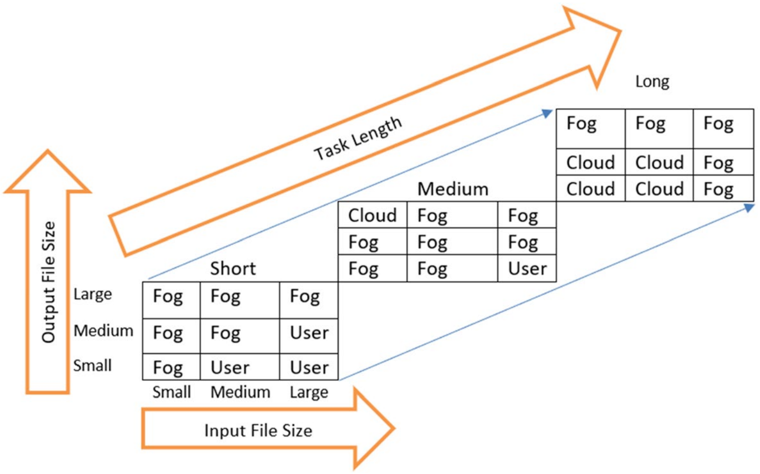


Fig. 5 Fuzzy rules for task–layer assignment

is carried out by the FL by comparing every task in the workflow using Eq. (12), which specifies the Computation Intensity (*CI*) of a task on a FN.

$$CI = \frac{CT}{f} \tag{12}$$

where *CT* stands for execution completion time of the task and *f* stands for the operating frequency of the FN, which is the combined clock speed of the processors running on that FN. It can be calculated by the aggregating computing power *CP* of the VMs associated with FN shown in Eq. (13) and can be measured in million instructions per second (MIPS).

$$f = \sum_{j=1}^{m} CP_j \tag{13}$$

Applying Eq. (12) yields two groups of values: (1) the group of light computation activities creates *CI* values that are trending toward 0 and less than 1, and (2) the group of computation-intensive activities generates *CI* values that are more than 1. In our study, each task is given a binary variable named *type* $(T_i) = 1$ or 0, which reflects the task's category. This *type* (T_i) variable is used in further steps for making optimal offloading decision. Equation (14) specifies the variable *type* of task T_i. Algorithm 3 illustrates the task segmentation phases that the FA will carry out.

$$type\left(T_i\right) = \begin{cases} 1, \text{ if } 0 < CI(T_i) < 1 \\ 0, \text{ if } CI(T_i) \geq 1 \end{cases} \tag{14}$$

Algorithm 3: Task Segmentation Algorithm

```
1  Input: $T(T_1,..., T_n)$, FN
2  Output: $T(T_1,..., T_n)$
3  /* $i$ is the task index in $T$
4  Begin
5   $i$=1
6  for each $T_i \in T$ do
7      if $(CI(T_i) \leq 1)$ then
8          $type(T_i) = 1$
9      else
10         $type(T_i) = 0$
11 End
```

4.1.3 Task sequencing

The tasks sequencing procedure, which is accomplished using the Topological Sorting Algorithm (TSA), is the second action taken by the FB [49]. We use the TSA to adhere to the restriction on task precedence. The goal is to produce an ordered list L that organizes the workflow tasks while taking into consideration the dependencies between the tasks in terms of priority. It should be noted every task gets added to the queue only once. When the TSA is applied to the workflow instance made up of five tasks, depicted by the DAG in Fig. 6, the order of tasks is given sequentially as $L = (a, c, d, b, e)$.

Each task belonging to the workflow is characterized by a list of parent tasks containing the previous tasks. Next, a list of parent tasks that include the preceding tasks is a characteristic of every task in the workflow. Then, using the set of parent tasks, a list named *edge-value* was constructed, which represents the total number of input edges of every task. The list edge-value determined for the example workflow depicted in Fig. 6 is illustrated in Table 2.

Firstly, the task with the list *edge-value* = 0 will be inserted into the list L and will then be labelled as visited. This task has no predecessor tasks and will be regarded as a root task. The subsequent tasks will then be appended to the list named queue. The result is a reduction of 1 in the total number of edges. The task will be recorded

Fig. 6 DAG representation of a workflow with five tasks

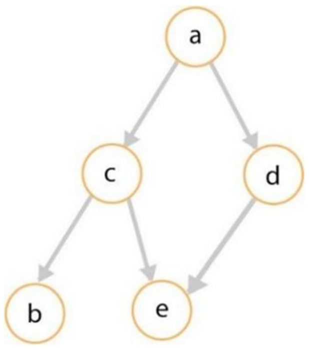


Table 2 Number of edges for tasks

Tasks	a	c	d	e	b
Edge-value	0	1	1	2	1

as visited and inserted into *L* whenever this *edge-value* of a task reaches 0. Once a task with no child is encountered, the algorithm terminates and all the tasks are then labelled as visited. The conclusive arrangement of workflow tasks by the *TSA* is shown in Fig. 7. The FB transmits the *L* list to the CA. To generate comprehensive and effective execution, the task scheduling algorithm is then applied to the workflow across both levels. In this procedure, the tasks whose value *type* = 0 are assigned to the FA, while the tasks whose value *type* = 1 are assigned to the CA.

4.1.4 Task scheduling

We describe the workflow scheduling method in this section. It attempts to bring together the competing goals of the user and service provider. We handle the workflow scheduling problem as a multi-objective optimization with constraint satisfaction problem. The workflow scheduling problem in FCI often aids in identifying the best scheduling that maximizes one or more QoS metrics. In this scenario, we present a resource scheduling strategy with multi-objective optimization for Makespan, cost, and resource utilization. These are described in Sect. 3.3. The Genetic Algorithm (GA), which we combined with Harmony Search algorithm (HSA) and to form Hybrid Meta-heuristic-based Optimized Resource Scheduling Algorithm (HORSA) for scheduling scientific workflow, will shorten Makespan, reduce costs, and improve resource utilization in the FL and CL. In this part, we describe each stage of the proposed algorithm.

HSA has a high convergence speed but a poor level of accuracy. The usage of GA is widespread and simple to adopt, although there are still certain gaps. Actually, the individual/chromosome in GA and the harmony in HS are two distinct representations of a viable solution. Additionally, populations in GA and harmonic memory in HS are two alternative formulations for a collection of practical solutions. These serve as the foundation for the introduction of HSA into GA. Harmony memory (*HM*) and population, as well as chromosomes and harmony, are properly interlinked. As a result, the novel HORSA algorithm is proposed.

The proposed HORSA algorithm's primary step is to first set up several parameters, such as maximum number of iterations (*MI*), population size (*PS*), crossover

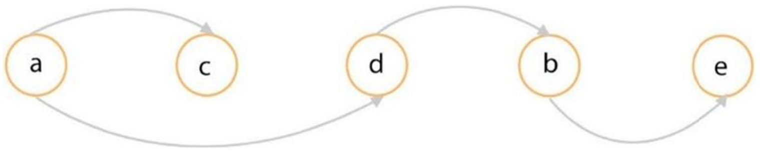


Fig. 7 The linear order of the graph

rate (*CR*), mutation rate (*MR*), harmony memory size (*HMS*), harmony memory considering rate (*HMCR*), pitch adjusting rate (*PAR*), number of iterations (*NI*), and displacement bandwidth (*BW*). Determine each individual's fitness value throughout the population. Then Create the subsequent population using selection, crossover and mutation. Consider the following population to be a HM, and each *individual* to be a *harmony*. Produce *New-Harmony* using HS. Recalculate the fitness value for each *individual* in the population. Until satisfactory outcomes are attained, the entire procedure is repeated. Figure 8 shows the flowchart of proposed HORSA algorithm.

When compared with GA, the novel HORSA algorithm improves the speed of convergence and capacity to find the best solution. In each cycle, if all the individuals of the entire population are processed by both GA and HS, then duration of operation and degree of complexity of the novel HORSA algorithm will be greater than that of GA. This paper gives two enhancements that take into account

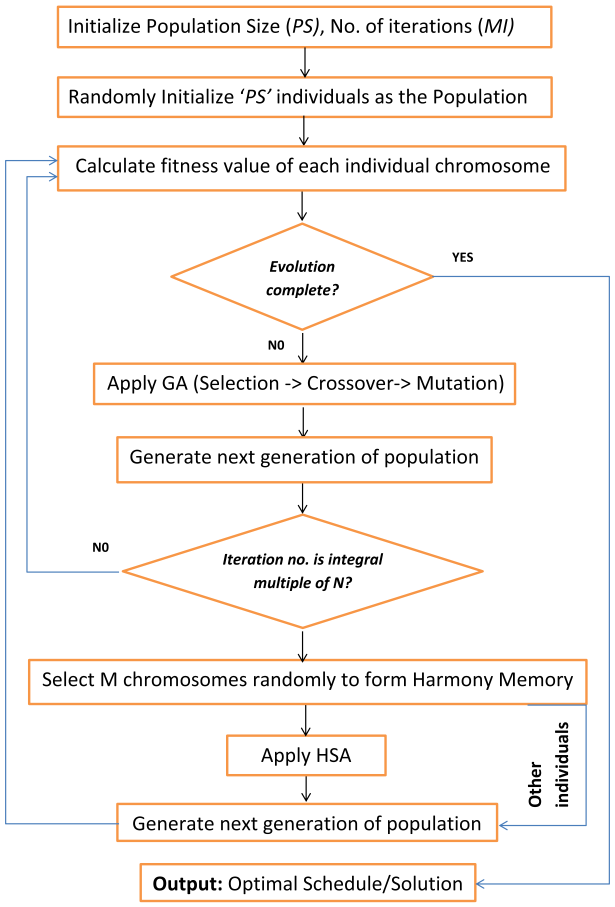

Initialize Population Size (*PS*), No. of iterations (*MI*)
Randomly Initialize '*PS*' individuals as the Population
Calculate fitness value of each individual chromosome
Evolution complete?
YES
NO
Apply GA (Selection -> Crossover-> Mutation)
Generate next generation of population
NO
Iteration no. is integral multiple of N?
Select M chromosomes randomly to form Harmony Memory
Apply HSA
Other individuals
Generate next generation of population
Output: Optimal Schedule/Solution

Fig. 8 Flowchart of proposed hybrid-heuristic-based optimized resource scheduling algorithm (HORSA)

the result quality, convergence rate, and complexity of the proposed algorithm. The first improvement of this new approach is that GA is utilized as a global heuristic throughout all iterations, but HS is executed only when the number of iterations is an integral multiple of N (N is an integer), to make the local heuristic more effective. Second improvement of this approach is to execute HS on only M (M is an integer) chromosomes after running GA on the next generated population in each round. These improvements reduce the amount and size of HS operations to make HORSA quicker and simpler.

- Initial population generation:

We utilize the genetic approach to initialize a population. Here w tasks are allocated VMs at random (w is greater than half of total number of tasks and less than the total number of tasks), while the remaining tasks are allocated to VMs such that exploitation of available resources is maximum. The calculation of resource utilization magnitude (RUM) is specified in Eq. (15), which was inspired by the research presented in [46, 50]. This condition is equivalent to when the task T_i is scheduled in the VM vm_j.

$$RUM\left(T_i, vm_j\right) = \sqrt{(MU_{i,j} - MA_{i,j})^2} \tag{15}$$

where $MU_{i,j}$ is determined using Eq. (16) which shows the total quantity of memory utilized by a VM vm_j, when a task T_i of size Z_i is assigned to the vm_j. $MA_{i,j}$ is determined using Eq. (17) shows the quantity of memory of vm_j that is still available if T_i is assigned to vm_j.

$$MU_{i,j} = RU^{j}_{vm_i^{\mathrm{Fog/Cloud}}} + Z_i \tag{16}$$

$$MA_{i,j} = MC_j - MU_{i,j} \tag{17}$$

The MC_j variable denotes the memory capacity of vm_j. The task-resource allocation which maximizes the RUM is the one which most effectively equalizes the RU in the VM. A positive or zero number is always used to quantify the RUM, just like any other metrics.

Evaluation of Fitness Function:

A multi-objective fitness function F that maximizes makespan, cost, and RUF is used in the GA assessment stage. As stated in Eq. (18), the fitness function maximizes the makespan by α, the cost values by β, and the RUF by γ.

$$\mathrm{F} = \alpha * \mathrm{Makespan}_{\mathrm{Normalized}} + \beta * \mathrm{Cost}_{\mathrm{Normalized}} + \gamma * \mathrm{RUF}_{\mathrm{Normalized}} \tag{18}$$

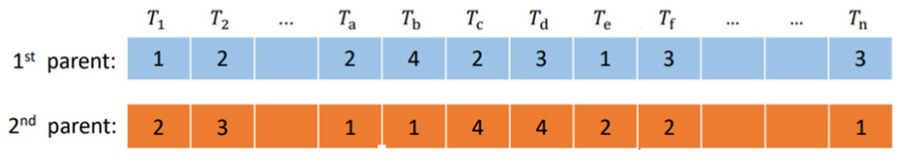
In the first phase, we assume that all accepted QoS measures are satisfied with weights that are equal. In accordance with the decision-maker's choices, we might give each goal or measure more or less weight. When F is applied to the resulting solutions, the GA will eventually reach at an efficient schedule that minimizes Makespan and cost and ensures the optimal utilization of VMs. The proposed approach seeks to optimize a collection of QoS measures with widely varied scales and values, whereas normalization is a crucial calculation. The *RUF* is one statistic that should be maximized, while cost and Makespan are two metrics that should be minimized. Equation (19) establishes the normalization formula [45].

$$\mathrm{QoS}_{\mathrm{Normalized}} = \begin{cases} \dfrac{\mathrm{QoS}_i}{\mathrm{maxQoS}_i}, & \text{if QoS is to be minimized} \\ 1 - \dfrac{\mathrm{QoS}_i}{\mathrm{maxQoS}_i}, & \text{if QoS is to be maximized} \end{cases} \tag{19}$$

where MaxQoS specifies the highest value of QoS attained till current iteration. Equation (19) determines the normalized value for a certain QoS measure after every iteration.

- Selection operation:

The selection process facilitates us in picking the pairs of *chromosomes* (solutions) that act as inputs for the crossover process. In our approach, we choose a pair of *individuals* (solutions) using a binary tournament. This includes choosing a couple randomly from the overall population and identifying the *best individual* (the one with maximum fitness value is selected in the case of maximization). This process will be performed once more. These two chosen *chromosomes* are designated as *parents* during the crossover procedure.



	T_1	T_2	...	T_a	T_b	T_c	T_d	T_e	T_f	...	...	T_n
1st parent:	1	2		2	4	2	3	1	3			3
2nd parent:	2	3		1	1	4	4	2	2			1

- Crossover operation:

The crossover operation is employed to produce a new solution using two-parent solutions. The goal is to produce new individual(offspring) in the present population by blending various components of existing individuals [47]. After picking two parent solutions(schedules) from the population namely *Parent*$_1$ and *Parent*$_2$, the crossover operator now chooses two random spots (task-ids) in the *Parent*$_1$ solution(schedule). The values at the chosen spots (resource-ids) are then swapped with the corresponding spots in *Parent*$_2$ solution(schedule).

A crossover operation using *Parent*$_1$ and *Parent*$_2$ is shown in Fig. 9 as an example. Initially, the tasks T_b, T_c, T_d and T_e are randomly selected from *Parent*$_1$. The resource-ids are then exchanged between both parent schedules to generate the new schedules, namely *Offspring*.

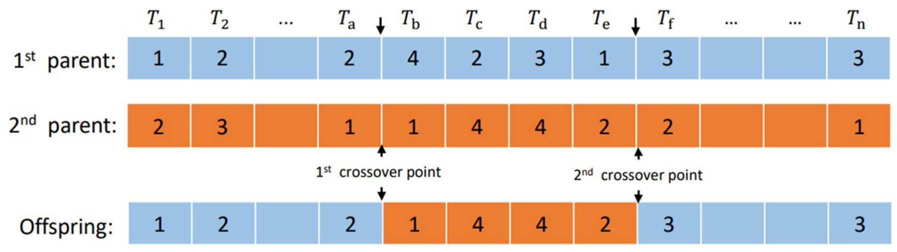

T_1 T_2 … T_a T_b T_c T_d T_e T_f … … T_n
1st parent: 1 2 2 4 2 3 1 3 3
2nd parent: 2 3 1 1 4 4 2 2 1
1st crossover point
2nd crossover point
Offspring: 1 2 2 1 4 4 2 3 3

Fig. 9 Illustration of crossover operator on *Parent*1 and *Parent*2

- Mutation operation:

The mutation operation is employed to reintroduce stochastic characteristics that are unrelated to any of the parent solutions, which prevents converging to a local optimum solution [45]. We apply permutation mutation as the mutation mechanism in our approach. The primary goal is to alter the order in which a solution completes tasks. The two primary phases of the mutation process are (1) selecting a task at random from the solution, and (2) selecting a different resource-id at random that replaces the selected task allocation [36]. Figure 10 shows an instance of the mutation process being applied to *Parent* using permutation to create new offspring, referred to as *Offspring*. The instance given involves the selection of T_C at random, followed by a random change in the allotted resources.

- Applying Harmony Search:

In 2001, Z. W. Geem proposed the harmony search algorithm. It is a brand-new, clever optimization algorithm that mimics the way musicians improvise music by constantly adjusting the pitches of their musical instruments in order to get the ultimate, ideal *harmony*. HSA correlates the *i*th decision variable with the *i*th musical instrument, the *j*th solution vector with the *Harmony* H_j $(j = 1, 2,\ldots, M)$ produced by all musical instruments, and the objective function with evaluation. The fundamental step in the HS procedure is to first set up the parameters *HMS, HMCR, PAR and BW*. Afterward, placing the initial solution vectors i.e. HMS in HM which were produced at random. For each *New-Harmony* component

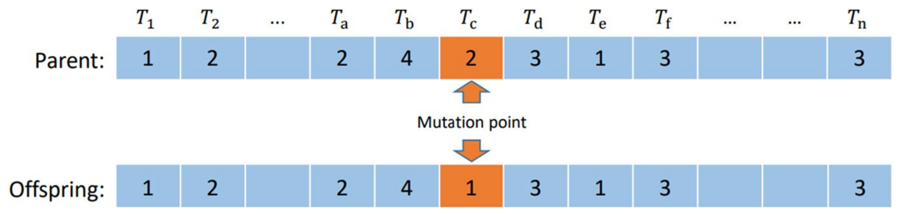

T_1 T_2 … T_a T_b T_c T_d T_e T_f … … T_n
Parent: 1 2 2 4 2 3 1 3 3
Mutation point
Offspring: 1 2 2 4 1 3 1 3 3

Fig. 10 Illustration of mutation operator on parent

(solution vector), we can either randomly find and pick in *HM* with a probability of *HMCR* or we can randomly find values throughout the range without taking *HM* into consideration with a probability of *1-HMCR*. When doing a search in *HM*, move the *New-Harmony* component with a probability of *PAR* to nearby values that are under a *BW* range. When the New Harmony has been created, if it is superior to the worst old harmony in *HM*, *HM* should now include the *New-Harmony* while excluding the worst old one. Until satisfactory results are attained, this procedure is repeated. The HSA flowchart is shown in Fig. 11. Algorithm 4 presents an illustration of the way the GA is used for workflow scheduling. The initial population is created in lines 4–8, and the produced population is evaluated in line 9. The instructions for using the genetic operators following the harmony search are provided in the remaining sections of the procedure. The proposed algorithm produces the optimal resource schedule as its result.

Algorithm 4: Hybrid Meta-heuristic-based Optimized Resource Scheduling Algorithm (HORSA)

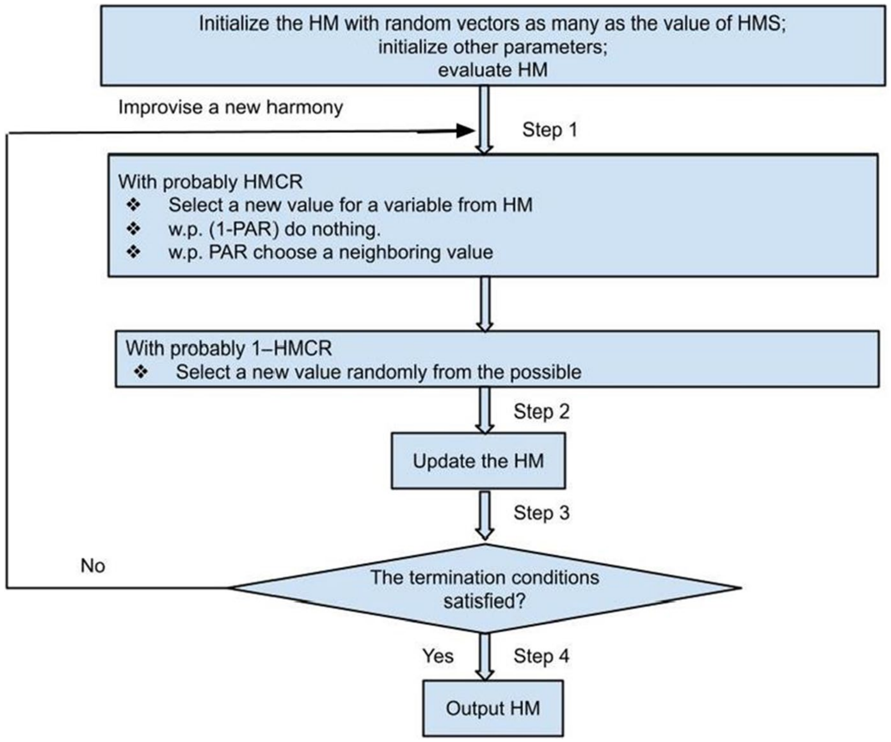

Initialize the HM with random vectors as many as the value of HMS;
initialize other parameters;
evaluate HM
Improvise a new harmony
Step 1
With probably HMCR
❖ Select a new value for a variable from HM
❖ w.p. (1-PAR) do nothing.
❖ w.p. PAR choose a neighboring value
With probably 1–HMCR
❖ Select a new value randomly from the possible
Step 2
Update the HM
Step 3
No
The termination conditions satisfied?
Yes
Step 4
Output HM

Fig. 11 Flowchart of harmony search algorithm

```
Algorithm 4: Hybrid Meta-heuristic based Optimized Resource Scheduling Algorithm (HORSA)
Input: Task with Precedence L, Resource List
Output: Optimal Resource Schedule
1 Begin
2 Define maximum-iteration, maximum-size, crossover-rate, mutation-rate.
3 Define HMCR, PAR, BW and pitch-limits
4 /*Generate initial population for task scheduling
5 for $T_0$ to $T_{n*w}$ do
6       Assign the task to a VM randomly
7 for $T_{n*w}$ to $T_n$ do
8       Assign task to a VM having minimum RUM
9       Calculate fitness of initial population using Eq. (18)
10      Set iteration-number = 0
11 while iteration-number $\leq$ maximum-iteration do
12      while population-size $\leq$ maximum-size do
13            /*Apply Genetic Algorithm
14            parent1= Selection()
15            parent2= Selection()
16            child1 = Crossover (parent1, parent2)
17            child2 = Mutate(child1)
18            population-size++
19      end while
20      Calculate fitness of current population
21      iteration-number++
22      if (iteration-number is multiple of N)
23            /*Apply Harmony Search Algorithm
24            Generate new-harmony by taking M fittest individuals from population
25            Adjust the pitch to get new-harmony(solutions)
26            if (rand > HMCR)
27                  Choose an existing harmony randomly
28            else if (rand >PAR)
29                  Adjust the pitch randomly within pitch-limits
30            else
31                  Generate new-harmony using randomization
32            end if
33            Accept the new-harmony (if better)
34      end if
35  end while
36 /*return the best solution in population/HM
37 return Optimal Resource Schedule.
38 End
```

5 Performance evaluation

5.1 Workflow datasets

The proposed technique is effectively evaluated in this research using five well-known scientific processes [51] from distinct scientific fields. The DAG XML files that describe each of these scientific operations are utilized as input for the simulations, which are released by the Pegasus project [52]. Inspired from the existing works [15, 16, 53–55], we have utilized five well-known workflow datasets which belong to different area of application and has different DAG structures. And the diversity of these workflow datasets contributes to validating the generality and applicability of our proposed algorithm across different domains and scenarios. These workflows are made up of a variety of tasks, dependencies, run-times, and data that must be passed from one task to another. A simple graphical representation of the workflows used is shown in Fig. 12.

The following is a description of these workflow datasets:

(1) *Montage workflow*: This illustrates an astronomy project that produces unique sky mosaics from a variety of input photographs.
(2) *CyberShake workflow*: This approach is used to describe the earthquake risks that exist in a specific region.
(3) *Epigenomics workflow*: This is used in bioinformatics to automate many steps involved in processing genome sequences.
(4) *LIGO (Inspiral) workflow:* This method is used to find gravitational waves.
(5) *Sipht workflow*: This is used to automatically search all bacterial replications for sRNA producing genes.

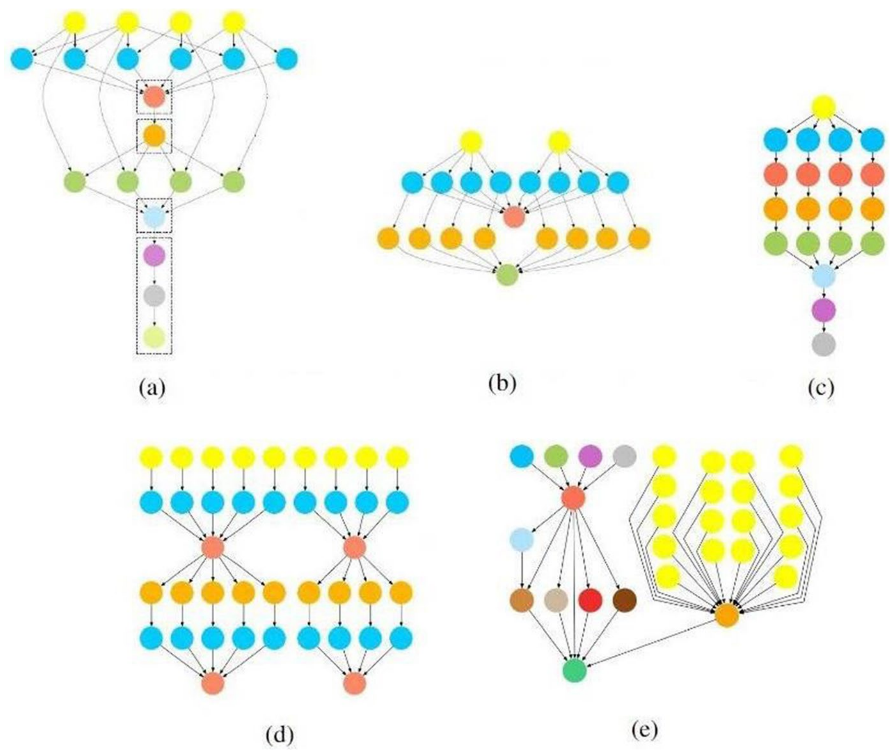


Fig. 12 Structure of scientific workflows: **a** Montage, **b** CyberShake, **c** Epigenomics, **d** LIGO, **e** SIPHT [53]

5.2 Simulation environment

The primary purpose of this study is to determine the efficacy of the proposed workflow offloading and task scheduling strategy in the heterogeneous FCI. Since FogWorkflowSim is the most widely used simulator and provides a full simulation environment for optimizing workflow execution in the FCI [54], we employ it with the Fuzzy logic toolkit to conduct our experiments. We have used a collection of five standard scientific workflows with variety of task sizes to replicate various workflow application in our simulation. First, we simulated some lightweight IoT applications using workflows of 20–80 interconnected tasks. Then, we simulated computationally expensive applications using workflow of 100–1000 tasks. As the FogWorkflowSim platform uses iFogSim to mimic the computing environment, we configured it with various configurations of resources to create heterogeneous environment and all FNs includes VMs with reduced performance than that of the CS. Inspired by the FAT-ETO technique [55], the simulated infrastructure comprises of 5 mobile devices, 5 CSs working with 5 FNs.

Table 3 lists the settings of simulation environment. The MIPS of the VMs at FN are kept between 1200 and 1600, and those of the CS between 1600 and 2000. We began by first implementing our proposed FDTCO techniques and evaluate the QoS improvement with and without using our strategy in terms of makespan, cost, energy consumption and resource utilization. We then implemented our proposed HORSA algorithm along with some existing resource scheduling techniques [56–58] in FCI to assess the QoS parameters in terms of makespan and cost. Here, the multi-objective fitness function of the optimization algorithm is evaluated by assigning makespan, cost and RUF with an equal weightage of 33.33% each. Figure 13 shows the parameter tuning of the proposed HORSA algorithm based on Makespan(MS),

Table 3 Simulation environment settings

Parameters	MD	FD	CS
Number of servers/devices	5	5	5
Processing Speed (MIPS)	MD1—1000, MD2—1000, MD3—1000, MD4—1000, MD5—1000	FN1—1200, FN2—1300, FN3—1400, FN4—1500, FN5—1600	CS1—1600, CS2—1700, CS3—1800, CS4—1900, CS5—2000
Task Execution Cost ($)	0	FN1—0.1, FN2—0.2, FN3—0.3, FN4—0.4, FN5—0.5	CS1—0.5, CS2—0.6, CS3—0.7, CS4—0.8, CS5—0.9
Communication cost ($)	0	0.01	0.02
Active power (MW)	700	800	1600
Idle power (MW)	30	40	1300
Uplink bandwidth (Mbps)	20	10	1
Downlink bandwidth (Mbps)	40	10	10

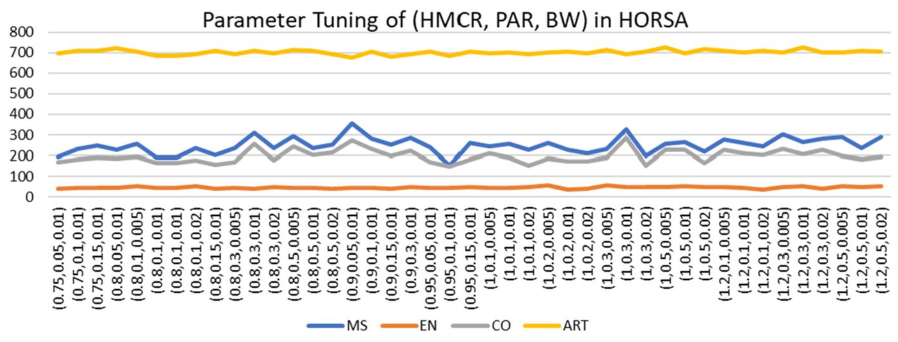


Fig. 13 Parameter tuning of (HMCR, PAR, BW) in proposed HORSA

Cost(CO) and Energy Consumption(EN) and Algorithm Running Time(ART). The parameters tuning of GA's algorithmic parameters like MaxIter, MaxSize, CR, and MR are done and also compared with [53, 59] before finalizing the parameter values used during the simulation. Table 4 enlists the optimal parameter settings after running the HORSA with a number of different combinations of algorithm parameter values.

5.3 Results and evaluation

The evaluation of our strategy in terms of its efficacy in the FCI and a comprehensive result comparison of our proposed approach with other existing strategies are covered in this section. Three performance metrics are used to evaluate our proposed techniques are makespan, cost, and resource usage. These performance metrics are described in Eqs. (2)–(7). We repeated each experiment 10 times and we have executed 100 iterations for each experiment. Finally, we compared average of makespan, cost, resource usage of our approach to those of the existing techniques. Our proposed MOTORS algorithm (combination of FDTCO and HORSA) outperformed existing approaches measured in terms of percentage variation in QoS parameters.

5.3.1 Evaluation of proposed FDTCO algorithm

The simulation assessment of our proposed offloading technique was done in terms of four QoS indicators on five benchmark workflow datasets having various sizes of workflows. First, we simulated all the five benchmark workflow datasets while increasing the number of tasks in the workflow from 20 to 1000. Figures 14, 15, 16, 17, 18 show the workflow execution outcomes in terms of percentage improvement in makespan and cost. The QoS metric improvement of our proposed offloading techniques is shown in Tables 5 and 6. We compare our proposed scheduling strategy with and without the proposed offloading algorithm in order to better understand the significance of offloading techniques that effectively distributes workflow tasks across FL and CL in heterogeneous FCI. The goal of this experiment is to determine if the proposed offloading strategy can enhance the collaboration between resources

Table 4 Algorithm parameters

Algorithm parameters	Values/range	Explanation
MaxIter	100	The total number of runs of the algorithm
For ACO algorithm		
Ant count	25	The number of ants, each ant represents a solution
PR	0.1	They the pheromone updating rate
CP	0.85	The choosing probability
Wa	0.95	The influence weights
For HPSOGWO algorithm		
Particle size	25	The number of particles, each particle represents a solution
C1, C2	2	They are the acceleration coefficients
Wp	0.1	It is inertia weight
Pack Size	25	The total number of wolves in a pack, each wolf is a potential solution
a	[2,0]	This parameter decreases from 2 to 0 during the iterations of algorithm
A	[-a, a]	This parameter is used to regulate the convergence rate, initially set to 0
when ($A > 1$) wolves diverge away from prey, when ($A < 1$) wolves converge toward prey		
C	[0,2]	This parameter is used to restrict falling into local optima, initially set to 0
D	Any value	It is an empirical model used to surround the prey
r1, r2	[0,1]	These are random coefficients. Their values vary from 0 to 1
For MAA algorithm		
N	25	The number of algal colonies, each algal colony represents a solution
e	0.3	The loss of energy
Δ	2	The shear force
Ap	0.5	The adaptation probability constant
For HORSA algorithm		

Table 4 (continued)

Algorithm parameters	Values/range	Explanation
MaxSize	100	Maximum size of population
CR	0.8	Crossover rate
MR	0.1	Mutation rate
HM	100	Harmony memory
HMCR	0.95	Harmony memory consideration rate
PAR	0.1	Pitch adjustment rate
BW	0.01	Distance bandwidth

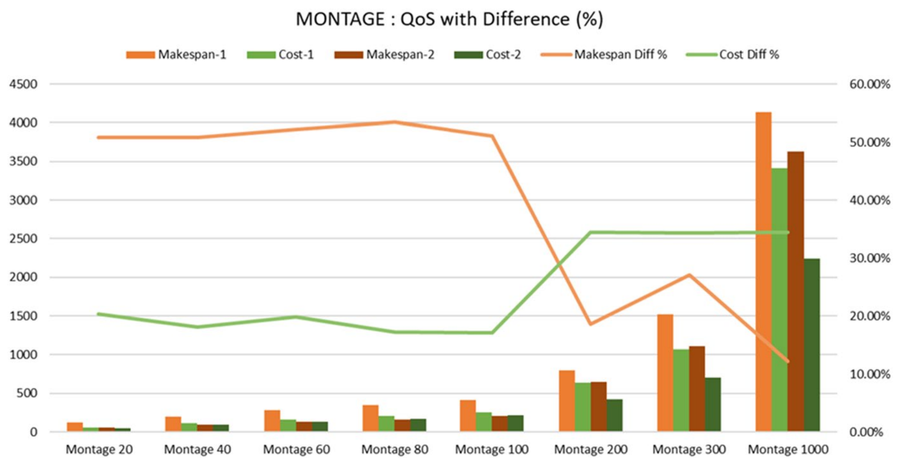


Fig. 14 Difference percentage between QoS (Makespan-Diff % and Cost-Diff %) generated with proposed FDTCO algorithm (Makespan-2 and Cost-2) and without offloading algorithm (Makespan-1 and Cost-1) using Montage Workflow

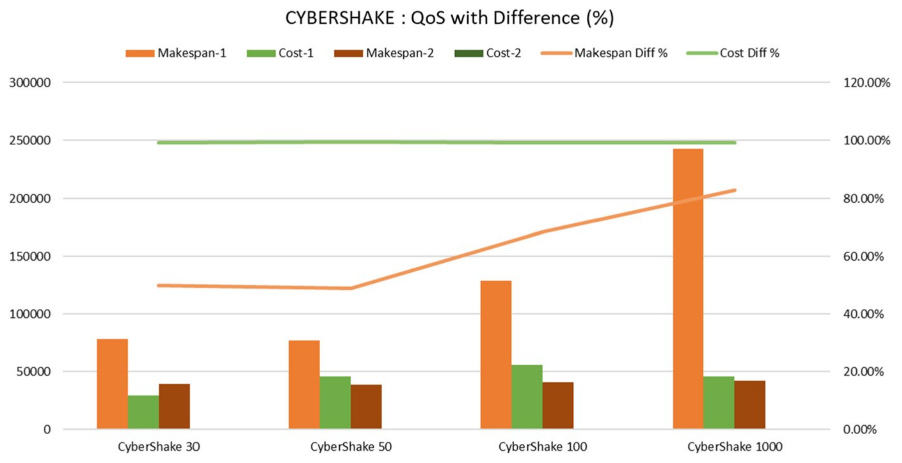


Fig. 15 Difference percentage between QoS (Makespan-Diff % and Cost-Diff %) generated with proposed FDTCO algorithm (Makespan-2 and Cost-2) and without offloading algorithm (Makespan-1 and Cost-1) using CyberShake Workflow

at CL and FL. For this, we used five standard scientific workflows with 20–1000 tasks to apply our resource scheduling method. We used a number of configurations of proposed HORSA algorithm with equal weights allocated to the makespan, cost, and RUF. Then, we conducted experiments with our resource scheduling strategy, first without taking the offloading method into consideration and then with it. First, our proposed technique is concurrently launched in CL and FL conditions. Then, the fitness function used in the HORSA implementation optimizes the quality of all chosen QoS indicators.

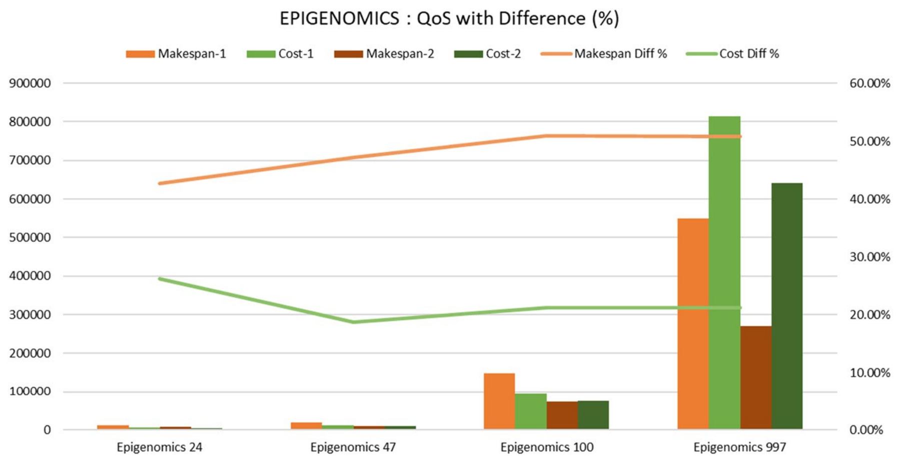


Fig. 16 Difference percentage between QoS (Makespan-Diff % and Cost-Diff %) generated with proposed FDTCO algorithm (Makespan-2 and Cost-2) and without offloading algorithm (Makespan-1 and Cost-1) using Epigenomics Workflow

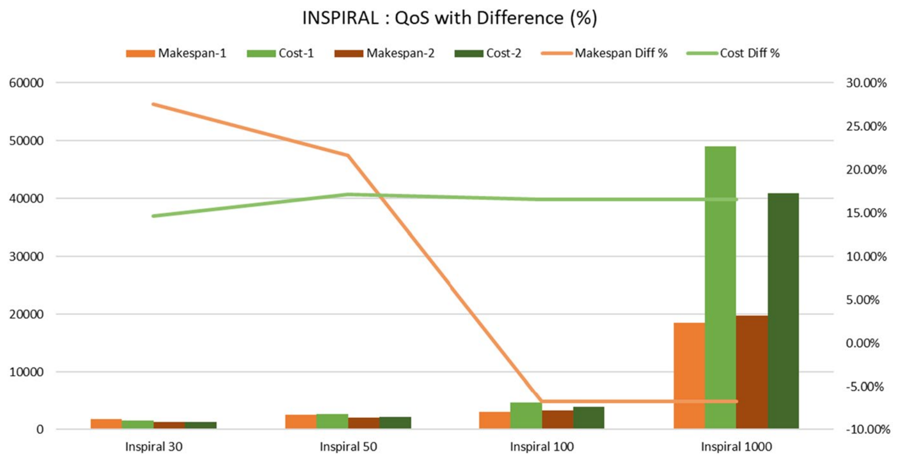


Fig. 17 Difference percentage between QoS (Makespan-Diff % and Cost-Diff %) generated with proposed FDTCO algorithm (Makespan-2 and Cost-2) and without offloading algorithm (Makespan-1 and Cost-1) using Inspiral Workflow

The outcomes of using our resource scheduling strategy without taking the offloading method into account are shown in Table 5. In all algorithm settings, we see that the number of tasks carried out by the FL is much larger than those carried out by the CL. Because the method gives the highest priority to the top-performing VMs, the resource scheduling solutions derived in this way are optimized in terms of makespan. As the technique maximizes resource utilization at FL and reduces the number of VMs utilized at CL, resource scheduling solutions are also optimized in terms of load balancing. The outcomes of testing our strategy using the FDTCO

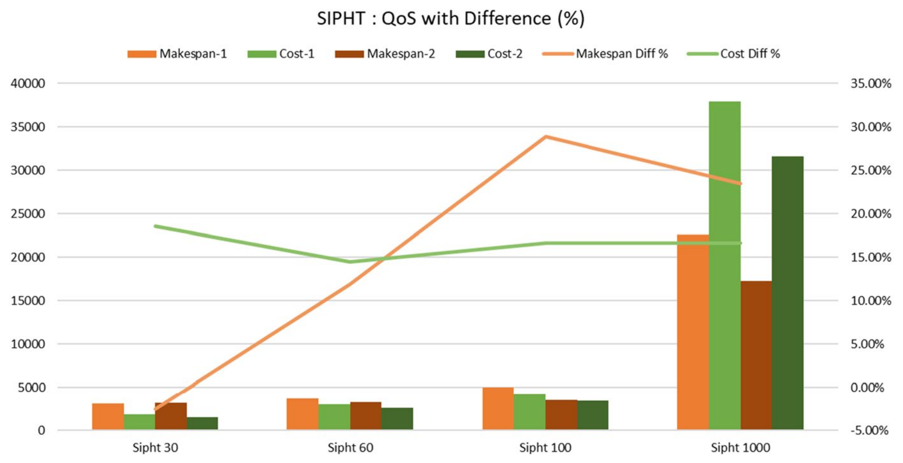


Fig. 18 Difference percentage between QoS (Makespan-Diff % and Cost-Diff %) generated with proposed FDTCO algorithm (Makespan-2 and Cost-2) and without offloading algorithm (Makespan-1 and Cost-1) using SIPHT Workflow

technique are shown in Table 6. According to the analysis, the quantity of tasks in the CL and FL varies drastically, which enhances intra-layer cooperation. The QoS values produced by this experiment are quite better than those of the previous trial. As there are heterogeneous VMs in all three layers, the metric values of makespan, energy consumption and cost vary drastically for all considered workflow datasets. We found that the number of tasks executed in the FL increased in direct proportion to the number of tasks in the workflow. As a result, carrying out more tasks in the FL instead of CL can have a big impact on the Makespan metric which exceed the delay constraint. Furthermore, the cost metric is also impacted by increasing the number of tasks in a computing layer.

The difference between the makespan of the solution produced using our HORSA technique and the FDTCO algorithm (Makespan-1) and the makespan of the solution produced without the offloading (Makespan 2) is shown in Figs. 14, 15, 16, 17, 18 in terms of percentage computed with percentage variation. The findings for montage, CyberShake and epigenomics workflow datasets show that the makespan difference surpass 50%, which may be regarded as a huge improvement in makespan. We highlight from the findings that average of makespan difference is approximately 10% and 15% for inspiral and SIPHT workflow, respectively, which is also significant enhancement in makespan. The cost of scheduling solutions for all workflow datasets is shown to be greatly reduced when the FDTCO technique is used, as shown in Figs. 14, 15, 16, 17, 18. In comparison to the first trial without the offloading technique, the difference in cost is reduced by 25%, 20%, 15% and 15% on average for montage, epigenomics, inspiral and SIPHT workflow, respectively. This reduction in difference in cost from the first experiment without the offloading technique is typically 100% for CyberShake workflow.

In summary, we can say that the cooperation between CS and FN resources to optimize QoS metric values is greatly improved by our proposed FDTCO technique.

Table 5 QoS and task distribution of proposed HORSA without offloading algorithm on benchmark workflow datasets

Workflow dataset	Makespan (s)	Energy consumed (J)	Cost ($)	RUF at CL	RUF at FL	No. of tasks in CL	No. of tasks in FL	No. of tasks in TL
Montage 20	122	7	59	0.4	3.2	2	16	3
Montage 40	193	10	111	0.4	7.2	2	36	3
Montage 60	278	14	161	0.4	11.2	2	56	3
Montage 80	350	18	204	0.4	15	2	75	4
Montage 100	415	21	257	0.4	19.2	2	96	3
Montage 200	794	38	639	0.4	39.2	2	196	3
Montage 300	1520	42	1072	0.4	59.2	2	296	3
Montage 1000	4140	126	3417	0.4	199.2	2	996	3
Cyber-Shake 30	78,250	593	29,664	2.4	1	12	5	14
Cyber-Shake 50	76,628	897	45,715	4.6	1	23	5	23
Cyber-Shake 100	128,817	1786	55,901	8.4	3	42	15	44
Cyber-Shake 1000	242,670	3596	45,821	85.8	22.4	429	112	460
Epigenomics 24	11,782	406	6029	0	2	0	10	15
Epigenomics 47	18,179	612	10,619	0	4.6	0	23	25
Epigenomics 100	149,075	4498	94,555	0	8	0	40	61
Epigenomics 997	549,290	35,419	813,211	0	90	0	450	548
Inspiral 30	1729	72	1513	1.4	4.6	7	23	1
Inspiral 50	2532	125	2630	2.4	7.6	12	38	1
Inspiral 100	2997	192	4627	4.8	9.2	24	46	1
Inspiral 1000	18,464	1904	48,981	50.2	149.8	251	749	1
Sipht 30	3060	109	1802	0.2	4.4	1	22	7
Sipht 60	3681	146	3019	0.4	10.2	2	51	6
Sipht 100	4964	147	4148	0.6	17.8	3	89	6

Table 5 (continued)

Workflow dataset	Makespan (s)	Energy consumed (J)	Cost ($)	RUF at CL	RUF at FL	No. of tasks in CL	No. of tasks in FL	No. of tasks in TL
Sipht 1000	22,572	1206	37,948	6.4	180.4	32	902	35

Consequently, to fulfil the rising demands of workflow applications, the FCI should be preferred under such circumstances. In comparison to resource scheduling without the offloading technique, our FDTCO approach produced resource scheduling solutions that are more cost and makespan effective.

5.3.2 Evaluation of proposed MOTORS Algorithm (combination of FDTCO and HORSA)

We evaluated the average makespan and cost produced by our proposed technique with those of other existing techniques in order to illustrate the efficiency of our approach in comparison to existing approaches. The average makespan of five benchmark workflows with tasks ranging between 20 and 1000 is shown in Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 along with a comparison with other existing methods. When compared to the ACO [56], HPSOGWO [57] and MAA [58] techniques our proposed technique provides a significant reduction in makespan. In comparison to ACO [56], HPSOGWO [57] and MAA [58], our technique often reduces the value of makespan by an average of 91%, 88% and 49%, respectively. Our method reduces the makespan by using both the fuzzy dominance-based task clustering and offloading approach and the multi-objective optimization using the hybrid approach of HSA and GA, which involves offloading tasks, optimizing the scheduling based on makespan, cost and RUF metrics, and then repeatedly running the HORSA until it reaches satisfactory values. The HPSOGWO technique also involves optimizing other QoS metrics, including makespan, cost, latency, reliability, and availability that have an impact on makespan values. However, as shown in Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, our strategy results in an average cost that is somewhat greater than that of other alternatives. The average difference between our outcomes and the ACO [56], HPSOGWO [57] and MAA [58] approaches is roughly 129%, 112% and 100%, respectively, this variation in cost that are still acceptable.

The optimization balance of our proposed technique when compared to others between makespan and cost metrics that we show previously is thereafter supported by Table 8. We use the average of QoS metrics to precisely compare how the algorithms vary from one another. We provide the average makespan produced by the four analyzed techniques in Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28. The findings indicate that the average values are noteworthy, particularly for montage workflows with 20 to 1000 tasks, demonstrating that our technique provides much lower makespan metrics than previous approaches. The average of the cost produced by the comparative algorithms are shown in Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28. According to the findings, the improvement is seldom as significant as it was in the

Table 6 QoS and task distribution of Proposed HORSA with FDTCO Algorithm on benchmark workflows datasets

Workflow dataset	Makespan (s)	Energy consumed (J)	Cost ($)	RUF at CL	RUF at FL	No. of tasks in CL	No. of tasks in FL	No. of tasks in TL
Montage 20	60	6	47	0.00	3.00	0	15	6
Montage 40	95	10	91	0.00	7.20	0	36	5
Montage 60	133	14	129	0.20	11.00	1	55	5
Montage 80	163	17	169	0.20	15.00	1	75	5
Montage 100	203	18	213	0.40	19.00	2	95	4
Montage 200	647	26	419	0.40	39.40	2	197	2
Montage 300	1108	38	704	0.40	59.40	2	297	2
Montage 1000	3633	111	2241	0.60	199.00	3	995	3
Cyber-Shake 30	39,149	1185	220	0.00	3.00	0	15	16
Cyber-Shake 50	39,078	1191	228	0.00	5.20	0	26	25
Cyber-Shake 100	40,818	1258	452	0.00	10.60	0	53	48
Cyber-Shake 1000	41,699	1615	371	0.00	100.40	0	502	499
Epigenomics 24	6756	220	4450	0.00	3.20	0	16	9
Epigenomics 47	9603	319	8629	0.00	5.40	0	27	21
Epigenomics 100	73,020	2221	74,491	0.00	9.40	0	47	54
Epigenomics 997	270,192	17,486	640,658	0.00	82.20	0	411	587
Inspiral 30	1252	44	1292	0.00	5.60	0	28	3
Inspiral 50	1985	76	2179	0.00	9.00	0	45	6
Inspiral 100	3200	130	3860	0.00	17.80	0	89	12
Inspiral 1000	19,716	1292	40,866	0.00	180.80	0	904	97
Sipht 30	3137	111	1468	0.20	1.60	1	8	21
Sipht 60	3245	132	2582	0.40	3.40	2	17	40
Sipht 100	3530	172	3457	0.60	4.20	3	21	74

Table 6 (continued)

Workflow dataset	Makespan (s)	Energy consumed (J)	Cost ($)	RUF at CL	RUF at FL	No. of tasks in CL	No. of tasks in FL	No. of tasks in TL
Sipht 1000	17,275	1412	31,623	6.40	46.20	32	231	706

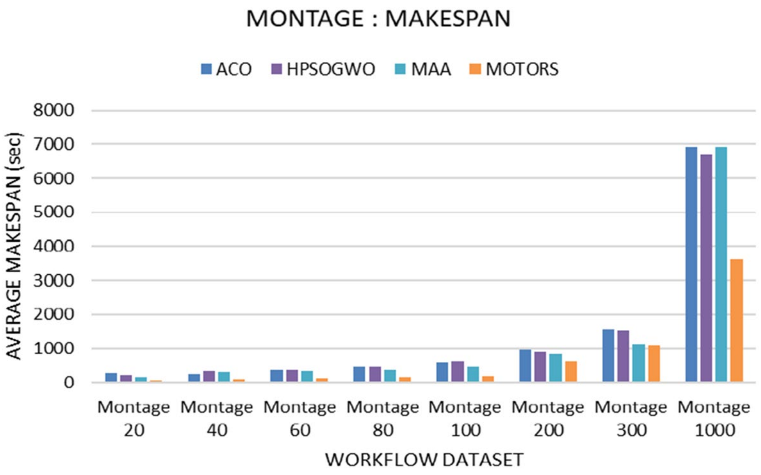


Fig. 19 Average Makespan of the Montage workflow

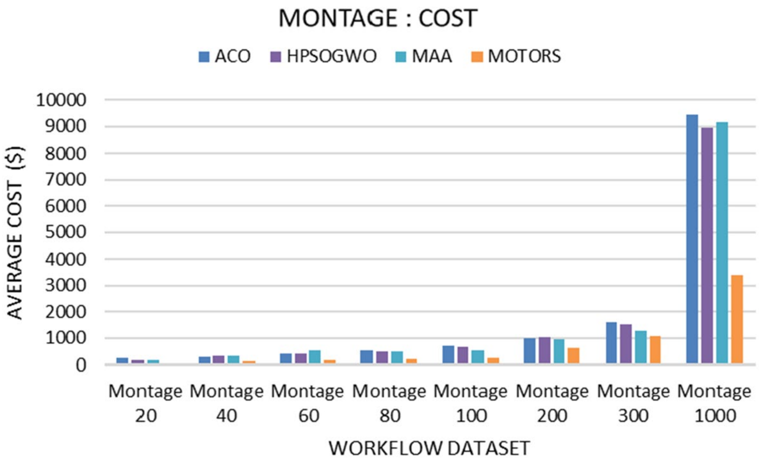


Fig. 20 Average Cost of the Montage workflow

makespan example. This indicates that our method produces cost metrics that are almost identical to those of the other existing methods. We come to the conclusion that our technique provides the best resource scheduling solutions in terms of cost

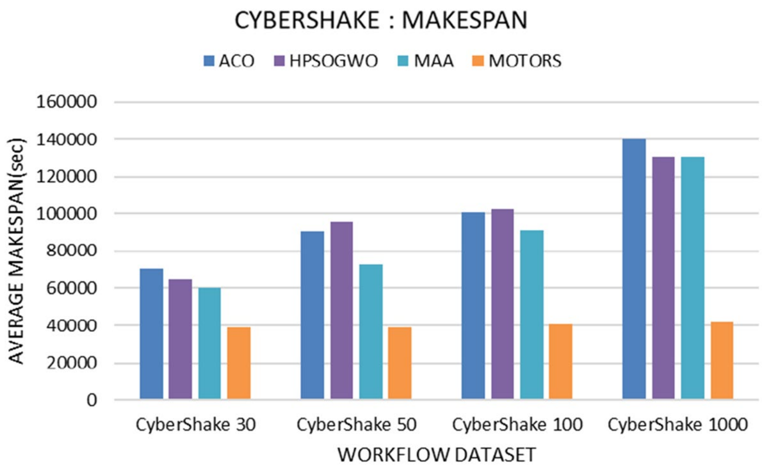


Fig. 21 Average Makespan of the CyberShake workflow

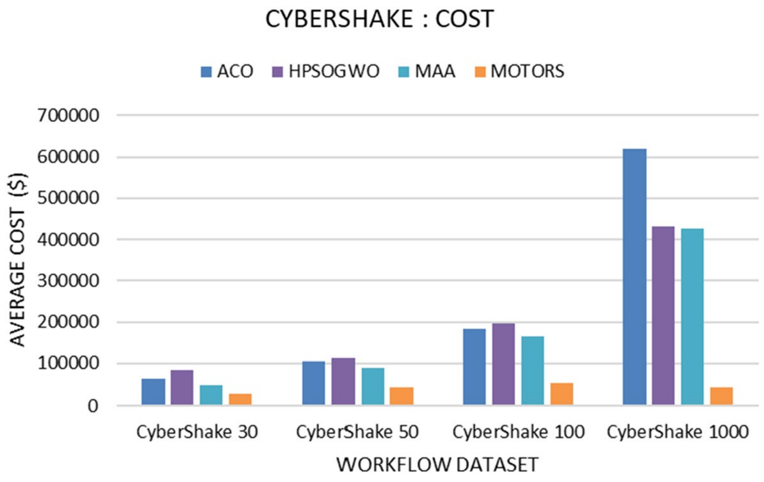


Fig. 22 Average Cost of the CyberShake workflow

and makespan after assessing its efficacy in comparison to ACO [56], HPSOGWO [57] and MAA [58] in multi-objective optimization.

All the QoS parameters of the existing techniques are compared with that of our proposed technique. Figures 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 show the makespan and cost metric outcomes after simulating all the five workflow datasets. As seen in all the simulation results, the makespan and the cost metric rises as number of tasks in workflow is increased from 20 to 1000 for all workflow datasets. Tables 7 and 8 demonstrates the proposed algorithm's makespan (shown in bold letters) and execution cost (shown in bold letters) of workflow, respectively, which yields better QoS metrics with respect to other existing techniques. This is because the number of tasks carried out

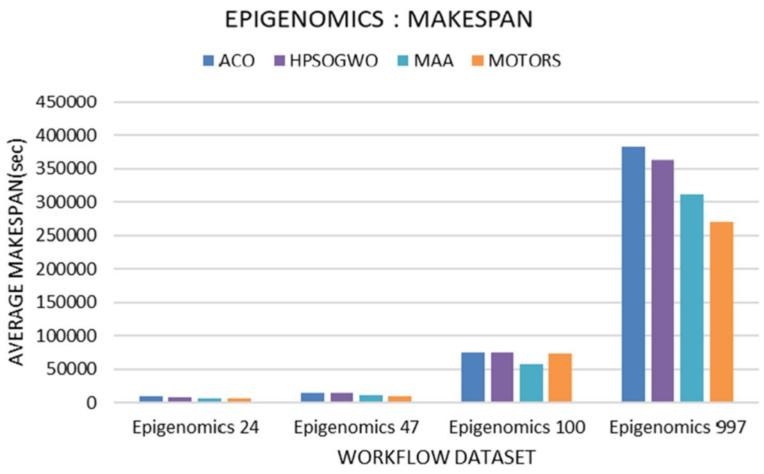


Fig. 23 Average Makespan of the Epigenomics workflow

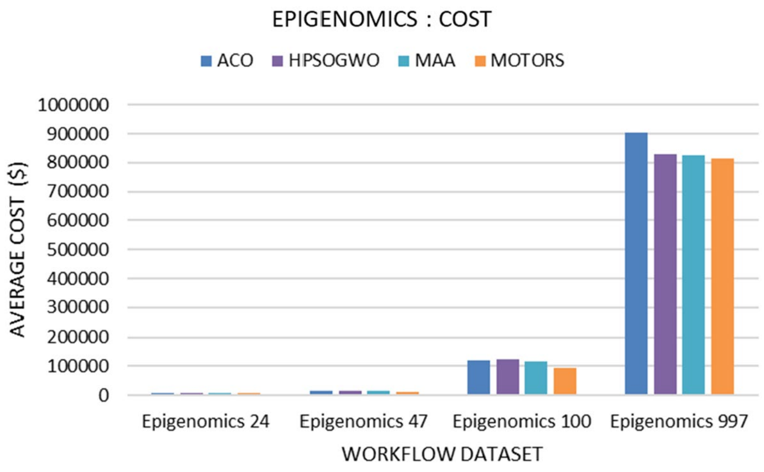


Fig. 24 Average Cost of the Epigenomics workflow

two tiers (FL and CL) are now divided into three tiers (TL, FL and CL). Makespan values have decreased because more VMs are being used to perform the same number of tasks. This facilitates task scheduling across an increasing number of VMs. This tendency may be seen as a slight decline in RUF.

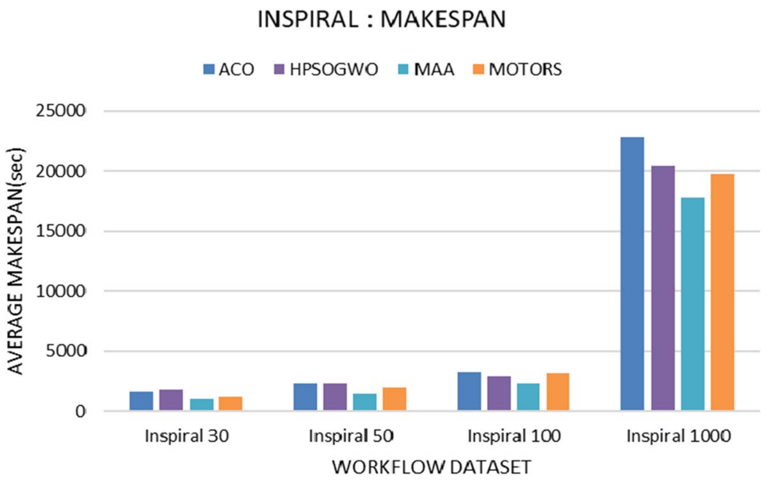


Fig. 25 Average Makespan of the Inspiral workflow

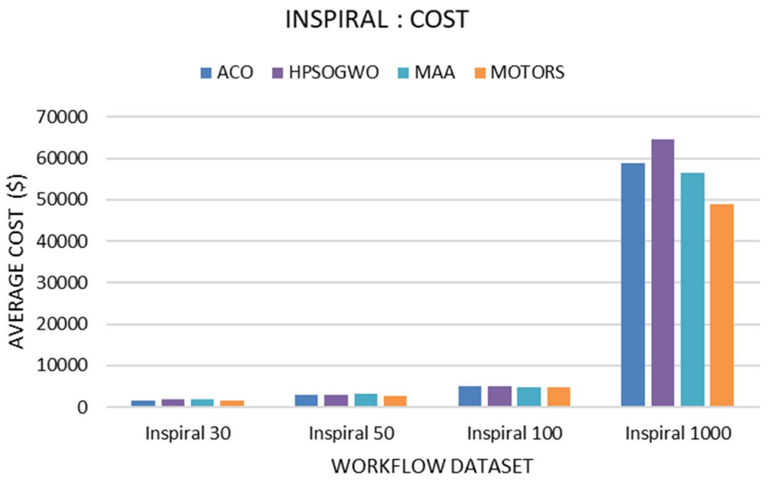


Fig. 26 Average Cost of the Inspiral workflow

6 Conclusion

In this paper, a fuzzy dominance-based task clustering and offloading technique (FDTCO) along with a hybrid optimization-based resource scheduling algorithm (HORSA) is proposed for workflow applications in FCI having heterogeneous resources with different communication costs. HORSA is based on hybridization of GA and HS. We have simulated five different workflow datasets: Montage, Cyber-Shake, epigenomics, LIGO (inspiral) and SIPHT and calculated average makespan, average cost, average RUF and average energy consumption. We have evaluated our proposed MOTORS algorithm, which is a combination of FDTCO and HORSA,

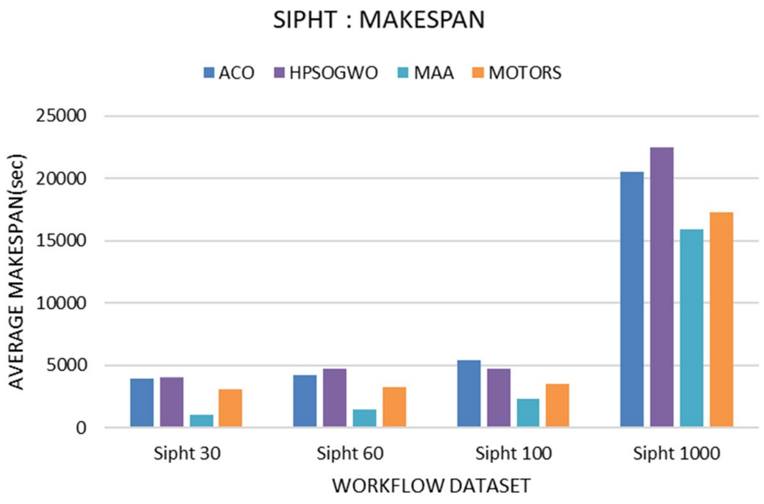


Fig. 27 Average Makespan of the SIPHT workflow

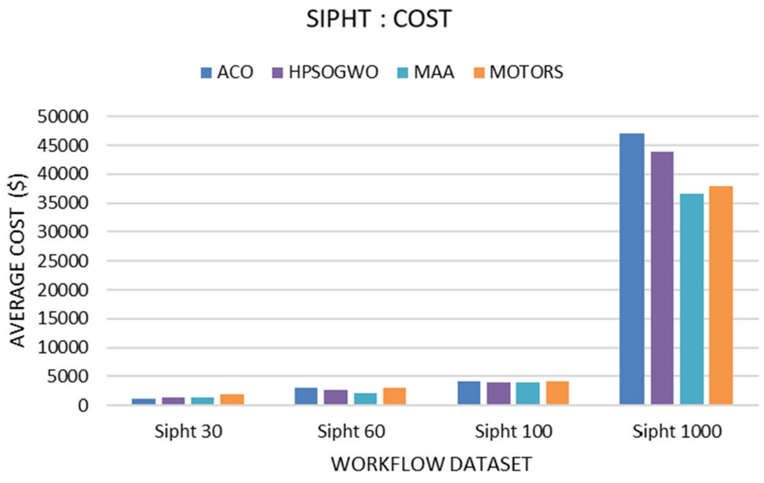


Fig. 28 Average Cost of the SIPHT workflow

against existing resource management techniques such as ACO, HPSOGWO and MAA algorithm. The proposed MOTORS algorithm effectively optimizes the task offloading and resource scheduling solution in terms of makespan, cost, resource utilization, and energy consumption. To maximize its ability to collaborate, the FL and CL have an optimal number of VMs.

In the future work, we will present an approach that support mapping the tasks on resources for communication intensive applications with more efficient outcomes. Further we will simulate our proposed method by scaling up the size of

Table 7 Makespan (in seconds) comparison of four WSA (proposed MOTORS with ACO, HPSOGWO and MAA) on benchmark workflows datasets, and difference percentage between Makespan of ACO, HPSOGWO and MAA with respect to proposed MOTORS algorithm

Workflow dataset	ACO	HPSOGWO	MAA	Proposed MOTORS	Diff % ACO	Diff % HPSOGWO	Diff % MAA
Montage 20	290	210	160	**60**	383.33	250.00	166.67
Montage 40	240	340	300	**95**	152.63	257.89	215.79
Montage 60	360	380	350	**133**	170.68	185.71	163.16
Montage 80	460	470	380	**163**	182.21	188.34	133.13
Montage 100	610	620	470	**203**	200.49	205.42	131.53
Montage 200	950	900	830	**647**	46.83	39.10	28.28
Montage 300	1540	1510	1130	**1108**	38.99	36.28	1.99
Montage 1000	6921	6716	6916	**3633**	90.50	84.86	90.37
CyberShake 30	70,430	64,400	60,280	**39,149**	79.90	64.50	53.98
CyberShake 50	90,500	95,630	72,620	**39,078**	131.59	144.72	85.83
CyberShake 100	100,790	102,130	90,790	**40,818**	146.93	150.21	122.43
CyberShake 1000	139,991	130,754	130,722	**41,699**	235.72	213.57	213.49
Epigenomics 24	9780	8090	6960	**6756**	44.76	19.75	3.02
Epigenomics 47	14,160	14,200	10,620	**9603**	47.45	47.87	10.59
Epigenomics 100	74,710	74,930	57,900	**73,020**	2.31	2.62	*−20.71*
Epigenomics 997	383,312	363,043	310,724	**270,192**	41.87	34.36	15.00
Inspiral 30	1660	1850	1080	**1252**	32.59	47.76	−13.74
Inspiral 50	2310	2340	1510	**1985**	16.37	17.88	−23.93
Inspiral 100	3280	2890	2340	**3200**	2.50	−9.69	−26.88
Inspiral 1000	22,855	20,422	17,814	**19,716**	15.92	3.58	−9.65
Sipht 30	3920	4060	1080	**3137**	24.96	29.42	−65.57
Sipht 60	4190	4680	1510	**3245**	29.12	44.22	−53.47
Sipht 100	5360	4750	2340	**3530**	51.84	34.56	−33.71
Sipht 1000	20,532	22,454	15,905	**17,275**	18.85	29.98	−7.93

Table 8 Cost (in dollars) comparison of four WSA (proposed MOTORS with ACO, HPSOGWO and MAA) on benchmark workflows datasets, and difference percentage between Cost of ACO, HPSOGWO and MAA with respect to proposed MOTORS algorithm

Workflow dataset	ACO	HPSOGWO	MAA	Proposed MOTORS	Diff % ACO	Diff % HPSOGWO	Diff % MAA
Montage 20	239	161	190	**59**	304.54	172.51	221.60
Montage 40	273	352	329	**111**	146.61	217.98	197.20
Montage 60	424	425	552	**161**	163.01	163.63	242.41
Montage 80	537	514	500	**204**	163.13	151.86	145.00
Montage 100	688	678	533	**257**	167.83	163.94	107.49
Montage 200	1002	1009	955	**639**	56.84	57.94	49.49
Montage 300	1589	1521	1275	**1072**	48.17	41.82	18.89
Montage 1000	9437	8962	9172	**3417**	176.21	162.31	168.46
CyberShake 30	65,997	82,856	50,756	**29,664**	122.48	179.31	71.10
CyberShake 50	104,854	112,564	90,129	**45,715**	129.36	146.23	97.15
CyberShake 100	182,732	195,005	163,653	**55,901**	226.89	248.84	192.76
CyberShake 1000	620,520	431,445	425,063	**45,821**	1254.23	841.59	827.66
Epigenomics 24	8472	7563	7021	**6029**	40.51	25.43	16.44
Epigenomics 47	14,669	16,510	15,188	**10,619**	38.14	55.47	43.02
Epigenomics 100	120,313	123,156	116,871	**94,555**	27.24	30.25	23.60
Epigenomics 997	904,523	830,216	826,514	**813,211**	11.23	2.09	1.64
Inspiral 30	1664	1755	1746	**1513**	9.96	15.98	15.38
Inspiral 50	2910	2970	3115	**2630**	10.64	12.93	18.44
Inspiral 100	4941	5080	4699	**4627**	6.79	9.80	1.56
Inspiral 1000	58,865	64,470	56,437	**48,981**	20.18	31.62	15.22
Sipht 30	1164	1408	1264	**1802**	−35.41	−21.86	−29.86
Sipht 60	3084	2613	2146	**3019**	2.15	−13.45	−28.92
Sipht 100	4029	3932	3899	**4148**	−2.87	−5.21	−6.00
Sipht 1000	47,114	43,846	36,590	**37,948**	24.15	15.54	−3.58

workflows for a larger number of tasks and other types of workflows datasets. We can also include the deadline and cost constraint. We can even utilize machine learning or deep learning techniques for efficient task offloading and optimal resource scheduling in FCI.

Authors' contributions Prashant Shukla contributed to conceptualization, methodology, implementation, validation, and writing—original draft, writing—review & editing. Sudhakar Pandey contributed to resources, conceptualization, and supervision.

Funding Not applicable.

Availability of data and materials Not applicable.

Declarations

Ethical approval Not applicable.

Coflict of interest The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

References

1. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for vm-based cloudlets in mobile computing. IEEE Pervasive Comput 8(4):14–23
2. Kwon YW, Tilevich E (2012) Energy-efficient and fault-tolerant distributed mobile execution. In: 2012 IEEE 32nd International Conference on Distributed Computing Systems. IEEE, pp 586–595
3. Shukla P, Pandey S, Agarwal D (2023) An efficient offloading technique using DQN for MEC-IoT networks. In: 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, pp 1–7
4. Shukla P, Gupta S, Pandey S (2023) ECO-RL-ECA: efficient computation offloading using reinforcement learning in edge-cloud architecture. In: 2023 OITS International Conference on Information Technology (OCIT), Raipur, India, pp 605–610
5. Wang Y, Tao X, Zhang X, Zhang P, Hou YT (2019) Cooperative task offloading in three-tier mobile computing networks: an ADMM framework. IEEE Trans Veh Technol 68(3):2763–2776
6. Li Z, Zhu Q (2020) Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing. Information 11(2):83
7. Huynh LNT, Pham Q-V, Pham X-Q, Nguyen TDT, Hossain MD, Huh E-N (2020) Efficient computation offloading in multi-tier multi-access edge computing systems: a particle swarm optimization approach. Appl Sci 10(1):203
8. Liu C, Bennis M, Poor HV (2017) Latency and reliability-aware task offloading and resource allocation for mobile edge computing. In: 2017 IEEE Globecom Workshops (GC Wkshps), pp 1–7
9. Chen M, Hao Y (2018) Task Offloading for mobile edge computing in software-defined ultra-dense network. IEEE J Sel Areas Commun 36(3):587–597
10. Yang L, Zhang H, Li M, Guo J, Ji H (2018) Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans Veh Technol 67(7):6398–6409
11. Chen M-H, Liang B, Dong M (2018) Multi-user multi-task offloading and resource allocation in mobile cloud systems. IEEE Trans Wirel Commun 17(10):6790–6805
12. Rodrigo GP, Östberg PO, Elmroth E, Antypas K, Gerber R, Ramakrishnan L (2018) Towards understanding hpc users and systems: a nersc case study. J Parallel Distrib Comput 111:206–221
13. Alvarez GPR, Östberg PO, Elmroth E, Antypas K, Gerber R, Ramakrishnan L (2016) Towards understanding job heterogeneity in hpc: anersc case study. In: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE, pp 521–526

14. Mukherjee M, Kumar S, Shojafar M, Zhang Q, Mavromoustakis CX (2019) Joint task offloading and resource allocation for delay-sensitive fog networks. In: Proceedings of IEEE International Conference Communication, pp 1–7
15. Shukla P, Pandey S (2023) DE-GWO: a multi-objective workflow scheduling algorithm for heterogeneous fog-cloud environment. Arab J Sci Eng 49:4419–4444
16. Shukla P, Pandey S (2023) Energy efficient workflow scheduling algorithm for latency-sensitive applications using cloud-fog collaboration. In: 2023 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Jaipur, India, pp 252–257
17. Kosta S, Aucinas A, Hui P, Mortier R, Zhang X (2012) ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Proceedings of the IEEE INFOCOM, pp 945–953
18. Ko SW, Huang K, Kim SL, Chae H (2017) Live prefetching for mobile computation offloading. IEEE Trans Wirel Commun 16(5):3057–3071
19. Wu Y, He Y, Qian LP, Huang J, Shen X (2018) Optimal resource allocations for mobile data offloading via dual-connectivity. IEEE Trans Mobile Comput 17(10):2349–2365
20. Jiao L, Yin H, Huang H, Guo D, Lyu Y (2018) Computation offloading for multi-user mobile edge computing. In: Proceedings of the IEEE 20th International Conference High Performance Computing and Communications. IEEE 16th International Conference on Smart City/IEEE 4th International Conference on Data Science Systems, pp 422–429
21. Yu S, Langar R, Fu X, Wang L, Han Z (2018) Computation offloading with data caching enhancement for mobile edge computing. IEEE Trans Veh Technol 67(11):11098–11112
22. Du J, Zhao L, Feng J, Chu X (2018) Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. IEEE Trans Commun 66(4):1594–1608
23. Chen M, Dong M, Liang B (2016) Joint offloading decision and resource allocation for mobile cloud with computing access point. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, pp 3516–3520
24. Chen M-H, Liang B, Dong M (2015) A semidefinite relaxation approach to mobile cloud offloading with computing access point. In: Proceedings of the IEEE 16th International Workshop Signal Processing Advances in Wireless Communications, pp 1–5
25. Liu Z, Yang X, Yang Y, Wang K, Mao G (2019) DATS: Dispersive stable task scheduling in heterogeneous fog networks. IEEE Internet Things J 6(2):3423–3436
26. Liu J, Zhang Q (2018) Offloading schemes in mobile edge computing for ultra-reliable low latency communications. IEEE Access 6:12825–12837
27. Liu J, Zhang Q (2019) Code-partitioning offloading schemes in mobile edge computing for augmented reality. IEEE Access 7:11222–11236
28. Shih Y-Y, Chung W-H, Pang A-C, Chiu T-C, Wei H-Y (2017) Enabling low-latency applications in fog-radio access networks. IEEE Netw 31(1):52–58
29. Mukherjee M, Liu Y, Lloret J, Guo L, Matam R, Aazam M (2018) Transmission and latency-aware load balancing for fog radio access networks. In: Proceedings of the IEEE global Communications Conference, pp 1–6
30. Xiao Y, Krunz M (2017) QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. In: Proceedings of the IEEE Conference on Computer Communications, pp 1–9
31. Mukherjee M et al (2019) Task data offloading and resource allocation in fog computing with multi-task delay guarantee. IEEE Access 7:152911–152918
32. Liu C-F, Bennis M, Debbah M, Poor HV (2019) Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. IEEE Trans Commun 67(6):4132–4150
33. Choudhari T, Moh M, Moh T-S (2018) Prioritized task scheduling in fog computing. In: Proceedings of the ACMSE 2018 Conference. ACM, p 22
34. Nguyen BM, Thi Thanh Binh H, Do Son B et al (2019) Evolutionary algorithms to optimize task scheduling problem for the iot based bagof-tasks application in cloud–fog computing environment. Appl Sci 9(9):1730
35. Rafique H, Shah MA, Islam SU, Maqsood T, Khan S, Maple C (2019) A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing. IEEE Access 7:115760–115773
36. Li X, Wan J, Dai H-N, Imran M, Xia M, Celesti A (2019) A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. IEEE Trans Ind Inf 15:4225–4234

37. Li L, Guan Q, Jin L, Guo M (2019) Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system. IEEE Access 7:9912–9925
38. Nikoui S, Balador A, Rahmani AM, Bakhshi Z (2020) Costaware task scheduling in fog-cloud environment. In: 2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST). IEEE, pp 1–8
39. Taneja M, Davy A (2017) Resource aware placement of iot application modules in fog-cloud computing paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, pp 1222–1228
40. Mebrek A, Merghem-Boulahia L, Esseghir M (2017) Efficient green solution for a balanced energy consumption and delay in the iot-fog cloud computing. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA). IEEE, pp 1–4
41. Guo S, Peters L, Surmann H (1996) Design and application of an analog fuzzy logic controller. IEEE Trans Fuzzy Syst 4(4):429–438
42. Ali HS, Rout RR, Parimi P, Das SK (2021) Real-time task scheduling in fog-cloud computing framework for IoT applications: a fuzzy logic based approach. In: 2021 International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, pp 556–564
43. Healy P, Nikolov NS (2001) How to layer a directed acyclic graph. In: International Symposium on Graph Drawing, Springer, pp 16–30
44. Yassa S, Sublime J, Chelouah R, Kadima H, Jo G-S, Granado B (2013) A genetic algorithm for multi-objective optimisation in workflow scheduling with hard constraints. Int J Metaheuristics 2(4):415–433
45. Yassa S, Chelouah R, Kadima H, Granado B (2013) Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Sci World J. 2013:350934
46. Mokni M, Hajlaoui JE, Brahmi Z (2018) MAS-based approach for scheduling intensive workflows in cloud computing. In: 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, IEEE, pp 15–20
47. Mokni M, Yassa S, Hajlaoui JE, Chelouah R, Omri MN (2021) Cooperative agents-based approach for workflow scheduling on fog-cloud computing. J Ambient Intell Humaniz Comput, 1–20
48. Wu CG, Li W, Wang L, Zomaya AY (2021) An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing. Futur Gener Comput Syst 117:498–509
49. Ajwani D, Cosgaya-Lozano A, Zeh N (2012) A topological sorting algorithm for large graphs. J Exp Algorithmics 17:1–3
50. Hassen FB, Brahmi Z, Toumi H (2016) VM placement algorithm based on recruitment process within ant colonies. In: 2016 international conference on digital economy, ICDEc. IEEE, pp 1–7
51. Bharathi S, Chervenak A, Deelman E, Mehta G, Su M-H, Vahi K (2008) Characterization of scientific workflows. In: Proceedings of the 3rd Workshop Workflows Support Large-Scale Science, pp 1–10
52. The Pegasus Website. Accessed: Jan. 10, 2022. Available: https://pegasus.isi.edu/
53. Subramoney D, Nyirenda CN (2022) Multi-swarm PSO algorithm for static workflow scheduling in cloud-fog environments. IEEE Access 10:117199–117214
54. Chen W, Deelman E (2012) WorkflowSim: a toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th International Conference on E-Science, pp 1–8
55. Shukla P, Pandey S, Hatwar P, Pant A (2023) FAT-ETO: Fuzzy-AHP-TOPSIS-based efficient task offloading algorithm for scientific workflows in heterogeneous fog–cloud environment. In: Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, pp 1–15
56. Hussein MK, Mousa MH (2020) Efficient task offloading for IoT-based applications in fog computing using ant colony optimization. IEEE Access 8:37191–37201
57. Arora N, Banyal RK (2020) HPSOGWO: a hybrid algorithm for scientific workflow scheduling in cloud computing. Int J Adv Comput Sci Appl 11(10):626–35
58. Shukla P, Pandey S (2023) MAA: multi-objective artificial algae algorithm for workflow scheduling in heterogeneous fog-cloud environment. J Supercomput 79:11218–60
59. Subramoney D, Nyirenda CN (2020) A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments. In: Proceedings of IEEE International Symposium on Computational Intelligence (SSCI), pp 760–767

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Prashant Shukla[1] · **Sudhakar Pandey**[1]

✉ Prashant Shukla
pshukla.phd2019.it@nitrr.ac.in

[1] Department of Information Technology, National Institute of Technology, Raipur, Chhattisgarh, India