Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Decentralized and scalable hybrid scheduling-clustering method for real-time applications in volatile and dynamic Fog-Cloud Environments

Masoumeh Hajvali[1], Sahar Adabi[2*], Ali Rezaee[1] and Mehdi Hosseinzadeh[3,4]

Abstract

A major challenge in Cloud-Fog settings is the scheduling of workflow applications with time constraints as the environment is highly volatile and dynamic. Furthermore, adding the complexities of handling IoT nodes, as the major owners of the workflow requests, renders the problem space even harder to address. This paper presents a hybrid scheduling-clustering method for addressing this challenge. The proposed lightweight, decentralized, and dynamic clustering algorithm is based on fuzzy inference with intrinsic support for mobility to form stable and well-sized clusters of IoT nodes while avoiding global clustering and recurrent re-clustering. The proposed distributed method uses Cloud resources along with clusters of mobile and inert Fog nodes to schedule time-constrained workflow applications with considering a proper balance between contradicting criteria and promoting scalability and adaptability. The Velociraptor simulator (version 0.6.7) has been used to throughtly examine and compare the proposed method in real workloads with two contemporary and noteworthy methods. The evaluation results show the superiority of the proposed method as the resource utilization is about 20% better and the schedule success rate is almost 21% better compared with the two other methods. Also, other parameters such as throughput and energy consumption have been studied and reported.

Keywords Fog computing, Workflow scheduling algorithm, Clustering, Fuzzy inference system

Introduction

Cloud computing is a promising model of the Internet of Things (IoT) in data processing and business applications [1, 2]. Using Cloud computing for IoT is not an easy task due to challenges such as synchronization, standardization, balancing between services and IoT needs [2–6].

Fog computing can be used as a viable choice for IoT providers with features such as less delay, more scalability, support for user and resource mobility, better performance for real-time interactive services, and data processing [7–9]. However, Fog computing needs to solve other issues such as the reliability and mobility of analytical data on edge devices. Another important challenge in Fog computing is resource allocation [10]. In Fog computing, nodes are differently based on input data and processing speed, which leads to problems on load balancing, when some Fog nodes are overloaded and some remain idle. These issues reduce the performance of the

*Correspondence:
Sahar Adabi
sahar_adabi@iau-tnb.ac.ir
[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2] Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran
[3] Health Management and Economics Research Center, Iran University of Medical Sciences, Tehran, Iran
[4] Computer Science, University of Human Development, Sulaymaniyah, Iraq

Fog environment [11–13]. Therefore, workflow scheduling is an important issue in a Fog environment.

Workflow is commonly used in a distributed computing environment [7, 8, 13–15]. A workflow is graphically performed using a Directed Acyclic Graph (DAG) to show the interdependence between workflow tasks, where nodes represent computational tasks on available resources and guided edges represent data flow dependency [8, 13–15]. In general, the runtime of a task in a particular resource is inversely proportional to the speed of that resource; therefore, optimal mapping of workflow tasks to computing resources available in Cloud/Fog is essential [16, 17].

In this paper, a distributed scheduling architecture for Cloud/Fog-based systems is presented. The proposed Cloud/Fog based scheduling management architecture has three hierarchical layers. The first layer is the IoT device layer. The second layer, has two sub-layers of Fog nodes and smart gateways, which are responsible for managing requests. This layer has a scheduling management system that receives all user requests, manages resources, processing and communication costs, and ultimately creates the most appropriate scheduling available. The third layer, the Cloud layer, hosts some computing nodes and provides external resources for the middle layer. In the proposed architecture, resources are mobile in the system, meaning that they can be in different places at different times. Also, resources have limited energy consumption and bandwidth. Therefore, computational resources should be clustered [18, 19].

In this paper, a new multi-criteria clustering algorithm is presented to manage resources in the Fog layer. The use of clustering algorithms in Cloud/Fog has been considered due to advantages such as request management, increased data transfer speed and energy efficiency. According to studies [18–22] in the field of clustering, the mobility of fog nodes and their stability in creating optimal clusters is very important. So, a proposed clustering algorithm is designed based on the rate of velocity, stability and mobility score that the fog layer nodes obtain using the fuzzy inference system. This makes the dynamic clustering algorithm more stable and efficient, and the requests are placed on closer and higher efficiency clusters.

This paper proposed a new scheduling algorithm. This algorithm includes three phases. The first phase focuses on extracting the Critical Path (*CP*) from the DAG. The CP algorithm extracts important paths based on the priority value of each task (*PRT*). The second phase is the new workflow scheduling algorithm (*SDMS* algorithm) that selects the best schedule according to the workflow deadline and definition of a new utility function. The determined deadline is crucial to improve Quality of Service (QoS) performance in the system. The third phase is task reservation.

Workflows are usually faced with a large number of tasks that must be processed anywhere. This requires that the provided scheduling methods are scalable and able to schedule tasks within a certain deadline. In this regard, many studies have been conducted in recent years on these methods due to the many advantages that cloud/fog environments have. However, there are shortcomings in this field, such as:

- Most of them are focused on unit workflow scheduling in Cloud or Fog environment [7, 8]. If the Cloud/Fog environment is considered at the same time, the benefits of these layers can be achieved.
- Since Fog nodes usually have mobility and the movement characteristics of nodes may lead to continuous changes in the network topology, it is very important to consider mobility. Many of the previous works have not considered it [8–13].
- Also, mobile sources in the Fog layer have limited energy and bandwidth. Therefore, clustering is one of the most effective methods to optimize energy consumption and resource management, a case that has not been addressed in many papers [23–26].
- The critical path is the longest path from the current task to an output task or an input task. Tasks on a critical path have priority over tasks on non-critical paths. Critical path helps to establish and control the schedule and minimize delays. This has been considered in few papers [25–29].

The above-mentioned challenges formed the main motivations of this research and consequently, the contributions of this paper are as follows:

- Providing a three-tier hierarchical architecture for Cloud/Fog-based systems. The proposed architecture introduces a sub-layer in the Fog to provide a scheduling management system with resource mobility.
- Extracting a Critical Path extraction method according to the priority of workflow tasks, which helps to create more accurate schedules.
- Considering the mobility behavior of the Fog layer nodes and calculating the mobility score of the Fog nodes using a multi-criteria fuzzy method and introducing a new dynamic clustering algorithm.
- Proposing a new workflow scheduling algorithm, which achieves the best scheduling using a new utility function.

The continuation of the paper is as follows: In the second section, related work is presented. In the third

section, the scheduling management system architecture is presented and in the fourth section, the scheduling model and related algorithms are presented. The fifth section deals with the evaluation and the sixth section with conclusions.

Related work

Workflow scheduling, which aims to meet different needs, has been extensively studied in recent decades. Many algorithms are proposed to find multi-purpose workflow scheduling scenarios [23]. In general, workflow scheduling algorithms are divided into three types: (1) meta-heuristic-based techniques, (2) heuristic, and (3) meta-heuristic hybrid techniques. Many studies of workflow scheduling are performed with multiple competing goals for integrated optimization or by combining multiple goals into a single goal or limiting various other criteria [24]. The main disadvantage of this type of approach is that only one computational solution may not be able to perform the user settings correctly. Recently, some practical multi-purpose solutions for computational scheduling have been developed and received a great deal of attention [23, 24].

A new problem is how to schedule workflows with data privacy restrictions while minimizing runtime and financial costs for large data applications in the Cloud [24]. In [7], such problems are modeled as a multi-objective optimization issue, and a Privacy-Aware multi-purpose scheduling workflow algorithm called MOPA is proposed. The results are compared with two other algorithms.

Arabnejad V et. al [8] explains the issue of scheduling scientific workflow in commercial environments. In this study, workflow scheduling is used to achieve low cost with appropriate response time in Cloud environments. Workflow scheduling in Cloud environments is different from network and cluster computing environments, which are mainly used in providing flexible resources and payment charging models for each user. Therefore, scheduling workflows in the Clouds requires a different approach to mapping tasks to resources while reducing costs.

In [12], a new algorithm for planning called the efficient planning workflow is introduced by establishing a trade-off between time and cost, which includes four main steps: task selection, evaluation of the range of all types of implicitly requested samples (IRITR), evaluation of the additional budget and selection of the virtual machine. IRITR evaluation is a new concept for scheduling, which aims to determine the different types of VM samples that are more suitable for workflow execution to avoid excessive bidding and negative bidding, which leads to budget violations and deadlines respectively. Compared to previous works, the simulation results prove the effectiveness of their approach, especially when there are many different sample types.

ChoonLee Y et. al [23] deals with the issue of resource-efficient workflow scheduling. For this purpose, the Significant Maximum Reduction (MER) algorithm provides a resource productivity solution that optimizes the resource utilization of a work schedule performed by each specific scheduling algorithm. The main innovation of MER depends on identifying its "near-optimal" point between increasing and decreasing resource use. Finding such a point is very important and can lead to improvement in resource utilization, reduction of resource supply, and energy saving. Another significant contribution is the widespread use of MER.

The focus in [25] is on schedule flows in Cloud computing. They present a new algorithm which comprises the Intelligent Water Drop (IWD) algorithm to optimize Cloud workflow scheduling. Their algorithm represents a significant improvement over classic timetable scheduling algorithms.

In [26] a multi-objective optimization problem is presented with a focus on reducing the power consumption of the whole system and delaying the execution of tasks. Overcoming permutation convergence in the initial population and genetic operators are effective factors in this algorithm. The performance of the proposed algorithm has been studied by extensive experiments and the obtained results show that this algorithm performs better than its counterparts in terms of various criteria. In this regard ch-PICEA-g algorithm is presented in [27] which is an innovative multi-objective developed algorithm. In their proposed method, first logistic maps and tents are used and then the improved bodily function is used to achieve the optimal solution.

Han P et. al [28] proposes an efficient exploratory method called CMSWC to solve the workflow scheduling problem by minimizing the cost and length of the workflow. CMSWC has two stages: ranking and mapping. In addition, CMSWC combines specifically for multi-objective challenges: (1) The scheduling phase to prevent the exploration of useless resources for tasks, which significantly limits the search space. (2) A new method for selecting non-dominant solutions, combining the non-dominant fast sorting approach and change-based density estimation. Extensive experiments on real workflows show that their approach can generate better costs than several advanced approaches.

Wang B et. al [29] provides an optimization-based scheduling framework in the Cloud environment. They use the virtual machine multiple queue model and the relationships between them along with the automatic encoder to improve noise cancelation to extract service

quality characteristics. In the evaluation results, their method effectively reduces energy consumption compared to others while guaranteeing the quality of service. In [30], due to service quality limitations, the problem of workflow scheduling in multiple grid environments has been considered. To achieve this goal, they have developed a two-tier scheduling strategy that includes general-level scheduling and local-level scheduling. Heterogeneity and all types of resources available in each grid cluster is an issue that significantly affects the complexity of scheduled workflows. Their proposed system consists of a cluster monitoring unit. The program management unit, and the cluster coordination unit. the cluster coordination unit uses the method of determining the quality of resources to prioritize the candidate's resources based on the multi-criteria decision-making problem space. Software simulation is used to evaluate the performance of the proposed system with other works.

Zhou X et. al [31] has proposed the Priority and Relative Distance (EPRD) efficiency algorithm to minimize work scheduling length for priority-limited workflow programs without violating the end-to-end deadline. The algorithm consists of two processes, first creating a task priority queue and then drawing a VM corresponding to its relative distance for a task. The proposed method can effectively improve VM performance and scheduling. Extensive detailed experiments based on workflow schedules are randomly generated and in the real world show that the amount of resource reduction and programming length of the EPRD algorithm significantly exceeds the existing algorithms. In terms of scheduling, scientific flows are compact data during which a large volume of medium data is generated. In the Cloud, some valuable intermediate data is stored for reuse. This storage is done manually according to the capacity of the system. Today, more mediocre data can be stored in the cloud due to the popularity of doing scientific workflow in the workplace.

In [32], they create a data dependency graph of the data resource in scientific flows. Meanwhile, this strategy considers user access due to delay. The results show that this strategy can significantly reduce the general scientific costs of academic operations. In [33], a four-layer architecture in a Fog environment is provided for delay and load balancing in scheduling. Minimum execution time, maximum completion time, arrival time and work size are considered as input criteria. In this study, the K-means + + clustering algorithm is used to cluster nodes in the Fog layer, and an artificial neural network is used to determine the extent to which a Fog node is used. Finally, they used iFogSim for simulation and the parameters of energy consumption, load balance, response time, scheduling time and latency were used to calculate the performance of the scheduling algorithm.

Pham X-Q et. al [34] considers workflow scheduling based on a Cloud/Fog computing system. Fog layer nodes are used to run large applications in collaboration with Cloud layer nodes. This allows for a good deal of agreement between implementation time and cost. In [35], task scheduling and data placement in systems embedded in the Cloud/Fog environment are investigated. Resource management is always an important issue for system performance. To deal with the complexity of the above calculations, an effective computational solution based on equation has been proposed and validated by extensive simulation studies.

Maio M et. al [36] focuses on scheduling extreme data. Extreme data is a redefinition of big data that is distinguished by the vast amount of information that needs to be analyzed. They use Fog calculations to schedule heavy data workflows that require accurate response times. They propose a beam-based method for loading tasks in the Fog called multi-objective workflow loading. Their algorithm considers three goals: cost, response time, and reliability. The simulation results show examples of the usage of severe data with precise delay requirements. Energy saving is one of the main goals in network, Cloud, and Fog computing.

In [37], an energy-aware method based on the DVFS technique for energy saving in Fog computing is proposed. Power savings are achieved using DVFS-enabled processors in the Fog, and compared to them, they can operate at lower voltages and frequencies at idle times. In addition, the IWO-CA hybrid evolution algorithm is used to arrange tasks without violating priority constraints. According to the results in the evaluation section, significant energy savings are achieved in the calculations of programs that have a predetermined deadline. Allocating and providing resources in the Fog-Cloud environment is a challenging task, given the dynamic changes in user needs and the limited resources available in Fog computing.

In [38], they propose a deadline-based algorithm for resource allocation and provision using resource ranking and resource provisioning in a combined and hierarchical manner. In this study, dynamic changes in user behaviors are investigated. The performance of the proposed algorithms in terms of total data processing time, sample cost, and network delay, with the increasing number of program submissions, are compared with available and better algorithms. The simulation results of their algorithm in CloudSim show the better behavior of this algorithm.

In [39] a workflow scheduling algorithm based on load balancing is proposed. Based on the cloud position, the execution time of the tasks is checked and

according to a queue model in the Cloud, the response time of the whole system is reduced. Next, they present their scheduling algorithm using the shortest path of the algorithm. The simulation results show the reduction of energy consumption and the increase of resource utilization. Konjaang KJ et. al [40] have proposed a virtual machine mapping algorithm. Their method is to efficiently manage resources in the Cloud to reduce runtime and cost in order to improve the quality of services and user requirement. The evaluation results show a reduction in execution time, cost and energy consumption.

Pham X-Q et. al [41] in a new distributed computing platform, task scheduling based on Cloud/Fog systems is presented. The main purpose of their algorithm is to balance the cost and performance of runtime, and the results show that the quality of system service is improved. Azizi S et. al [42] presents the scheduling method in a Fog environment in order to reduce total energy consumption and achieve high service quality, while minimizing deadline violations. Through simulation experiments, the superiority of the proposed algorithm is shown in the number of tasks, reducing the deadline violations and optimizing energy consumption.

Rodriguez M-A et. al [43] analyzed various features of Cloud workflow scheduling algorithms. In particular, the planning model, application, and resources have been studied. Since extensive researches have been done on the field of planning in general, there are accepted and accurate classifications for these features. Some of these are planning decisions, planning goals, and optimization strategies. The software model is viewed from the point of view of workflow multiplicity, i.e. the number and type of workflows that algorithms can process. Zaman Khalid, et al. [44] presents a technique for efficient distribution of resources in all nodes in a centralized manner. Their model is established in terms of cost-effectiveness using threshold values. The proposed system reduces cloud copies and costs. Azure's $6 static solution has three nodes. The adaptive data replication management system reduces the number of copies from three to two, saves memory, and lowers the price from $6 to $4. MATLAB and other programming languages are used for simulation. Azure is compared to Date Replication and Reproducer.

In cloud computing, services play a key role. Services are well-defined and independent components. Today, the demand for using fuzzy inference as a service in the field of complex and critical systems is increasing. In such systems, along with software development, the cost of diagnosing and fixing software defects increases. Therefore, the use of formal methods, which provide a clear, concise and mathematical interpretation of the system, is very important for the design of these fuzzy systems. To achieve this goal, [45] introduces a fuzzy inference cloud service (FICS) and proposes a new discipline for its formal modeling. FICS provides fuzzy inference services to consumers. Four new formal verification tests are also provided, which allow detailed analysis of certain behavior discipline patterns in FICS. In [46], a new approach to combine partitioning, sequencing and scheduling algorithms and multi-objective optimization is presented. Their approach integrates a distributed resolution with a multi-agent system and a fuzzy inference system to deal with the uncertainty problem. They compare their approach with the main related approaches and the analysis of the obtained results validates the performance of the approach as well as its effectiveness in scheduling workflows on fog-cloud computing considering resource utilization.

In IoT-based environments, workflows are usually faced with a number of tasks that need to be processed anywhere. This requires that the provided scheduling methods are scalable and able to schedule tasks within a certain time limit. In this regard, many studies have been investigated in recent years on these methods considering the advantages of Cloud/Fog environments. However, there are gaps in this regard. Most of the studies have focused on workflow scheduling in Cloud or Fog environment [8–13]. This paper has examined both layers in order to achieve their benefits.

Also, the mobility of nodes in the Fog layer to optimally allocate tasks to resources has become an important challenge in scheduling methods. On the other hand, they have limited energy and bandwidth resources. Therefore, clustering is one of the most effective methods that simultaneously considers the mobility of nodes and their management. In this paper, a new clustering algorithm that takes these challenges into account is presented. The next gap is to consider the critical path for tasks in a workflow [30]. Scheduling tasks on the critical path is very important because the critical path helps to establish and control the schedule and minimize delays [47]. This paper calculates the priority of tasks by presenting a critical path algorithm, which helps in better and faster scheduling.

Table 1 shows the comparison between the proposed work and related works.

Scheduling management system architecture

In this paper, a workflow scheduling management system in a Cloud/Fog computing system is proposed. In these systems, due to the large volume and increasing demand for services, as well as the production of big data and their storage, the workload of Cloud servers increases [43]. Thus, it is desirable to consider the distribution management of services in the Fog layer.

Table 1 Comparison of the proposed framework with related work

Work	Target system	Scheduling Type	Minimum Makespan	Minimum Cost	Privacy	Delay	Deadline	Task priority	Energy consumption	Resource Management	Resource Mobility
[7] **(2020)**	Cloud	Workflow Scheduling	Yes	Yes	Yes	No	No	No	No	No	No
[8] **(2017)**	Cloud	Scientific Workflow Scheduling	Yes	Yes	No	No	Yes	Yes	No	No	No
[13] **(2020)**	Cloud	Workflow Scheduling	Yes	Yes	No	No	Yes	No	No	No	No
[24] **(2018)**	Cloud	Task Scheduling	Yes	Yes	No	No	No	Yes	No	No	No
[25] **(2018)**	Cloud	Workflow Scheduling	Yes	Yes	No	No	No	No	No	No	No
[26] **(2021)**	Fog	Workflow Scheduling	Yes	Yes	No	Yes	No	No	Yes	No	No
[27] **(2021)**	Cloud	Workflow Scheduling	Yes	Yes	No	No	No	No	Yes	No	No
[28] **(2021)**	Cloud	Workflow Scheduling	Yes	Yes	No	No	No	No	Yes	No	No
[29] **(2021)**	Cloud	Task Scheduling	Yes	Yes	No	No	No	No	Yes	No	No
[33] **(2019)**	Fog	Task Scheduling	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No
[34] **(2016)**	Fog	Task Scheduling	Yes	Yes	No	No	No	Yes	No	No	No
[35] **(2016)**	Fog	Task Scheduling	Yes	Yes	No	No	No	No	No	No	No
[36] **(2020)**	Fog	Task offloading	Yes	No	No	Yes	No	No	No	No	No
[37] **(2020)**	Fog	Task Scheduling	Yes	Yes	No	No	No	No	Yes	No	No
[38] **(2019)**	Fog/ Cloud	Task Scheduling	Yes	Yes	No	Yes	No	No	No	No	No
[39] **(2020)**	Cloud	Workflow Scheduling	Yes	Yes	No	No	No	No	Yes	Yes	No
[41] **(2017)**	Fog/ Cloud	Workflow Scheduling	Yes	Yes	No	No	Yes	Yes	No	No	No
[42] **(2022)**	Fog	Task Scheduling	Yes	Yes	No	No	Yes	No	Yes	Yes	No
This Work	Fog/ Cloud	Workflow Scheduling	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes

On the other hand, the QoS (e.g. delay) of end-users is important [48]. This is because the delayed response can have catastrophic consequences. For this purpose, it is studied how to distribute the workload among the Fog layer nodes at different time intervals to provide service and achieve an optimal schedule in the system. The type of received tasks and the priority of their execution are also considered. As a result, higher priority tasks need to be scheduled sooner [47, 49].

The proposed Cloud/Fog computing system has three hierarchical layers, as shown in Fig. 1. The first layer is the device layer, which includes IoT devices. These devices act as an intermediary and send user requests to higher layers. The second layer, called the Fog layer, has a set of nodes that receives and processes part of the workload of user requests. The third layer is also responsible for providing external resources to execute the workload sent from the middle layer and hosts computing machines. Because the system computing resources are distributed to Cloud and Fog nodes, there is a smart gateway in the middle layer that receives all user requests, manages resources in Cloud and Fog nodes and processing and communication costs, along with data request results from nodes. It also creates the most appropriate schedule for an input workflow [50, 51].

Execution of workflows in the middle layer is done through a workflow management system [29–31]. When implementing workflow scheduling in a Cloud/Fog environment, two issues need to be considered. The first one, known as resource provision, involves the selection and provision of computational resources that are used to perform tasks [36]. The second one is scheduling or task allocation, in which each task is scheduled on the appropriate resources. A runtime estimation component is performed using historical data, protocol data, or time series forecasting models, among other methods, to evaluate the performance

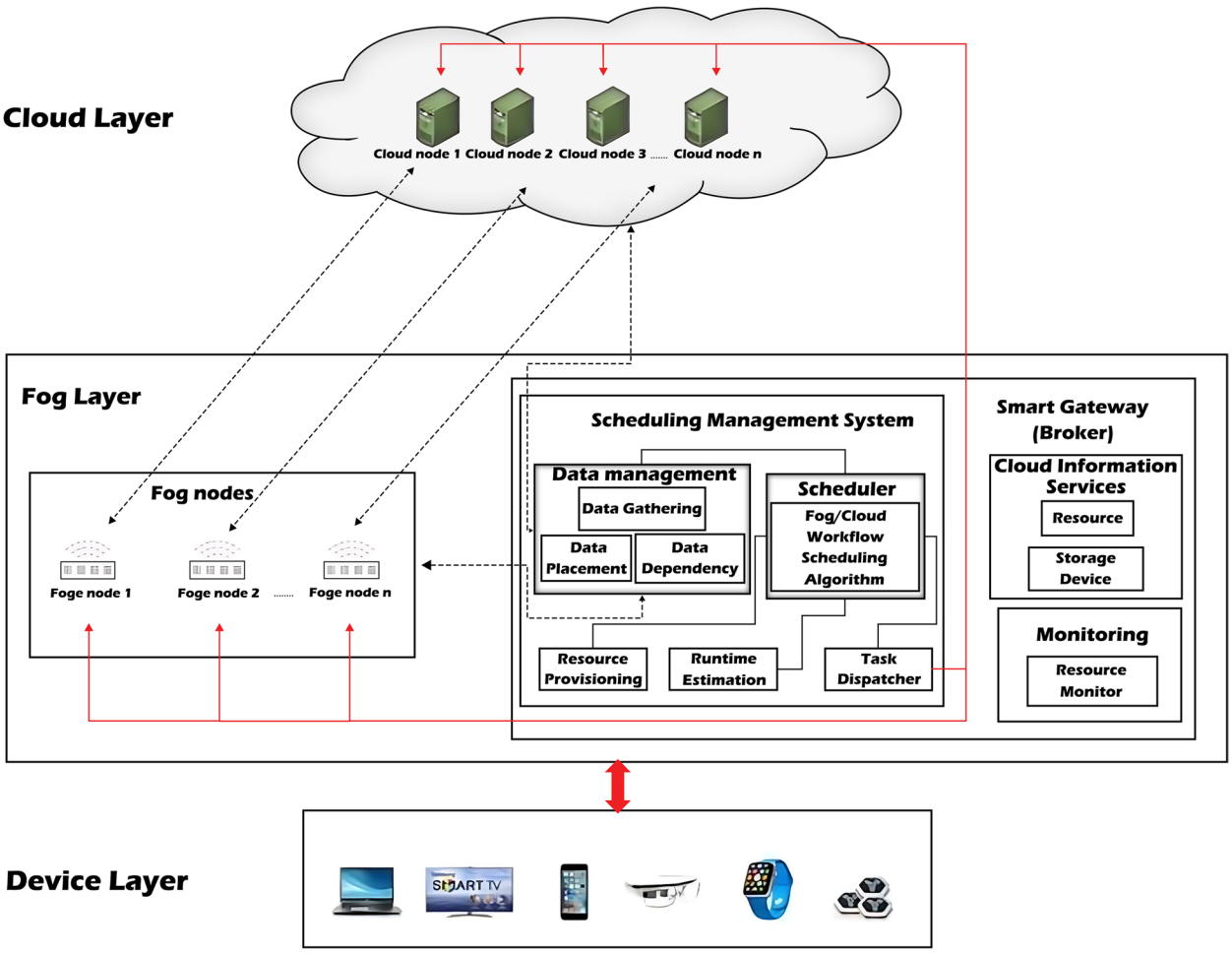


Fig. 1 General architecture of the proposed Cloud/Fog-based scheduling management system

of Cloud resources and the amount of time it takes to perform tasks in different resources [52–54]. This data is used by the resource supply component and the scheduling component to make accurate and efficient decisions about task allocation. The data management component in the workflow management system manages the motion, placement, and storage of data as needed to perform task operations. Finally, the task distributor is responsible for interacting with Cloud APIs to send tasks ready to run on existing resources. Figure 1 shows the general architecture of the proposed Cloud/Fog-based scheduling management system.

Proposed components for the Smart Gateway are explained as follows:

- **Cloud/Fog information services**

This component provides a variety of information for storage devices and resources in the Cloud/Fog, including their features and characteristics, cost of use, location, and other information needed to select resources and make decisions about optimal scheduling.

- **Scheduling management system**

The core of the system is the scheduling management system that is responsible for managing the actual execution of the workflow. It includes the following components:

- Resource provisioning

The *resource provisioning component* is responsible for selecting and supplying Cloud/Fog resources and is done according to the requirements of the scheduling system and service quality requirements.

- Runtime estimation

When a task is entered into the system, preprocessing is performed and a deadline is assigned to each task, which is provided in the system by the *runtime estimation component.*

- Task dispatcher

This component is responsible for interacting with Cloud APIs to send tasks ready to run.

- Data management

The *data management component* in the scheduling management system manages the collection, motion, placement, storage, and dependency between data when needed to perform task operations.

- Scheduler

This component performs the task of scheduling according to the information and factors obtained from other components to make accurate and efficient decisions, using scheduling algorithms.

- Monitoring

The monitoring includes components that enable dynamic and continuous monitoring of workflow tasks and resource performance, as well as resource management.

This paper aims to extend the scheduling component in this architecture, and for this purpose, the following parameters in Table 2 which are described in detail in Evaluation and Discussion section, are considered to evaluate the proposed method.

Problem model

Task graph model

A task graph is represented by a Directed Acyclic Graph (DAG), $G = (V, E, w, c)$, where V represents a set of tasks

Table 2 Performance evaluation parameters for proposed method

Performance Evaluation parameters	Description
Resource Utilization	Resource utilization in the Fog and Cloud can be defined as the amount of resource time used to perform scheduled tasks on it
Success Rate	Success rate is defined in terms of the number of successful requests in relation to the total number of tasks, and represents the fraction of tasks that have been successfully scheduled and completed before the deadline
Time related scheduling performance	
Parallelism Degree	Parallelism Degree is the number of tasks running in parallel
MakeSpan	Makespan is the amount of time from providing a workflow to completing it
Waiting Time	Waiting time is the average time it takes for a task to complete the scheduling process
Energy Consumption	Energy consumption report in each stage in the average and total mode
Packet Delivery Report	Package delivery report in three modes: Wi-Fi, cellular and lost
Throughput	Throughput rate during the scheduling process
Alive node	Number of live nodes during the scheduling process

and E represents a direct relationship between tasks. Each graph has a weight that specifies the workload of the task (number of instructions), and each edge indicated by e_{ij} is the cost of communication between the two tasks i, j (t_i, t_j), which represents the amount of communication data transmitted from the t_i task and is used as input data for the t_j task. *pred* (t_i) means tasks that are placed directly before t_i, and *succ* (t_i) refers to direct successors to t_i. If t_i has no precedent, *pred* (t_i) = 0, and if a task without any substitutes, *succ* (t_i) = 0, in which case the output task is called t_{exit}. It is assumed that a task cannot start until all its inputs are sufficiently collected [30].

Resource clustering algorithm

Clustering is one of the most effective algorithms that can be used to optimize energy consumption and management [18–21]. The basis of dynamic clustering is to create stable clusters. Considering the inherent mobility behavior of the nodes of the Fog layer and the speed and matching of the movement of these nodes, the purpose of clustering is to create more concentrated clusters with nodes close to each other in order to provide faster execution of the algorithm on the nodes of the Fog layer [22]. Creating centralized clusters with a proportional number of members makes the nodes have better local communication [23]. It also increases the throughput rate, the lifetime of the nodes, reduces the overhead of network computing, reduces the transmission delay and, as a result, delivers more optimal packets.

The purpose of this section clustering, in addition to maximizing the nodes' lifetime, is to put together nodes that are in the same condition in terms of mobility while minimizing the number of clusters. To achieve this goal, a score-based clustering algorithm is proposed. In this algorithm, each node calculates its score based on four parameters and distributes the score. The cluster head node selects the cluster member nodes (Fog nodes) based on the score received from neighboring nodes. Therefore, the cluster head determines its members in a distributed manner with the minimum computational overhead required and based on the latest information on the current condition of neighboring nodes.

Here it is assumed that the cluster heads are fixed and are assigned to a cluster according to the points they gain. The other nodes then join the clusters according to their mobility score. Since the system is dynamic and architectural elements can be entered to or exited during the execution of the system, and also due to the mobility of the nodes, the clusters are constantly changing, and as a result, the proposed clustering algorithm is defined dynamic too. Three important parameters are considered for selecting cluster heads. These parameters are fully described below. Each cluster node as well as each cluster member node calculates a value for itself using the proposed fuzzy-based method to indicate its suitability for a cluster. Table 3 shows the term descriptions and Table 4 shows messages used in the clustering algorithm.

The main structure of the proposed algorithm (Algorithm 1) for selecting the cluster head and its members has the following steps:

- *The first step* (calculating the cluster head score).

Initially, the score of each cluster head is calculated using a multi-criteria method based on the fuzzy inference method (described in Input and output variables of the fuzzy SoCH system section).

- *Step 2* (calculating the mobility score for each node).

In this step, a multifunctional method based on fuzzy inference is proposed to calculate the mobility score of Fog layer nodes. In Input and output variables of the fuzzy SoMob system section this method is fully explained.

Table 3 Term description

Term	Description
SoCH	Scored cluster head node point
SoMob	Cluster member node mobility score
BW_{CH}	Bandwidth of a cluster relative to the number of its members
RT_{CH}	Response time for messages for the cluster head node
PP_{CH}	Processing power of a cluster depends on the number of its members
V_M	Speed of each member node relative to a member of the cluster
ST_M	Stability of the position of each node of the cluster relative to the cluster head

Table 4 Message format

Message name	Message format
CHmsg	{Id_M, SoMob score, Avl list}
JOINmsg	{Id_M, SoMob score}
ACCEPTmsg	{Id_CH, size of CH, SoCH score}

- *Step 3* (creating a list of available cluster heads for each node).

At this point, each node creates a list of cluster heads in its location by sending a *"CHmsg"* message to its neighbor cluster heads.

- *Step 4* (distributing the global message).

Each node sends the message *"JOINmsg"* announcing its request to connect to the clusters in its range (available list).

- *Step 5* (selecting the cluster numbers).

At this stage, the cluster head checks the number of its members and if the number of members in that cluster has not reached the maximum, by sending the message *"ACCPETmsg"*, it selects the node as a member of the cluster and connects the node to the cluster.

Fuzzy inference system for calculating scores of the cluster head

A fuzzy decision-making controller consists of the following [30]:

1) Input and output variables. These variables are usually determined based on the knowledge of experts.
2) Fuzzification interface. This interface converts input variables into fuzzy sets.
3) Fuzz rules. The fuzzy inference process uses these rules.
4) De-fuzzification interface. This interface translates fuzzy linguistic values into a clear real number that is the output of the fuzzy inference process.

Input and output variables of the fuzzy SoCH system In this section, a fuzzy-based method for calculating the score of the cluster head and members of each cluster is presented.
This paper uses three parameters BW_{CH}, RT_{CH}, and PP_{CH} as input parameters in the fuzzy inference system (Eqs. 1, 2 and 3). The output of the method is used as a qualitative criterion for each node of the cluster head. The output variable will also be called (*SoCH*), which shows the score of each node of the cluster and the score of each cluster in general. The input variables are as follows:

$$BW_{CH} = \frac{\frac{1}{n}\sum_{i=1}^{n} \text{Bandwidth}}{\text{Number of members}} \quad (1)$$

$$RT_{CH} = \text{Delay of network(Response Time)} \quad (2)$$

$$PP_{CH} = \frac{\sum_{i=1}^{n} \text{Processing power}}{\text{Number of members}} \quad (3)$$

where BW_{CH} is the average bandwidth of cluster members, RT_{CH} is the response time, and PP_{CH} is the average processing power of cluster members.

Input: Resources in Cloud and Fog layers
Output: Clustering of resources in Cloud/Fog

```
void RCFC(Resources[0..n] array of resources) {
 1:  var N[i]  array of neighbors;
 2:  var Avl[i] array of available list;
 3:  var Time_Cluster
 4:  var CH (Id_H, Size, SoCH, List_Neighbors) // CH refers to ClusterHead, the size is the number of
members, the SoCH is score.
 5:  var member (Id_M, CH, SoMob) // the CH is the head of the cluster of each resource and SoMob is the
Score of Mobility of each member.
 6:  for (int j=0; j<count.CH; j++) {
 7:    BW = Band width/NumMembers;
 8:    RTi = ∑c∈N(i){dist(i, c)} ;
 9:    PP = ProssPow / NumMembers;
10:    Float SoCH (BW, RT, PP). //using the Fuzzy SoCH Inference System
      }
11:  for (int i=0; i<=n; i++) {
12:    STi = ∑Tt=1 TRP − TRL;
13:    Vi = 1/T ∑Tt=1 √((Xt − Xt−1)² + (Yt − Yt−1)²)
14:    Float SoMobi =FIS (ST, V, SoCH); //using the Fuzzy SoMob Inference System
      }
15:  Repeat
      {
16:  for (int i=0; i<=n; i++) {
17:      for (int j=0; j<=count.CH; j++) {
18:    if (SoMob[i] is in the range of SoCH[j]) {Avl[i] = CH[j]. Id_H;} // Due to that the score of the member
is within the score of CH, the Id_H of CH add to the available list (Avl []).
       }
19:    for (int j=0; j<count.CH; j++) {
20:       send the message "CHmsg" by CH to its neighbors (N[CH]);
21:       int k = Count (N [CH]);
22:       for (int i=0; i<=k; i++) {
23:          if (member[i] receives the message) {
24:                if (CH.Id ∈ Avl[i] ){
25:                member[i] sends a message "JOINmsg" to CH with Max(SoCH) ; }
      }
26:       if (CH.Size < Threshold_Max ) {
27:          CH sends a message "ACCEPTmsg" to member[i];
28:          CH.Size = CH.Size + 1;
29:          member[i].CH = Id_H;
      }
    }
  }
30:  Until (CH.Size=Threshold_Max) or expired (Time_Cluster); }
} //End function
```

Algorithm 1. Resource Clustering in Fog/Cloud RCFC Function()

Fuzzy inference, de-fuzzy inference, and membership functions for the fuzzy SoCH system The concept of fuzzy inference is the use of fuzzy logic to adapt a given set of inputs to a given output. For this, fuzzy logic, fuzzy rules, linguistic variables and fuzzy sets are the main members. Fuzzy variables can take values, such as low, medium or high, etc., the degree of membership that the numerical between 0 to 1 can obtain with the membership function. To express the membership function can be used a curve or linear shape. The membership functions designed for three

Table 5 Membership functions and linguistic values of the fuzzy system

	Variable	Membership function	Linguistic Value
Input variable	BW_{CH}	$\mu(x) = \begin{cases} 1 & x < 0.2 \\ \frac{0.4-x}{0.2} & x \in [0.2.0.4] \\ 0 & x > 0.4 \end{cases}$	L
		$\mu(x) = \begin{cases} 0 & x < 0.27 \\ \frac{x-0.27}{0.13} & x \in [0.27.0.4] \\ 1 & x \in [0.4.0.6] \\ \frac{0.8-x}{0.2} & x \in [0.6.0.8] \\ 0 & x > 0.8 \end{cases}$	M
		$\mu(x) = \begin{cases} 0 & x < 0.62 \\ \frac{0.9-x}{0.28} & x \in [0.62.0.9] \\ 1 & x > 0.9 \end{cases}$	H
	RT_{CH}	$\mu(x) = \begin{cases} \frac{0.4-x}{0.4} & x \in [0.0.4] \\ 0 & x > 0.4 \end{cases}$	W
		$\mu(x) = \begin{cases} 0 & x < 0.1 \\ \frac{x-0.1}{0.4} & x \in [0.1.0.5] \\ \frac{0.8-x}{0.3} & x \in [0.5.0.8] \\ 0 & x > 0.8 \end{cases}$	M
		$\mu(x) = \begin{cases} 0 & x < 0.6 \\ \frac{x-0.6}{0.4} & x \in [0.6.1] \end{cases}$	S
	PP_{CH}	$\mu(x) = \begin{cases} \frac{0.35-x}{0.35} & x \in [0.0.35] \\ 0 & x > 0.35 \end{cases}$	L
		$\mu(x) = \begin{cases} 0 & x < 0.15 \\ \frac{x-0.15}{0.35} & x \in [0.15.0.5] \\ \frac{0.72-x}{0.22} & x \in [0.5.0.72] \\ 0 & x > 0.72 \end{cases}$	M
		$\mu(x) = \begin{cases} 0 & x \in [0.0.6] \\ \frac{x-0.6}{0.4} & x > 0.6 \end{cases}$	H
Output variable	SoCH	$\mu(x) = \begin{cases} 1 & x < 0.15 \\ \frac{0.22-x}{0.07} & x \in [0.15.0.22] \\ 0 & x > 0.22 \end{cases}$	B
		$\mu(x) = \begin{cases} 0 & x < 0.18 \\ \frac{x-0.18}{0.18} & x \in [0.18.0.27] \\ 1 & x \in [0.27.0.4] \\ \frac{0.5-x}{0.1} & x \in [0.4.0.5] \\ 0 & x > 0.5 \end{cases}$	M
		$\mu(x) = \begin{cases} 0 & x < 0.35 \\ \frac{x-0.35}{0.15} & x \in [0.35.0.5] \\ 1 & x \in [0.5.0.7] \\ \frac{0.78-x}{0.08} & x \in [0.7.0.78] \\ 0 & x > 0.78 \end{cases}$	G
		$\mu(x) = \begin{cases} 0 & x < 0.58 \\ \frac{0.8-x}{0.28} & x \in [0.58.0.8] \\ 1 & x > 0.8 \end{cases}$	E

inputs as well as one output are summarized and shown in Table 5. A graphical representation of the fuzzy system membership functions is shown in Fig. 2 and the output surfaces of a fuzzy system with two inputs and one output are shown in Fig. 3 (a, b, c).

Input variables of the fuzzy method for selecting the cluster head:

1. BW_{CH}. Low(L), Medium(M) and High(H) are defined as a fuzzy set.
2. RT_{CH}. Weak(W), Moderate(M), Strong(S) are defined as a fuzzy set.
3. PP_{CH}. Low(L), Medium(M), High(H) are defined as a fuzzy set.

The fuzzy method output variable for selecting the cluster head:

1. *SoCH*. This output variable has four fuzzy sets which are defined as Bad (B), Moderate (M), Good (G), and Excellent (E).

Fuzzy rule base for the fuzzy SoCH system Existing knowledge about the problem is stored in the "*if*" language rules as shown in Table 6. These rules guide system behavior. An "*if–then*" rule can make the usual human decisions using language tags and membership functions. Each "*if–then*" rule, on the other hand, defines the behavioral dynamics of the target system.

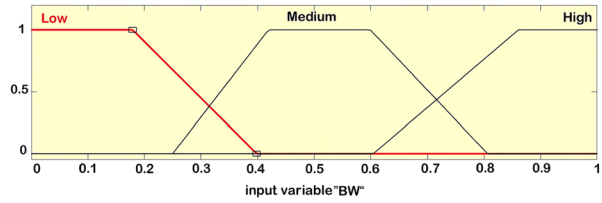
- *Rule 10*: *if* (BW_{CH} *is Moderate*) *and* (RT_{CH} *is Weak*) *and* (PP_{CH} *is Low*) *then* (*SoCH is Bad*)
- *Rule 18*: *if* (BW_{CH} *is NOT Medium*) *and* (RT_{CH} *is Strong*) *and* (PP_{CH} *is High*) *then* (*SoCH is Good*)

Input and output variables of the fuzzy SoMob system The proposed clustering algorithm for calculating the mobility score of cluster members considers Stability (ST_M) and Speed (V_M), and scores obtained from the nodes of the cluster head (calculated in Input and output variables of the fuzzy SoCH system section) where each node is given a score based on these parameters, which is the main basis for clustering by cluster heads. The input variables are as follows:

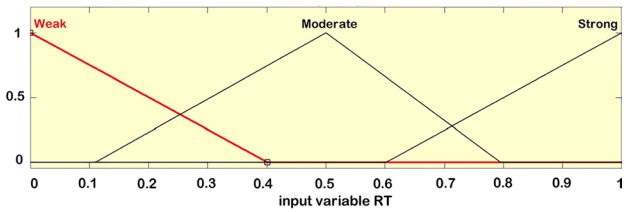
1. Speed of each node (V_M). The average velocity for each node up to the current time T is calculated by Eq. 4:

$$V_M = \frac{1}{T}\sum_{t=1}^{T}\sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2} \tag{4}$$

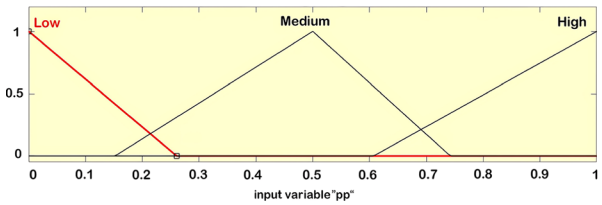
where (X_t, Y_t) and (X_{t-1}, Y_{t-1}) are the coordinates of node v at time (t) and $(t—1)$ respectively.



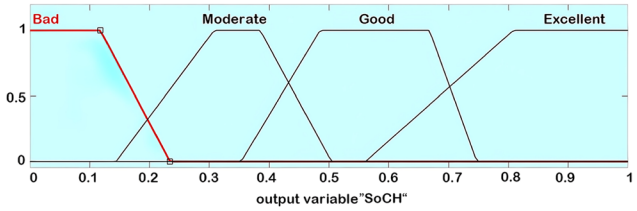
(a)Membership function for the input variable: BW_{CH}



(b)Membership function for the input variable: RT_{CH}

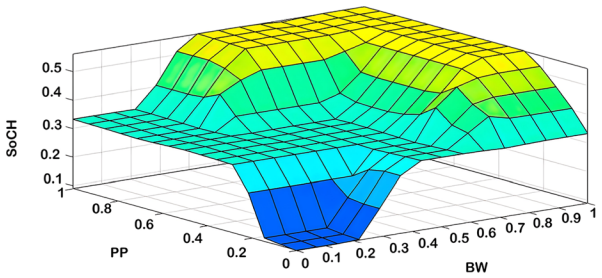


(c)Membership function for the input variable: PP_{CH}

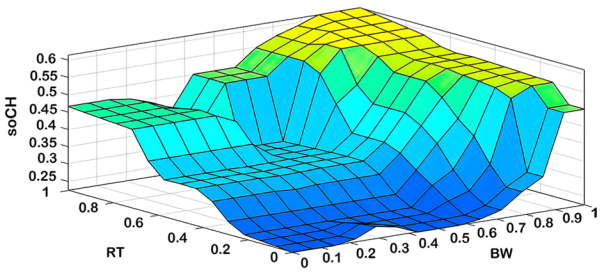


(d)Membership function for the output variable: $SoCH$

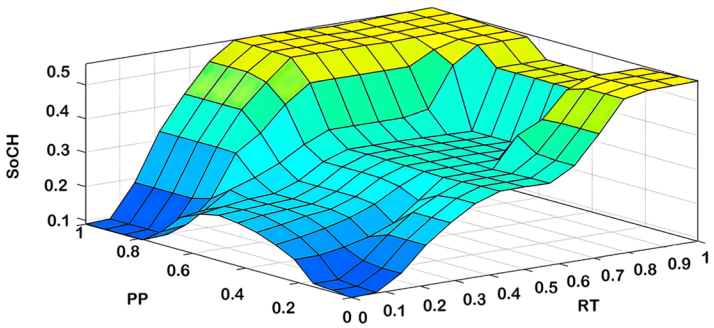
Fig. 2 Graphical presentation of the membership functions of the *fuzzy SoCH System*. **a** Membership function for the input variable: *BW*. **b** Membership function for the input variable: RT_{CH}. **c** Membership function for the input variable: PP_{CH}. **d** Membership function for the output variable: *SoCH*



(a) inputs: PP_{CH} and BW_{CH}



(b) inputs: RT_{CH} and BW_{CH}



(c) inputs: PP_{CH} and RT_{CH}

Fig. 3 Output surfaces of the *fuzzy SoCH system* with two inputs and one output. **a** inputs: PP_{CH} and BW_{CH}. **b** inputs: RT_{CH} and BW_{CH}. **c** inputs: PP_{CH} and RT_{CH}

Table 6 Fuzzy rules for the output variable to select the cluster head

Rule no	BW_{CH}	RT_{CH}	PP_{CH}	SoCH
1	L	W	L	**B**
2	L	W	M	**B**
3	L	W	H	**B**
4	L	∼W	L	**B**
5	L	M	M	**M**
6	L	M	H	**M**
7	L	S	∼M	**M**
8	L	S	M	**G**
9	∼L	S	H	**G**
10	M	W	L	**B**
11	M	W	M	**B**
12	M	W	∼L	**B**
13	M	M	L	**M**
14	M	M	M	**M**
15	M	M	H	**G**
16	M	S	L	**G**
17	∼M	S	M	**M**
18	∼M	S	H	**G**
19	∼H	W	L	**B**
20	∼H	W	M	**M**
21	H	W	H	**G**
22	H	M	L	**M**
23	H	M	M	**G**
24	H	M	∼L	**G**
25	H	S	∼H	**G**
26	H	S	M	**E**
27	H	S	H	**E**
28	H	∼S	H	**G**

2. Stability (ST_M). Stability refers to the stability of the cluster member node relative to the cluster head nodes in its range. A higher stability number indicates that a node has been in the same range for a longer time, which means that the intended node is in a more stable state. Stability is calculated by Eq. 5.

$$ST_M = \sum_{i=1}^{n} T_{RF} - T_{RL} \tag{5}$$

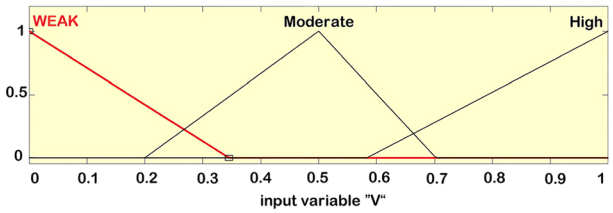
where T_{RF} is the time of the first packet received and T_{RL} is the time of the last packet received, and n represents the number of neighbors of a node.

3. Cluster head node score (*SoCH*). It is calculated in Input and output variables of the fuzzy SoCH system section.

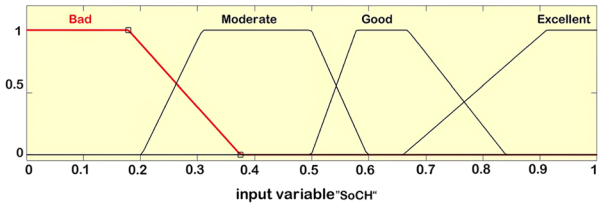
Fuzzy, de-fuzzy, and membership functions for the fuzzy SoMob system In a fuzzy model, it is important to select

Table 7 Fuzzy system membership functions and linguistic values

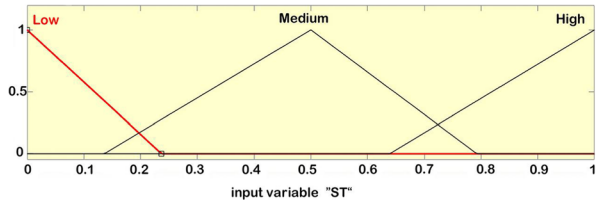
	Variable	Membership function	Linguistic Value
Input variable	V_M	$\mu(x) = \begin{cases} \frac{0.35-x}{0.35} & x \in [0.0.35] \\ 0 & x > 0.35 \end{cases}$	**W**
		$\mu(x) = \begin{cases} 0 & x < 0 \\ \frac{x-0.2}{0.3} & x \in [0.2.0.5] \\ \frac{0.8-x}{0.3} & x \in [0.5.0.8] \\ 0 & x > 0.8 \end{cases}$	**M**
		$\mu(x) = \begin{cases} \frac{x-0.6}{0.4} & x < 0.6 \\ 0 & x \in [0.6.1] \end{cases}$	**S**
	ST_M	$\mu(x) = \begin{cases} \frac{0.35-x}{0.35} & x \in [0.0.35] \\ 0 & x > 0.35 \end{cases}$	**L**
		$\mu(x) = \begin{cases} 0x < 0 \\ \frac{x-0.2}{0.3} & x \in [0.2.0.5] \\ \frac{0.8-x}{0.3} & x \in [0.5.0.8] \\ 0 & x > 0.8 \end{cases}$	**M**
		$\mu(x) = \begin{cases} \frac{0.35-x}{0.35} & x \in [0.0.35] \\ 0 & x > 0.35 \end{cases}$	**H**
	$SoCH_M$	$\mu(x) = \begin{cases} 0 & x < 0.65 \\ \frac{x-0.65}{0.17} & x \in [0.65.0.82] \\ 1 & x > 0.82 \end{cases}$	**B**
		$\mu(x) = \begin{cases} 0 & x < 0.32 \\ \frac{x-0.32}{0.18} & x \in [0.32.0.5] \\ 1 & x \in [0.5.0.7] \\ \frac{0.82-x}{0.08} & x \in [0.7.0.82] \\ 0 & x > 0.82 \end{cases}$	**M**
		$\mu(x) = \begin{cases} 0 & x < 0.32 \\ \frac{x-0.32}{0.18} & x \in [0.32.0.5] \\ 1 & x \in [0.5.0.7] \\ \frac{0.82-x}{0.08} & x \in [0.7.0.82] \\ 0 & x > 0.82 \end{cases}$	**G**
		$\mu(x) = \begin{cases} 0 & x < 0.65 \\ \frac{x-0.65}{0.17} & x \in [0.65.0.82] \\ 1 & x > 0.82 \end{cases}$	**E**
Output variable	$SoMob_M$	$\mu(x) = \begin{cases} 1 & x < 0.1 \\ \frac{x-0.1}{0.05} & x \in [0.1.0.15] \\ 0 & x > 0.15 \end{cases}$	**VL**
		$\mu(x) = \begin{cases} 0 & x < 0.08 \\ \frac{x-0.08}{0.07} & x \in [0.08.0.15] \\ 1 & x \in [0.15.0.25] \\ \frac{0.32-x}{0.07} & x \in [025.0.32] \\ 0 & x > 0.32 \end{cases}$	**L**
		$\mu(x) = \begin{cases} 0 & x < 0.2 \\ \frac{x-0.2}{0.15} & x \in [0.2.0.35] \\ 1 & x \in [0.35.0.5] \\ \frac{0.65-x}{0.15} & x \in [0.5.0.65] \\ 0 & x > 0.65 \end{cases}$	**M**
		$\mu(x) = \begin{cases} 0 & x < 0.45 \\ \frac{x-0.45}{0.15} & x \in [0.45.0.6] \\ 1 & x \in [0.6.0.75] \\ \frac{0.9-x}{0.15} & x \in [0.75.0.9] \\ 0 & x > 0.9 \end{cases}$	**H**
		$\mu(x) = \begin{cases} 0 & x < 0.75 \\ \frac{x-0.75}{0.1} & x \in [0.75.0.85] \\ 1 & x > 0.85 \end{cases}$	**VH**



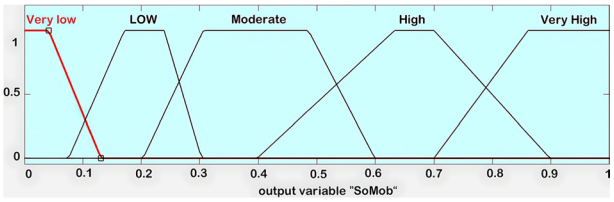
(a)Membership function for the input variable: V_M



(b)Membership function for the input variable: $SoCH_M$

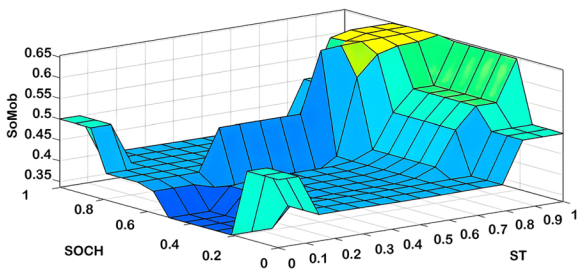


(c)Membership function for the input variable: ST_M

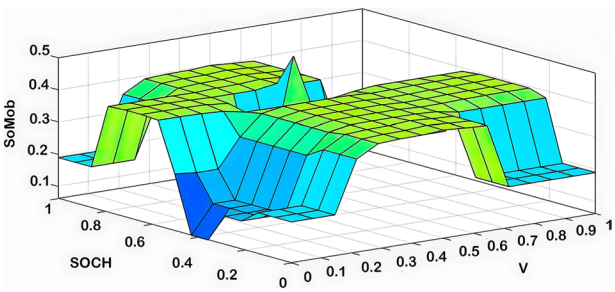


(d)Membership function for the output variable: $SoMob_M$

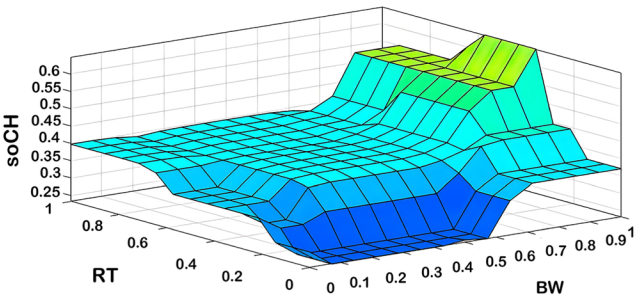
Fig. 4 Graphical presentation of the membership functions of the *Fuzzy SoMob System*. **a** Membership function for the input variable: V_M. **b** Membership function for the input variable: $SoCH_M$. **c** Membership function for the input variable: ST_M. **d** Membership function for the output variable: $SoMob_M$



(a) inputs: $SoCH_M$ and ST_M



(b) inputs: $SoCH_M$ and V_M



(c) inputs: ST_M and V_M

Fig. 5 Surface output plots of the *Fuzzy SoMob System*. (**a**) inputs: $SoCH_M$ and ST_M. **b** inputs: $SoCH_M$ and V_M. c inputs: ST_M and V_M

the appropriate language values to provide input and output variables and to select membership functions to determine the scope of the program. As explained above, the next step is to convert the explicit values of the input variables to fuzzy sets. Table 7 shows the membership functions and linguistic values of the desired fuzzy system. A graphical representation of fuzzy system membership functions is shown in Fig. 4 and surface output plots of the fuzzy SoMob system with two inputs and one output are shown in Fig. 5 (a, b, c).

Input variables of the fuzzy method for selecting a member of a cluster:

1. V_M. Three fuzzy sets are defined for this input variable: Weak(W), Moderate(M), Strong(S)
2. ST_M. Three fuzzy sets are defined for this input variable: Low(L), Medium(M), High(H)
3. *SoCH*. Four fuzzy sets are defined for this input variable: Bad(B), Moderate(M), Good(G), Excellent(E)

The output variable of the fuzzy method for selecting a cluster member:

1. *SoMob*. This output variable has five fuzzy sets which are defined as Very Low (VL), Low (L), Moderate (M), High (H), Very High (VH).

Fuzzy rule base for the fuzzy SoMob system In the next step, determine the behavior of the system using fuzzy rules is determined. The rules for calculating the output are given in Table 8.

Figure 6 shows the sequence diagram of resource clustering in the Cloud/Fog architecture.

Table 8 Fuzzy rules for the output variable to select the cluster member

Rule no	V	ST	SoCH	SoMob
1	W	L	B	**VL**
2	W	M	B	**VL**
3	W	H	~B	**M**
4	~W	L	B	**VL**
5	M	~M	B	**L**
6	M	M	B	**M**
7	~M	H	B	**M**
8	M	H	B	**M**
9	S	L	M	**M**
10	S	M	M	**M**
11	W	H	M	**M**
12	W	~L	M	**VL**
13	W	~M	M	**L**
14	~W	M	M	**L**
15	M	H	M	**H**
16	M	M	M	**M**
17	M	L	M	**L**
18	~M	M	~M	**L**
19	M	~H	M	**M**
20	W	L	G	**L**
21	W	M	G	**M**
22	~S	H	G	**H**
23	M	M	~G	**M**
24	~M	L	G	**L**
25	~S	H	G	**M**
26	S	M	G	**M**
27	W	L	G	**M**
28	~W	M	G	**H**
29	M	H	E	**H**
30	M	~L	E	**H**
31	~S	M	E	**M**
32	S	H	E	**VH**
33	S	M	E	**VH**

Scheduling algorithm

The purpose of this paper is to present a distributed scheduling algorithm for scheduling the tasks of a workflow to a set of Cloud/Fog resources. Since the workflow consists of a set of dependent tasks to ensure that priority is limited between tasks, each task must begin when all previous tasks have been completed. Also, the runtime of a schedule should not exceed the set time for it, so the purpose of the scheduling algorithm is to select the appropriate resource for assigning tasks by observing the workflow deadline, the data dependency between tasks and also resource mobility. Figure 7 shows an overview of the workflow scheduling steps that will be described below. The goal in this paper is to provide an algorithm for allocating workflow to a set of resources in the Cloud/Fog environment. This algorithm tries to select the best resource with the priority of Fog resources while guaranteeing the deadline of the workflow.

The proposed workflow scheduling algorithm consists of three phases: The first phase is to obtain a critical path to find higher priority tasks. In the next phase, according to the clustering done on the resources (described in Fuzzy inference system for calculating scores of the cluster head section), *m* clusters are selected from the available clusters for the workflow. Then, by using the exhaustive search method, all possible schedules are generated on the clusters and the best schedules are selected according to the utility function. The third phase is to reserve resources for tasks on each of the selected schedules. In this phase, if a task fails to make a reservation on a resource, the entire scheduling process fails.

- **Phase 1; Critical path extraction algorithm**

In the DAG scheduling process, one of the most important keys is identifying important tasks. This paper uses the Critical Path extraction algorithm (*CP*) to generate the critical path. This algorithm obtains the task priority (*PRT*) factor using the Multi-Criteria Decision-Making method (*MCDM*) according to the following parameters:

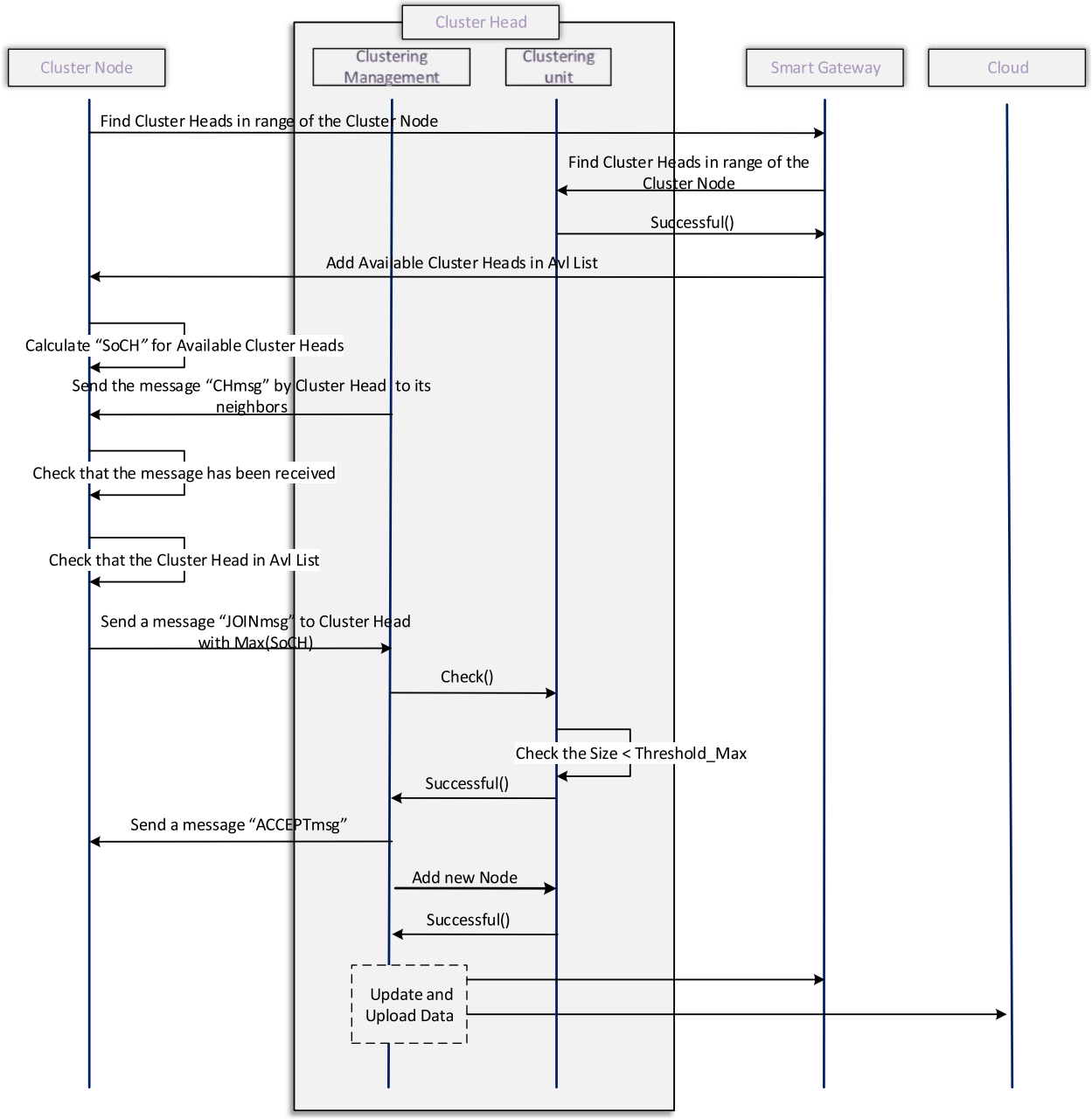


Fig. 6 Sequence diagram of the proposed clustering algorithm

• Communication Cost (CC_j^i)

The cost of communication between two tasks, t_i and t_j, is the amount of data that must be transferred between them. This value can be measured in bytes, kilobytes, etc. The higher the cost of communication between related tasks, the more important the tasks become, and therefore these highly dependent tasks can be set to the same cluster or near clusters (according to the mobility score) which are available with the fast connection link.

• Execution Cost of Task ($ECT_{t_i}^{RS}$)

This factor is the cost of performing the t_i task on all available clusters for that task, which has the type of resource required for the task and based on the estimated processor cycles required to perform the t_i task, and the average resource computing power with the type required in all clusters are calculated. Calculating the execution cost according to the processor cycles instead of the runtime makes it more accurate in estimating the runtime of each task on each resource based on the resource computing power. $ECT_{t_i}^{RS}$ is calculated by Eq. 6.

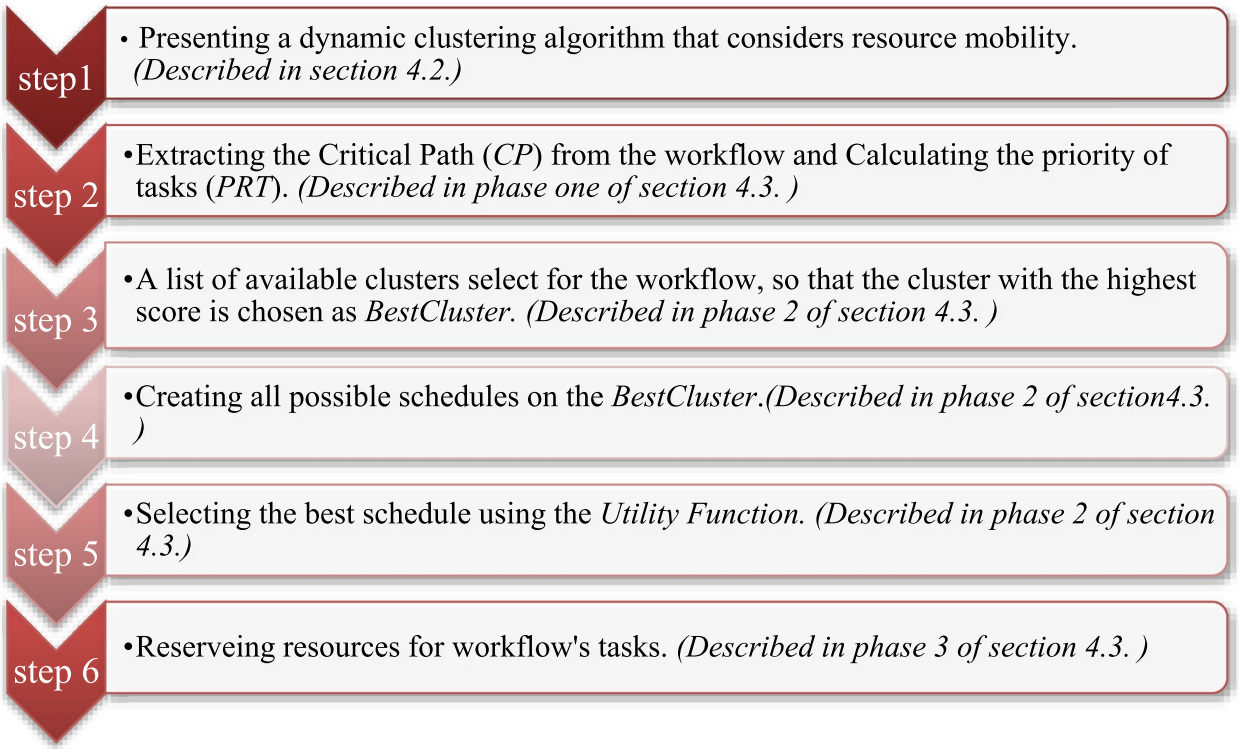


Fig. 7 An overview of the workflow scheduling steps

$$ECT_{t_i}^{RS} = \frac{t_i \cdot Length}{CCP_{RS}} \tag{6}$$

where CCP_{RS} is the average cluster computing power for resources of the type required for each task and is calculated by Eq. 7.

$$CCP_{RS} = \frac{1}{M} \sum_{R_j \in Cluster_{RS}} R_j \cdot ComputationPower \tag{7}$$

where $Cluster_{RS}$ is the set of all the resources of the type required by each task in the cluster and M is the number of resources in the $Cluster_{RS}$ set. A task becomes more critical with increasing the execution cost.

• Time Pressure of Scheduler (TPS_{t_i})

This factor indicates how much pressure the scheduler has to perform a task, and its value is between zero and one. Let $EST(t_i.RS_n)$ be defined as the task start time estimation t_i, which is estimated before the actual execution of the task on the resource. When the value of TPS_{t_i} is closer to one, the program management unit is under more pressure to schedule the task t_i. This factor is determined by Eq. 8.

$$TPS_{t_i} = 1 - \frac{(EST(t_i.RS_n) - \alpha)}{Max(\cup_{t_{i \in DAG \cdot Tasks}}(EST(t_i.RS_n) - \alpha))} \tag{8}$$

where α is the current time of the system.

Since the Cloud and Fog environment are heterogeneous and the calculation time of tasks varies from one resource to another resource, accurate calculation $EST(t_i.RS_n)$ is not possible.

In addition, factors such as bandwidth between clusters and the amount of data transfer between tasks affect the data transfer time. Therefore, for any unscheduled task, data transfer must be performed and the execution time of each task must be calculated approximately. If t_{entry} is the input node in the DAG of the application, the $EST(t_{entry}.RS_n)$ is calculated by Eq. 9.

$$EST\left(t_{entry} \,.\, RS_n\right) = \alpha - \varepsilon \tag{9}$$

where ε is the time when the scheduler needs t_i to schedule the task.

$EST(t_i.RS_n)$ is calculated by Eq. 10.

$$EST\left(t_i.RS_n\right) = Max_{t_j \in pred(t_i)} \begin{cases} Min(DL_{t_j}.\ EFT\left(t_j\ .\ RS_m\right) + \frac{CC_j^i}{BW} & if\ t_j\ is\ not\ scheduled \\ AFT\left(t_j\ .\ RS_n\right) + \frac{CC_j^i}{BW} & otherwise \end{cases} \tag{10}$$

Let $EFT\left(t_j.RS_n\right)$ be the estimated end time of t_j without considering the actual resource of the task processing, and be calculated by Eq. 11.

$$EFT\left(t_j\ .\ RS_n\right) = EST\left(t_j\ .\ RS_n\right) + AEC\left(t_j\ .\ RS_n\right) \tag{11}$$

after scheduling $EST(t_i.RS_n)$ and$EST(t_i.RS_n)$, with the actual start time of $AST\left(t_j.RS_n\right)$ and the actual end time of $AFT\left(t_j.RS_n\right)$ being replaced respectively.

• Number of Task Successors ($Num \cdot Suc_{t_i}$)

If a task has more successors, it indicates that the immediate successors of this task are waiting for their main tasks to be performed. As a result, this task usually depends on more resources for communication, and this makes it even more important.

• Source Request Rate ($SRR_{t_i}^{RS}$)

This factor is defined according to the type of resources requested for each task. $SRR_{t_i}^{RS}$ specifies the average availability of the user's requested resource type to the number of requests given to that resource. the more amount that is available compared to the number of requests in this type of source then the more critical path containing the tasks that requested this resource. This amount is calculated for all clusters containing this resource in the form of Eq. 12:

$$SRR_{t_i}^{RS_{RT}} = \frac{1}{P} \sum_{RT\ \in\ Cluster_{RS}} SAR_{t_i}^{RS_{RT}} \tag{12}$$

where P is the number of clusters that have the type of intended resource and $SAR_{t_i}^{RS_{RT}}$ is the resource rate for the type of required resource t_i that is calculated in the form of Eq. 13.

$$SAR_{t_i}^{RS_{RT}} = \frac{\min(1, RS \cdot\ Request\ Rate)}{RS \cdot Capacity} \tag{13}$$

Now the priority of each task can be calculated according to Eq. 14.

$$PRT^{t_i} = w_1 \times \left(\frac{\sum_{t_k \in DAG \cdot Succ_{t_i}} \left(((CC)^N + PRT_{DAG}^{t_k} \right)}{Max\left(2 \times Num \cdot Suc_{t_i}.\ 1\right)} \right) + w_2 \times TPS_{t_i} + w_3 \times Num \cdot Suc_{t_i} + w_4 \times \mathrm{ECT}\left(t_j\ .\ RS_n\right) + w_5 \times SRR_{t_i}^{RS_{RT}} \tag{14}$$

where w_i *(i = 1,2,3,4,5)* is the weighting factor that determines the effect of the value i-th on the priority of the task.

Algorithm 2 shows how to extract the critical path.

This algorithm first uses Depth First Search (*DFS*) to find all possible paths in the DAG (line 2). The priority of each task is then calculated if it is not scheduled. The *PRT* value is calculated according to Eq. 14 (lines 4 to 7). The algorithm then obtains the critical path from the sum of the *PRT* values of the tasks. Thus the sum of the *PRT* values of each path is calculated and *CP* selects the path with the highest *PRT* sum as the critical path. The scheduling algorithm is then called (line 18) and the result determines whether the scheduling is successful or not.

Input: Workflow
Output: Boolean (true or false)

```
boolean CP(workflow){
1: Pa[] as array of Path;
2: Pa[] = GenerateAll_DFS_Paths(workflow);
3: boolean RSA=true;
4:     for(int i = 0; i < workflow.tasks.count; i + +){
5:             Calculate PRT for each task via Eq.14 }
6:   for(int i = 0; i < Pa.count; i + +){
7:     Pa[i].SumPRT = 0;
8:   for(int j = 0; j < Pa[i].count; j + +){
9:     Pa[i].SumPRT= Pa[i].SumPRT +Pa[i].tasks[j].PRT;} }
10: Sort Pa[] in descending order by SumPRT;
11: Select the first element of the Pa[] array.
12: for(int i = 1; i < CP.tasks.count; i + +){
13:     CP + = Pa[i]; }
14: SDMS();
15: if (!RSA){
16:         RollBackUpdates();
17:         Return (false); }
18: Return(true) } //End function
```

Algorithm 2. CP Function()

The second phase; the Soft Deadline Mobility Scheduling (SDMS) algorithm

The proposed scheduling algorithm is implemented hierarchically. In the first step, a list of available clusters is selected for the workflow, and then the best clusters are selected as candidate clusters ($CL \cdot Candidates_{WF}$) to perform the workflow. These clusters are examined according to the factors of distance of the workflow to the desired cluster and the velocity vector matching of each cluster. Then, using the Multi-Criteria Decision-Making method (*MCDM*), the factor $CL \cdot Candidates_{WF}$ is calculated according to the following parameters.

- Workflow Distance to Cluster ($Distance_{WF}^{CL_i}$)

The shorter the distance from the workflow to a cluster, the better the choice. To obtain this distance, Eq. 15 can be used, which considers the relative velocity of two nodes:

$$\overrightarrow{V_{WF}} - \overrightarrow{V_{CL_i}} \tag{15}$$

so that the velocity vector of a cluster is equal to the vector sum of all resources within a cluster, which is calculated by Eq. 16.

$$\overrightarrow{V_{CL_i}} = \sum_{k=1}^{n} \overrightarrow{V_{RS}}_{RS \in Cluster_i} \tag{16}$$

where K is the number of resources within a cluster. The distance parameter is calculated according to Eq. 17.

$$Distance_{WF}^{CL_i} = \sqrt{\left(X_T^{CL_i} - X_T^{WF}\right)^2 + \left(Y_T^{CL_i} - Y_T^{WF}\right)^2} \tag{17}$$

where $(X_T^{CL_i}.Y_T^{CL_i}($ and $(X_T^{WF}.Y_T^{WF})$ are the coordinates of the cluster CL_i and WF at time T respectively.

- Cluster Velocity Vector Matching with the Workflow ($SVM_{WF}^{CL_i}$)

This parameter shows the matching of the velocity vector of a cluster with the workflow. Since the layer nodes of the device are fixed, their velocity is also considered zero.therefore, if the result of Eq. 18 is a positive number, it indicates that the cluster will be in the direction of the workflow, otherwise, there will be a mismatch. In this regard, the matching of the velocity vector of a cluster with the workflow will be equal to the size of the velocity vector of the cluster.

$$SVM_{WF}^{CL_i} = \left\{ \begin{matrix} \left|V_{CL_i}\right| & \text{if cluster matches to the workflow} \\ (-1) \times \left|V_{CL_i}\right| & \text{if cluster does not match to the workflow} \end{matrix} \right\} \tag{18}$$

According to the defined parameters, the equation for selecting the best cluster ($CL \cdot Candidates_{t_i}$) is then defined as Eq. 19.

$$CL \cdot Candidates_{WF} = w_1 \times Distance_{WF}^{CL_i} + w_2 \times SVM_{WF}^{CL_i} \tag{19}$$

that w_i *(i=1,2)* is the weighting factor that determines the effect of the value i-th on the candidate clusters.

Then the cluster with the highest score is selected as the *BestCluster* for the workflow. In the second phase of the scheduling algorithm, using the exhaustive search method, all possible scheduling for the workflow is created. At this stage, the best schedules are selected using the utility function (*ScheduleScore*). The desired workflow deadline is also selected and it will be sent to the next stage for reservation. To obtain the utility function, this paper uses the following factors to score the execution time of a workflow in parallel and serial mode.

1. The execution time of a workflow in parallel mode depends on the extent to which the tasks of a workflow can be performed in parallel mode which is defined as follows:

Job Execution Time on Most Powerful Resource with Max Parallel($ETMP^{CL_x}$)

2. If No.1 fails to be scheduled, all tasks must be performed consecutively which is defined as follows:

Job Execution Time on Most Powerful Resource with Serial($ETMS^{CL_x}$)

3. This parameter considers the state in which the workflow is executed on the best cluster that can do its work in parallel, which is defined as follows:

Job Execution Time on with Max Parallel ($ETMP^{CL_{mp}}$)

4. The last parameter is related to when the workflow is executed on the best cluster that can do its work in serial, which is defined as follows:

Job Execution Time on with Serial ($ETMS^{CL_{ms}}$)
The utility function is calculated by Eq. 20.

$$Utility\ Function = \frac{ETMP^{CL_x}}{ETMP^{CL_{mp}}} + \frac{ETMS^{CL_x}}{ETMS^{CL_{ms}}} \tag{20}$$

The SDMS algorithm is presented in Algorithm 3.

In this algorithm, the scheduler first selects the m cluster of available resource clusters for each task of all the workflow tasks (lines 1 to 7) and lists them. Next, the best clusters are selected for the workflow (lines 11 to 14), and the *EST* and *EFT* factors are calculated for each task (lines 15 to 18). In the following step, all

possible schedules are created for the workflow, which also takes into account the considered deadline for the workflow. Next, using a utility function (*ScheduleScore*), the schedules with the highest score (minimum utility function) are included in the list of the best scheduling (lines 20 to 26). Then on line 28, the *Task Reservation* function is called and the reservation result is returned. If a task can successfully reserve a resource, the *EST* and *EFT* values will be replaced by the *AST* and *AFT* values respectively (line 35).

Input: CP as path , D as deadline of workflow
Output: – (affects globally shared variables) // *result of advanced reservations*

void **SCH**(Task CP[0..*n*], Edge E[0..*m*]){
1: Cluster AvailableClusters[] =**GetAvailableClusters()**;
3: Cluster TargetClusters[0..CP.tasks.count-1];
4: **for** (int *i* = 0; *i* < CP.count; i++){
5: **for** (int *j* = 0; *j* < AvailableClusters.count; j++){
6: **if** (CP[i].RS **in** AvailableClusters[j].AvailableResourceTypes){
7: TargetClusters[i].Add(AvailableClusters[j]);
8: }
9: }
10: }
11: **for** (int i=0 ; i<= AvailableClusters.Count ; i++) {
12: $CL \cdot Candidates_{WF} = w_1 \times Distance_{WF}^{CL_i} + w_2 \times SVM_{WF}^{CL_i}$ //*Eq. (19)*
13: }
14: $CL \cdot Candidates_{WF}$ *with* ***highest*** *score*.**Add**(*BestCluster*)
15: **for** (int i=0 ; i< SCH.task.Count ; i++) {
16: $EST(t_i.RS_n) = Max_{t_j \in pred(t_i)} \begin{cases} Min(DL_{t_j}.\ EFT(t_j.RS_m) + \frac{CC_j^i}{BW} & if\ t_j\ is\ not\ scheduled \\ AFT(t_j.RS_n) + \frac{CC_j^i}{BW} & otherwise \end{cases}$
17: $EFT(t_i.RS_n) = EST(t_i.RS_n) + AEC(t_i.RS_n)$
18: }
19: Schedule SCH[] =**Generate_Schedules(CP)** *with* ***Deadline*** *aware*;
20: **for** (int j=0; j<=SCH[].count ; j++){
21: Schedule ScheduleScore = $\frac{ETMP^{CL_x}}{ETMP^{CLmp}} + \frac{ETMS^{CL_x}}{ETMS^{CLmp}}$ //*Eq. (20)*
22: }
23: **for** (int j=0; j<=SCH[].count ; j++){
24: **Sort**(SCH[].ScheduleScore,descending); }
25: **for** (int i=0; i<= P ; i++){
26: Bestscheduling[].**Add** SCH[i]; }
27: **for** (int *ix* = 0; *ix* < P; ix++){
28: *Task_Reservation*();
29: }
30: **for**(int *i* = 0; *i* < Bestscheduling.task.count; i++){
31: **if** (Resevation.task[i]==false) {
32: **RSA**= false;
33: **Return**;
34: }
35: **Update**(dag, Resevation[i].task#, Resevation[i].ClusterId, Resevation[i].AFT, Resevation[i].AST_);
36: }
} //*End function*

Algorithm 3. SDMS Function()

- **Phase 3; Task reservation**

Once the scheduling is done and the best ones have been selected, the time has come to reserve resources for the tasks. First, a schedule is selected from the list of best schedules, and a resource is reserved for a task. At any stage of the reserving process, if a task cannot be mapped to a resource, the entire schedule will fail and the algorithm consider the next schedule. Finally, if no scheduling is set, the *RSA* value that determines the scheduling result will change to *False*.

For this purpose, this paper uses the resource reservation algorithm in [30]. This algorithm receives the workflow as a set of tasks and a list of the best schedules and examines the reservation of resources on the tasks. For each best schedule, tasks are sorted in descending order, and for each task, a reservation is made. In each *repetition*, an unplanned task with the highest *PRT* is selected. If at any stage the resource reservation fails, the whole process for that schedule is canceled and the function will move on to the next schedule in the list of best schedules. If all tasks in the *STL* array are successfully scheduled, this function returns the *True* value and indicates that the entire workflow has been successfully scheduled.

Evaluation and discussion

In this section, the scheduling algorithm presented in the previous section is evaluated. Velociraptor simulator [55], which is a domain-specific simulator software for Cloud/Fog environments is used to evaluate the performance of the proposed method. The performance of the proposed method in comparison with GRP-HEFT [56] and MOODS [57] is shown and the MATLAB tool is also used to implement the proposed fuzzy inference system. The duration of each simulation was 300 time steps and the simulation process was repeated 10 times for each method. Finally, this paper reported the average of all experiments. To correctly compare the proposed method with MOODS and GRP-HEFT, the following modifications should be considered:

- In the proposed method, budget constraint is not considered. Since these two methods MOODS and GRP-HEFT have this constraint, it is not considered.
- GRP-HEFT and MOODS do not support Fog resources for scheduling, but it has been added to both.
- GRP-HEFT uses a greedy scheme.
- In the proposed algorithm and MOODS, the first fit scheme is used.
- The implementation of GRP-HEFT and MOODS in the performed experiments has some minor differences compared with the original work as it was necessary to make the implementation feasible.
- **Configuration of environment**

This paper used the information published by Facebook in the OpenCompute project https://www.opencompute.org/ to configure the cloud environment. The number of hosts in the clusters in this project is divided into 20 and the parameter "Cloud provider count" has been added to mimic a multi-cloud environment. Table 9 shows the Fog environment configuration and Table 10 the Cloud environment configuration. Processor capacity is estimated

Table 9 Configuration of Fog layer for simulation experiments

Network			**Physical hosts**		
Communication Locality	**Fog-point to WAN Bandwidth**	**Inter-Fog point Bandwidth**	**RAM(GB)**	**Processing Capacity (TIPS)**	**Processor type**
70%—75%	80–200 Mbps	unlimited	16–64	221—412	• Intel i64_Corei9 • Intel i64_Corei7
Environment Structure					
Total count of Fog hosts		**No. of Hosts in Fog-points**	**No. of Fog-Points of each Fog Provider**		No. of Fog Providers
120–299		5–9	6–8		4

Table 10 Configuration of Cloud environment for simulation experiments

Network			**Physical hosts**		
Communication Locality	**Datacenter to WAN Bandwidth**	**Inter-DS Bandwidth**	**RAM(GB)**	**Capacity (TIPS)**	**Processor type**
80%-90%	1 Tbps	unlimited	144–192	749—2,356	• Xeon 5500 • Xeon 6500 • AMD Ryzen9 • AMD Magny-Cours • AMD Threadripper
Environment Structure					
Total count of Cloud hosts		**Count of nodes in each cluster**	**No. of Clusters in each Data-center**	**No. of Data-centers owns by Cloud Providers**	**Count of Cloud Providers**
48– 288		4–8	2–3	2–4	3

by Cisco https://www.cisco.com/c/dam/global/da_dk/assets/docs/presentations/vBoo.amp_Performance_Benchmark.pdf in a million instructions per second (MIPS) and a trillion instructions per second (TIPS).

- **Data-sets**

The DS Lab. Workflow Jobs Dataset [58] has been used, which contains 100,000 DAGs created in a hybrid form based on Epigenomics and Montage scientific workflows structures with different job sizes and communication costs in the experimental studies.

An overview of the data-set configuration is given in Table 11. This paper used this data-set for input workloads to compare the performance of the proposed method and the two works GRP-HEFT and MOODS. Details of Montage and Epigenomics workflows are discussed in [59]. Figure 8 shows the Montage and Epigenomics workflows.

Notice: In each simulation round, the simulator software randomly selected 70 DAGs from the dataset and used these DAGs as the input workload.

- **Consumption of energy**

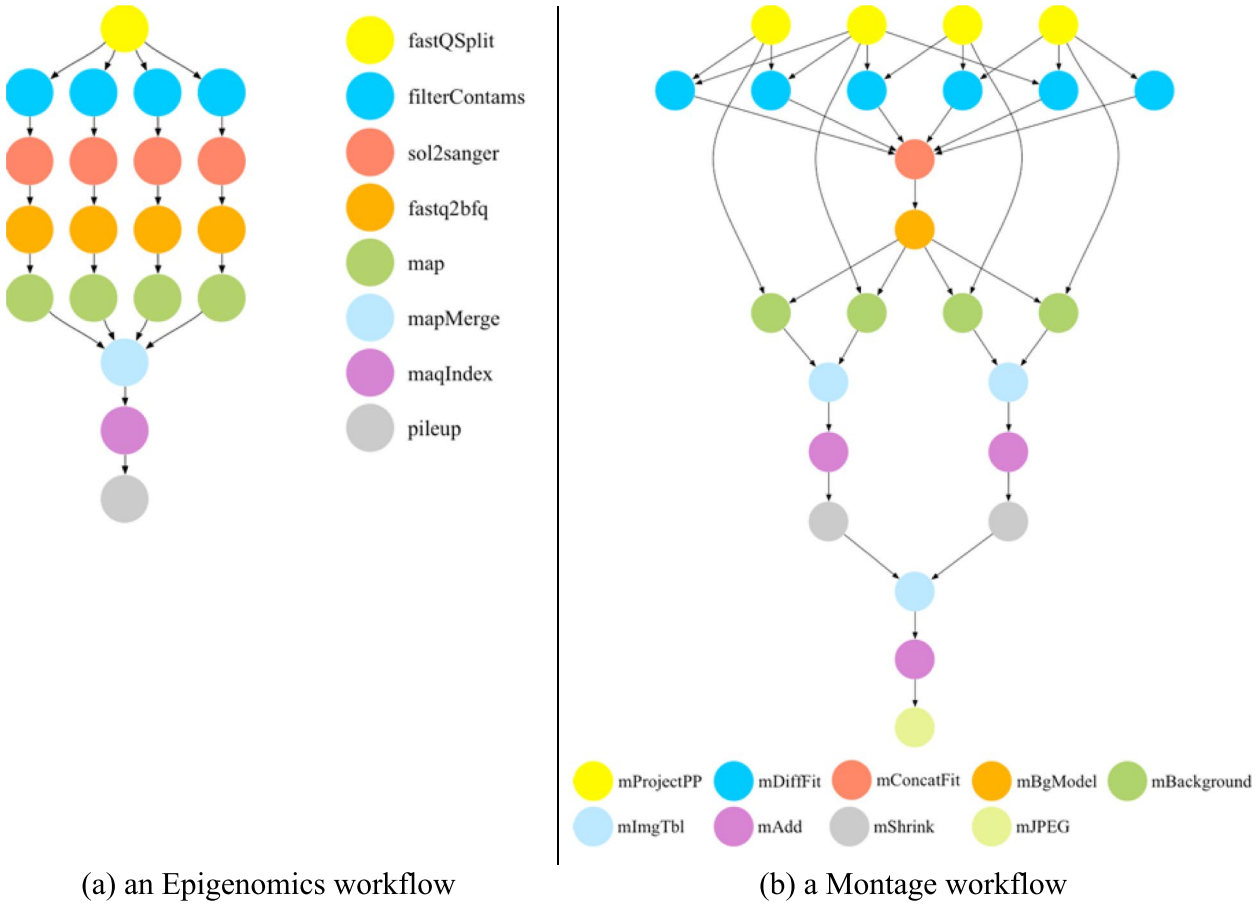
In this section, consumption of energy is used based on a simplified model in [60] for Cloud and Fog hosts. For this purpose, Eq. 21 and assumptions in [60, 61] have been used.

$$P = I + \sum_{i=0}^{N-1} \alpha_N \rho_N(i) + \sum_{J=0}^{C-1} \alpha_C \rho_C(j) + \sum_{K=0}^{D-1} \alpha_D \rho_D(k) + \psi_m \left(\sum_{j=0}^{C-1} \rho_C(j) \right) + \psi_M \left(\sum_{j=0}^{C-1} \rho_C(j) \right) \quad (21)$$

where I indicate idle power consumption. According to [60], each subsystem produces linear power consumption according to its individual utilization. In Eq. 21, the power consumption of a core is obtained by multiplying the factors (ρC), (ρD), (ρN) by a constant coefficient (αC, αD, αN) which are the core usage, disk usage and network usage, respectively. These factors do not depend on the workload of the program. In this equation, there is no separate subsystem for memory and the power consumed

Table 11 Job dataset configuration

Nodes	Edges	Count of DAGs	Task size (Billion Instructions)	Data dependency weight (Mega Bytes)	RAM requirement (Mega Bytes)
6–30	8–46	100,000	9,000—4,500	3–80	20–3100



(a) an Epigenomics workflow (b) a Montage workflow

Fig. 8 **a** an Epigenomics workflow, **b** a Montage workflow [59]

by access to memory is included in the calculations of power consumed by other subsystems. CPU instruction execution tends to highly correlate to memory accesses in most applications.

- **Fault model**

To run the simulation, a simplified fault model is considered during its execution to the environment. Information about the main probability values of the error activation model is given in Table 12, which is based on [62] and some intuitions.

Some random deletion errors are indicated by *P (NodeHasOmissionFault)* which are activated during runtime. If an error is triggered that is irreparable, the node will no longer be available and all tasks in its advanced backup list will be removed without notice. This makes it highly likely that all tasks will be scheduled or rescheduled. *P (taskFailedAndDeadlinePassed | fullFailurePermanent)* fail on other resources within

Table 12 Failure model settings

P(CloudDatacenterHostNodeHasOmissionFault)	0.01
P(fullFailureRecoverable | NodeHasOmissionFault)	0.3
P(fullFailurePermanent | NodeHasOmissionFault)	0.1
P(noFailure | NodeHasOmissionFault)	0.6
P(taskFailedAndDeadlinePassed | fullFailureRecoverable)	0.3
P(taskFailedAndDeadlinePassed | fullFailurePermanent)	0.7
P(taskFailedAndDeadlinePassed | NodeHasOmissionFault)	0
P(FogNodeHasOmissionFault)	0.05

Table 13 Physical environment specifications

Movement Model	Minimum Speed	Maximum Speed	Environment Dimension
Random Way Point	*8 m/s*	*18 m/s*	**20*km* × 20*km***

their timeframe because the cluster head loses effective window control and scheduling of failed node tasks.

- **Location and movement model of nodes**

The dimensions of the environment and the motion model of IoT nodes are given in Table 12. Random Way-Point (RWP) [63] is used as the motion model. In this section, it is assumed that if a node starts a path, according to Table 13, it immediately reaches a random speed between 8 m/s to 18 m/s and moves at this constant speed to reach the target. After a node has reach its target, it selects a new destination and a new speed.

In the first step of the simulation with a uniform distribution of a fixed position in the environment for each data centre of Cloud and Fog-point is selected, which is shown in Fig. 9 the movements of a single node in the environment. The dimensions of the environment in all experiments are considered to be 20 km × 20 km.

Figure 10 illustrates the location of datacenters in one of the simulation settings in the environment. According to Table 9, there are three cloud providers *Alabamacom, TravoSystems* and *Comtroniz*. For instance, if the *ID* of a data center in Fig. 10 is *Datacenter_0_Alabamacom*, that data center belongs to the *Alabamacom* Cloud provider (*sequence number* = *0*). These names do not belong to any real commercial or non-commercial organization. Figure 11 shown the location of Fog-points in the environment. According to Table 8, there are four Fog providers *Ney*, *Tabib*, *Aseman*, and *Peymaneh*. For instance, if the *ID* of a Fog-point in Fig. 11 is *F_9_Ney*, that Fog-point belongs to the *Ney* Fog provider company (*sequence number* = *7*). These names do

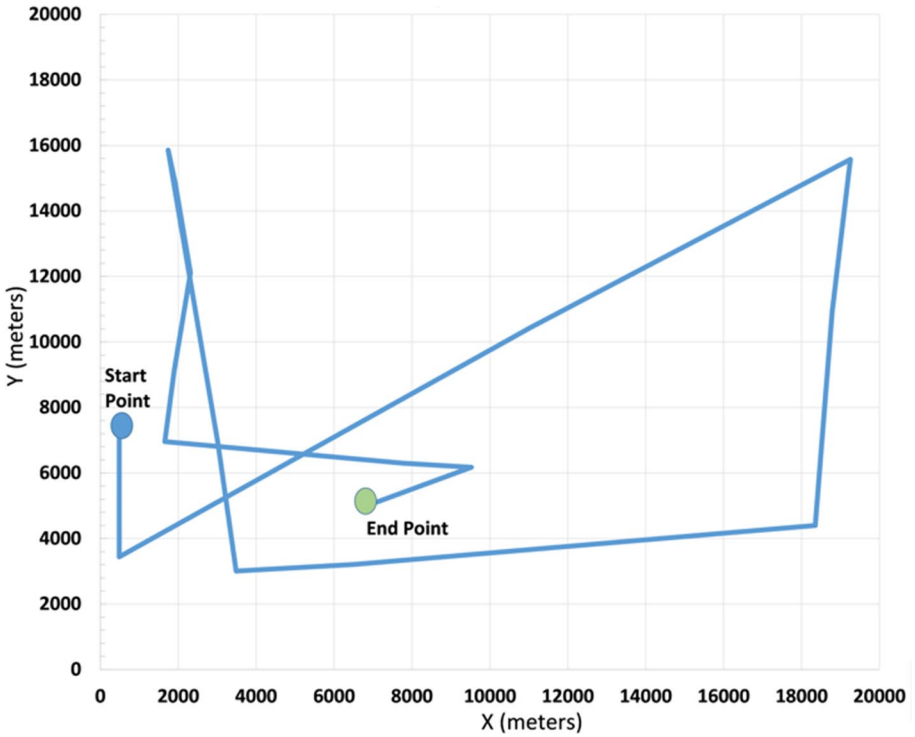


Fig. 9 An example of a node path in a 20*km* × 20*km* environment that has no predetermined path or obstacles and is implemented with the RWP algorithm

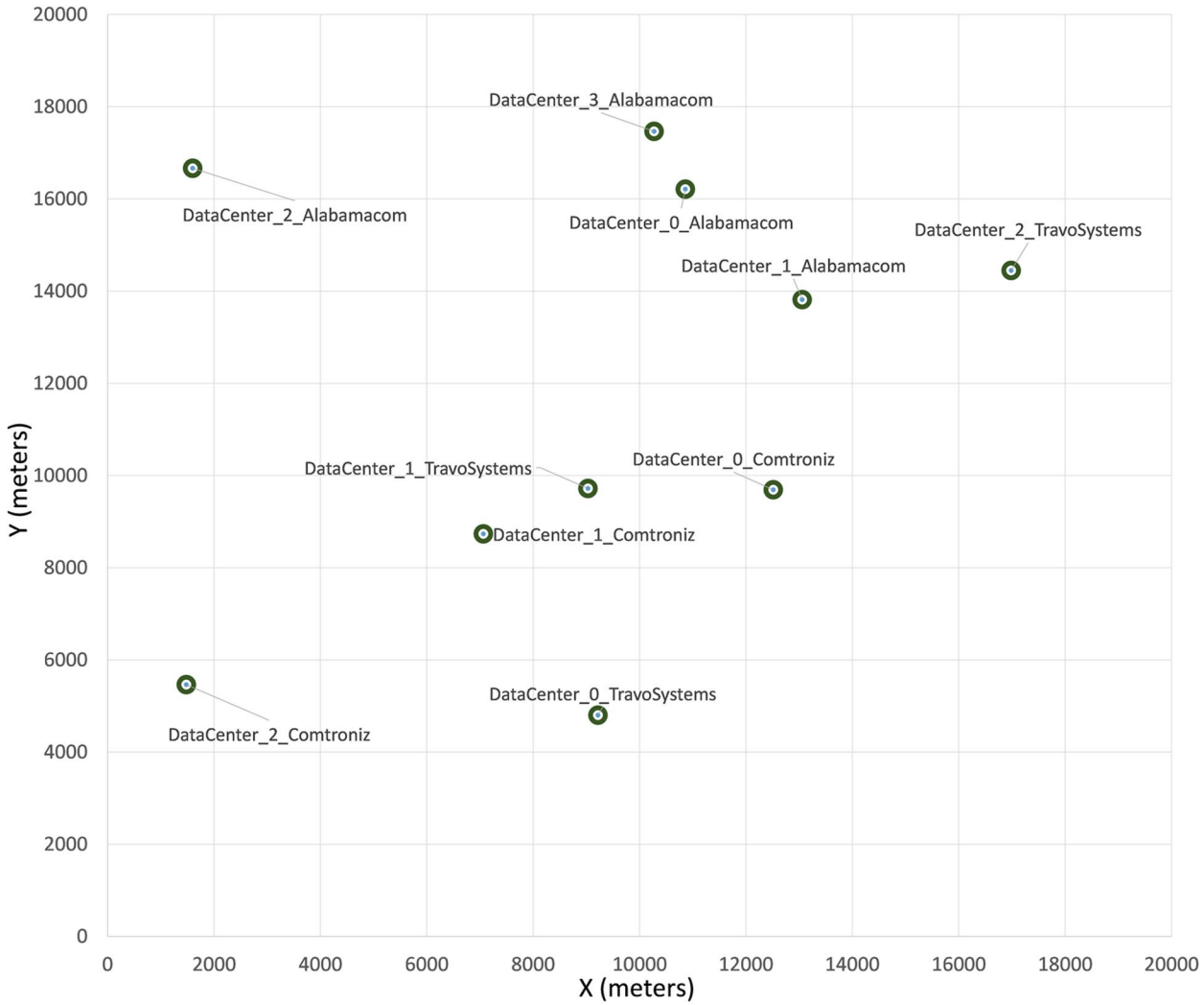


Fig. 10 Datacenters location in the environment

not belong to any real commercial or non-commercial organization.

Experimental results

In this subsection, the results of comparing the proposed method, with the GRP-HEFT [56] and MOODS [57] are reported.

Resource utilization rate and scheduling success rate are important parameters in our evaluation that affect other parameters. In order to have a better resource utilization rate, resources should be given more work, and this will increase the workload of the resources and increase the rejection rate and waiting time. On the other hand, the higher the resource utilization rate and the higher the success rate of scheduling, the more energy will be spent, but the waiting time will be shorter.

Likewise, if the number of alive nodes are many during execution and we can have clusters with closer and more stable resources, as a result, we will have better and faster data transfer, and this will cause higher throughput and a lower failure rate. But having more alive nodes will not necessarily increase the throughput, and by not having an advantage in the number of alive nodes, it is possible to have a significant throughput. It can be concluded that usually the parameters are in conflict and the better of one does not necessarily make the other better. In general, in this paper, it has been tried to put the contradicting parameters in a trade-off and as the results show, our proposed method has performed better compared to the other two methods.

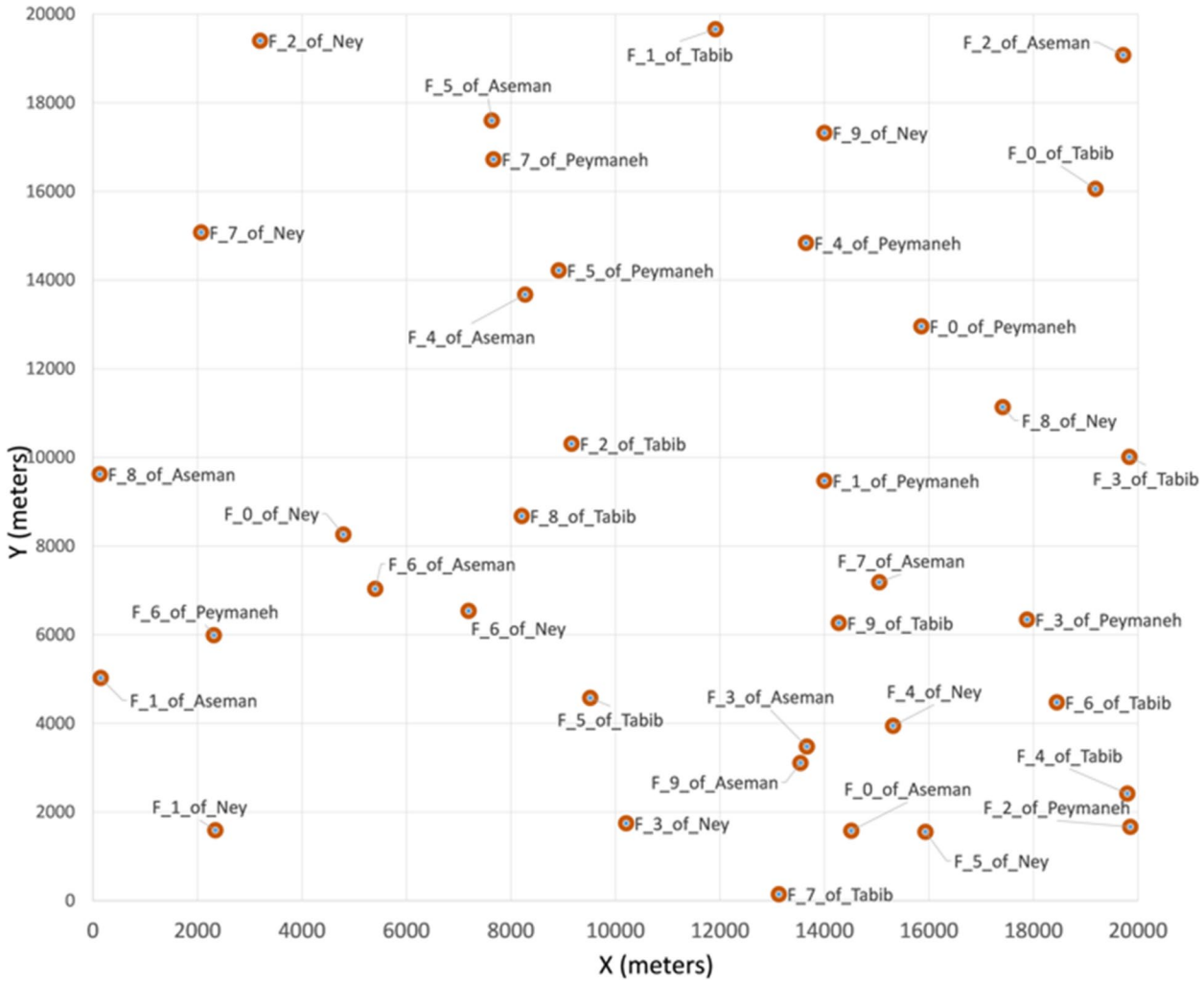


Fig. 11 Fog-points location in the environment

- **Fog/Cloud resource utilization for Scheduling performance**

Resource utilization in the Fog and Cloud can be defined as the amount of resource time used to perform scheduled tasks on it, which is defined as Eq. 22. Table 14 shows the resource utilization in the proposed method compared to GRP-HEFT and MOODS. The proposed method has a much higher resource utilization than the other two methods.

$$Resource\ Utilization = \frac{busyTime}{totalUpTime} \times 100 \tag{22}$$

Table 14 Resource utilization

Method	Average Utilization of Fog and Cloud Hosts
Proposed Method	~71%
GRP-HEFT	~58%
MOODS	~55%

Table 15 Job scheduling success rate

Method	Average Job Scheduling Success Rate
Proposed Method	~82%
GRP-HEFT	~65%
MOODS	~62%

- **Success rate for scheduling performance**

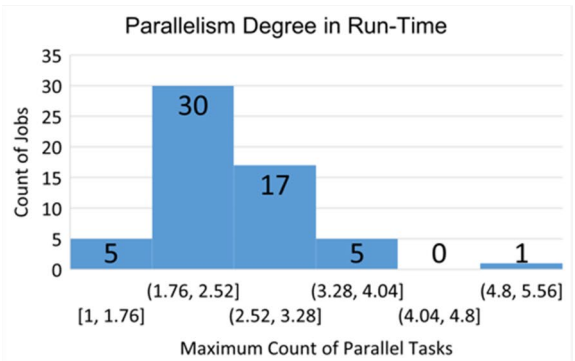
The success rate is defined in terms of the number of successful requests in relation to the total number of tasks, and represents the fraction of tasks that have been successfully scheduled and completed before the deadline. In the proposed scheduling algorithm, all scheduling is done according to the deadline and the success rate of the algorithm is due to the successful completion of the workflow before the deadline. Table 15 shows the success rate of resources in the proposed method compared to GRP-HEFT and MOODS. This rate is significantly higher in the proposed method than in the other two works, indicating that the proposed algorithm performed better.

Getting a better result in resource utilization and scheduling success rate shows that schedules are selected using the utility function that are better in terms of completion time and as a result, the selected cluster is a better cluster for workflow tasks. In fact, the better rate of resource utilization shows that the clustering algorithm has worked very successfully.

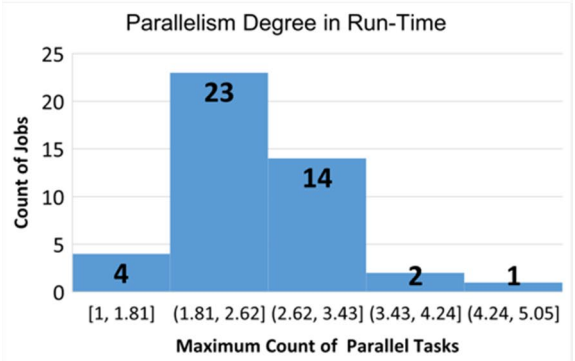
- **Time related parameters for scheduling performance**

This paper has examined the parameters degree of parallelism, makespan and waiting time as time related parameters in relation to scheduling performance.

(a)Proposed Method



(b)GRP-HEFT



(c)MOODS

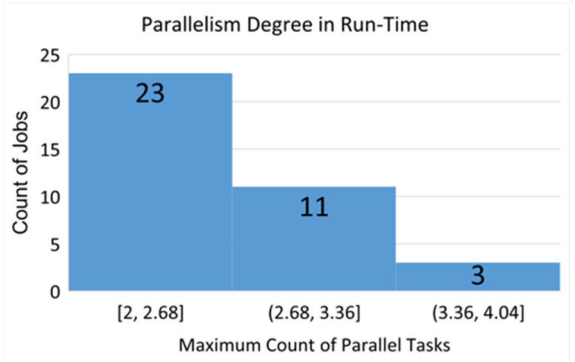


Fig. 12 Parallelism degree. **a** Proposed Method. **b** GRP-HEFT. **c** MOODS

■ Parallelism degree

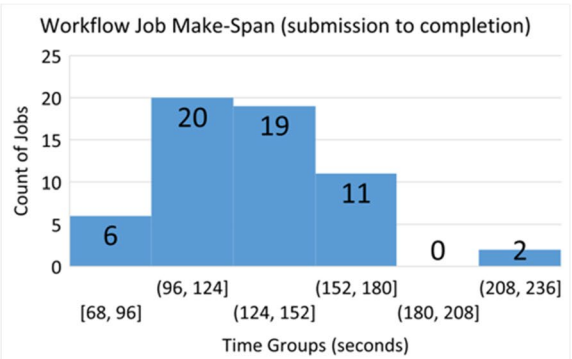
In this part, the degree of parallelism (number of tasks running in parallel) for GRP-HEFT and MOODS is compared to the proposed algorithm, which is shown in Figure 12. As can be seen in Figure 12.a, the proposed algorithm has performed better in performing tasks in parallel, and in a period of time this number has reached a maximum of 30 tasks, while in GRP-HEFT and MOODS this value eventually reaches 23 tasks.

Due to dynamic clustering and creating clusters with closer and more stable resources, it can be concluded that more tasks are executed at the same time, and this causes us to consume more energy.

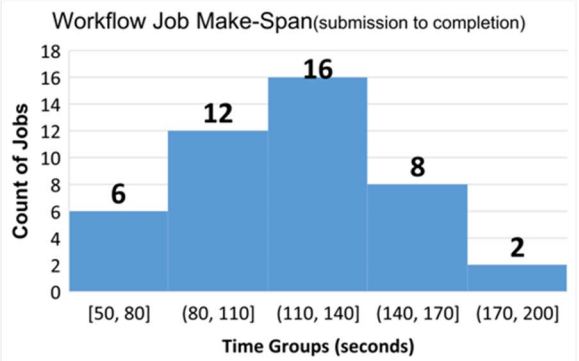
■ MakeSpan

Makespan is the amount of time from providing a workflow to completing it. Figure 13 shows a comparison between the Makespan diagram of the proposed algorithm with GRP-HEFT and MOODS. As has been seen in Figure 13(a, b, c), the proposed method and two other methods performed similarly in the initial time frame, but later the proposed scheduling algorithm performed better (Figure 13.a) and more tasks were completed. It should also be noted that all completed work has been completed before the deadline. There is no guarantee that the work will be done, but if it is done, it will be done before the deadline.

(a)Proposed Method



(b)GRP-HEFT



(c)MOODS

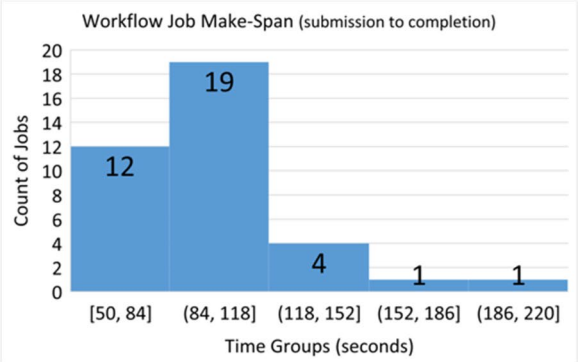
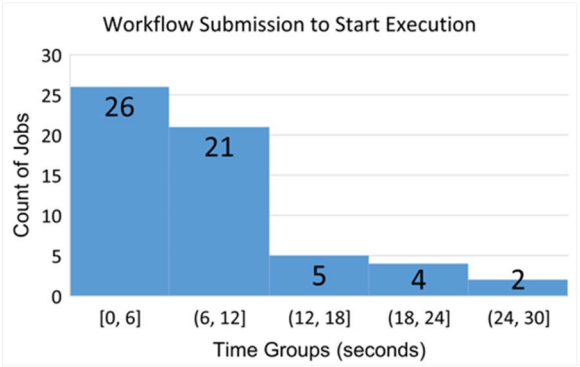
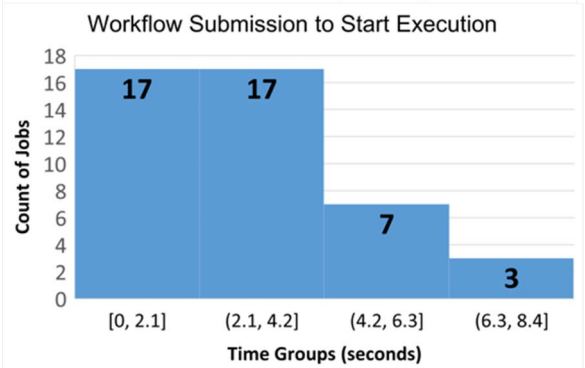


Fig. 13 MakeSpan parameter. **a** Proposed Method. **b** GRP-HEFT. **c** MOODS

(a)Proposed Method



(b)GRP-HEFT



(c)MOODS

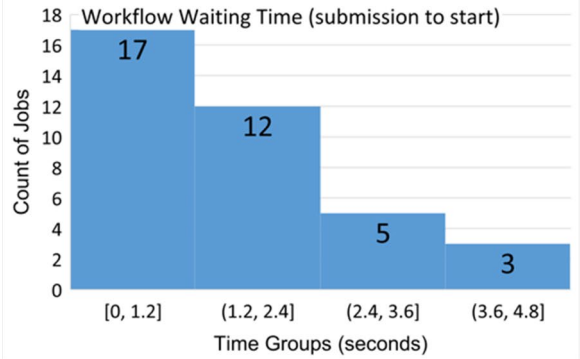


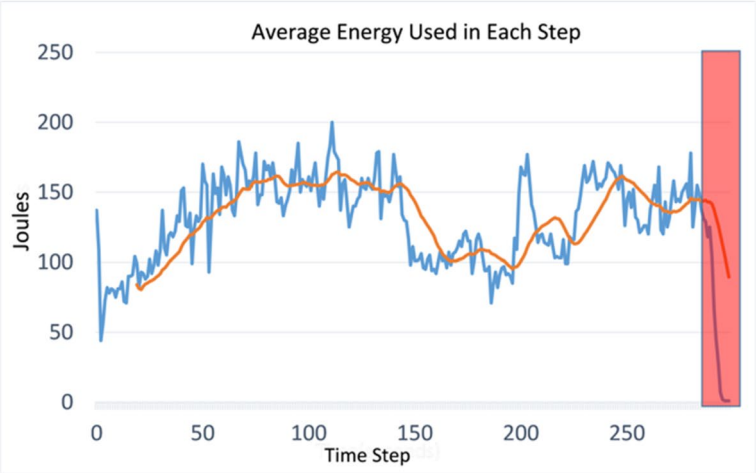
Fig. 14 Waiting Time parameter. **a** Proposed Method. **b** GRP-HEFT. **c** MOODS

■ Waiting time

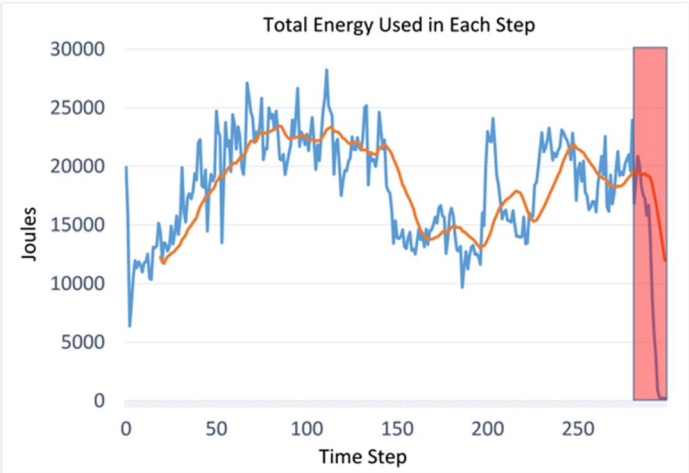
Waiting time is the average time it takes for a task to complete the scheduling process. Figure 14 shows the workflow waiting time. The waiting time for the proposed method (Figure 14.a) is longer than the other two works (Figure 14.b and c). Given that the success rate and resource utilization of the proposed method is higher, this causes more work to be done, so that naturally the waiting time will be longer.

- **Energy consumption**

The energy consumption for the proposed scheduling algorithm, MOODS and GRP-HEFT is examined, and the results are shown in Figs. 15, 16 and 17. Figure 15.a shows the average energy consumption per step by the physical Cloud /Fog host, and the diagram in Fig. 15.b shows this for the total energy consumption at each step by all physical hosts. In the proposed algorithm, the average energy consumption in some time stages reaches 200, while this number will be in GRP-HEFT and MOODS 140 and 250 respectively. The total energy consumption at each stage in some cases is higher than 28,000, while in the GRP-HEFT algorithm, this value is less than 18,000 in the usual steps. Also, in MOODS, the total energy reaches over 34,000. It must be kept in mind that the more resources used, the more energy will be consumed. As in the proposed scheduling algorithm, more tasks are successfully scheduled, so higher energy consumption values are expected to be seen.



(a) Average energy consumption in each step by a Fog/Cloud physical host

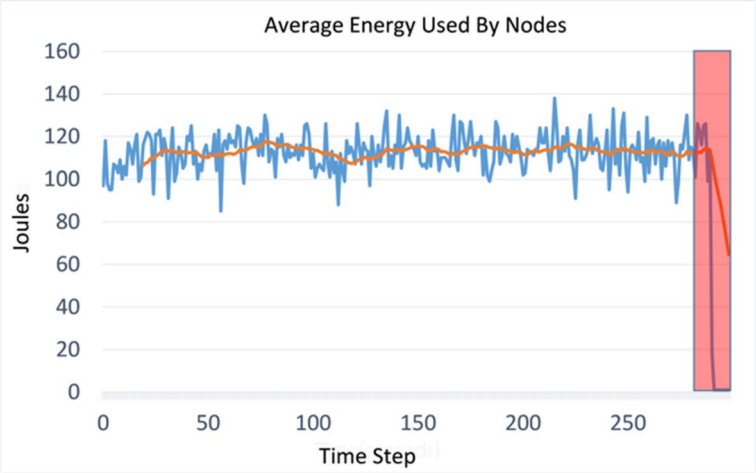


(b) Total energy consumption in each step by all physical host

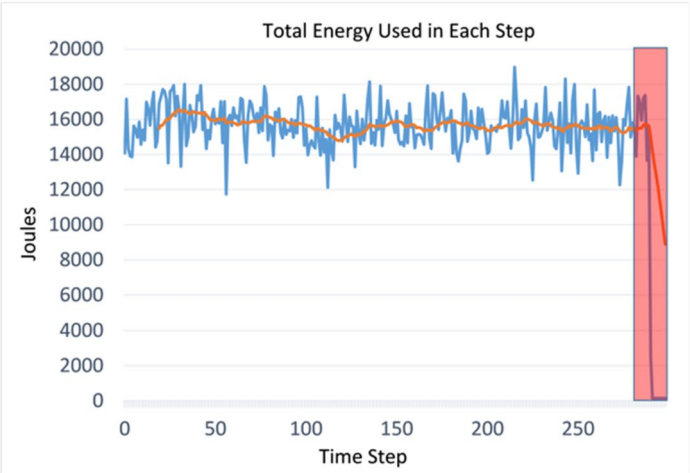
Fig. 15 Energy consumption report for the proposed Method. **a** Average energy consumption in each step by a Fog/Cloud physical host. **b** Total energy consumption in each step by all physical host

Figure 18 shows the percentage of clusters and their members compared to the proposed scheduling algorithm, GRP-HEFT and MOODS. As can be seen in Fig. 18.a, 88% of the clusters formed in the proposed algorithm have less than 11 members, and only 1% have more than 20 members, while in GRP-HEFT (Fig. 18.b) 26% of the clusters have between 15 and 20 members and 18% have more than 20 members. In MOODS, 71% of the clusters have less than 6 members, 19% have between 6 and 10 members, 9% have 11 to 20 members and 1% have more than 20 members (Fig. 18c). Figure 19 also shows the package delivery report for the proposed method and the other two methods. As can be seen in the figure, in total a larger amount of information was transmitted in the proposed method than in the other two methods, of which 72.4% were related to receiving via Wi-Fi and only 9.1% of the data were lost. These numbers in GRP-HEFT are 27.8% and 16.6% respectively. MOODS also accounted for 61.7% of its data transmission via Wi-Fi and 10.3% of its data were lost through network failure.

Figure 20 shows the throughput and Fig. 21 shows the number of alive nodes in the simulation of the proposed algorithm, GRP-HEFT and MOODS. The throughput of the proposed algorithm (Fig. 20.a) in some time stages reaches a number close to 4000 MBps, while this number is in GRP-HEFT 1400 and in MOODS 1800 MBps. This demonstrates the superiority



(a) Average energy consumption in each step by a Fog/Cloud physical host



(b) Total energy consumption in each step by all physical hosts

Fig. 16 Energy consumption report for GRP-HEFT. **a** Average energy consumption in each step by a Fog/Cloud physical host. **b** Total energy consumption in each step by all physical hosts

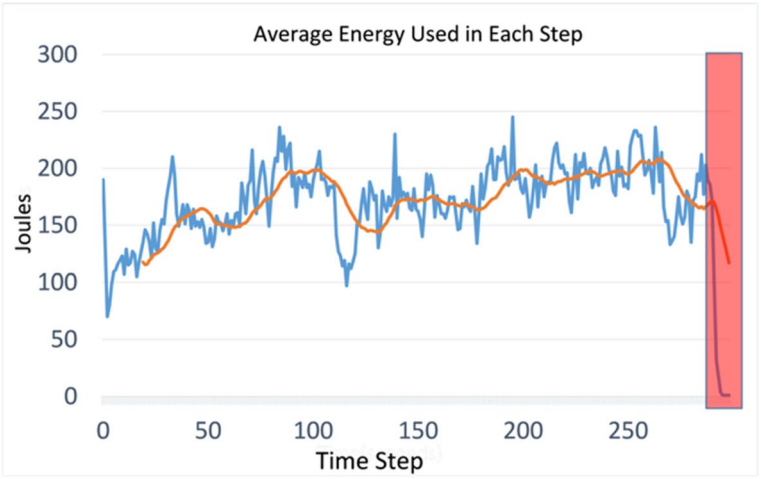
of the proposed algorithm in throughput. Since the number of alive nodes in this method is not much different from the other two methods, the reason for the high throughput rate can be considered the success of the clustering algorithm in creating more stable and optimal clusters. In the number of alive nodes, the proposed algorithm works well and the number of nodes stays alive over time and their number decreases with a lower slope.

In fact, when there are more nodes in the network, as a result, more data is sent in the network. Because in the proposed method, the scheduling is done on a cluster and also more data is sent via Wi-Fi, this makes the data transfer speed higher and at the same time the failure rate decreases. This issue also has a very positive effect on throughput.

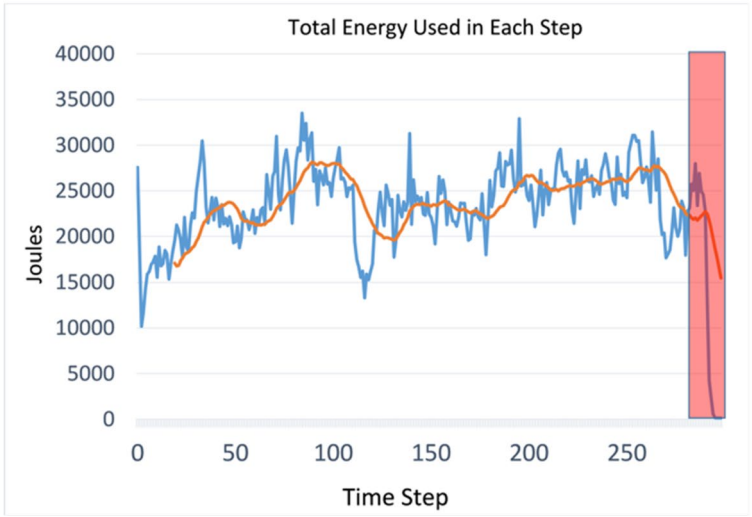
Conclusion

In this paper, a hybrid scheduling-clustering method for Cloud/Fog-based systems was proposed. The interaction of Fog computing and Cloud computing significantly increases the performance of real-time interactive applications. This Cloud/Fog computing system has three hierarchical layers. The second layer (Fog layer), which is responsible for managing requests, has a scheduling management system that manages all user requests under the heading of workflow and creates the most appropriate available scheduling for the workflow.

A decentralized and dynamic fuzzy inference based clustering algorithm was proposed that operates with the inherent mobility of nodes to form stable clusters with an appropriate number of members while avoiding global clustering. In this algorithm, first, the node

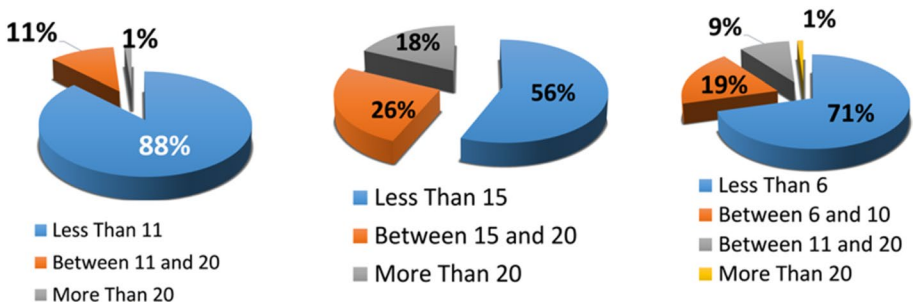


(a) Average energy consumption in each step by a Fog/Cloud physical host



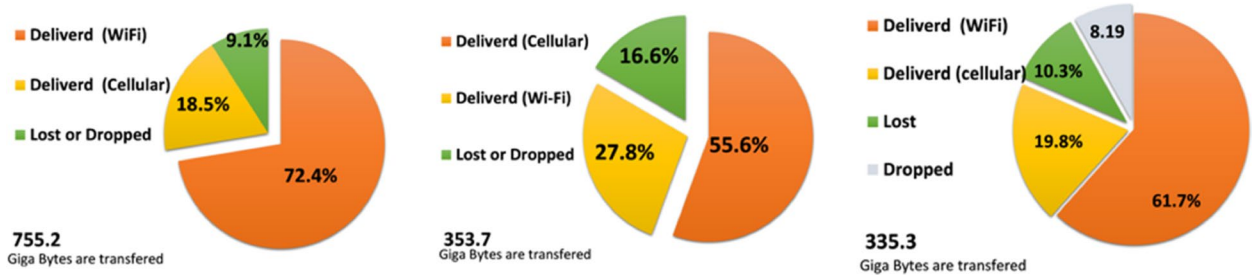
(b) Total energy consumption in each step by all physical hosts

Fig. 17 Energy consumption report for MOODS. **a** Average energy consumption in each step by a Fog/Cloud physical host. **b** Total energy consumption in each step by all physical hosts



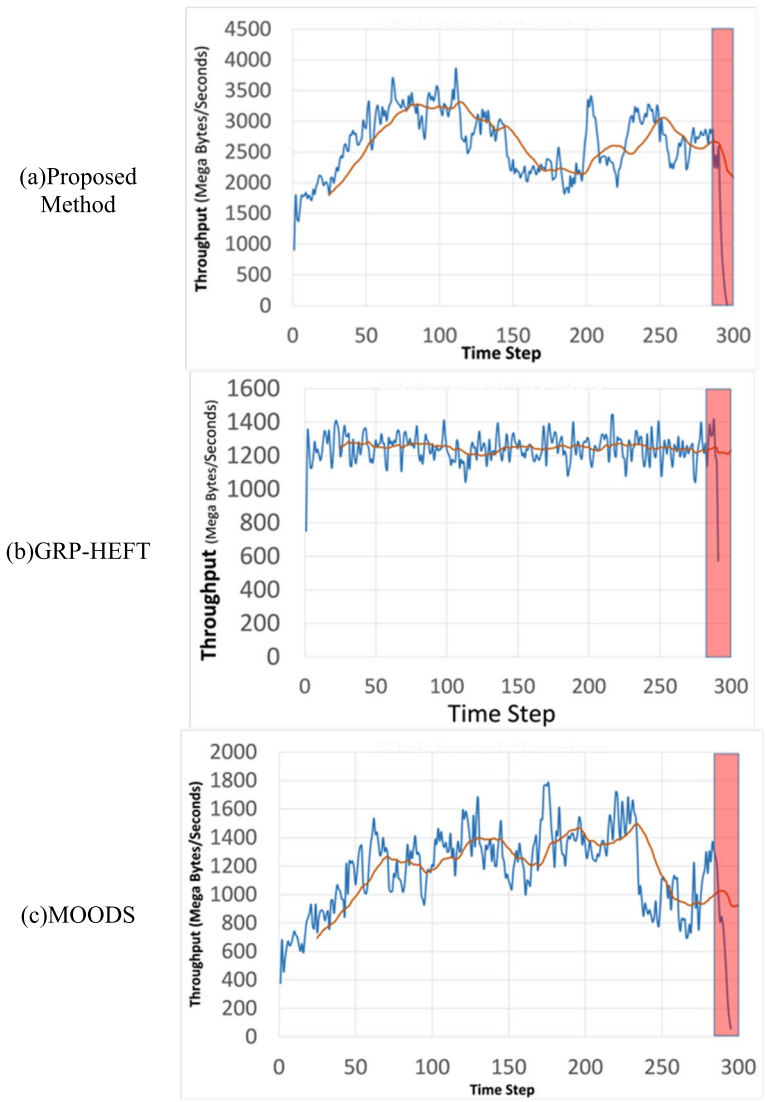
(a) Proposed Method (b) GRP-HEFT (c) MOODS

Fig. 18 Cluster formation. **a** Proposed Method. **b** GRP-HEFT. **c** MOODS



(a)Proposed Method (b)GRP-HEFT (c)MOODS

Fig. 19 Packet delivery report. **a** Proposed Method. **b** GRP-HEFT. **c** MOODS



(a)Proposed Method

(b)GRP-HEFT

(c)MOODS

Fig. 20 Throughput. **a** Proposed method. **b** GRP-HEFT. **c** MOODS

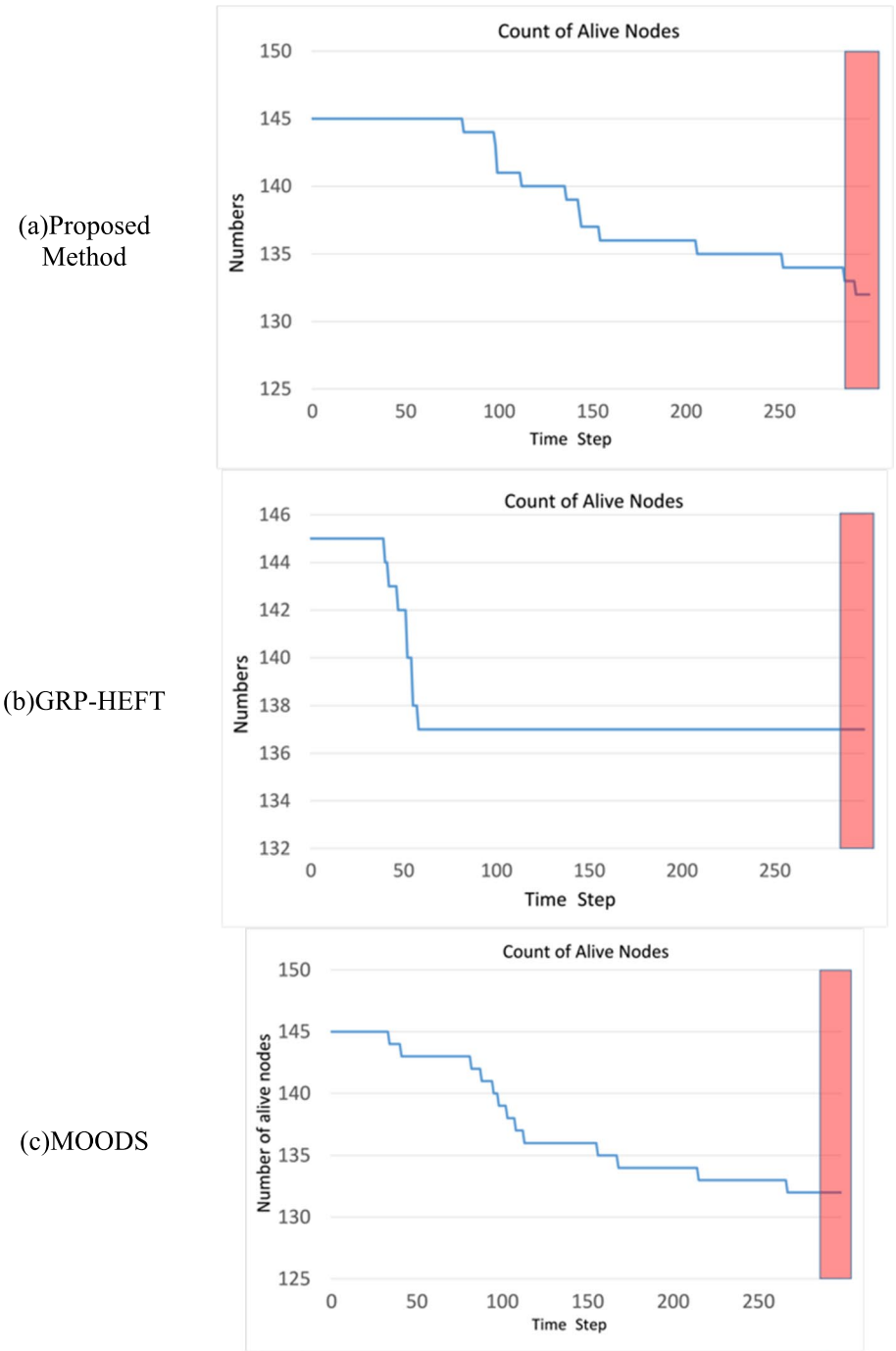


Fig. 21 Alive nodes. **a** Proposed method. **b** GRP-HEFT. **c** MOODS

score of the cluster heads were calculated using fuzzy inference system. Then, each node member of the cluster obtained its mobility score in the same way and used four important parameters, and according to the message it sent to connect to the cluster head nodes in its range, it is connected to a cluster. This allows the nodes with the closer mobility score to connect to their corresponding clusters.

The scheduling management component in the proposed architecture in the Fog layer performs scheduling workflow in three phases. In the first phase, using a critical path algorithm, important paths were

extracted based on the priority value of each task. In the next step, all possible schedules were created using a scheduling algorithm and the best one was selected using a new utility function. In the last step, tasks were assigned to resources.

In this regard, the contributions of this paper can be summarized as follows:

- Devising a three-layer architecture that simultaneously takes into account the advantages of the cloud and fog layers to manage the scheduling of workflow tasks and also manage the mobility of fog nodes.
- Presenting a dynamic, light-weight, multi-criteria, and decentralized clustering algorithm to create clusters considering the mobility of nodes.
- Extending the scheduling component and introducing a new scheduling algorithm that considers: priority of tasks obtained by using a critical path algorithm, a novel utility function, and considering workflow deadlines.

This paper used extensive software simulation to compare the performance of proposed method with two new methods in real workload. The evaluation results showed that proposed method has performed significantly better in success rate, completion time of jobs and utilization of the resources.

For future research we will focus on adding inherent support for workload prediction mechanisms in IoT layer to allow pro-active distributing of workloads and to enable future-aware scheduling of resource-intensive workloads.

Authors' contributions

Mrs. Masoumeh Hajvali contributed to the main idea and wrote the main manuscript text. Dr. Sahar Adabi is the corresponding author, introduced the core idea and organized the manuscript. Dr. Ali rezaee supervised the research and conducted the simulation studies. Dr. Mehdi Hoseinzadeh reviewed the paper and supervised the evaluation process. The author(s) read and approved the final manuscript.

Funding

Not applicable. No funding in any form is received for this manuscript. Also, There are no financial or non-financial competing interests.

Availability of data and materials

The datasets that has been used in this research is available online as mentioned in reference [58] of this manuscript.

Declarations

Consent for publication

Not applicable. This manuscript does not have any individual person data.

Competing interests

The authors declare no competing interests.

Received: 4 June 2022 Accepted: 18 March 2023
Published online: 28 April 2023

References

1. Pallewatta S, Kostakos V, Buyya R (2022) QoS-aware placement of microservices-based IoT applications in Fog computing environments. FGCS 131:121–36
2. Bagies E, Barnawi A, Mahfoudh S, Kumar N (2022) Content delivery network for IoT-based Fog Computing environment. Comput Netw 205:108688
3. Hamdi AM, Hussain FK, Hussain OK. (2022) Task offloading in vehicular fog computing: State-of-the-art and open issues. FGCS 133:201-212
4. Puliafito C, Vallati C, Mingozzi E, Merlino G, Longo F (2021) Design and evaluation of a fog platform supporting device mobility through container migration. Pervasive Mob Computing 74:101415
5. Potu N, Bhukya S, Jatoth CH, Parvataneni P (2022) Quality-aware energy efficient scheduling model for fog computing comprised IoT network". Comput Electr Eng 97:107603
6. Hilburg JC, Zapater M, Moya M, Ayala J (2022) Energy-aware task scheduling in data centers using an application signature. Comput Electr Eng 97:107630
7. Wen Y, Liu J, Dou W, Xu X, Cao B, Chen J. (2020) Scheduling workflows with privacy protection constraints for big data applications on cloud. FGCS, 108:084-1091
8. Arabnejad V, Bubendorfer K, Ng B (2017) Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. FGCS 75:348–364
9. Gill SS, Garraghan P, Buyya R (2019) ROUTER: Fog Enabled Cloud based Intelligent Resource Management Approach for Smart Home IoT Devices. J Syst Softw 154:125–138
10. Hajvali M, Adabi S, Rezaee A, Hosseinzadeh M. (2022) Software architecture for IoT-based health-care systems with cloud/fog service model," Cluster Computing 25:91-118
11. Kumari A, Tanwar S, Tyagi S, Kumar N, Parizi MR, Choo RKK (2019) Fog data analytics: A taxonomy and process model. J Netw Comput Appl 128:90–104
12. Li Z, Ge J, Yang H, Huang L, Hu H, Hu H, Luo B (2016) A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Future Gener Comput Syst 65:140–152
13. Mboula J, Kamla V, Djamegni C (2020) Cost-time trade-off efficient workflow scheduling in cloud. Simul Model Pract Theory 103:102107
14. Xiao Y, Zhou CA, Yang X, He B (2022) Privacy-preserving workflow scheduling in geo-distributed data centers. FGCS 130:46–58
15. Bu B (2022) Multi-task equilibrium scheduling of Internet of Things: A rough set genetic algorithm. Comput Commun 184:42–55
16. Hossain R, Whaiduzzaman M, Barros A, Tuly S, Mahi JN, Roy S, Fidge C, Buyya R (2021) A scheduling-based dynamic fog computing framework for augmenting resource utilization. Simul Model Pract Theory 111:102336
17. Niu Ch, Wang L (2022) Big data-driven scheduling optimization algorithm for Cyber-Physical Systems based on a cloud platform. Comput Commun 181:173–181
18. Asensio A, Masip-Bruin X, Durán R, Miguel I, Ren G, Daijavad S, Jukan A (2020) Designing an Efficient Clustering Strategy for Combined Fog-to-Cloud Scenarios. FGCS 109:392–406
19. Amine D, Nassreddine B, Bouabdellah K. (2014) Energy Efficient and Safe Weighted Clustering Algorithm for Mobile Wireless Sensor Networks. Procedia Computer Science 34:63-70
20. Abdolkarimi M, Adabi S, Sharifi A (2018) A new multi-objective distributed fuzzy clustering algorithm for wireless sensor networks with mobile gateways. Int J Electron Commun (AEÜ) 89:92–104
21. Kandali KH, Bennis L, Bennis H (2021) A New Hybrid Routing Protocol Using a Modified K-Means Clustering Algorithm and Continuous Hopfield Network for VANET. IEEE Access 9:47169–83
22. Sharifi SA, Babamir SM. (2020) A Clustering Algorithm for Efficient Energy Management in Mobile Ad-Hoc Networks. Comput Netw .166:06983
23. Lee YC, Han H, Zomaya AY, Yousif M (2015) Resource-Efficient Workflow Scheduling in Clouds. Knowl Based Syst 80:153–162

24. Dubey K, Kumar M (2018) Modified HEFT Algorithm for Task Scheduling in Cloud Environment. Procedia Comput Sci 125:725–732
25. Elsherbiny SH, Eldaydamony E, Alrahmawy M, Reyad A-E (2018) An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment. Egypt Inform J 19(1):33–55
26. Keshavarznejad M, Rezvani M, Adabi S. (2021) Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Cpomuting. 24:1825-1853
27. Paknejad P, Khorsand R, Ramezanpour M. (2021) Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment. FGCS 117:2-28
28. Han P, Du C, Chen J, Ling F, Du X (2021) Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J Syst Archit 112:101837
29. Wang B, Liu F, Lin W, Ma Z, Xu D (2021) Energy-efficient collaborative optimization for VM scheduling in cloud computing. Comput Netw 201:108565
30. Adabi S, Movaghar A, Rahmani A-M (2014) Bi-level fuzzy based advanced reservation of Cloud workflow applications on distributed Grid resources. J Supercomput 67:175–218
31. Zhou X, Zhang G, Sun J, Zhou J, Wei T, Hu S (2019) Minimizing cost and Makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. Future Gen Comput Syst 93:278–89
32. Yuan D, Yang Y, Liu X, Zhang G, Chen J (2012) A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. Concurr Comput Pract Exp 24(9):956–76
33. Sharma S, Saini H (2019) A Novel Four-Tier Architecture for Delay Aware Scheduling and Load Balancing in Fog Environment. SUSCOM 24:100355
34. Pham X-Q, Huh E-N. (2016) Towards task scheduling in a cloud-fog computing system. The 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). https://doi.org/10.1109/APNOMS.2016.7737240
35. Zeng D, Gu L, Guo S, Cheng Z, Yu S (2016) Joint Optimization of Task Scheduling and Image Placement In Fog Computing Supported Software-Defined Embedded System. IEEE Trans Comput 65(12):3702–12
36. Maio M, Kimovski D (2020) Multi-objective scheduling of extreme data scientific workflows in Fog. FGCS 106:171–84
37. Hosseinioun P, Kheirabadi M, KamelTabbakh SR, Ghaemi R (2020) A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. J Parallel Distrib Comput 143:88–96
38. Naha KR, Garg S, Chan A, Battula KS (2019) Deadline-based dynamic resource allocation and provisioning algorithms in Fog-Cloud environment. FGCS 104:131–41
39. Li C, Tang J, Ma T, Yang X, Luo Y (2020) Load balance based workflow job scheduling algorithm in distributed cloud. J Netw Comput Appl 152:102518
40. Konjaang KJ, Murphy J, Murphy L (2022) Energy-efficient virtual-machine mapping algorithm (EViMA) for workflow tasks with deadlines in a cloud environment. J Netw Comput Appl 203:103400
41. Pham XQ, Man ND, Tri NDT, Thai NQ, Huh EN (2017) A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. IJDSN 13(11):1550147717742073
42. Azizi S, Shojafar M, Abawajy J, Buyya R (2022) Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. J Netw Comput Appl 201:103333
43. Rodriguez MA, Buyya R (2017) A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments ". Concurr Comput Pract Exp 29(8):e4041
44. Zaman K et al (2022) Cost-effective data replication mechanism modelling for cloud storage. Int J Grid Util Comput 13(6):652–69
45. Rezaee A, Rahmani MA, Movaghar A, Teshnehlab M (2014) Formal process algebraic modeling, verification, and analysis of an abstract Fuzzy Inference Cloud Service. J Supercomput 67:345–83
46. Mokni M, Yassa S, Hajlaoui J, Omri NM, Chelouah R (2023) Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog-Cloud computing. Simul Model Pract Theory 123:102687
47. Davami F, Adabi S, Rezaee A, et al. (2022) Fog-based architecture for scheduling multiple workflows with high availability requirement. J Comput 104:69-208
48. Islam SM, Kumar A, Hu YC (2021) Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. J Netw Comput Appl 180:103008
49. Chang H, Hari A, Mukherjee S, Lakshman T-V (2014) Bringing the Cloud to the edge. in in Proc. IEEE Conf. Infocom Wkshps. https://doi.org/10.1109/INFCOMW.2014.6849256
50. Botta A, Donato W, Persico V, Pescapé A (2016) Integration of Cloud computing and Internet of Things: A survey. Future Gen Comput Syst 56:684–700
51. Zhang L, Zhou L, Salah A (2020) Efficient scientific workflow scheduling for deadline constrained parallel tasks in cloud computing environments. Info Sci 531:31–46
52. Elaziz M, Abualigah L, Attiya I (2021) Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. FGCS 124:142–54
53. Tang X, Shi C, Deng T, Wu Z, Yang L (2021) Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems. Appl Soft Comput 113:107914
54. Dubey K, Sharma SC (2021) A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. SUSCOM 32:100605
55. Velociraptor simulator https://github.com/simulatie-oplossingen/Velociraptor
56. Faragardi HR et al (2020) GRP-HEFT: A Budget-Constrained Resource Provisioning Scheme for Workflow Scheduling in IaaS Clouds. IEEE Trans Parallel Distrib Syst 31(6):1239–1254
57. Calzarossa MC et al (2021) Multi-Objective Optimization of Deadline and Budget-Aware Workflow Scheduling in Uncertain Clouds. IEEE Access 9:89891–89905. https://doi.org/10.1109/ACCESS.2021.3091310
58. Rezaee A, Adabi S (2022) 101K workflow jobs dataset (1.2 million tasks); a composition of Epigenomics and Montage workflows . Zenodo. https://doi.org/10.5281/zenodo.6373666.
59. Juve G, Chervenak.A, Deelman.E, Bharathi.S, Mehta.G, Vahi.K (2013) Characterizing and profiling scientific workflow", Future Generation Computer Systems. 29(3) : 682-692
60. Dayarathna M et al (2016) Data Center Energy Consumption Modeling: A Survey. IEEE Communication Surveys & Tutorials 18(1):732–94
61. Lent R (2013) A model for network server performance and power consumption. Sustainable Comput Informat Syst 3(2):80-93.
62. Shahid AM, Islam.N, Alam.M, Mazliham.M, Musa.SH (2021) Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. Computer Science Review. 40:100398
63. Baradaran AA, Navi.K (2020) HQCA-WSN: High-quality clustering algorithm and optimal cluster head selection using fuzzy logic in wireless sensor networks. Fuzzy Sets and Systems. 389:114-144

Publisher's Note