Received: May 28, 2022. Revised: June 29, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Modified Workflow Scheduling Algorithm for Cloud Computing Environment

Sara Ahmed[1]* **Fatma A. Omara[1]**

[1]*Computer Science Department, Faculty of Computers and Artificial Intelligence, Cairo University, Giza-12611, Egypt*
* Corresponding author's Email: asara4372@gmail.com

Abstract: Cloud computing has gained many attentions. Workflow scheduling one of the most important issues in cloud computing. It involves mapping tasks onto cloud resources – Virtual machines (VMs), to improve scheduling performance. Because the existing heterogeneous earliest finish time (HEFT) algorithm is considered one of the best algorithm, so the work in this paper propose a new algorithm based on HEFT algorithm; called modified heterogeneous earliest finish time (M-HEFT); to reduce the tradeoff among make span, resource utilization, and load balance. The proposed M-HEFT consists of two phases; task prioritization and task-VM mapping. In Task prioritization phase, a priority will be provided to each task in directed acyclic graph (DAG) as in the original HEFT algorithm. According to task-VM phase, tasks allocate to resources according to length of tasks and the load of available VMs with considering load balance. To evaluate the performance of the proposed algorithm, a comparative study has been done among the proposed algorithm and three existed algorithms. The experimental results show that the proposed algorithm outperforms the other algorithms by minimizing make span by 29%, improve resource utilization by 53% and load balance by 18% in average.

Keywords: Cloud computing, Task scheduling, Workflow scheduling, Heft, Make span, Resource utilization, Load balance.

1. Introduction

Cloud computing is a new technology become more popular among individual and organizations. Cloud computing is internet-based computing, where sharing resource software and information are provided with computers and other devices on demand and followed by paying as you go model.

There are main five characteristics of cloud computing such as on-demand self-service, where user can access the required services as needed automatically, broad network access, services available over the internet using desktop, laptop, PDA, mobile phone, resource pooling where resources are shared among several users, rapid Elasticity where user can scale in /out resource capacity to fulfill the increasing /decreasing demands, and measured service where resources measured and billing of the usage are delivered[1].

There are four deployment models of cloud computing, Public cloud, which allows systems and services to be easily accessible to the public. Private cloud, which allows systems and services to be easily accessible within an organization. Hybrid 8 cloud is considered a mixture of private and public clouds, but each one can remain as separate entities, where critical activities are performed in Private cloud while not critical activities are performed in public cloud [2].

Three services could be provided by cloud computing; software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [3].Due to the popularity of the cloud, most of the scientists execute their works on the cloud computing environment.

Scheduling of scientific workflows under quality of service (QoS) constraint is considered one of the main problems in the cloud environment. The workflow scheduling problem concerns about allocating each task of the scientific workflow to the suitable computing resources while meeting set of

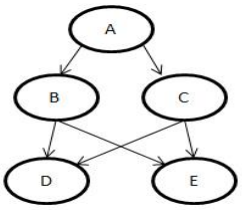

Figure. 1 Simple workflow

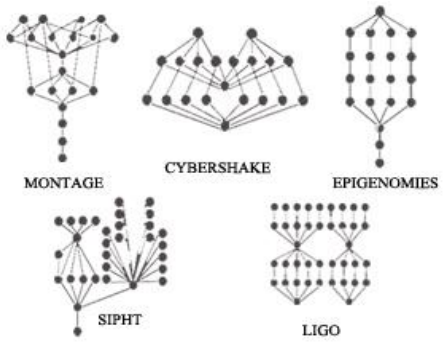

Figure. 2 Scientific workflow

QoS constraints like make span, resource utilization rate, and load balance [4].Resource allocation involves effective, appropriate selection of resources that minimize the application execution time, as well as, maximize the percentage of resource utilization [5]. So, efficient workflow scheduling technique is needed to increase resource utilization to assign large number of tasks in a balanced way within a short time in the cloud. Therefore, to utilize cloud computing environment efficiently, a good combination of optimized scientific workflow scheduling and resource allocation is needed.

There are two types of workflow; simple and scientific. Simple workflow represents real work which consists of group of tasks with sequence of activities and mechanisms used to perform individual or group tasks (see Fig. 1). Scientific workflow represents scientific applications which depend on other tasks with complexity in execution. These applications require several analysis tools for data processing. These applications have time constraints and require supercomputing support of heterogeneous computing resources (see Fig. 2) [6].

There are common scientific workflows used as a benchmark to evaluate the performance of the task scheduling algorithms such as Montage, CYBERSHAKE, SIPHT, LIGO and EPIGENOMICS [7]. The work in this paper uses LIGO and EPIGENOMICS workflows as a benchmark, where **LIGO** (Astrophysics) application is a memory intensive application used in the physics field with the aim of detecting gravitational waves, and **EPIGENOMICS** (Bioinformatics) application is a CPU intensive application that automates the execution of various genome-sequencing operations.

1.1 Workflow structure

A popular representation of a workflow application is the directed acyclic graph (DAG), *G (T, E)*, where *T* is a set tasks and *E* is a set of directed edges that represent inter-task data dependencies. Each node represents an individual application's task with a certain amount of computation workload W with million instructions (*MI*) as unit of measurement. Each edge e_{ij} represents a precedence constraint that indicates that task t_i should complete executing before task t_j can start. If there is data transmission from t_i to t_j, the t_j can start only after all the data from t_i has received [8].

1.2 Scheduling scientific workflow

Task scheduling is the process of allocating an application's tasks to suitable resources with considering dependency between them to reduce make span, maximize resource utilization, improve load balance, and achieve QoS parameters. Therefore, task scheduling algorithm is used to utilize resources more efficiently by reducing the overall execution time of tasks and satisfying load balance on various computing resources [9].

Despite the heterogeneous earliest finish time (HEFT) algorithm is considered the most popular algorithm, it suffers from load imbalance and not satisfy utilization of resources. Therefore, a modified heterogeneous earliest finish time (M-HEFT) algorithm has been introduced to improve the performance of the HEFT algorithm by reducing the tradeoff among load balancing, resource utilization and make span. M-HEFT algorithm consists of two phases; task prioritization and task-VM mapping. The task prioritization is implemented as the original HEFT algorithm. According to task-VM mapping phase, the tasks allocate to VMs based on the length of tasks and the load of available VMs. If the length of the ready task is less than or equal to the average length of all allocated tasks, it will be allocated to the most idle VM and, in the same time, guarantees earliest finish time. Otherwise, the task allocates to VM that guarantees earliest finish time.

Paper is organized as follows; related work is presented in section 2. section 3 illustrates the principles of the proposed task scheduling algorithm. The experiment results of the proposed M-HEFT algorithm are discussed in section 4. The performance evaluation of the proposed M-HEFT algorithm using the WorkflowSim simulator is illustrated in section 5. Finally, section 6 includes conclusion & future work.

2. Related work

Generally, workflow scheduling is represented by directed acyclic graph (DAG), where application program is represented by DAG in which each task represents by node and their communication link depicts by edges. Generally, task scheduling is known as DAG scheduling [10].

Scheduling tasks and/or jobs on the data center is very difficult because the number of Jobs/tasks requesting is very large and require extra resources to execute. Therefore, the schedule such jobs on the cloud must be optimal and good enough so that each request by the user gets response on time, and every task/job gets proper resources for its execution [11]. On the other hands, the scheduling of workflow needs to be care about task precedence constrains and Virtual Machines (VMs) configurations in the data center. Scheduling the tasks of DAG on VMs with different configuration needs to be aware about computation and communication costs. Scheduling algorithms classified to heuristic and Meta heuristic. Heuristic algorithms dependent on the problem and try to find the solutions by applying problem features in a complete way. Their solution is based on learning and exploration in which a comprehensive and scientific search for finding an optimal response and speeding to response process is applied [12]. Meta heuristic algorithms are independent problem, and they used to handle different type of problems. [13] In cloud computing, heuristic algorithms are designed to resolve the problematic issues faster than meta-heuristic algorithms, when their performance is too slow. Also, heuristic algorithms are used to find an optimum solution, when meta-heuristic algorithms failed to discover the precise or optimal solution [14]. This paper focuses on heuristic algorithm.

In [15], an algorithm, called scheduling service workflow for cost optimization in hybrid cloud is introduced. The goal of this algorithm is to reduce make span and deadline. This algorithm consists of two phases. In the first phase, the workflow is scheduled in the private cloud using the private resources using the path clustering heuristics (PCH) algorithm. After that, the schedule make span is determined and compared with the deadline. If deadline is not matched, the second phase will be executed. In the second phase, resources from the public cloud with reasonable cost will be reserved to execute a part of the workflow. Path clustering heuristics (PCH) algorithm select a path from the DAG and the nodes on this path will be allocated on the same processor [16], it is a combination of clustering and list scheduling heuristics [17]. The limitations of this algorithm is that if the problem is solved using the first phase only, this will lead to load imbalance, but if solved with second phase, Here it is important to decide when and what resource need to borrow because any mistake with this decision will make execution cost very high.

An improved max-min task scheduling algorithm has been introduced [18]. The goal of this algorithm is reducing make span. This algorithm calculates the average of execution time for all tasks in the workflow. Next max-min is used when receiving a task with execution time is smaller than the average. Otherwise a task with execution time greater than or equal to the average is assigned to the VM with minimum completion time among all the VMs regardless of VM availability. Where the completion time represents machine's ready time with task's execution time. More metrics need to be considered to prove the efficiency of this algorithm.

An efficient workflow scheduling algorithm (EWSA) is introduced [19]. The goal of this algorithm is maximize resource utilization and while meeting the deadline. This algorithm is designed to schedule scientific workflow. The algorithm consists of two phases; update, and task-VM mapping. The objective of the update phase is to trace each path in the DAG and set the execution time for each task, and then define the VM with needed capability to execute each task. In task-VM mapping phase, the tasks are scheduled on proper VMs. This algorithm does not concern load balance among VMs.

A scheduling algorithm, called MaxChild, is proposed [20]. The main objective of this algorithm is to improve the system throughput with proper resource utilization and high performance by obeying the required QoS parameters which specified by the user. According to this algorithm, the task that has maximum number of Childs is scheduled first to guarantee that maximum number that tasks could be available for the next schedules and resource are utilized properly. The problem of this strategy is that after a job is submitted to the resource and this resource is not available, this may affect makes pan. Also, the status of VMs is not concerned.

An algorithm called deadline–budget workflow scheduling (DBWS) has been introduced [21]. It aims to find a feasible schedule within a budget and deadline constraints. The algorithm consists of two phases; task selection and resource selection. According to task selection phase, the DAG tasks will be selected according to their priorities. To assign a priority to a task in the DAG, the upward rank is computed. This rank represents, for a task, the length of the longest path from this task to the exit node including average execution time of the task over all

resources and average communication time. The resource selection phase consists of two steps; in the first step, all tasks are divided in different levels based on their depth in the graph. Then, the user deadline will be distributed among all levels and sub deadline for each level will be computed, where all tasks belonging to the same level have the same sub-deadline. In the second step, the task will be allocated to the resource which has closer finish time with respect to its sub -deadline, and in the same time, has low cost. This algorithm suffers from computation overhead.

Heterogeneous earliest finish time algorithm (HEFT) one of the most popular algorithms for scheduling workflow [22]. The ultimate objective of HEFT is to reduce make span. The algorithm consists of two phases; task prioritization and processor selection. In the Task Prioritization phase, the priorities of all tasks are assigned by computing the rank for each task, which is based on mean computation time and mean communication cost. Then, the tasks list is ordered in descending order. In the Processor Selection phase, the tasks are scheduled on the processors that give the earliest finish time (EFT) for the task. The HEFT algorithm is similar as the EFT, in addition to, scales resources elastically at runtime. Therefore, it obtains an optimized execution time. HEFT suffers from load imbalance.

A comparative study has done among four heuristic algorithms, minx–min, random, suffrage, and heterogeneous earliest finish time (HEFT), while using static and dynamic scheduling schemes, and considered some features that are Number of machines [23] According to the results of the comparative study; it is found that the HEFT algorithm outperforms the other algorithms because it is a list-based scheduling strategy that considers the DAG as a whole, while the other algorithms (non-list scheduling algorithms) consider the nodes ready to be executed only.

A modification has been done to the heterogeneous earliest finish time (HEFT) algorithm to enhance the performance on the cloud environment [24]. According to this modification, the priority for every task in the DAG has been defined by calculating the order of execution; average of task on all the processor + max (order of task value of predecessor task of current task) + communication cost between predecessor task node to current node) starting with the last node in the DAG. By this modification, the algorithm outperforms the HEFT algorithm with respect to make span. This algorithm not concerns load balance metrics.

An efficient task scheduling algorithm for DAG in cloud computing environment has been proposed [25]. The goal of this algorithm is reduce make span. The algorithm consists of two phases; task priority and resource selection. According to the task priority phase, the priority of the tasks is defined using critical path and static level (CPS) Attributes. Then, the tasks are sorted in decreasing order. In the resource selection phase, the selection of resource is based on the earliest start time (EST) and the earliest finish time (EFT). The algorithm outperforms the HEFT algorithm with respect to make span. This algorithm suffers from load imbalance.

Multi-Objective Workflow Optimization Strategy (MOWOS) has been introduced [26]. The goal of this algorithm is reducing execution cost and makes span. MOWOS Strategy consists of three sub algorithms; task spiriting algorithm, minimum VM (MinVM) selection algorithm, and maximum VM (MaxVM) selection algorithm. MOWOS Strategy uses a task-splitting mechanism to break down large tasks into smaller chunks to reduce workflow schedule. This algorithm suffers from load imbalance.

A new min-min algorithm called optimized min-min (OMin-Min) for scientific workflow has been introduced [27]. The goal of this algorithm is to reduce make span and try to avoid neglecting long task as min-min algorithm. According to this algorithm, tasks that have minimum and maximum execution times (MinT and MaxT) will be defined, and then the task with minimum execution time will be assigned to resource that produces minimum execution time. Otherwise, the task with minimum execution time assigns to resource that produces minimum execution time. This algorithm not concerns load balance metrics.

Unfortunately, most of the existed algorithms have problem with respect to resource utilization and load balancing among VMs in the distributed systems. Therefore, a modified heterogeneous earliest finish time (M-HEFT) algorithm has been introduced by the work in this paper to overcome the limitations of other algorithms (i.e., load balance, and resource utilization).

Table 1 shows a comparison between the aforementioned algorithms.

3. The proposed task scheduling algorithm

The proposed task-scheduling algorithm is based on the existed HEFT algorithm with some modifications to improve resource utilization, and load balance, in addition to, make span. The proposed algorithm is called M-HEFT. The goal of M-HEFT is to make no idle VM which will lead to maximize resource utilization, and make resources more balanced and reduce make span as well.

The proposed M-HEFT algorithm consists of two

Table 1. Comparison of workflow scheduling algorithms

Scheduling algorithm	Scheduling parameters	Finding	Environment
Scheduling Service Workflow for cost optimization in hybrid cloud [15]	Make span and deadline	This algorithm can reduce the make span comparing to the local execution, as well as, the cost comparing to the execution in the public cloud. We can note that in the private cloud, the higher the number of slices, the higher the execution time, as well as cost.	Cloud environment
An improved Max-Min task scheduling [18]	Make span	Improved Max-Min algorithm outperforms the Max-Min algorithm in most of the cases with respect to make span.	Cloud environment
An Efficient Workflow Scheduling Algorithm (EWSA) [19]	Resource utilization and deadline	This algorithm maximizes the resource utilization and meet the deadline of the application	Cloud environment
MaxChild [20]	Make span and resource utilization	MaxChild was found to be the most efficient algorithm with respect to make span and resource utilization comparing to FCFS, MAX-MIN, and MAX-MAX algorithm.	Cloud environment
Deadline–Budget Workflow Scheduling (DBWS) has been introduced [21]	Deadline and budget	This algorithm achieves better rates of successful schedules compared to other heuristic-based approaches for the real world applications consider.	Cloud environment
Heterogeneous Earliest Finish Time Algorithm (HEFT) [22]	Make span	This algorithm reduces make span	Cloud environment
A modification has been done to the Heterogeneous Earliest Finish Time (HEFT) algorithm [24]	Make span	This algorithm reduces the make span and satisfies load balancing compare to existing HEFT and CPOP algorithms.	Cloud environment
An efficient task scheduling algorithm for DAG in cloud computing environment [25]	Make span, speed, efficiency and scheduling length ratio.	The algorithm outperforms the HEFT algorithm with respect to make span, speed, efficiency and scheduling length ratio.	Cloud environment
Multi-Objective Workflow Optimization Strategy (MOWOS) [26]	Make span ,cost and resource utilization	The proposed MOWOS algorithm has less execution cost, better execution make span, and utilizes the resources than the existing HSLJF and SECURE algorithms.	Cloud environment
Optimized Min-Min (OMin-Min) [27]	Make span	The algorithm outperforms the Round Robbin, Modified Max-Min (MMax-Min) Min-Min and Max-Min algorithms with respect to make span.	Cloud environment

Algorithm 1: M-HEFT scheduling algorithm

Input: DAG and VMs configuration
Output: Mapping scheme for the requested tasks cloudlets on the available resources VMs.

1: set the computation cost for each task on each resource $CCT_{i,j}$
2: set the communication cost between tasks and their successors $C_{i,k}$
3: for each task $i=1$ to T_i in DAG
4: calculate rank value for every task in DAG
$rank\ of\ task\ (T_i) = \frac{\sum CCT_i}{VmNum} + max\ (rank\ (T_k) + C(T_i, T_k))$
5: end for
6: arrange tasks in a list in decreasing order based on their rank value of (T_i).
7: compute Average task length (*AL*) for all tasks in the DAG
8: for each task in ready list
9: check if task length greater than or equal *AL*
10: map task to VM which has the earliest finish time
11: else if the task length less than *AL*
12: map task to the most idle VM which has earliest finish time
13: end for
14: end

phases; Task prioritization phase to assign priority for each task in the DAG, and Task-VM mapping phase to allocate task to suitable VM.

The novelty of the M-HEFT algorithm is in Task-VM mapping phase where it based on average length of tasks and load balance on VMs. According to Task-VM mapping phase, the average length for all tasks is calculated, and then the ready task is mapped on the most idle VM which guarantees earliest finish time if its length is less than or equal average length. Otherwise ready task maps to the VM that guarantees earliest finish time.

In this section will describe each phase in details.

3.1 Task prioritization phase

In this phase, a priority will assign to each task in the workflow DAG according to rank value. The phase starts by computing computation cost *(CCT)* for each task in DAG (Ti) on each VM_j using Eq. (1).

$$CCT\ (Ti, VM_j) = \frac{T_i.length}{VM_j.MIPS} \quad (1)$$

Where, $T_i.length$ is the needed time to execute T_i, and $VM_j.MIPS$ is the speed of VM_j.

Then, communication cost is calculated between tasks and their successors $C\ (T_i,T_k)$. Then, rank value for each task (T_i) *is* calculated which equal to the average computation cost of the task on all VMs + max (rank value of successor task of the current task + communication cost between successor task and current task) (see Eq. (2)).

$$Rank\ (T_i) = \frac{\sum CCT_i}{VmNum} + max\ (rank\ (T_k) + C(T_i, T_k)) \quad (2)$$

Finally, sort the tasks in a list in decreasing order based on their rank value.

3.2 Task – VM mapping phase

In this phase, M-HEFT algorithm tries to select the best VM for each task by calculating the average task length (AL) for all tasks using Eq. (3).

$$AL = \frac{\sum T_i\ length}{TaskNum} \quad (3)$$

Where, *Ti* length is the needed computation time of task *Ti*, and taskNum is the number of tasks in the DAG.

If the length of the ready task less than *(AL)*, the task will be mapped to the idlest VM (that has the most available time), in the same time, it has the earliest finish time. Else, map the task to VM that has earliest finish time using Eq. (4), Eq. (5).

$$Finish\ Time\ (T_i, VM_j) = CCT\ (Ti,\ VM_j) + ST\ (T_i, VM_j) \quad (4)$$

$$Start\ Time\ (T_i, VM_j) = max\ \{T_{avil}(VM_j), FT\ (T_k) + C\ (T_K,T_i)\ \} \quad (5)$$

Remove the task from list and update available time for each VM. Repeat these steps till all tasks allocate to VMs.

3.3 Pseudo code of the proposed M-HEFT algorithm

The pseudo code of the proposed M-HEFT algorithm is described in Algorithm. 1.

3.4 Flowchart of the proposed M-HEFT algorithm

The flowchart represents a process of scheduling tasks of DAG by proposed M-HEFT algorithm which described in Fig. 3 to explain more how the proposed algorithm works.

4 Experimental results

4.1 Performance metrics

Three metrics are used to evaluate the performance of the proposed M-HEFT algorithm; make span, resource utilization rate, and ideal load

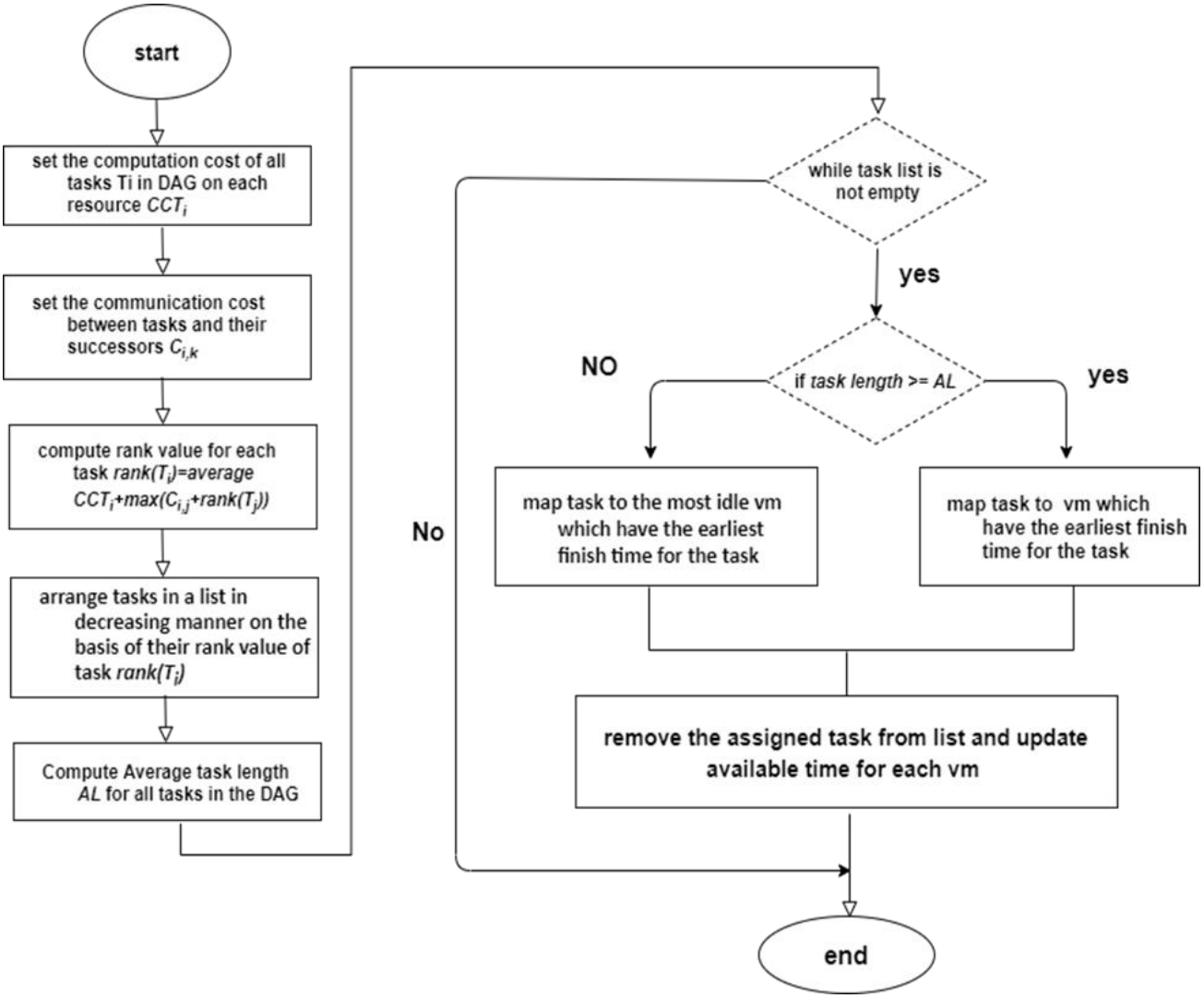

Figure. 3 The flow chart of the proposed M-HEFT algorithm

balance.

Make span is the maximum time required for completion of the whole DAG tasks [21]. Make span should be minimized. Eq. (6) is used to calculate make span.

$$Make\ span = max\ \{CT_i\} \quad (6)$$

where CT_i is the completion time of the longest task T_i.

Resource utilization rate *(RUR)* is the ratio between the total busy time of VM_i and the make span of the parallel application in percentage (see equation Eq. (7), and Eq. (8)) [28]. Resource utilization should be maximized.

$$RUR\ (VM_j)\ \% = \left(\frac{\sum vm_j\ Busy\ Time}{Make\ span}\right) X\ 100 \quad (7)$$

$$RUR\ for\ DAG = \frac{\sum RUR\ (VM_j)}{VmNum} \quad (8)$$

Ideal load balance *(ILB)* is the ratio between the total number of tasks and the number of VMs and it determines using Eq. (9) [29].

$$Ideal\ Load\ Balance\ (ILB) = Number\ of\ Tasks/\ Number\ of\ usedVM \quad (9)$$

Difference from ideal rate of load balance (DLB) is the difference between actual load balance and the ideal load in VMi. It is calculated using Eq. (10) [28].

$$DLB\ (VM_j)\ \% = \sum Number\ of\ tasks\ (VM_j) - ILB\ (VM_j) \quad (10)$$

Average difference from ideal rate of load balance *(ADLB) is* the ratio between the total summations of DLB for each VMi over their number. It is calculated using Eq. (11) [30].

$$ADLB\ (VM_j) = \frac{\sum_{j=1}^{m} DLB\ (VM_j)}{VmNum} \quad (11)$$

Table 2. Vm configuration and used workflow

Entities		Values
Workflows	Ligo	100, 1000
	Epigenomics	100,1000
Data center	1	1
VMs	Quantity	5,10,15,20
	Speed	50-1000
CPU	Quantity	1
	Ram	512
	Bandwidth	1000Mbps

Improvement rate in terms of used metrics (make span, resource utilization and load balance), it is the performance improvement rate of the proposed M-HEFT algorithm over the current HEFT and proposed strategies mentioned in [25, 27]. It is computed using Eq. (12).

$$IR_m = \left(\frac{ABS(M(existing\ algorithm) - M(proposed\ algorithm))}{M(existing\ algorithm)}\right) x\ 100 \quad (12)$$

4.2 Experimental environment

The proposed algorithm has been simulated using WorkflowSim1.0 toolkit integrated into net beans IDE 8.0.2 with the configurations shown in Table 2. WorkflowSim is an open source workflow simulator, which is an extension of the CloudSim framework [31]. The experiments have done using two workflows; Ligo (Astrophysics), and epigenomics (Bioinformatics).

5 Performance evaluation of the proposed M-HEFT algorithm

To evaluate the performance of the proposed algorithm, a comparative study has been contacted among the proposed algorithm, the heterogeneous earliest finish time (HEFT) algorithm [24], the algorithm mentioned in [25] and the algorithm mentioned in [27] with respect to make span, resource utilization, and load balancing metrics. This study has been implemented with considering heterogeneous environment using WorkflowSim, and two benchmarks, Ligo and epigenomics with 100 and 1000 tasks, and 5, 10, 15 and 20 VMs.

5.1 Make span evaluation

The implementation results of the comparative study among our proposed M-HEFT algorithm, the algorithm mentioned in [25] and the algorithm mentioned in [27] with respect to make span with considering Ligo and epigenomics benchmark with 100 and 1000 tasks using 5, 10, 15 and 20 VMs are discussed as the follow.

5.1.1. Make span for 100 and 1000 tasks of ligo

Make span results for 100 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 3, and Fig. 4.

Make span results for 1000 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 4, and Fig. 5.

According to the comparative results in Table 3 and Fig. 4, it is found that the proposed M-HEFT algorithm improves make span by 14% with respect to HEFT algorithm, and 40% with respect to algorithm in [25], and 44% with respect to algorithm in [27] in average with considering 100 tasks in Ligo. According to the comparative results in Table 4 and Fig. 5, it is found that the proposed M-HEFT algorithm improves make span by 46% with respect to HEFT algorithm, and 11% with respect to algorithm in [25], and 13% with respect to algorithm in [27] in average with considering 1000 tasks in Ligo.

5.1.1 Make span for 100 and 1000 tasks of epigenomics

Make span results for 100 tasks of epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 5, and Fig. 6. Make span results for 1000 tasks of epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 6, and Fig. 7.

According to the comparative results in Table 5 and Fig. 6, it is found that the proposed M-HEFT algorithm improves make span by 15% with respect to HEFT algorithm, and 41% with respect to algorithm in [25], and 48% with respect to algorithm in [27] in average with 100 tasks in Ligo. According to the comparative results in Table 6 and Fig. 7, it is found that the proposed M-HEFT algorithm improves make span by 54% with respect to HEFT algorithm, and 12% with respect to algorithm in [25], and 14% with respect to algorithm in [27] in average with 1000 tasks in epigenomics.

5.2 Resource utilization evaluation

The implementation results of the comparative study among our proposed M-HEFT, propose algorithm [25], and algorithm in [27], and HEFT algorithms with respect to resource utilization with considering Ligo and epigenomics benchmark with 100 and 1000 tasks using 5, 10, 15 and 20 VMs are discussed as the follow.

Table 3. Make span results for 100 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	43195.33	10043.61	4784.83	3360.4
Algorithm [25]	34479.03	17224.36	9383.23	6204.83
Algorithm [27]	34674.23	17598.23	10173.00	7820.00
Proposed M-HEFT	31710.37	9023.57	4646.04	2836.96

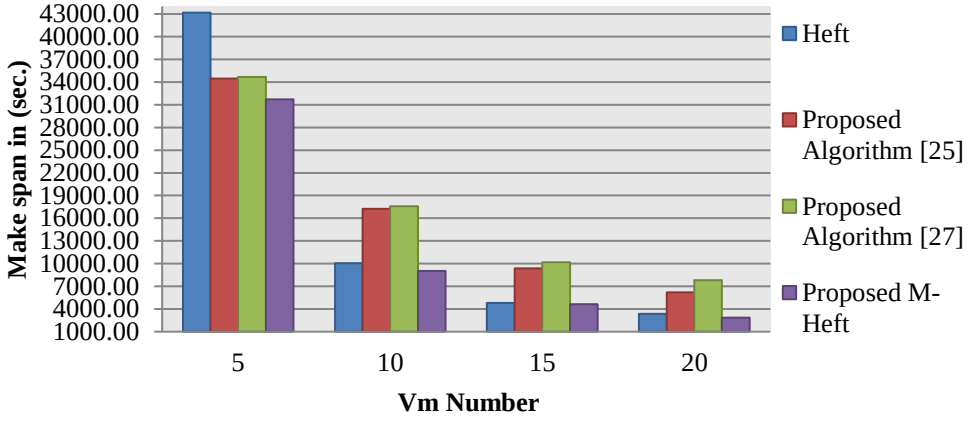

Figure. 4 Make span results for 100 tasks of ligo

Table 4. Make span results for 1000 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	316515.75	255772.8	62360.67	105685.8
Algorithm [25]	307040.59	87822	50968.25	29652.74
Algorithm [27]	308075.18	87945.32	51598.46	30757.41
Proposed M-HEFT	304147.05	87225.67	40624.22	22723.3

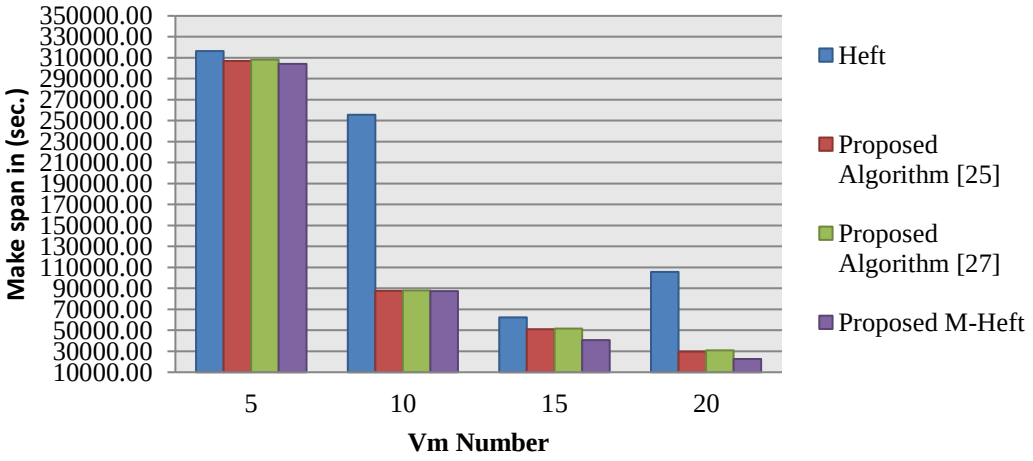

Figure. 5 Make span results for 1000 tasks of ligo

Table 5. Make span results for 100 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	569754.69	193400	141876	114244.72
Algorithm [25]	656837.59	411632	189023	164725.11
Algorithm [27]	675509.87	413795.00	225311.52	250584.03
Proposed M-HEFT	5448	183010	111433	81473.38

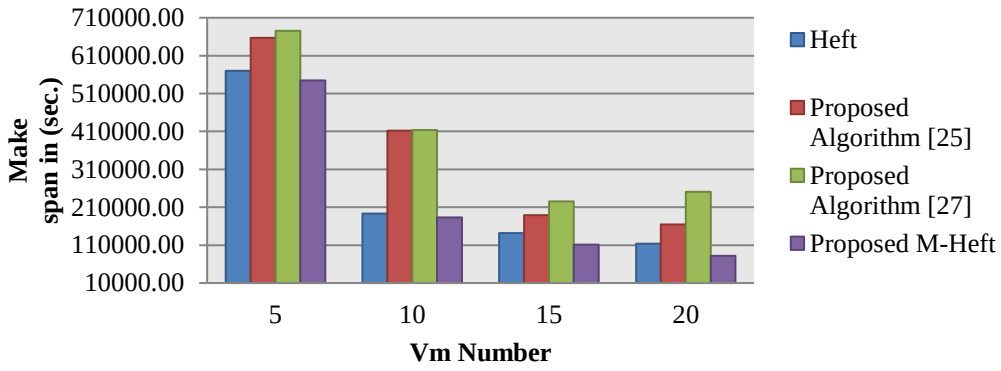

Figure. 6 Make span results for 100 tasks of epigenomics

Table 6. Make span results for 1000 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	5178814.39	2921064	4249865	2230593.1
Algorithm [25]	5191570.65	867429	867429	532190.83
Algorithm [27]	5202232.30	1691359.69	898372.57	559629.85
Proposed M-HEFT	5155803	1420167	714205	451127.66

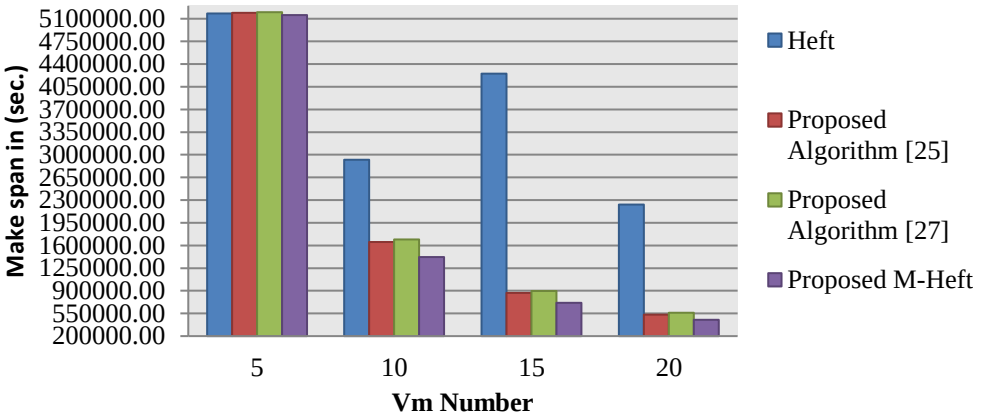

Figure. 7 Make span results for 1000 tasks of epigenomics

Table 7. Resource utilization rate results for 100 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	38.93	41.86	40.24	31.28
Algorithm [25]	48.79	24.41	19.91	16.94
Algorithm [27]	47.58	24.31	18.62	15.34
Proposed M-HEFT	53.03	46.59	42.24	37.05

5.2.1. Resource utilization for 100 and 1000 tasks of ligo

Resource utilization results for 100 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 7,and Fig. 8. Resource utilization results for 1000 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 8, and Fig. 9.

According to the comparative results in Table 7 and Fig. 8, it is found that the proposed M-HEFT algorithm improves resource utilization by 18% with respect to HEFT algorithm, and 82% with respect to algorithm in [25], and 93% with respect to algorithm in [27] in average with 100 tasks in Ligo. According to the comparative results in Table 8 and Fig. 9, it is found that the proposed M-HEFT algorithm improves resource utilization by 97% with respect to HEFT algorithm, and 15% with respect to algorithm in [25], and 18% with respect to algorithm in [27] in average with 1000 tasks in Ligo.

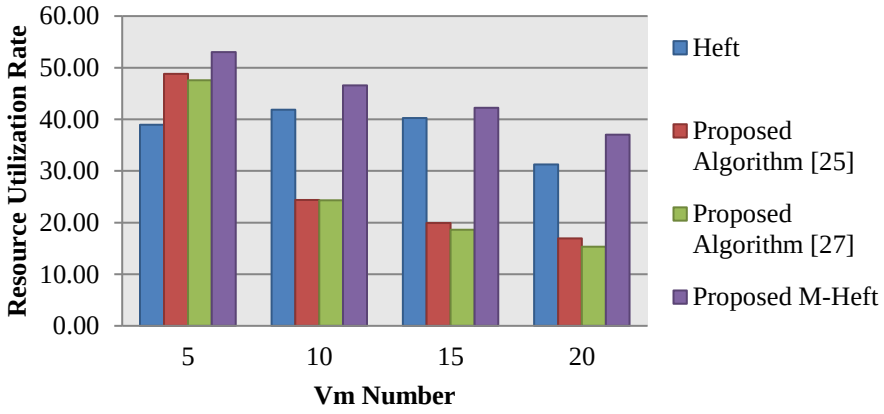

Figure. 8 Resource utilization rate results for 100 tasks of ligo

Table 8. Resource utilization rate results for 1000 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	57.55	20.80	32.45	18.18
Algorithm [25]	58.73	51.38	39.71	38.39
Algorithm [27]	58.14	50.95	38.56	37.15
Proposed M-HEFT	60.05	52.249	50.53	50.10

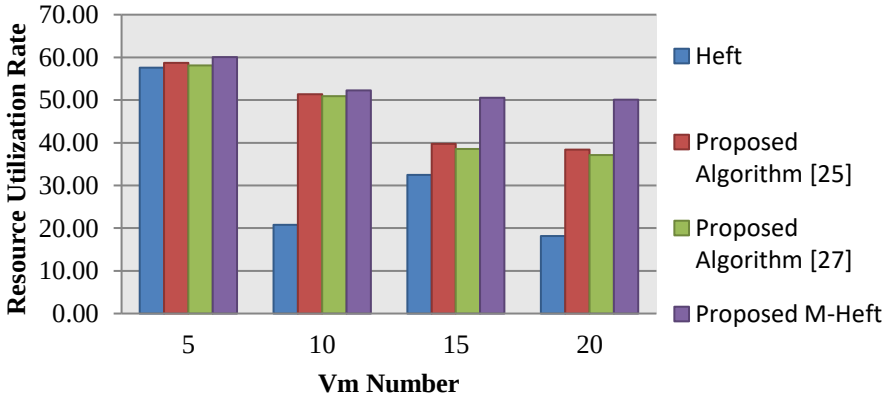

Figure. 9 Resource utilization rate results for 1000 tasks of Ligo

5.2.2. Resource utilization for 100 and 1000 tasks of epigenomics

Resource utilization results for 100 tasks of epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 9, and Fig. 10. Resource utilization results for 1000 tasks of epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 10, and Fig. 11.

According to the comparative results in Table 9 and Fig. 10, it is found that the proposed M-HEFT algorithm improves resource utilization by 19% with respect to HEFT algorithm, and 79% with respect to algorithm in [25], and 96% with respect to algorithm in [27] in average with 100 tasks in epigenomics. According to the comparative results in Table 10 and Fig. 11, it is found that the proposed M-HEFT algorithm improves resource utilization by 96% with respect to HEFT algorithm, and 14% with respect to algorithm in [25], and 20% with respect to algorithm in [27] in average with 1000 tasks in Epigenomics.

5.3 Load balance rate

The implementation results of the comparative study among our proposed M-HEFT, proposed algorithm [25], and algorithm [27], and HEFT algorithms with respect to load balance rate with considering Ligo and Epigenomics benchmark with 100 and 1000 tasks using 5, 10, 15 and 20 VMs are discussed as the follow.

Table 9. Resource utilization rate results for 100 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	56.64	41.72	25.27	17.66
Algorithm [25]	49.13	19.60	18.97	12.24
Algorithm [27]	48.76	18.81	16.42	10.56
Proposed M-HEFT	59.23	44.09	32.18	24.76

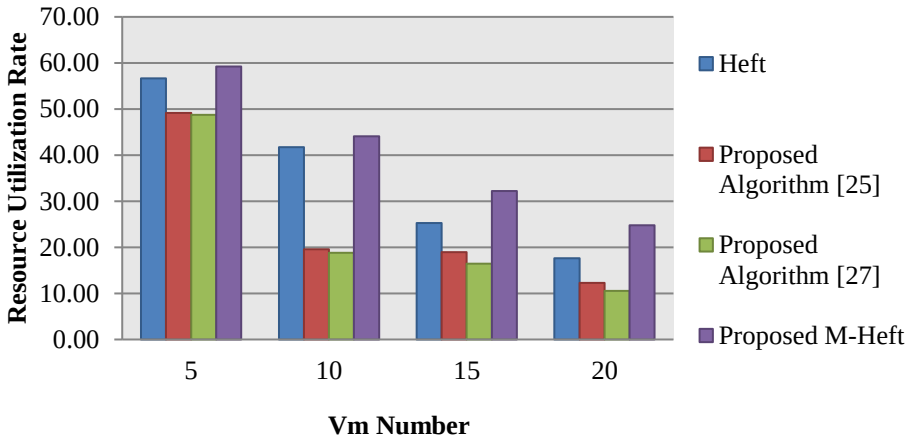

Figure. 10 Resource utilization rate results for 100 tasks of epigenomics

Table 10. Resource utilization rate results for 1000 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	59.15	26.39	20.06	18.04
Algorithm [25]	59.40	46.63	39.50	36.22
Algorithm [27]	59.13	46.24	36.32	33.48
Proposed M-HEFT	60.45	54.29	47.98	42.72

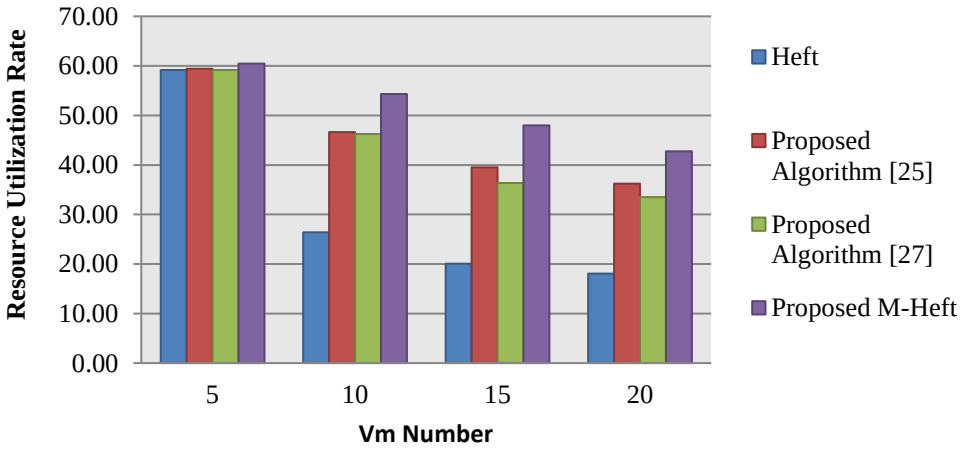

Figure. 11 Resource utilization rate results for 1000 tasks of epigenomics

Table 11. The average difference from ideal load balance (ILB) results for 100 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	8.8	4	3.20	2.70
Algorithm [25]	7	3.8	2.3	1.4
Algorithm [27]	7.1	3.9	2.5	1.7
Proposed M-HEFT	6.8	2	2	1

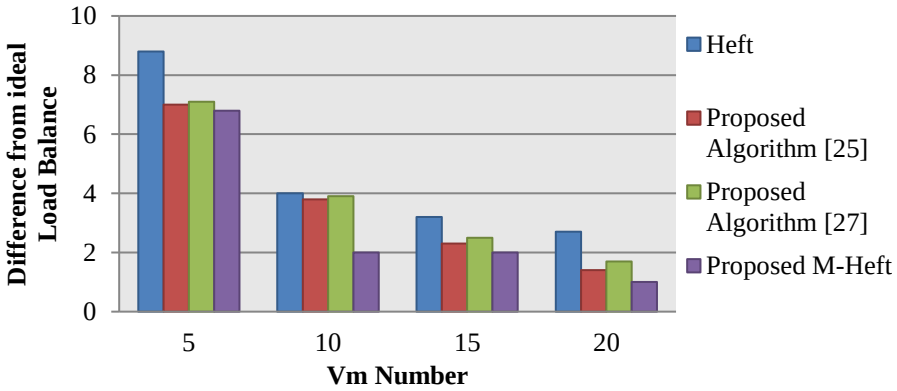

Figure. 12 The average difference from ideal load balance (ILB) results for 100 tasks of ligo

Table 12. The average difference from ideal load balance (ILB) results for 1000 tasks of ligo

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	79.5	63.4	33.3	34.3
Algorithm [25]	78.6	38	30.3	22.5
Algorithm [27]	78.8	38.2	30.5	22.8
Proposed M-HEFT	78.4	36.3	27.4	19.4

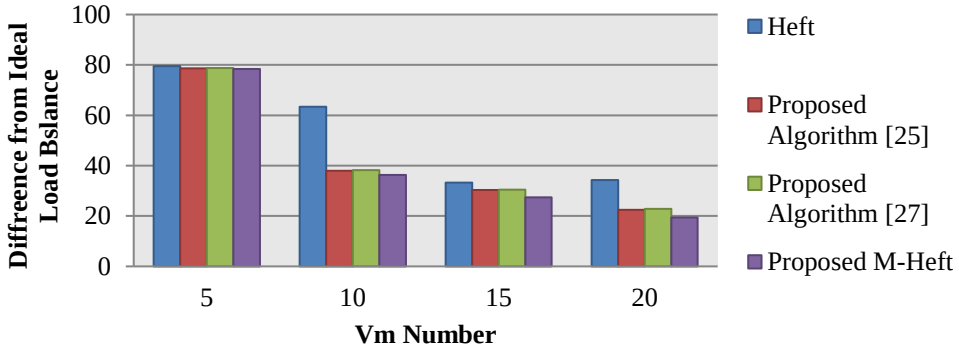

Figure. 13 The average difference from ideal load balance (ILB) results for 1000 tasks of ligo

5.3.1. Load balance rate for 100 and 1000 tasks of ligo

Load balance rate results for 100 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 11, and Fig. 12. Load balance rate results for 1000 tasks of Ligo using 5, 10, 15 and 20 VMs are discussed in Table 12, and Fig. 13.

algorithm in [25], and 29% with respect to algorithm in [27] in average with 100 tasks in Ligo. According to the comparative results in Table 12 and Fig. 13, it is found that the proposed M-HEFT algorithm improves load balance by 26% with respect to HEFT algorithm and 7% with respect to algorithm in [25], and 8% with respect to algorithm in [27] in average with 1000 tasks in Ligo.

Table 13. The average difference from ideal load balance (ILB) results for 100 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	10.8	4.6	2.70	2.1
Algorithm [25]	9.6	5	3.00	2.1
Algorithm [27]	9.7	5.1	3.2	2.3
Proposed M-HEFT	8	4.4	2.6	2.1

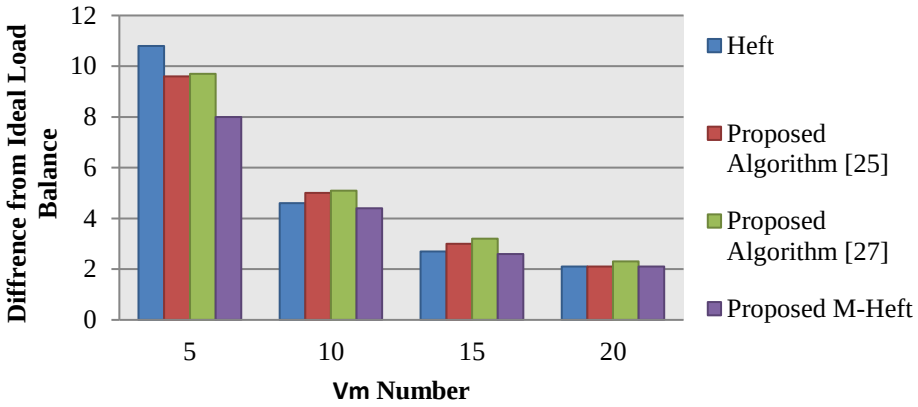

Figure. 14 The average difference from ideal load balance (ILB) results for 100 tasks of epigenomics

Table 14. The average difference from ideal load balance (ILB) results for 1000 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT	140.2	80.3	49.6	36.95
Algorithm [25]	77.6	49.7	33.4	23.75
Algorithm [27]	77.5	50.55	34.46	23.55
Proposed M-HEFT	77.4	39.7	28.16	23.65

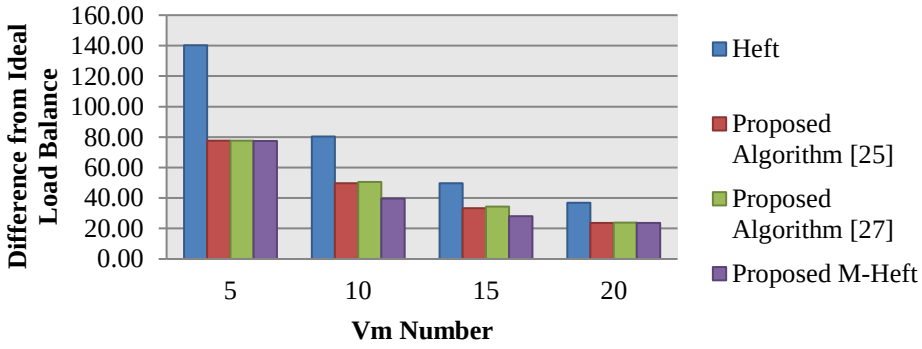

Figure. 15 The average difference from ideal load balance (ILB) results for 1000 tasks of epigenomics

5.3.2. Load balance rate for 100 and 1000 tasks of epigenomics

Load balance rate results for 100 tasks of epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 13, and Fig. 14. Load balance rate results for 1000 tasks of Epigenomics using 5, 10, 15 and 20 VMs are discussed in Table 14, and Fig. 15

According to the comparative results in Table 13

and Fig. 14, it is found that the proposed M-HEFT algorithm improves load balance by 8% with respect to HEFT algorithm, and 4% with respect to algorithm in [25], and 5% with respect to algorithm in [27] in average with 100 tasks in epigenomics. According to the comparative results in Table 14 and Fig. 15, it is found that the proposed M-HEFT algorithm improves load balance by 44% with respect to HEFT algorithm, and 9% with respect to algorithm in [25], and 10% with respect to algorithm in [27] with 1000 tasks in epigenomics.

6. Conclusion and future work

Task Scheduling is one of the main issues for achieving good performance over a cloud environment. In this paper, a modified task scheduling algorithm, called M-HEFT, has been introduced to improve the performance of the existing HEFT with respect to make span, resource utilization, and load balance. To evaluate the performance of the proposed M-HEFT algorithm, a comparative study has been contacted using two benchmarks, LIGO and EPIGENOMICS, with 100 and 1000 tasks and implemented on WorkflowSim simulator considering 5, 10, 15 and 20 VMs.

According to the implementation results, it is found that the proposed M-HEFT improves the make span algorithm by 32% in average with respect to the original HEFT algorithm, and by 26% in average with respect to the proposed algorithm [25], and by 30% in average with respect to the algorithm in [27]. The resource utilization has been improved using the proposed M-HEFT algorithm by 58% in average with respect to the original HEFT algorithm, and by 48% in average with respect to the proposed algorithm [25], and by 54% in average with respect to the algorithm in [27]. In addition, the load balance has been improved using the proposed M-HEFT algorithm by 31% in average with respect to the original HEFT algorithm, and by 11% in average with respect to the proposed algorithm [25], and by 13% in average with respect to the algorithm in [27].

As a future work, there is a need to enhance our M-HEFT algorithm by considering extra performance parameters such as budget, power consumption, and deadline.

Conflicts of interest

There is no conflict of interest.

Author contributions

Conceptualization, Fatma A. Omara, Sara Ahmed; methodology, Sara Ahmed; software, Sara Ahmed; validation, Fatma A. Omara and Sara Ahmed; formal analysis, Sara Ahmed; investigation, Fatma A. Omara, Sara Ahmed; resources, Fatma A. Omara, Sara Ahmed; data curation, Fatma A. Omara, Sara Ahmed; writing original draft preparation, Sara Ahmed; writing review and editing, Fatma A. Omara, Sara Ahmed; visualization, Fatma A. Omara, Sara Ahmed; supervision, Fatma A. Omara.

References

[1] R. Jain and D. Nagpal, "A Review on Cloud Computing and its Models", *International Journal of Advanced Trends in Computer Applications (IJATCA)*, Vol. 1, No. 2, pp. 1-4, 2020.

[2] A. Sharma and S. Tyagi, "Task Scheduling in Cloud Computing", *International Journal of Scientific & Engineering Research*, Vol. 7, No. 12, pp. 1-5, 2016.

[3] A. A. Motlagh, A. Meagher, and A. Masoud, "Task scheduling mechanisms in cloud computing: A systematic review", *International Journal of Communication System*, Vol. 33, No. 6, pp. 1-23, 2020.

[4] J. Meena and M. Vardhan, "Comparative analysis of scientific workflow scheduling algorithms in Cloud under Budget constraint", In: *Proc. of the 3rd International Conf. on Advances in Internet of Things and Connected Technologies (ICIOTCT-2018)*, Jaipur, India, 2018.

[5] Dr. T. L. A. Beena, "Resource Utilization of Workflow Scheduling Algorithms in Public Cloud", *International Journal of Scientific Research in Science, Engineering and Technology*, Vol. 4, No. 1, pp. 1132-1139, 2018.

[6] S. Jaybhaye and V. Attar, "A review on scientific workflow scheduling in cloud computing", In: *Proc. of the 2nd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2017.

[7] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments", *Concurrency and Computation-practice and experience*, Vol. 29, No. 8, pp. 1-32, 2017.

[8] F. Wu, Q. Wu, and Y. Tan," Workflow scheduling in cloud: a survey", *Journal of Supercomputing*, Vol. 71, No. 9, pp. 3373–3418, 2015.

[9] P. Rani1 and P. Nagpal, "Optimized Task Scheduling Algorithm for cloud computing environment", *International Journal of*

Emerging Trends & Technology in Computer Science (IJETTCS), Vol. 6, No. 5, pp. 39-47, 2017.

[10] W. Zheng, C. Wang, and D. Zhang, "A Randomization Approach for Stochastic Workflow Scheduling in Clouds", *Hindawi Publishing Corporation Scientific Programming*, Vol. 2016, No. 8, pp. 1-13, 2016.

[11] R. Singh and P. Pateriya, "Workflow Scheduling in Cloud Computing," *International Journal of Computer Applications*, Vol. 61, No. 11, pp. 38-40, 2013.

[12] N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey", *International Journal of Computer Network and Information Security*, Vol. 9, No. 8, pp. 16-22, 2017.

[13] A. Gade, M. N. Bhat, and N. Thakare," A Review on Meta-heuristic Independent Task Scheduling Algorithms in Cloud Computing", In: *Proc. of International Conference On Computational Vision and Bio Inspired Computing (ICCVBIC)*, pp. 1165–1180, Coimbatore, India, 2018.

[14] S. Madni, M. A. Latiff, M. Abdullahi, S. Abdulhamid, and M. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment", *Journal Plos One*, Vol. 12, No. 5, pp. 1-26, 2017.

[15] L. F. Bittencour, C. R. Senna, and E. Madeira, "Scheduling service workflows for cost optimization in hybrid clouds", In: *Proc. of International Conference on Network and Service Management (CNSM)*, Niagra, Canada, pp. 394–397, 2010.

[16] L. F. Bittencourt, E. R. Madeira, F. Cicerre and, L. Buzato, "A path clustering heuristic for scheduling task graphs onto a grid", In: *Proc. of 3rd International Workshop on Middleware for Grid Computing (MGC05)*, Grenoble, France, p. 1, 2005.

[17] S. Kaur, P. Bagga, R. Hans and, H. Kaur, "Review - Computer Engineering and Computer science Quality of Service (QoS) Aware Workflow Scheduling (WFS) in Computing: A Systematic Review", *Arabian Journal for Science and Engineering*, Vol. 44, No. 4, pp. 2867–2897, 2019.

[18] A. S. A. A. Haboobi, "Improving Max-Min scheduling Algorithm for Reducing the Makespan of Workflow Execution in the Cloud", *International Journal of Computer Applications*, Vol. 177, No. 3, pp. 5-7, 2017.

[19] M. Adhikari and T. Amgoth, "Efficient algorithm for workflow scheduling in cloud computing environment", In: *Proc. of 9th International Conference on Contemporary Computing (IC3)*, Noida, India, pp. 1-7, 2016.

[20] J. P. Pinto, A. Hukkeri, and S. B, "A Study On Workflow Scheduling Algorithms In Cloud", *International Journal of Latest Trends in Engineering and Technology*, Special Issue, pp. 43-48, 2017.

[21] M. Ghasemzadeh, H. Arabnejad, and J. Barbosa, "Deadline-Budget constrained Scheduling Algorithm for Scientific Workflows in a Cloud Environment", In: *Proc. of 20th International Conference on Principles of Distributed Systems (OPODIS)*, Dagstuhl, Germany, pp. 1–16, 2016.

[22] N. Almezeini and A. Hafez, "An Enhanced Workflow Scheduling Algorithm in Cloud Computing", In: *Proc. of 6th International Conference on Cloud Computing and Services Science*, Rome, Italy, pp. 67–73, 2016.

[23] S. Kaur, P. Bagga, R. Hans, and H. Kaur, "Quality of Service (QoS) Aware Workflow Scheduling (WFS) in cloud Computing: A Systematic Review", *Arabian Journal for Science and Engineering*, Vol. 4, No. 4, pp. 2867–2897, 2019.

[24] K. Dubeya, M. Kumarb, and S. C. Sharmaa, "Modified HEFT Algorithm for Task Scheduling in Cloud Environment", In: *Proc. of 6th International Conference on Smart Computing and Communications (ICSCC-2017)*, Kurukshetra, India, pp. 725-732, 2017.

[25] N. Rajak and D. Shukla, "Ambient Communications and Computer Systems", *Springer*, Singapore, Vol. 1097, 2020.

[26] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing", *Journal of Cloud Computing*, Vol. 10, No. 1, pp. 1-19, 2021.

[27] S. S. Murad, R. Badeel, N. S. A. Alsandi, R. F. Alshaaya, R. A. Ahmed, A. Muhammed, and M. Derahman, "Optimized MIN-MIN Task Scheduling Algorithm for Scientific Workflows in a Cloud Environment", *Journal of Theoretical and Applied Information Technology*, Vol. 100, No. 2, pp. 480-506, 2022.

[28] A. A. Rahayfeh, S. Atiewi, A. Abuhussein, and M. Almiani, "Novel approach to task scheduling and load balancing using the dominant sequence clustering and mean shift clustering algorithms", *Future Internet*, Vol. 11, No. 5, pp. 1-19, 2019.

[29] D. M. Abdelkader and F. A. Omara, "Dynamic task scheduling algorithm with load balancing for heterogeneous computing system", *Egyptian*

Informatics Journal, Vol. 13, No. 2, pp. 135-145, 2012.

[30] F. Adifard and S. M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment", *Cluster Computing Journal,* Vol. 24, No. 2, pp. 1075-1101, 2021.

[31] W. Chen and E. Deelman, "Workflosim: A tooklit for simulating scientific workflows in distributed environments", In: *Proc. of 8th International conference in E-sciences (e-Science)*, Chicago, USA, pp. 1-8, 2012.