Received: September 18, 2022. Revised: October 5, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Enhanced 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 worldwide. Workflow systems become a significant method for develop scientific applications. Therefore, workflow scheduling is considered one of the most important issues in cloud computing. It concerns about mapping tasks on cloud resources (i.e., Virtual machines (VMs)), to improve scheduling performance. Because the existing heterogeneous earliest finish time (HEFT) algorithm is considered one of the best and simplest algorithms, many algorithms have been proposed to improve the performance of the HEFT algorithm. According to our previous work, a modification has been done to HEFT algorithm to enhance the performance, called modified heterogeneous earliest finish time (M-HEFT). The goal of M-HEFT algorithm is to reduce make span, maximize resource utilization and increase load balance. According to the work in this paper, an enhancement has been added to our previous M-EFT algorithm to reduce the tradeoff among make span, resource utilization, and load balance, called enhanced modified heterogeneous earliest finish time (EM-HEFT). The enhanced EM-HEFT algorithm 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) by introducing new factors in priority value to be more aware about task requirements. According to task-VM phase, tasks are allocated to resources as in our previous M-HEFT algorithm. To evaluate the performance of the proposed EM-HEFT algorithm, a comparative study has been done among the proposed algorithm and four existed algorithms (HEFT, Efficient scheduling algorithm use critical path and static level attribute, Optimized Min-Min (OMin-Min) and our previous M-HEFT). The experimental results show that the proposed algorithm outperforms other algorithms by minimizing make span by 25%, improving resource utilization by 43% and load balance by 14% in average.

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

1. Introduction

High performance computing (HPC) is the ability to process data and perform complex calculations at high speeds by using concurrent processing for running application programs efficiently, reliably and quickly [1]. Cloud computing refers to the remote manipulation, configuration and access hardware and software resources. It provides online data storage, infrastructure and application [2].

There are four deployment models of cloud computing; public, private, hybrid, and community [3]. Public cloud allows systems and IT services to be easily accessible to the public. Private cloud allows systems and IT services to be easily accessible within an organization. Hybrid cloud It is

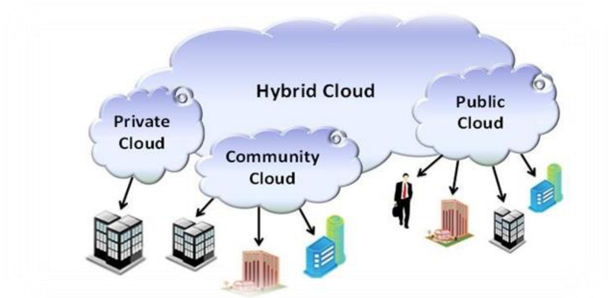
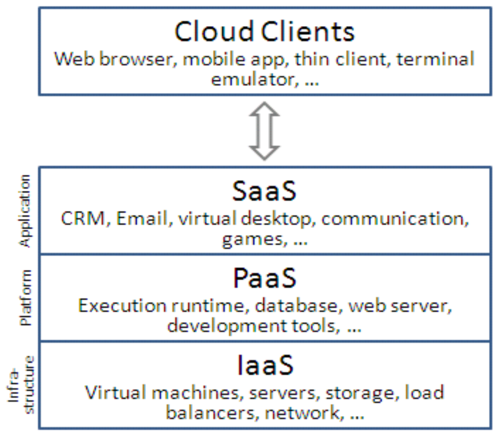


Figure.1 Cloud computing deployment models [5]

Received: September 18, 2022. Revised: October 5, 2022.


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure
Figure.2 Cloud computing service models [7]

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 [4]. Community cloud is a collaborative effort made for sharing infrastructure between multiple organizations (See Fig. 1) [5].

Cloud computing marked by popularity among scitists because of its services. Three services could be provided by cloud computing; software as a service (SaaS) where a software is deployed over the internet, platform as a service (PaaS) by providing a computing platform that allows user to design web applications quickly and easily, and infrastructure as a service (IaaS) by distributing cloud computing infrastructure such as servers, storage, network and operating systems on-demand service instead of buying them (See Fig. 2) [6].

One of the most important challenges of cloud computing is the lack of resources where organizations upload more workloads on the cloud while cloud technologies continue to rapidly advance to keep up with these needs. Resource Management is the process of allocating computing resource such as virtual machine (VMs), storage, networking and indirectly energy resources to a set of applications, to jointly meet the performance objectives of the infrastructure providers, the users of the cloud resources and applications [8]. Resource management includes different issues. One of the most important issues is task scheduling.

1.1 Task scheduling

Task scheduling is the fundamental issue in cloud environment. It is the process of allocating an application's tasks to suitable resources with considering dependency/independency between them to reduce make-span, maximize resource utilization, improve load balance, and achieve QOS parameters [9]. Based on the task dependency, the tasks can be classified as independent and dependent tasks. The tasks which do not require any communication with other tasks are called independent tasks. The dependent tasks differ from the independent tasks as the former have precedence order to be followed during the scheduling tasks, called workflow [10].

1.2 Applications workflow

Workflow considers the applications' tasks which presents in directed acyclic graph (DAG). The nodes in the DAG graph represent the problem's tasks and edges represent inter task dependencies. Each task in the workflow can communicate with other tasks [11]. There are two types of workflow; simple workflow represents real work which consists of group of tasks with sequence of activities and mechanisms used to perform individual or group tasks. Scientific workflow represents scientific applications which depend on other tasks with complexity in execution [12]. The work in this paper focuses on scientific workflows. There are common scientific workflows would be used as benchmark to evaluate the performance of the task scheduling algorithms such as MONTAGE, CYBERSHAKE, SIPHT, LIGO and EPIGENOMICS [13]. The work in this paper uses LIGO and EPIGENOMICS workflows as benchmark to implement and evaluate the proposed EM-HEFT algorithm

1.2.1. Workflow structure

A workflow is modeled by $G\ (T, E)$ where T is a set of nodes or tasks $\{t_1, t_2, t_3\ldots\ldots\ldots, t_n\}$ and E is the set of directed edges $\{\ eij \mid (T_i,T_j)\ \in\ E\}$ representing the dependencies between the tasks. Each task is a workflow task with an associated computation workload wl_i. Each edge eij represents T_i as the parent task of T_j and T_j is said to be the child task of T_i. After the complete execution of the parent task, a child task can be executed. If there is data transmission from T_i to T_j, the T_j can start only after all the data from Ti has been received. A task which does not have parent task is called input task, and a task which entry does not have child task is called exit task [14].

1.2.2. Workflow scheduling

Workflow scheduling is one of the prominent issues in cloud computing which is aimed at complete execution of workflows by considering their QoS requirements [15]. Therefore, an efficient scheduling algorithm is needed to reduce the tradeoff between execution time of tasks, resources utilization and satisfying load balance on various computing resources [16].

1.2.3. Workflow scheduling algorithms

Task scheduling algorithms can be categorized as Heuristic and Meta Heuristic, Heuristic algorithms which is problem-based that attempt to find solutions by applying the features of the problem in a complete manner. Their solution is based on learning and exploration where comprehensive scientific research is applied to find an optimal response and speed up the response process, Meta Heuristic algorithms unlike heuristic algorithms, these algorithms are problem independent, and are used to deal with different types of problems [17]. The work in this paper focuses on heuristic algorithms.

Despite huge numbers of algorithms are introduced to solve task scheduling, there are many updates can be added to improve the algorithms. In this paper, an algorithm has been introduced, called EM-HEFT, to improve the performance of our previous M-HEFT algorithm by reducing the tradeoff among load balancing, resource utilization and make span [18]. Enhanced EM-HEFT algorithm consists of two phases; task prioritization and task-VM mapping. The task prioritization is implemented with new factors added to the rank equation to be more aware about task characteristics which makes the algorithm performs better. 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; literature review is presented in section 2. Section 3 illustrates the principles of the proposed task scheduling algorithm. The Performance criteria which used to evaluate the enhanced EM-HEFT algorithm using the WorkflowSim simulator are illustrated in section 4. The experiment results of the enhanced EM-HEFT algorithm are discussed in section 5. Finally, section 6 includes conclusion & future work.

2. Literature review

Workflow scheduling assigns tasks based on their dependencies on the shared resources that the workflow scheduler controls. However, it is important to assign task to certain resources in order to provide high quality of service [19]. Therefore, workflow scheduling is a challenging optimization problem with a lot of research in the last years. Several evolutionary algorithms have been introduced to solve this problem [20].

In [21], heterogeneous earliest finish time (HEFT) algorithm was proposed. The goal of this algorithm is to reduce make span. It works on two phases; task ranking phase and task–VM mapping phase. According to task ranking phase, a rank assign for each task based on average execution time of each task and the average communication time between the resources of two tasks. After that the tasks will be sorted in descending order in a list. In the task–VM mapping phase, the highest rank task will be assigned to the VM that produces earliest finish time. Finally, this task is removed from the list and the process is repeated until each task is assigned. The main limitation of this algorithm is that the algorithm cares only to minimize make span only.

In [22], min-min scheduling algorithm was proposed. The goal of this algorithm is to reduce make span. It works by scheduling the task with minimum size to the resource that has the minimum completion time (MCT). Finally, this task is removed from the set of unassigned tasks and the process is repeated until each task is assigned. However, the limitation of this algorithm is that it schedules the small tasks at first and leave large one at the end of scheduling process. It works fine if the number of smaller tasks is greater than the number of larger ones.

In [23], resource awareness scheduling algorithm (RASA) was proposed. The goal of this algorithm is to reduce make span. RASA is a hybrid algorithm composed of two traditional scheduling algorithms; max-min and min-min. RASA uses the advantages of max-min and min-min algorithms and covers their disadvantages. Based on concept of completion time of each task, it works on two phases. In the first phase, the expected completion time for each task is calculated. In the second phase, max-min and min-min algorithms are applied alternatively to schedule task according to resource ID where if resource ID is odd min-min is applied first otherwise max-min is applied. The algorithm advantages of this algorithm are no longer waiting of VM for larger tasks or smaller tasks and satisfying load balance. However, the limitation of this algorithm is that it unable to load balance when the number of larger tasks increases. Another issue is that it only concerned with the number of the resources to be odd or even.

In [24], an improved max-min task scheduling algorithm was proposed. The goal of this algorithm is reducing make span. It calculates the average of execution time for all tasks in the workflow. Then, max-min is used when receiving a task with execution time smaller than the average of execution time. Otherwise, a task with execution time greater than or equal to the average of execution time is assigned to the VM with minimum completion time among all the VMs regardless of VM availability. The limitation of this algorithm is that it cares about make span only.

In [25], an efficient workflow scheduling algorithm (EWSA) was proposed. The goal of this algorithm is to maximize resource utilization and meeting the deadline. 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. The limitation of this algorithm is that is not considered load balance among VMs.

In [26], a MaxChild algorithm was proposed. The goal of this algorithm is to improve the system throughput with appropriate resource utilization and high performance by obeying the required QoS parameters which specified by the user. MaxChild start with the task which has maximum number of child to be scheduled first to guarantee that the maximum number of tasks could be available for the next schedules and resource are utilized properly. However, the limitation of this algorithm 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.

In [27], a modification has been done to the heterogeneous earliest finish time (HEFT) algorithm to enhance the performance on the cloud environment. The goal of this algorithm is to reduce make span. It works on two phases; first phase is task ranking phase where assign a rank for each task. Then, sort them descendingly based on their rank. Second phase is task – VM mapping phase which concerns about assigning the task to resource that produce earliest finish time. According to this modification, the priority for each task in the DAG has been defined by calculating the order of execution which define by the result of (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. The algorithm outperforms the HEFT algorithm with respect to make span. The limitation of this algorithm is that the algorithm cares only to minimize make span only.

In [28], a combination between heterogeneous earliest finish time ranking algorithm and modified balance minimum completion time resource selection algorithm was proposed for concurrent workflow. The goal of this algorithm is to reduce make span and make resources more balanced. The modified heterogeneous earliest finish time ranking algorithm considers a communication time of parent task that plays a significant role when tasks have rich communication in the workflow. Then, the modified balance minimum completion time resource selection algorithm is used to check load of all machines to move the tasks between the highly loaded machines to the lightly loaded machines. By this modification, the algorithm outperforms the HEFT algorithm with respect to make span. The limitation of this algorithm is that other QoS metrics didn't considered.

In [29], an efficient task scheduling algorithm for DAG in cloud computing environment has been proposed. The goal of this algorithm is reduce make span. The algorithm works on of two phases; task priority phase and resource selection phase. In task priority phase, the priority of the tasks is defined using critical path and static level (CPS) attributes. Then, the tasks are sorted in descending order. In 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. However, the limitation of this algorithm is suffered from load imbalance on VMs.

In [30], multi-objective workflow optimization strategy (MOWOS) was proposed. The goal of this algorithm is reducing execution cost with make span for workflow. In the other words, the aim of MOWOS is all tasks executed on their deadlines with reduced time and budget. MOWOS include three sub algorithms; task spiriting algorithm to break down large tasks into smaller chunks to reduce their schedule length, and two task allocation algorithms, minimum VM (MinVM) selection algorithm, and maximum VM (MaxVM) selection algorithm. However, the limitation of this algorithm is suffered from load imbalance on VMs.

In [31], a list scheduling with task duplication (LSTD) algorithm was proposed. The goal of this algorithm is to minimize the make span of workflow applications. LSTD algorithm mainly consists of three steps. In the first step, the rank of the tasks is calculated for deciding the scheduling order. The second step is responsible for duplicating the entry task on the processor only if it increases the overall efficiency and avoids processor overloading. Finally, in the last step, the processor is assigned to the tasks based on the popular insertion-based policy that attempts to insert the task among two earlier assigned tasks on a given processor in earliest idle time. LSTD outperforms other existing algorithms with respect to make span. Similar to other proposed algorithms, the limitation of this algorithm is that more QoS metrics didn't considered.

In [32], an enhanced heterogeneous earlier finish time based on rule (EHEFT-R) task scheduling algorithm was proposed. The goal of this algorithm is to optimize task execution efficiency, quality of service (QoS) and energy consumption. This algorithm, works with task ranking and VM allocation phases in parallel. After one or more tasks determine their execution order, they are immediately arranged on the VM with respect to the earliest completion time. After all tasks are sorted, the virtual machine selection process is also completed. The rank value does not completely guarantee that tasks allocate on the optimal VM in all situations if tasks have different priority levels. But, at the same priority level, remapping rules are applied start from the task with the highest rank value. If two adjacent tasks are not in the same layer, the VM assignments of the two tasks will not be changed. Otherwise, if two adjacent tasks are in the same layer and their earliest completion times fall on different VM, the virtual machine allocation will not be changed; if two adjacent tasks are in the same layer and their earliest completion time falls on the same VM, compare the EFT rank of the two tasks, the one with the larger value will be assigned to this VM. The limitation of this algorithm is that load balance metrics need to be considered.

In [33], a new Min-Min algorithm called Optimized Min-Min (OMin-Min) algorithm. OMin-Min is designed for scientific workflow. The goal of this algorithm is to reduce make span and try to avoid neglecting execution time. OMin-Min define tasks that have minimum and maximum execution times (MinT and MaxT), and then the task with minimum execution time will be assigned to resource that produces minimum execution time. Otherwise, the task with maximum execution time assigns to resource that produces minimum execution time. The limitation of this algorithm is that load balance metrics need to be considered.

In [34], an improvement of the existing HEFT and Enhancement-HEFT (E-HEFT) algorithms has been introduced, called Load Balance HEFT (LB-HEFT). The goal of this algorithm is to reduce make span, improve load balance and reduce resource utilization. The principle of the proposed LB-HEFT algorithm is to allocate the application's tasks to VMs by considering the heterogeneous clusters. It works on two phases; tasks ranking and Task-VM matching phases. In the task ranking phase, a rank is assigned to each task. In task-VM matching phase schedules tasks on appropriate VMs taking into account load balancing along with the earliest execution time and optimizing resource usage. The limitation of this algorithm is more metrics need to be considered such cost. This algorithm would be improved by considering the task's weight in the Task Ranking phase. So, the work in this paper concerns this issue.

In [18], a modification has been done to the heterogeneous earliest finish time (HEFT) algorithm to enhance the performance on the cloud environment. Called modified heterogeneous earliest finish time (M-HEFT). The goal of this algorithm is to reduce make span, maximize resource utilization and increase load balance. It works on two phases; first phase is task ranking phase where assign a rank for each task. Then, sort tasks descendingly with respect to their ranks. Second phase is task–VM mapping phase where, average task length (AL) for all nodes is calculated. Then, if the task length is greater than or equal AL, it will be mapped to resource that produce earliest finish time. Else, the task will be mapped to the laziest VM that produce earliest finish time. These will be repeated till all tasks are assigned to specific resource. However, the limitation of this algorithm is that make span need to be improved.

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

Table 1 list the comparison between the aforementioned algorithms.

3. The proposed task scheduling algorithm

The proposed task-scheduling algorithm is based on our previous M-HEFT algorithm with some modifications to improve resource utilization, and load balance, in addition to, make span [18]. The proposed algorithm is called enhanced modified HEFT (EM-HEFT). The goal of EM-HEFT is to make the task' rank to be more powerful by adding new factors in the rank equation which will lead to maximize resource utilization, improve load balance,

Table 1. Comparison of workflow scheduling algorithms

Scheduling algorithm	**Scheduling parameters**	**Finding**	**Environment**
HEFT [21]	Make span	This algorithm efficient for reducing the make span	Grid environment
Min-Min [22]	Make span	This algorithm efficient for reducing the make span with small task length	Grid environment
RASA [23]	Make span	This algorithm outperforms Min-Min and Max-Min and opportunistic load balancing (OLB) algorithms with respect to make span.	Grid environment
An improved Max-Min task scheduling [24]	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) [25]	Resource utilization and deadline	This algorithm maximizes the resource utilization and meet the deadline of the application	Cloud environment
MaxChild [26]	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
A modification has been done to the Heterogeneous Earliest Finish Time (HEFT) algorithm [27]	Make span	This algorithm reduces the make span and satisfies load balancing compare to existing HEFT and CPOP algorithms.	Cloud environment
A modification has been done to the Heterogeneous Earliest Finish Time (HEFT) algorithm [28]	Make span and scheduling length ratio	This algorithm outperforms HEFT with respect to make and schedule length ratio.	Cloud environment
An efficient task scheduling algorithm for DAG in cloud computing environment [29]	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) [30]	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
List Scheduling with Task Duplication (LSTD) algorithm [31]	Make span and scheduling length ratio	The LSTD outperforms other existing algorithms with respect to make span and schedule length ratio.	Cloud environment
Enhanced Heterogeneous Earlier Finish Time Based on Rule (EHEFT-R) [32]	Make span, energy consumption and QOS	The proposed EHEFT-R algorithm has better make span and energy consumption than HEFT and NSGAII algorithms.	Cloud environment
Optimized Min-Min (OMin-Min) [33]	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
Load Balance Heterogeneous Earlier Finish Time (LB-HEFT) [34]	Make span, resource utilization and load balance	The algorithm outperforms the HEFT, and E-HEFT with respect to make span, resource utilization and load balance	Cloud environment
M-HEFT [18]	Make span, resource utilization and load balance	The algorithm outperforms the HEFT, algorithm in [30], and algorithm in [33] with respect to make span, resource utilization and load balance	Cloud environment

Received: September 18, 2022. Revised: October 5, 2022.

and reduce make span.

The proposed EM-HEFT algorithm consists of two phases; task prioritization phase to assign priority for each task in the DAG, and Task-VM mapping phase to allocate each task in the DAG to suitable VM. The Task-VM phase is implemented as in our previous M-HEFT algorithm [18]. The novelty of EM-HEFT is in task prioritization phase that will be discussed in details.

3.1 Task prioritization phase

A modification has been done to enhance task prioritization phase of our previous M-HEFT algorithm with respect to make span. The modification has been done based on four algorithms max child algorithm [26] which concerns the number of the childs to define the rank of tasks, load balancing scheduling algorithm for concurrent workflow algorithm [28] which concerns average computation cost of the task on all VMs and the communication cost between predecessor tasks with respect to current task, modified HEFT algorithm for task scheduling in cloud environment [27] which concerns the predecessor of each task, and an efficient list scheduling algorithm with task duplication for scientific big data work flow in heterogeneous computing environments algorithm [31] which concerns the weight speed of successor task beside communication.

According to the work in this paper, the task's rank is defined by summation of weight speed as in [31], maximum of predecessor rank and its communication [28], maximum of communication of task and its successor as in [27] and the number of task's successor [26]. Therefor the task's rank is calculated using Eq. (1).

$$Rank(T_i) = w_i + \max_{tk\in tpred\{t_i\}} \left(c(i,j) + rank(T_j)\right) + \max_{Tk\in tsucc\{T_i\}} c(i,k) + number\ of\ successors\ of\ T_i \quad (1)$$

Where, ***W***$\boldsymbol{W_i}$ is the difference between the highest and lowest computation time of task $\boldsymbol{T_i}$ on $\boldsymbol{VM_m}$ and $\boldsymbol{VM_n}$, respectively divided by speedup of these VMs. $\boldsymbol{T_{pred}}$ is the set of predecessors of $\boldsymbol{T_i}$, ***c (i, j)*** is the average communication cost between task $\boldsymbol{T_i}$ and $\boldsymbol{T_j}$, $\boldsymbol{T_j}$ is the successor of $\boldsymbol{T_i}$, $\boldsymbol{T_{succ}}$ is the set of successors of $\boldsymbol{T_i}$, and ***C (i, k)*** is the average communication cost between task $\boldsymbol{T_i}$ and $\boldsymbol{T_k}$.

Therefore, the proposed EM-HEFT algorithm is become more knowledgeable because it concerns communication cost for task's successors and predecessors and number of successors, to make the rank value more valuable and effective.

The task priority phase starts by computing computation cost ***(CCT)*** for each task $\boldsymbol{T_i}$ in the DAG on each $\boldsymbol{VM_j}$ using Eq. (2).

$$CCT(T_i, VM_j) = \frac{T_i.length}{VM_j.MIPS} \quad (2)$$

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

Then, the communication cost is calculated between tasks and their successors $\boldsymbol{C\ (T_i,T_k)}$, and their predecessor $\boldsymbol{C\ (T_i,T_j)}$. Then, set number of children for each task. After that, the rank for each task $\boldsymbol{T_i}$ is calculated using the Enhanced M-HEFT algorithm (see Eq. (1)). Finally, the tasks are sorted in a list in decreasing order based on their rank value.

Algorithm 1: Task ranking phase of EM-HEFT algorithm.

Input: DAG and VMs configuration.
Output: list of tasks in decreasing order based on their rank

```
1: set the computation cost for each task on each resource $CCT_{i,j}$
2: set the communication cost between tasks and their successors $C_{i,k}$ , and their predecessor $C_{i,j}$
3: set the number of successors for each task $T_i$ in DAG
4: for each task $i=1$ to $T_i$ in DAG
    $Rank(T_i) = w_i + \max_{tk\in tpred\{t_i\}} (c(i,j) + rank(T_j)) + \max_{Tk\in tsucc\{T_i\}} c(i,k) + number\ of\ successors\ of\ T_i.$
5: end for
6: arrange tasks in a list in decreasing order based on their rank
```

3.1.1. Pseudo code of the task ranking phase of EM-HEFT algorithm

The pseudo code of the task ranking phase is described in Algorithm. 1.

3.2 Task – VM mapping phase

The task-VM Mapping phase in our previous M-HEFT algorithm will be used to select the best VM for each task by calculating the average task Length (***AL***) for all tasks using Eq. (3) [18].

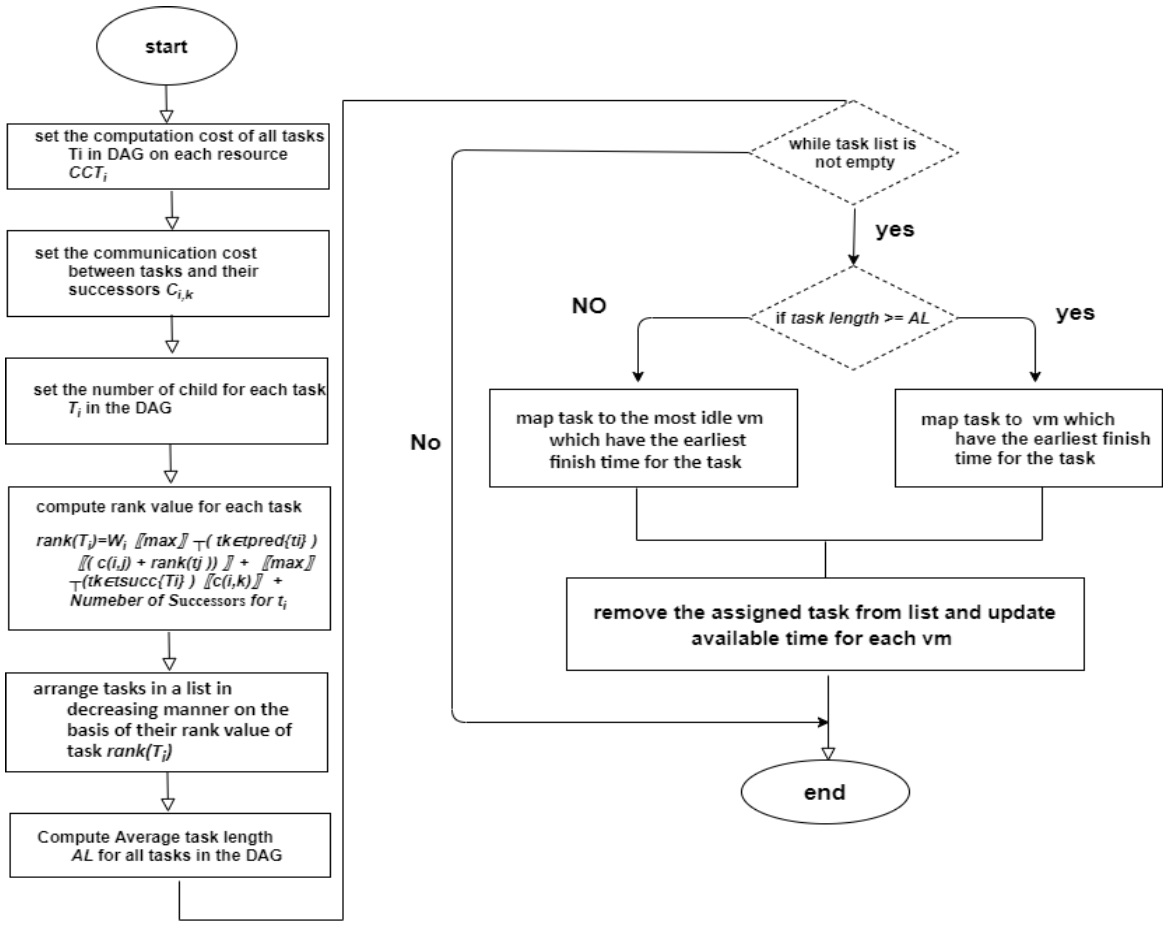

start
set the computation cost of all tasks Ti in DAG on each resource CCT_i
set the communication cost between tasks and their successors C_i,k
set the number of child for each task T_i in the DAG
compute rank value for each task
rank(T_i)=W_i ⟦max⟧ ┬(tk∈tpred{ti}) ⟦(c(i,j) + rank(tj)) ⟧ + ⟦max⟧ ┬(tk∈tsucc{Ti}) ⟦c(i,k)⟧ + Numeber of Successors for t_i
arrange tasks in a list in decreasing manner on the basis of their rank value of task rank(T_i)
Compute Average task length AL for all tasks in the DAG
while task list is not empty
yes
NO
if task length >= AL
yes
map task to the most idle vm which have the earliest finish time for the task
map task to vm which have the earliest finish time for the task
No
remove the assigned task from list and update available time for each vm
end

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

$$AL = \frac{\sum T_i\, length}{TaskNum} \tag{3}$$

Where, T_i *length* is the needed computation time of task $\boldsymbol{T_i}$, and ***taskNum*** is the number of tasks in the DAG.

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

$$Finish\ Time(T_i, VM_j) = CCT(T_i, VM_j) + ST(T_i, VM_j) \tag{4}$$

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

After assigning the task on suitable VM according to Eqs. (4), (5), remove the task from the list. Repeat the steps till all tasks are assigned to the VMs.

3.2.1 Pseudo code of the task-VM mapping phase of EM-HEFT algorithm

The pseudo code of the task-VM mapping phase of EM-HEFT algorithm is described in Algorithm. 2.

Algorithm 2: Task-VM mapping phase.

Input: List of tasks based on their rank and VMs configuration.

Output: Mapping scheme for the requested tasks cloudlets on the available resources VMs.

```
1: compute Average task length (AL) for all tasks in the DAG
2: for each task in ready list
3: check if task length greater than or equal AL
4: map task to VM which has the earliest finish time
5: else if the task length less than AL
6: map task to the most idle VM which has earliest finish time
7: end for
8: end
```

3.3 Flowchart of the proposed EM-HEFT algorithm

Fig. 3 represents the flowchart of scheduling tasks of DAG according to the proposed EM-HEFT algorithm.

4. Performance evaluation of the proposed EM-HEFT algorithm

4.1 Performance metrics

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

Make span is the maximum time required to complete the entire DAG tasks. Make span should be reduced. Eq. (6) is used to calculate make span [35].

$$Make\ span = max\{CT_i\} \tag{6}$$

Here $\boldsymbol{CT_i}$ is the completion time of the longest task $\boldsymbol{T_i}$.

Resource utilization rate ***(RUR)*** is the ratio between the total occupied time of $\boldsymbol{VM_i}$ and the make span of the parallel application in percentage (see equation Eq. (7), and Eq. (8)) [37]. Resource utilization should be maximized.

$$RUR(VM_j)\% = \left(\frac{\sum vm_j\ Busy\ Time}{Make\ span}\right)X100 \tag{7}$$

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

Ideal load balance ***(ILB)*** is the ratio between the total number of tasks and the number of VMs, which is calculated by Eq. (9) [36].

$$Ideal\ Load\ Balance(ILB) = Number\ of\ Tasks/Number\ of\ used\ VM \tag{9}$$

Difference from ideal rate of load balance (***DLB***) is the difference between actual load balance and the ideal load balance ***(ILB)*** in $\boldsymbol{VM_i}$. It is calculated using Eq. (11) [36]. ***DLB*** should be minimized.

$$DLB(VM_j)\% = \sum Number\ of\ tasks(VM_j) - ILB\ (VM_j) \tag{10}$$

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. (12) [38].

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

Improvement rate (***IRm***) in terms of used metrics (make span, resource utilization and load balance), this ratio will define the improvement rate with respect to used metrics (***M)*** using EM-HEFT relative to the current HEFT and the strategies mentioned in [21, 29, 33, 18]. It is computed using Eq. (12).

$$IR_m = \left(\frac{ABS(M(existing\ algorithm) - M(proposed\ algorithm))}{M(existing\ algorithm)}\right)X100 \tag{12}$$

Here ***ABS*** is the absolute value that neglects the sign of the number.

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

4.2 Experimental environment

The proposed algorithm has been implemented using WorkflowSim1.0 toolkit integrated into Net Beans IDE 8.0.2 with the configurations shown in Table 2. WorkflowSim is an extension of the CloudSim framework [39]. The experiments have done using two type of workflows; Ligo, and Epigenomics.

5. Performance evaluation of the proposed EM-HEFT algorithm

To evaluate the performance of the proposed algorithm, a comparative study has been contacted among the proposed EM-HEFT algorithm, the heterogeneous earliest finish time (HEFT) algorithm [21], the algorithm mentioned in [29], the algorithm mentioned in [33] and our previous M-HEFT algorithm [18] with respect to make span, resource utilization, and load balancing metrics. This study has been implemented with considering heterogeneous environment using WorkflowSim,

Table 3. Make span results for 100 tasks of Ligo

Algorithm	**5 VMs**	**10 VMs**	**15 VMs**	**20 VMs**
HEFT [21]	43195.33	10043.61	4784.83	3360.40
Algorithm [29]	34479.03	17224.36	9383.23	6204.83
Algorithm [33]	34674.23	17598.23	10173.00	7820.00
M-HEFT[18]	31710.37	9023.57	4646.04	2836.96
Enhanced EM-HEFT	31267.74	8972.39	4017.25	2817.72

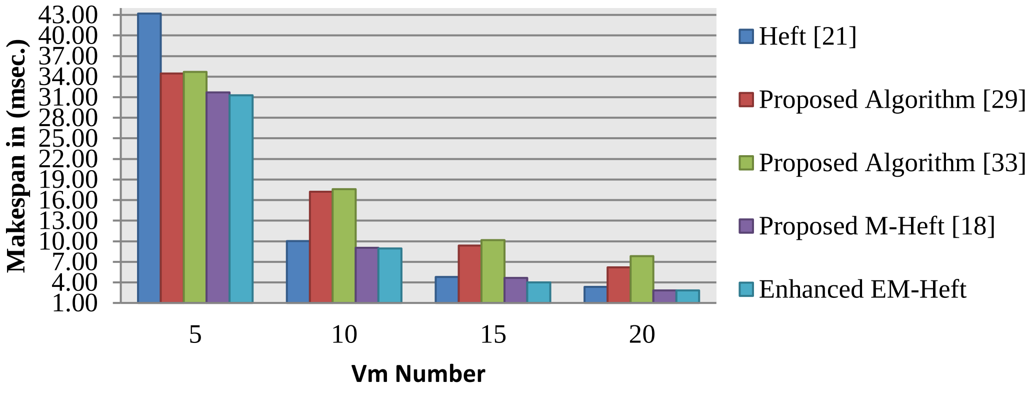


Figure. 4 Make span results for 100 tasks of Ligo

Table 4. Make span results for 1000 tasks of Ligo

Algorithm	**5 VMs**	**10 VMs**	**15 VMs**	**20 VMs**
HEFT [21]	316515.75	255772.84	62360.67	105685.84
Algorithm [29]	307040.59	87822.00	50968.25	29652.74
Algorithm [33]	308075.18	87945.32	51598.46	30757.41
M-HEFT [18]	304147.05	87225.67	40624.22	22723.30
Enhanced EM-HEFT	295315.59	79820.98	39227.75	21491.95

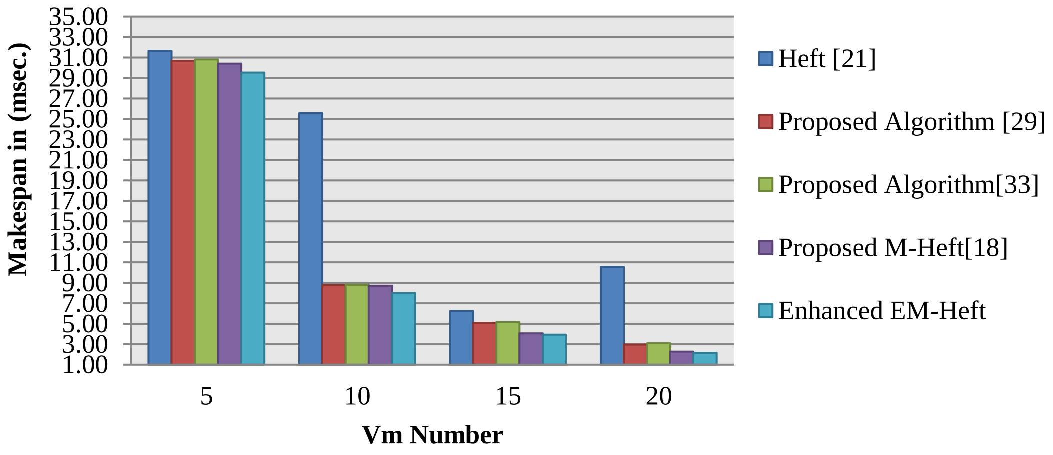


Figure. 5 Make span results for 1000 tasks of Ligo

Table 5. Make span results for 100 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	569754.69	193399.72	141876.45	114244.72
Algorithm [29]	656837.59	411631.95	189023.29	164725.11
Algorithm [33]	675509.87	413795.00	225311.52	250584.03
M-HEFT [18]	544848.60	183010.05	111433.28	81473.38
Enhanced EM-HEFT	533676.02	165486.02	106814.13	79425.13

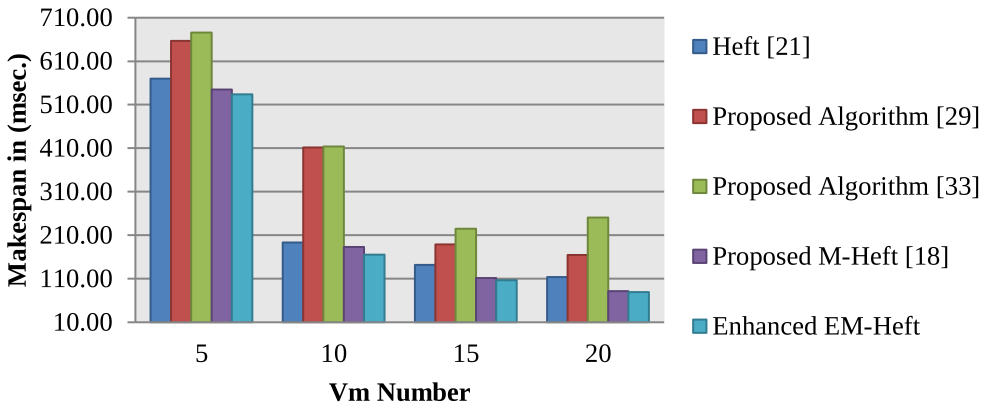


Figure. 6 Make span results for 100 tasks of epigenomics

Table 6. Make span results for 1000 tasks of epigenomics

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	5178814.39	2921063.95	4249865.15	2230593.05
Algorithm [29]	5191570.65	1653208.62	867429.26	532190.83
Algorithm [33]	5202232.30	1691359.69	898372.57	559629.85
M-HEFT [18]	5155803.00	1420166.50	714205.34	451127.66
Enhanced EM-HEFT	5155713.76	1418864.24	696630.71	407408.56

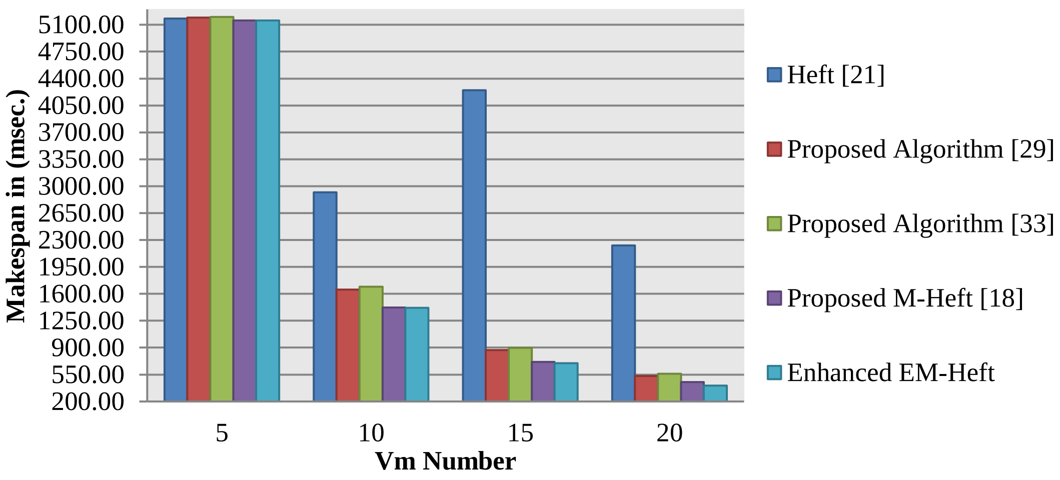


Figure. 7 Make span results for 1000 tasks of epigenomics

and two benchmarks, Ligo and Epigenomics with 100 and 1000 tasks, and 5, 10, 15 and 20 VMs.

5.1 Make span

The implementation results of the comparative study among our enhanced EM-HEFT algorithm, HEFT [21], the algorithm in [29], the algorithm in [33], and our previous M-HEFT algorithm in [18] 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 follows.

5.1.1. For 100 and 1000 tasks of ligo

Make span for 100 tasks of Ligo are presented in Table 3, and Fig. 4. While make span for 1000 tasks of Ligo are presented in Table 4, and Fig. 5.

By considering 100 tasks in Ligo and the implementation results presented in Table 3 and Fig. 4, it is found that the enhanced EM-HEFT algorithm improves make span by 18% with respect to HEFT algorithm [21], 42% with respect to algorithm in [29], 46% with respect to algorithm in [33], and 4% with respect to our previous M-HEFT algorithm [18] in average. A By considering 100 tasks in Ligo and results presented Table 4 and Fig. 5, the enhanced EM-HEFT algorithm improves make span by 48% with respect to HEFT algorithm [21], 16% with respect to algorithm in [29], 17% with respect to algorithm in [33], and 5% with respect to our previous M-HEFT algorithm in [18] in average.

5.1.2. 100 and 1000 tasks of epigenomics

Make span for 100 tasks of Epigenomics are presented in Table 5, and Fig. 6. While make span for 1000 tasks of Epigenomics VMs are presented in Table 6, and Fig. 7.

By considering 100 tasks in Epigenomics and implementation results presented in Table 5 and Fig. 6, it is found that the enhanced EM-HEFT algorithm improves make span by 19% with respect to HEFT algorithm [21], 43% with respect to algorithm in [29], 50% with respect to algorithm in [33] and 5% with respect to our previous M-HEFT algorithm in [18] in average. By considering 1000 tasks in Epigenomics and results presented results in Table 6 and Fig. 7, it is found that the enhanced EM-HEFT algorithm improves make span by 54% with respect to HEFT algorithm [21], 15% with respect to algorithm in [29], 14% with respect to algorithm in [33] and 3% with respect to our previous M-HEFT algorithm in [18] in average.

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

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	38.94	41.87	40.25	31.28
Algorithm [29]	48.80	24.41	19.92	16.94
Algorithm [33]	47.58	24.31	18.62	15.34
M-HEFT [18]	53.04	46.60	42.25	37.05
Enhanced EM-HEFT	53.45	47.01	43.40	37.20

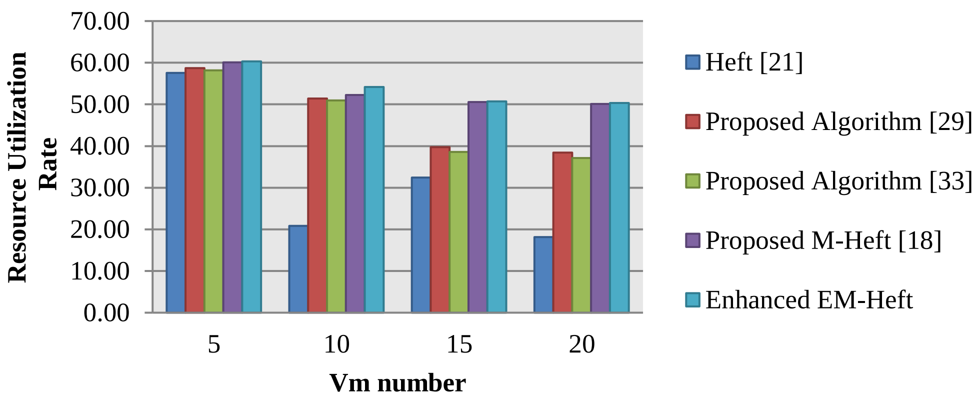


Figure. 8 Resource utilization rate results for 100 tasks of Ligo

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

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	57.55	20.81	32.46	18.18
Algorithm [29]	58.74	51.38	39.71	38.39
Algorithm [33]	58.14	50.95	38.56	37.15
M-HEFT [18]	60.05	52.25	50.54	50.10
Enhanced EM-HEFT	60.35	54.20	50.75	50.30

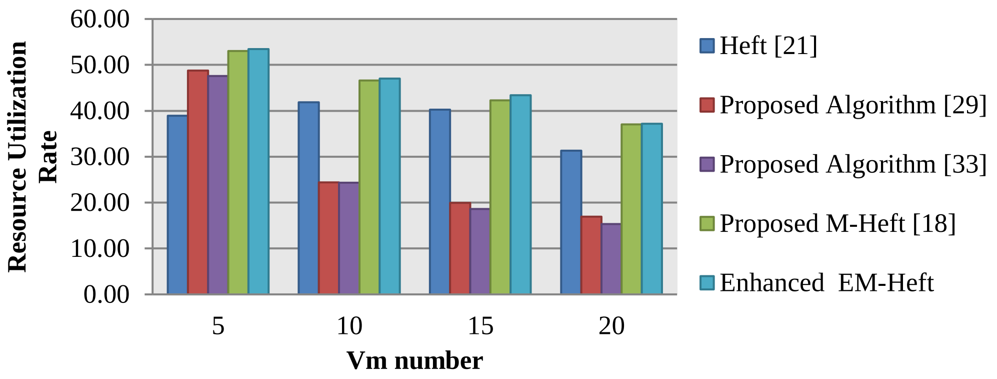


Figure. 9 Resource utilization rate results for 1000 tasks of Ligo

5.2 Resource utilization evaluation

The implementation results of the comparative study among our enhanced EM-HEFT, HEFT [21], algorithm in [29], algorithm in [33], and our previous M-HEFT algorithm in [18] 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.

5.2.1. For 100 and 1000 tasks of Ligo

Resource utilization results for 100 tasks of Ligo are presented in Table 7, and Fig. 8. While resource utilization results for 1000 tasks of Ligo are presented in Table 8, and Fig. 9.

By considering 100 tasks in Ligo and resource utilization results presented in Table 7 and Fig. 8, it is found that the enhanced EM-HEFT algorithm improves resource utilization by 19% with respect to HEFT algorithm [21], 85% with respect to algorithm in [29], 95% with respect to algorithm in [33], and 1% with respect to our previous M-HEFT algorithm in [18] in average. By considering 1000 tasks in Ligo and resource utilization presented in Table 8 and Fig. 9, it is found that the enhanced EM-HEFT algorithm improves resource utilization by 99% with respect to HEFT algorithm [21], 17% with respect to algorithm in [29], 19% with respect to algorithm in [33], and 1% with respect to our previous M-HEFT algorithm in [18] in average.

5.2.2. For 100 and 1000 tasks of epigenomics

Resource utilization results for 100 tasks of Epigenomics are presented in Table 9, and Fig. 10. While resource utilization results for 1000 tasks of Epigenomics are presented in Table 10, and Fig. 11.

By considering 100 tasks in Epigenomics and resource utilization results presented in Table 7 and Fig. 8, it is found that the enhanced EM-HEFT algorithm improves resource utilization by 22% with respect to HEFT algorithm [21], 83% with respect to algorithm in [29], 100% with respect to algorithm in [33], and 8% with respect to our previous M-HEFT algorithm in [18] in average. By considering 1000 tasks in Epigenomics and results presented Table 8 and Fig. 9, the enhanced EM-HEFT algorithm improves resource utilization by 99% with respect to HEFT algorithm [21], 16% with respect to algorithm in [29], 22% with respect to algorithm in [33], and 2% with respect to our previous M-HEFT algorithm in [18] in average.

5.3 Load balance rate

The implementation results of the comparative study among our enhanced EM-HEFT, HEFT [21],

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

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	56.64	41.72	25.27	17.66
Algorithm [29]	49.13	19.60	18.97	12.24
Algorithm [33]	48.76	18.81	16.42	10.56
M-HEFT [18]	59.23	44.09	32.18	24.76
Enhanced EM-HEFT	59.80	45.00	32.90	25.50

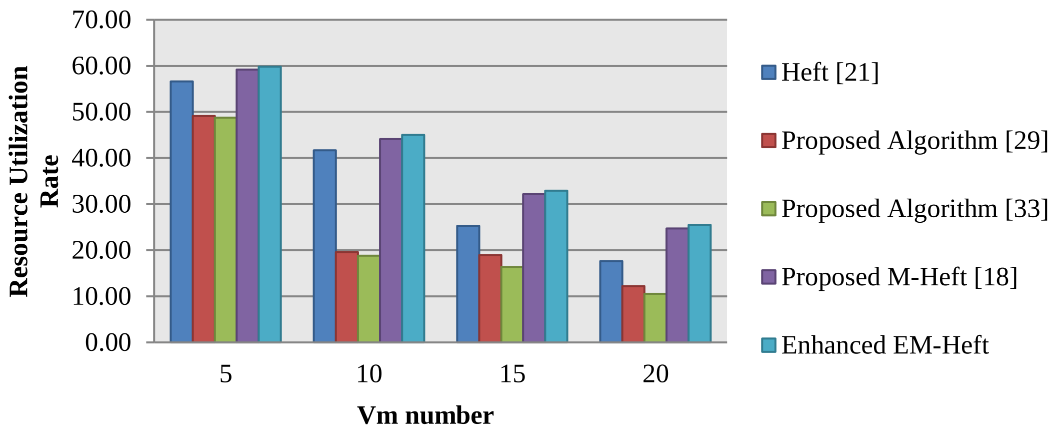


Figure. 10 Resource utilization rate results for 100 tasks of epigenomics

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

Algorithm	5 VMs	10 VMs	15 VMs	20 VMs
HEFT [21]	59.15	26.39	20.06	18.04
Algorithm [29]	59.40	46.63	39.50	36.22
Algorithm [33]	59.13	46.24	36.32	33.48
M-HEFT [18]	60.45	54.29	47.98	42.72
Enhanced EM-HEFT	60.32	54.10	49.10	44.20

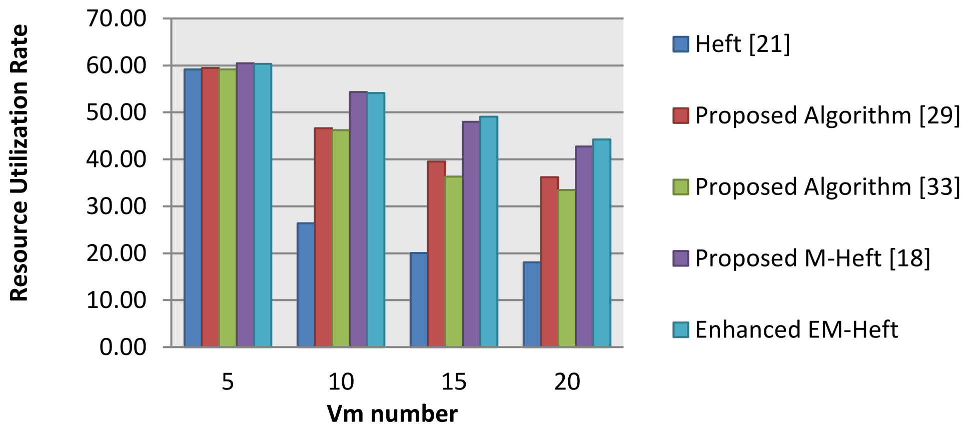


Figure. 11 Resource utilization rate results for 1000 tasks of epigenomics

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 [21]	8.8	4.0	3.20	2.70
Algorithm [29]	7.0	3.8	2.3	1.4
Algorithm [33]	7.1	3.9	2.5	1.7
M-HEFT [18]	6.8	2.0	2.0	1.0
Enhanced EM-HEFT	6.7	1.9	1.9	1.0

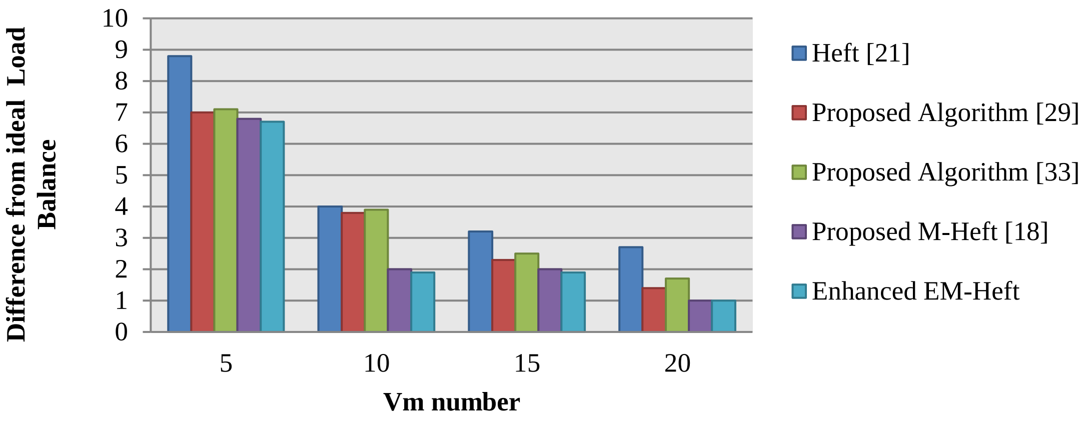


Figure. 12 The average difference from ideal load balance (ILB) results for 100 tasks of Ligo

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 [21]	79.5	63.4	33.3	34.3
Algorithm [29]	78.6	38.0	30.3	22.5
Algorithm [33]	78.8	38.2	30.5	22.8
M-HEFT [18]	78.4	36.3	27.4	19.4
Enhanced EM-HEFT	78.2	35.8	27.21	19.2

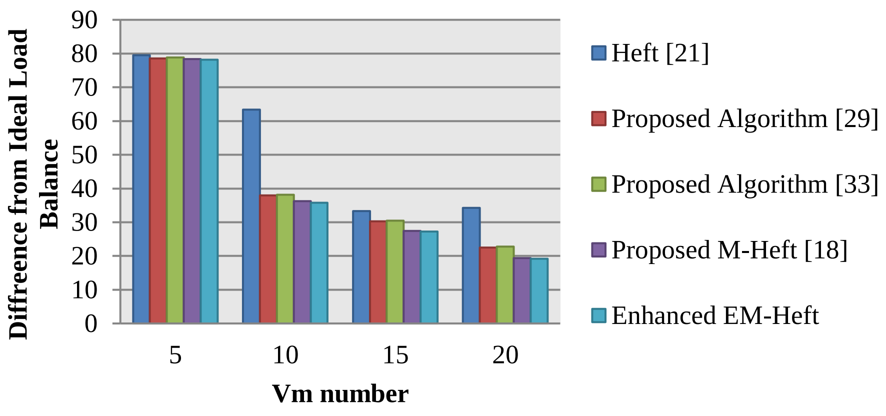


Figure. 13 The average difference from ideal load balance (ILB) results for 1000 tasks of Ligo

algorithm in [29], algorithm in [33], and our previous M-HEFT algorithm in [18] 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.

5.3.1. For 100 and 1000 tasks of ligo

Load balance rate results for 100 tasks of Ligo are presented in Table 11, and Fig. 12. While load balance rate results for 1000 tasks of Ligo are presented in Table 12, and Fig. 13.

According to the results presented in Table 11 and Fig. 12, it is found that the enhanced EM-HEFT algorithm improves load balance by 45% with respect to HEFT algorithm [21], 25% with respect to algorithm in [29], 31% with respect to algorithm in [32], and 3% with respect to our previous M-HEFT algorithm in [18] in average by considering 100 tasks in Ligo. According to the results presented in Table 12 and Fig. 13, the enhanced EM-HEFT algorithm improves load balance by 27% with respect to HEFT algorithm [21], 8% with respect to algorithm in [29], 8% with respect to algorithm in [32], and 0.8% with respect to our previous M-HEFT algorithm in [18] in average by considering 1000 tasks in Ligo.

5.3.2. For 100 and 1000 tasks of epigenomics

Load balance rate results for 100 tasks of Epigenomics are presented in Table 13, and Fig. 14. While load balance rate results for 1000 tasks of Epigenomics are presented in Table 14, and Fig. 15.

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 [21]	10.8	4.6	2.70	2.10
Algorithm [29]	9.6	5.0	3.00	2.1
Algorithm [33]	9.7	5.1	3.2	2.3
M-HEFT [18]	8.0	4.4	2.6	2.1
Enhanced EM-HEFT	7.9	4.0	2.4	2.0

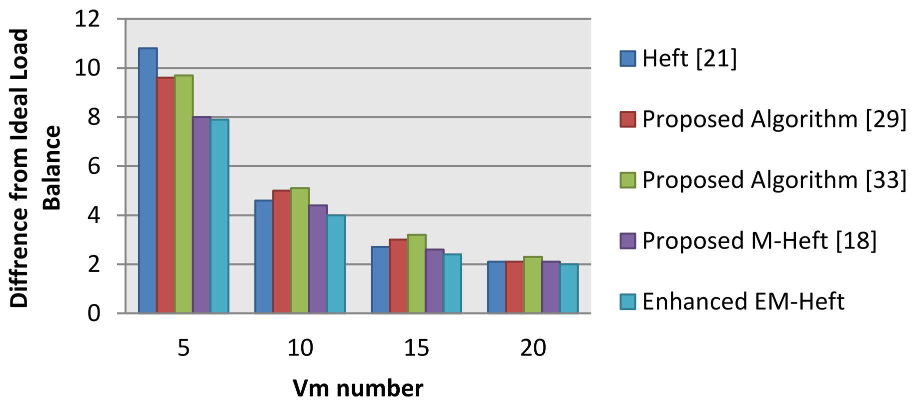


Figure. 14 The average difference from ideal load balance (ILB) results for 100 tasks of epigenomics

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 [22]	140.20	80.30	49.60	36.95
Algorithm [30]	77.60	49.70	33.40	23.75
Algorithm [33]	77.70	50.55	34.46	23.85
M-HEFT [35]	77.40	39.70	28.16	23.65
Enhanced M-HEFT	77.30	39.60	26.40	22.31

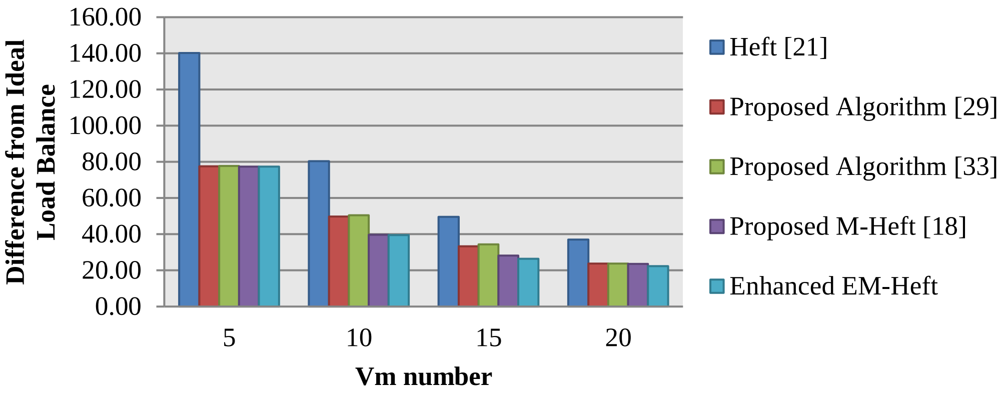


Figure. 15 The average difference from ideal load balance (ILB) results for 1000 tasks of epigenomics

According to the results presented in Table 11 and Fig. 12, the enhanced EM-HEFT algorithm improves load balance by 14% with respect to HEFT algorithm [21], 6% with respect to algorithm in [29], 7% with respect to algorithm in [33], and 7% with respect to our previous M-HEFT algorithm in [18] in average by considering 100 tasks in Epigenomics. According to the results presented in Table 12 and Fig. 13, the enhanced EM-HEFT algorithm improves load balance by 45% with respect to HEFT algorithm [21], 12% with respect to algorithm in [29], 13% with respect to algorithm in [33], and 3% with respect to our previous M-HEFT algorithm in [18] in average by considering 1000 tasks in Epigenomics.

Conclusion and future work

Workflow scheduling is one of the eminent issues that work on allocating workflows tasks on VMs based on different requirements related to tasks and VMs in cloud computing. In this paper, an enhanced task scheduling algorithm, called EM-HEFT, has been introduced to improve the performance of our previous M-HEFT [18] with respect to make span, resource utilization, and load balance. To evaluate the performance of the proposed EM-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 enhanced EM-HEFT improves the make span algorithm by 35% in average with respect to the original HEFT [21] algorithm, by 29% in average with respect to the proposed algorithm [29], by 33% in average with respect to the algorithm in [33], and by 4% in average with respect to our previous M-HEFT algorithm in [18]. The resource utilization has been improved using the enhanced EM-HEFT algorithm by 50% in average with respect to the original HEFT [21] algorithm, by 34% in average with respect to the proposed algorithm [29], by 53% in average with respect to the algorithm in [33], and by 35% in average with respect to our previous M-HEFT algorithm in [18]. In addition, the load balance has been improved using the enhanced EM-HEFT algorithm by 26% in average with respect to the original HEFT [21] algorithm, by 11% in average with respect to the proposed algorithm [29], by 8 % in average with respect to the algorithm in [33], and by 11% in average with respect our previous M-HEFT the algorithm in [18].

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

Table of abbreviations

Name	Abbreviation
VM	Virtual machine.
T	***T*** is stand for Task.
w_i	Weight of the task which is the difference between the highest and lowest computation time of task T_i on VM_m and VMn, respectively divided by speedup of these VMs.
$C_{i,j}$	Communication cost between two tasks.
$CCT_{i,j}$	Computation cost for task T_i on VM_j.
AL	Average length for all tasks in the DAG.
T_{avil}	It is the available time of VM.

ST	Start Time of task on VM.
FT	Finish time of task on VM.
CT$_i$	Is the completion time of the longest task ***T$_i$***.
RUR	Resource Utilization Rate.
ILB	Ideal Load Balance.
DLB	Difference from Ideal Rate of load balance.
ADLB	Average difference from Ideal Rate of load balance.
IRM	Improvement rate in terms of used metrics (make span, resource utilization and load balance).
ABS	It is the Absolute value of a number.
M (proposed algorithm)	M is stand for metrics such as (make span, resource utilization rate and load balance rate).

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] M. R. Prasad, R. L. Naik, and V. Bapuji, "Cloud Computing: Research Issues and Implications", *International Journal of Cloud Computing and Services Science (IJ-CLOSER),* Vol. 2, No. 2, pp. 134-140, 2013.

[2] M. K. Aery, *Cloud Computing Introduction*, 2018.

[3] P. Srivastava and R. Khan, "A Review Paper on Cloud Computing", *International Journal of Advanced Research in Computer Science and Software Engineering,* Vol. 8, No. 7, pp. 17-20, 2018.

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

[5] P. Srivastava and R. Khan, "A Review Paper on Cloud Computing", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 8, No. 7, pp. 17-20, 2018.

[6] D. C. V. Raghavendran, D. G. N. Satish, D. P. S. Varma, and D. G. J. Moses, "A Study on Cloud Computing Services", *International Journal of Engineering Research and Technology (IJERT-ICACC),* Vol. 4, No. 34, pp. 1-6, 2016.

[7] A. Kanwal, A. Kumar, and C. O. V, "Review paper on Cloud Computing", *International Journal of Research in Science & Engineering (IJRSE),* Vol. 1, No. 1, pp. 60-65, 2011.

[8] E. M. Kumar, "Cloud Computing in Resource Management", *International Journal of Engineering and Management Research,* Vol. 8, No. 6, pp. 93-98, 2018.

[9] A. Kanwal, A. Kumar, and C. O. V, "Review paper on Cloud Computing", *International Journal of Research in Science & Engineering (IJRSE),* Vol. 1, No. 1, pp. 60-65, 2011.

[10] R. A. J., A. B. W., and Shriram, "A Taxonomy and Survey of Scheduling Algorithms in Cloud: based on Task Dependency", *International Journal of Computer Applications*, Vol. 82, No. 15, pp. 20-26, 2013.

[11] N. Sharma and S. Tyagi, "Task Scheduling in Cloud Computing", In: *Proc. of Vivechana: National Conf. on Advances in Computer Science and Engineering (ACSE),* Kurukshetra, pp. 249-252, 2016.

[12] S. Jaybhaye and V. Attar, "A review on scientific workflow scheduling in cloud computing", In: *Proc. of the 2nd International Conf. on Communication and Electronics Systems (ICCES),* Coimbatore, India, pp. 218-223, 2017.

[13] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments", *Journal of Concurrency Computat*, Vol. 29, No. 8, pp. 1-23, 2017.

[14] A. Thushara, "Scientific Workflow Scheduling in Cloud Computing Environment: A Survey", *International Journal of Computer Engineering and Technology*, Vol. 9, No. 6, pp. 83-91, 2018.

[15] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis",

Journal of Network and Computer Applications, Vol. 66, pp. 64-82, 2016.

[16] A. Kanwal, A. Kumar, and C. O. V, "Review paper on Cloud Computing", *International Journal of Research in Science & Engineering (IJRSE)*, Vol. 1, No. 1, pp. 60-65, 2011.

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

[18] S. Ahmed and F. A. Omara, "A Modified Workflow Scheduling Algorithm for Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 5, pp. 336-352,2022, doi: 10.22266/ijies2022.1031.30.

[19] N. Almezeini and A. Hafez, "An Enhanced Workflow Scheduling Algorithm in Cloud Computing", In: *Proc. of the 6th International Conf. on Cloud Computing and Services Science (CLOSER)*, Rome, Italy, pp. 67-73, 2016.

[20] A. Mohammadzadeh and M. Masdari, "Scientifc workfow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm", *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-21, 2021.

[21] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment", *Sigmod Record*, Vol. 34, No. 3, pp. 56-62, 2005.

[22] M. Rahman, R. Hassan, R. Ranjan, and R. Buyya, "Adaptive Workflow Scheduling For Dynamic Grid And Cloud Computing Environment", *Journal of Concurrency and Computation on Practice and Experience*, Vol. 25, No. 13, pp. 1816-1842, 2013.

[23] S. Parsa and R. E. Maleki, "RASA: A new task scheduling algorithm in grid environment", *Journal of World Applied Sciences*, Vol. 4, No. 3, pp. 152-160, 2009.

[24] 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.

[25] 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.

[26] 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.

[27] 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)*, Kurukshetra, India, pp. 725-732, 2018.

[28] S. Singhal and J. Patel, "Load Balancing Scheduling Algorithm for Concurrent Workflow", *Computing and Informatics*, Vol. 37, No. 2, pp. 311–1326, 2018.

[29] N. Rajak and D. Shukla, "An Efficient Task Scheduling Strategy for DAG in Cloud Computing Environment", *Ambient Communications and Computer Systems*, Vol. 1097, pp. 273–289, Springer, Singapore, S.2020.

[30] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing", *Complex & Intelligent Systems*, Vol. 10, No. 11, pp. 1–19, 2021.

[31] W. Ahmed and B. Alam, "An efficient list scheduling algorithm with task duplication for scientific big data work flow in heterogeneous computing environments", *Journal of Concurrency Computat*, Vol. 33, No. 5, pp. 1-18, 2021.

[32] H. Zhang, Y. Wu, and Z. Sun, "EHEFT-R R: multi-objective task scheduling scheme in cloud computing", *Journal of Theoretical and Applied Information Technology*, Vol. 100, No. 2, pp. 480-506, 2021.

[33] 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.

[34] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment", *Journal of Cluster Computing*, Vol. 24, No. 4, pp. 3405-3419, 2021.

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

[36] S. Abrishami and M. Naghibzadeh, "Deadline-constrained workflow scheduling in software as a service cloud", *Scientia Iranica*, Vol. 19, No.

3, pp. 680-689, 2012.

[37] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 7, pp. 1787-1796, 2014.

[38] M. Rahman, S. Venugopal, and R. Buyya, "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids", In: *Proc. of IEEE International Conf. on EScience and Grid Computing*, Bangalore, India, pp. 35-42, 2007.

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