Multiobjective Task Scheduling in Cloud Environment Using Decision Tree Algorithm

HADEER MAHMOUD[1,2]**, MOSTAFA THABET**[3]**, MOHAMED H. KHAFAGY**[1]**, AND FATMA A. OMARA**[4]**, (Member, IEEE)**

[1]Department of Computer Science, Faculty of Computers and Information, Fayoum University, Fayoum 63514, Egypt
[2]Faculty of Information Systems and Computer Science, October 6 University, Giza 12585, Egypt
[3]Department of Information System, Faculty of Computers and Information, Fayoum University, Fayoum 63514, Egypt
[4]Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo 12613, Egypt

Corresponding author: Hadeer Mahmoud (hadeer.mhmoud.csis@o6u.edu.eg)

ABSTRACT In recent years, Cloud computing has been developed and become the foundation of a wide range of applications. It allows users to access a catalog of standardized services and respond to their business needs flexibly and adaptively, in the event of unforeseen demands, paying solely for the consumption they have made. Task scheduling problem is considered one of the most critical cloud computing challenges. The problem refers to how to reasonably order and allocate the applications tasks provided by the users to be executed on virtual machines. Furthermore, the quality of scheduling performance has a direct effect on customer satisfaction. The task scheduling problem in cloud computing must be more accurately described in order to improve scheduling performance. In this paper, a multi-objective task scheduling algorithm is proposed based on the decision tree in a heterogenous environment. We introduce a new Task Scheduling-Decision Tree (TS-DT) algorithm for allocating and executing an application's task. To evaluate the performance of the proposed TS-DT algorithm, a comparative study was conducted among the existing algorithms; Heterogeneous Earliest Finish Time (HEFT), Technique for Order of Preference by Similarity to Ideal Solution that incorporates the Entropy Weight Method (TOPSIS-EWM), and combining Q-Learning with the Heterogeneous Earliest Finish Time (QL-HEFT). Our results show that the proposed TS-DT algorithm outperforms the existing HEFT, TOPSIS-EWM, and QL-HEFT algorithms by reducing makespan by 5.21%, 2.54%, and 3.32%, respectively, improving resource utilization by 4.69%, 6.81%, and 8.27%, respectively, and improving load balancing by 33.36%, 19.69%, and 59.06%, respectively in average.

INDEX TERMS Cloud computing, task scheduling, data dependency, decision tree, makespan, resource utilization, load balancing, energy consumption.

I. INTRODUCTION

Cloud computing is a modern computer technology that employs virtualized infrastructure to provide secure and reliable services to end-users in a complicated environment. Because it provides essential information technology (IT) services such as computing resources in the form of virtual machines (VMs), cloud computing has gotten a lot of attention as a computing model [1], [2].

Unfortunately, Cloud computing has some challenges such as performance, resource management, cost, etc. [3]. On the other hand, task scheduling on cloud computing is the allocation of users' tasks on the available resources to optimize the execution time, enhance load balancing, and increase

resource efficiency. Task scheduling depends on the existence of dependencies among the tasks. The problem of scheduling dependent tasks in a heterogeneous environment has drawn many attention of researchers in this area. Among the most studied area is the Directed Acyclic Graph (DAG), which is the most common graph that shows the dependability of the application's tasks. This dependent task scheduling is also called DAG scheduling.

Some of the DAG types are Montage, the Srna Identification Protocol using High-throughput Technology (SIPHT), Cyber-Shake, Epigenomics, and the Laser Interferometer Gravitational Wave Observatory (LIGO) workflow applications [4]–[6]. (See Figure 1).

In DAG scheduling, the workflow is presented by $G = (T)$, where $T = \{t_1, t_2 \ldots,\}$ is the set of tasks and $E = \{e1, e2 \ldots,\}$ is the set of edges. Each task $t_i \in T$ denotes
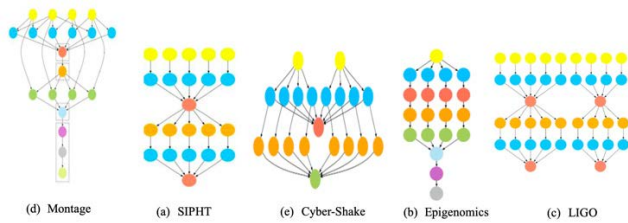
FIGURE 1. Structure of DAG workflow [7].

an application task, and each E $(i, j) \in E$ represents the communication cost between the independent tasks, where task t_i is should be executed before task t_j. A structure of the DAG of some workflow types is shown in Figure 1.

Scheduling and allocation of the applications' tasks are considered Non-deterministic Polynomial (NP)-Complete problems [8]. Therefore, optimization approaches are used to solve these problems by considering performance parameters such as makespan, load balance, resource usage, cost, power consumption, etc., to solve these problems [9].

Machine learning is currently being used in various fields such as speech recognition, data classification, and face recognition [10]. It has played a significant role in task scheduling and several other areas in computer science technology. Among the popular machine learning techniques used for developing and visualizing predictive models and algorithms is the decision tree.

In this paper, a Task Scheduling-Decision Tree (TS-DT) algorithm, which is a task scheduling algorithm based on the decision tree is introduced. The performance of the proposed task scheduling is evaluated using load balance parameters such as makespan, resource utilization, and power consumption in a heterogeneous environment. By the work in this paper, the following significant contributions are satisfied:

- A new task scheduling algorithm based on the decision tree method for scientific workflows is proposed.
- Extensive simulation tests on various performance metrics are used to evaluate the proposed algorithm.

The rest of this paper is organized as follows; Section 2 discusses the related works. In Section 3, the principles of the proposed algorithm are described. Section 4 illustrates the proposed algorithm in detail. The CloudSim simulator's configuration and the proposed Decision Tree Algorithm are discussed in Section 5. The comparative study among the proposed algorithm and other existing algorithms like HEFT, TOPSIS-EWM, and QL-HEFT are discussed in Section 6. Finally, the conclusion and future work are given in Section 7.

II. RELATED WORK

Task scheduling in distributed, parallel, and cloud computing environments have become an important research topic. Its purpose is to ensure an effective distribution of computing resources to provide high performance. In traditional distributed and parallel computing environments, a set of scheduling strategies has been proposed by many researchers.

K. Naik *et al.* [11] have described a new hybrid multi-objective heuristic method that combines Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and Gravitational Search Algorithm (GSA) called as NSGA-II & GSA to assist with VM selection for application scheduling. NSGA-II has the ability to expand the search space through exploration, while GSA has the ability to exploit the good solution to discover the best solution and so avoid the algorithm getting trapped in local optima. This hybrid algorithm is designed to achieve the fastest response time and lowest cost for scheduling a larger number of tasks with the minimum total energy consumption. Unfortunately, there is no load balancing between VMs.

S.Pang and colleagues [12] have developed a hybrid scheduling algorithm based on the Estimation of Distribution Algorithm (EDA) and Genetic Algorithm (GA). The algorithm initially generates some feasible solutions using EDA operations, then utilizes GA operations to generate new solutions based on the great solutions selected in the previous phase to expand the search range of solutions, and finally selects the best solution. The purpose of this technique is to reduce the task completion time and enhance load balancing. However, this paper does not consider the dynamics and uncertainties of the cloud computing environment.

S.H.H. Madni *et al.* [13] have presented a novel Multi-objective Cuckoo Search Optimization (MOCSO) technique for dealing with the cloud computing resource scheduling problem. The goal of this technique is to explore the multi-objective resource scheduling problem in an Infrastructure as a service (IaaS) cloud computing environment by maximizing resource consumption. The load balancing across VMs is considered a big flaw in this technique.

Y.Q. Han and Q. Li. Jun [14] have solved the flexible task scheduling problem in a cloud system by proposing a hybrid discrete Artificial Bee Colony (ABC) algorithm. The suggested algorithm includes three categories of artificial bees; employed, onlooker, and scout bees, as in the classical ABC algorithm. The proposed ABC algorithm reduces completion time and improves balancing machine loads. One of this algorithm's major flaws is resource utilization.

Avinash Kaur *et al.* [15] have proposed a new workflow scheduling scheme by integrating the Deep Q-learning mechanism and the HEFT algorithm is called DQ-HEFT. The scheme is considered the most common heuristic scheduling in literature. The algorithm consists of two phases: gaining the task's execution order at each stage and allocating the task to the processor with a significantly higher volume of data. It is worthy to note that the DQ-HEFT algorithm can achieve better makespan and speed. However, the main drawback of the DQ-HEFT algorithm is that excessive value updates of the Q-table are performed in large-scale task optimization problems, which slow down the scheduling process.

A. Al-maamari and F. Omara [16] have proposed a task scheduling algorithm for the cloud computing environment.

The algorithm is considered a fusion of the Cuckoo Search (CS) algorithm and the Dynamic PSO (DAPSO) algorithm, which has been modified to increase the population. According to this algorithm, tasks are assigned to virtual machines (VMs) to minimize makespan and maximize resource uses. However, there is no load unbalancing between VMs.

Arabi Keshk *et al.* [17] have introduced Modified Ant Colony Optimization for Load Balancing (MACOLB) algorithm to allocate the incoming jobs to the virtual machines (VMs). The tasks are allocated to the VMs based on the processing powers (i.e., tasks are allocated in descending order, starting from the most powerful VM, and so on) by considering balancing VMs' loads. The MACOLB is used to find the proper resource allocation for batch tasks in the cloud system, minimize the makespan, and achieve better system load balance. However, the resource utilization between VMs is considered a crucial flaw of this algorithm.

Jae-Min Yu1 and colleagues [18] have proposed a decision tree-based method for scheduling flexible workshops with multiple process plans. For static and dynamic flexible job shops, two decision tree-based scheduling mechanisms were created. All jobs were provided in advance in the static case, and the decision tree is used to select a priority dispatching rule to process all of them. In the dynamic scenario, jobs arrive over time. The decision tree is used to select a priority rule in real-time according to a rescheduling strategy using a decision tree that is modified regularly. The objectives considered in this method are makespan, total flow time, and total delay, but the load balancing between VMs was not considered.

Liu Yuan, Dong Yinggang, etc. [19] have proposed a static HEFT task scheduling algorithm, called ST-HEFT. The algorithm consists of two key steps; task sorting and task mapping. According to the sorting step, tasks are sorted based on the maximum communication cost between them and their direct VMs. The task mapping step is assigned to the VM that provides the earliest execution time. The proposed algorithm has achieved better performance by reducing the development threshold for parallel computing programs and increasing the utilization of various computing devices' capabilities in the heterogeneous computing environment. On the other hand, load balancing and sleek time are the critical weaknesses of this algorithm.

Sambit Kumar Mishra *et al.* [20] have suggested an Adaptive Task Allocation (ATAA) algorithm in the cloud environment. This algorithm uses the Expected Time to Completion (ETC) matrix to solve the heterogeneous environment problem, including completing all tasks on VMs. The author uses a technique that reduces energy consumption and minimizing the makespan of the system. Also, the major weakness of this algorithm is the load balancing between VMs.

Atyaf Dhari *et al.* [21] have proposed a cloud computing environment load balancing decision algorithm, called (LBDA), to enhance load balancing among virtual machines and reduce makespan. The algorithm consists of three steps.

The first step is used to calculate the VM's capacity and load (under-full VM, balanced VM, high-balance VM, and overloaded VM). In the second step, for each VM, the required time is determined to execute the task. In the third step, based on the VM state and the task time, a decision is made to spread tasks. Unfortunately, resource utilization between VMs is considered a critical weakness of this algorithm.

Zeshan Iqbal *et al.* [22] have proposed an algorithm called Parental Prioritization Earliest Finish Time (PPEFT) for a heterogeneous distributed environment. The algorithm consists of two phases; the prioritization of the tasks and the processor's assignation. First, the tasks are scheduled in the Parental Priority Queue (PPQ) based on the descending Rank and parental priority in the task prioritization phase. Then, the Processor Assigning Phase assigns each task in the PPQ queue to a processor that guarantees fast execution (i.e., minimum computation cost). Experimentally, the PPEFT scheduling algorithm performs substantially better concerning cost and schedule makespan than other algorithms. Unfortunately, load balancing between VMs is a critical weakness of this algorithm.

S.C. Sharma *et al.* [23] have modified the HEFT algorithm to effectively distribute the workload between processors and effectively reduce completion time. This algorithm analyzes various algorithms for the task scheduling, parameters, tools, improvement, and algorithm limitations. This algorithm reduces makespan and improves load balancing by comparing it to the existing HEFT and the Critical Path on a Processor Algorithm (CPOP) [24] algorithms. The critical weakness of this algorithm is the sleek time.

In this paper, the decision tree has been used to optimize the multi-objective task scheduling problem by minimizing makespan, satisfying load balancing among virtual machines, and maximizing resource utilization.

III. OVERVIEW

In this paper, we proposed a task scheduling algorithm based on the decision tree for a heterogeneous cloud environment has been proposed. Also, we evaluated the performance of the proposed algorithm through a comparative study among the HEFT, TOPSIS-EWM, and QL-HEFT algorithms, which are widely used algorithms for task scheduling in the cloud computing environment. Based on the above-mentioned goals, we now discuss the principles of the existing algorithms in the following sections.

A. THE HETEROGENEOUS EARLIEST FINISH TIME ALGORITHM

In the HEFT algorithm [25], the tasks presented in the DAG are scheduled to a series of heterogeneous machines. This algorithm consists of two phases; ranking and processor selection phases. The goal of the ranking phase is to provide a priority for each task. The Processor Selection phase concerns about allocating each task to a suitable processor. This phase will be repeated until all tasks will be scheduled for the available processors [26]. In the ranking

phase, the upward (ranking) function is used to define the priority of each task which is defined recursively by using Equation (1) [27]:

$$\mathbf{Rank}(t_i) = \overline{w_I} + \mathbf{max}_{t_j \in succ(t_i)} \left(\overline{c_{I,J}} + \mathbf{Rank}(t_j) \right) \quad (1)$$

where $\mathbf{W_i}$ is the average of the computation cost of the task $\mathbf{t_i}$, $\mathbf{c_{i,j}}$ is the average of the communication cost between the edges from $\mathbf{e_i}$ to $\mathbf{e_j}$, and succ ($\mathbf{t_i}$) is the set of successors of the task $\mathbf{t_i}$. It's important to remember that Rank ($\mathbf{t_i}$) is determined by the computation of all its children's Rank ($\mathbf{t_j}$).

In the ''processor selection'' phase, tasks are sorted in descending order according to their rank values. Then, the processors assign tasks by selecting the processor with the shortest finish time for each task. However, the HEFT algorithm always considers the processor with the earliest finish time to allocate tasks, but it does not consider load balancing among the processors [28].

B. A MULTI-CRITERIA DECISION-MAKING APPROACH

A Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) workflow scheduling algorithm in a cloud environment has been introduced that incorporates the Entropy Weight Method (EWM) called (TOPSIS-EWM) [13]. The proposed algorithm aims to reduce makespan, cost, and energy consumption while increasing reliability. According to the ROPSIS-EWM algorithm, EWM is used to determine the input weight of the attributes schedule length (EFT), cost, reliability, and energy consumption. The TOPSIS approach is then used to choose the optimal virtual machine for each task. The research takes into consideration a cloud environment with dynamic voltage scaling (DVS) and pay-per-use heterogeneous VM instances. The MIPS of the VM instances is used in the simulations directly proportional to VM pricing. The (TOPSIS-EWM) algorithm does not consider any user preferences, such as deadlines, load balance, and resource utilization.
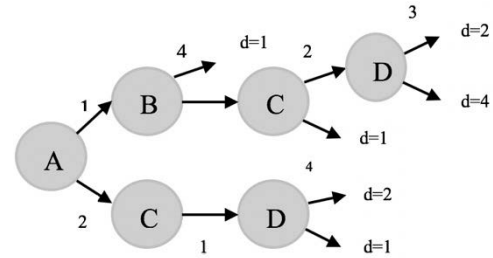
C. QL-HEFT ALGORITHM

Authors in [29], proposed a novel task scheduling algorithm that reduces the makespan by combining Q-Learning with the HEFT algorithm is called (QL-HEFT). The algorithm uses the upward rank value of HEFT as the immediate reward. In the Q-learning framework, the agent can obtain better learning results through the self-learning process to update the Q-table. The QL-HEFT algorithm is divided into two phases: the task sorting phase and the processor allocation phase. The task sorting phase uses Q-learning to find the best order of the tasks, while the processor allocation phase uses the earliest finish time strategy. However, the authors discovered that using the QL-HEFT algorithm to solve large-scale task optimization problems has some drawbacks, such as an excessively large Q-table that causes long update times.

D. A DECISION TREE DEFINITION

A decision tree is a hierarchical data structure that uses a divide-and-conquer technique to represent data [30]. On the other hand, the decision tree is a rooted tree having leaf and non-leaf nodes. The decision criteria for classification and regression trees distinctly depend on the decision tree. Meanwhile, a decision tree is a rooted tree with leaf and non-leaf nodes. The leaf nodes represent the classification or decision-making, whereas the non-leaf nodes represent the selection options by dividing the instance space into two or more subspecies based on a discrete function of input attribute values (See Figure 2) [31].

FIGURE 2. Decision tree structure [32].

The decision tree is a popular method for creating and visualizing predictive models and algorithms [23], [33]. As explained earlier, the static approach uses the decision tree to select a priority rule combination to process the set of given tasks, i.e., no rule changes over the scheduling horizon. Hence, it can be used for planning purposes. The static decision tree-based mechanism suggested in this study is shown in the next section.

IV. PROBLEM STATEMENT

Multi-objective optimization has a significant impact on scheduling issues in the cloud computing environment. We covered three relatively approaches for resolving task scheduling performance, each has its own set of restrictions:

- When allocating tasks, the HEFT method always considers the processor with the earliest finish time, but it ignores load balancing among the processors.
- The (TOPSIS-EWM) algorithm considers the input weight of the attributes schedule length like (EFT), cost, reliability, and energy consumption. Then, the TOPSIS technique is used to select the best virtual machine for each task without considering any user preferences like deadlines, load balance, and resource utilization.
- The QL-HEFT approach for solving large-scale task optimization issues has some limitations, such as a big Q-table that causes long update times that effect on the Makespan in the task schedule.

V. THE PROPOSED TASK SCHEDULING DECISION TREE (TS-DT) ALGORITHM

In this section, we introduce the TS-DT algorithm to reduce the makespan, enhance load balance, and maximize utilization of the resource. The algorithm consists of three phases: the priority task, the resource matrix, and the resource allocation phase. First, the task priority phase is used to assign

a rank for each task. The resource matrix phase is used in collecting the tasks' features in the form of a matrix, while the resource allocation phase is where tasks are scheduled on the proper VMs using the decision tree. The principles of each phase will be explained in the following sections.

A. TASK PRIORITY PHASE

According to the task priority phase, a rank is assigned for each task in the given workflow (i.e., DAG). By considering tasks set T = {T1, T2, . . . , Tn}. If $T_i < T_n$, then T_i is the parent of T_n. Equation (1) has been changed by adding task length (T_L), which indicates the length of the instruction of a cloudlet (i.e., task) to be processed in the virtual machine (VM), and the number of child's (N_c) (See Equation (2)). Therefore, the Rank of each task in the given workflow is defined using Equation (2).

$$\text{Rank}(t_i) = \overline{w_l} + \boldsymbol{max}_{t_j \in \text{succ}(t_i)} \left(\overline{c_{l,j}} + \text{Rank}(t_j) \right) + T_l + N_c$$
(2)

After assigning priority to each task, the tasks will be sorted in descending order according to their Rank value and stored in the Rank [T] list. As a result, the most important task will be executed first. The pseudo-code of the Task Priority phase is as follows:

TABLE 1. Shows the pseudo-code of the tasks priority phase.

Algorithm 1. Tasks Priority of the Proposed TS-DT Algorithm
Input: DAG workflow and set of virtual machines.
1. **For** each task **t_i** in DAG of the workflow, **Do**
2. Calculate the average execution time of t_i on each VM
3. **If** a task t_i is the last task in DAG, **Then**
4. *Rank (t_i)* = average execution time of the task on all VMs
5. **Else**
6. $\text{Rank}(t_i) = \overline{w_i} + max_{t_j \in succ(t_i)} \left(\overline{c_{i,j}} + \text{Rank}(t_j) \right) + T_l + N_c$
7. **End If**
8. **End For**
9. Arrange tasks in descending order in the Rank [T] list.

B. RESOURCE MATRIX PHASE

This phase is used to collect the features of the selected task from the Rank [T] list and store them in the task's matrix **(T).** In the **T** matrix, columns represent the number of needed resources, while rows represent four features for each task as follows:

Feature 1: Computation cost (**CP**) of each task on each VM.
Feature 2: Assigning the task to VM by selecting VM with Earliest Finish Time (**EFT**).

Feature 3: Total length of tasks (**TTL**) assigned to each VM. This refers to the length of the instructions of tasks on VM.
Feature 4: Showing the VM-based task parent (**TP**) (i.e., Parent location (0/1)), where **one** is typed if there is a parent for the task; otherwise, **zero** is typed. This feature considers the communication cost between tasks.

For example, a structure of a task structure with its features is summarized in a matrix (T) by considering five VMs with four features, as shown in Figure 3.

$$T = \begin{bmatrix} & VM1 & VM2 & VM3 & VM4 & VM5 \\ \textit{Feature 1} & CP & CP & CP & CP & CP \\ \textit{Feature 2} & EFT & EFT & EFT & EFT & EFT \\ \textit{Feature 3} & TTL & TTL & TTL & TTL & TTL \\ \textit{Feature 4} & TP & TP & TP & TP & TP \end{bmatrix}$$

FIGURE 3. A Structure of a tasks' matrix.

C. RESOURCE ALLOCATION PHASE

In this phase, the proper VMs are selected to execute tasks in the Rank [T] list, which contains the tasks which order in descending order according to their priorities.

For the task that plays a role in the Rank [T] list (i.e., the task with high priority), the decision tree is constructed to represent its features from its task matrix. In the case of leaf nodes, a test is done to check if the task's parent is on the same VM or not according to Feature 4. If the answer is "yes", the communication cost remains zero. In the case of "no", the communication cost is considered between the parent and the successor.

The output in the leaf nodes is the summation of the task's features in the task's matrix, which is defined using Feature 1 to Feature 3 (i.e., CP, EFT, and TTL) with considered the summation of the communication cost from the DAG workflow if the task's parent is not on the same VM. If the parent task in the same VM, it is compute using Equation (3). When the parent is not in the same VM, it is calculated using Equation (4).

$$\text{VM} = CP_i + EFT + TTL$$
(3)
$$\text{VM} = CC_{(i,j)} + CP_i + EFT + TTL$$
(4)

where $\textbf{CC}_{(i,j)}$ is the communication cost between ti that presented in Rank [T] list with t_j is the last task in VM. ($\textbf{CP}_i$) is the computation cost for the task ti among all the VMs. (**EFT**) is the earliest finish time of all tasks in VM. (**TTL**) is the total length of tasks assigned to each VM.

Finally, the task is assigned to the VM, which has the lowest value that will come out of the tree's leaf nodes (See Figure 4). The pseudo-code for the Resource Allocation phase is shown in Table 2.

Example: To explain how the proposed TS-DT algorithm works, a sample of the task graph and the computational task costs on each VM with considering 3 VMs are depicted in Figure (5a, b).
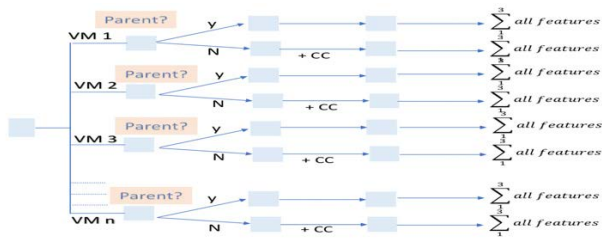
FIGURE 4. The proposed TS-DT algorithm using decision tree.

**TABLE 2.** Shows the pseudo-code of the resource allocation phase.

Algorithm 2: The Resource Allocation phase of the proposed TS-DT Algorithm.

Input: Rank [T$_i$] list and [VM] list

1. **For** t =1 to n, **Do**   //n is the number of tasks
2. **For** v =1 to m, **Do**   // m is the number of VMs
3.   **If** (T$_i$ has parent ()), **Then**
          // the task parent in the same vm
4.   Communication Cost = 0
5.   **Else**
6.   Communication Cost = Communication Cost.Parent();
7.   **End If**
8. Result [vm.no]   = Communication Cost;
9. Result [vm.no] + = Computation Cost ();
10. Result [vm.no] + = VM.no.get EFT ();
11. Result [vm.no] + = VM.no.get total tasks length ();
12. **End for**
13. Task. allocate (get smallest value (Result [ ]));
14. **End for**
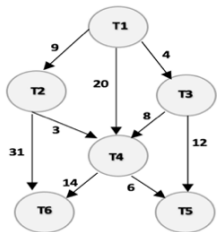
(a) Task Graph   (b) Task Computation Cost on each VM

FIGURE 5. A task graph and computational cost example.

1) PHASE 1: TASK PRIORITY PHASE

By applying this phase, a rank is assigned to each task using Equation (2), and the tasks are sorted in descending order in Rank [T] (See Table 3).

2) PHASE 2: RESOURCE MATRIX PHASE

In this phase, the features of the selected task from the Rank [T] list will be collected and stored in the task's matrix by considering **T$_1$** in Figure 5, Feature2 (i.e., EFT), Feature3 (i.e., TTL), and Feature4 (i.e., TP) are considered zero on all VMs because it is the entry task. Therefore, the features of (**T$_1$**) in its matrix are presented in Figure 6.

TABLE 3. Calculation Rank [T] for Figure 5.

Task	w_i (A)	Max ($\overline{c_{i,j}}$ + Rank (t_j)) (B)	T_l (C)	N_c (D)	Rank= A+B+C+D	Rank [T] = Order of task
T_1	15	2103	200	3	2321	T1
T_2	21	1783.67	100	1	1906.67	T3
T_3	13.33	1776.67	300	2	2092	T2
T_4	14.67	750	1000	1	1766.67	T4
T_5	36.33	0	500	0	536.33	T6
T_6	40	0	700	0	740	T5

$$T_1 = \begin{bmatrix} & VM1 & VM2 & VM3 \\ Computation\ Cost & 22 & 13 & 10 \\ EFT & 0 & 0 & 0 \\ Total\ Task\ Length & 0 & 0 & 0 \\ Parent\ Location & 0 & 0 & 0 \end{bmatrix}$$

FIGURE 6. The resource features of task T$_1$.

3) PHASE 3: RESOURCE ALLOCATION PHASE

In this phase, each task is assigned to a suitable VM based on its decision tree. The cost of communication value between tasks and their parent and successor will be assessed based on the matrix that constructs in Phase 2. According to the decision tree of task (T1), the summation value of VM3 is considered the lowest value (i.e., 0+10=10) (See Figure 7). So, VM 3 is assigned to execute the task (T1).
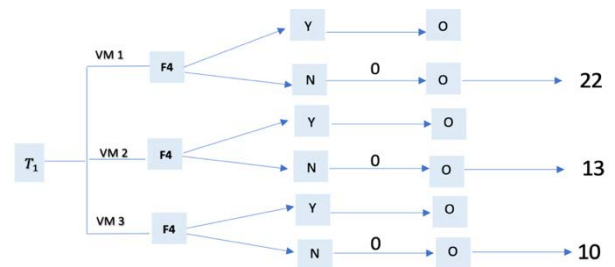
FIGURE 7. The decision tree of task T$_1$.

Phases two and three (i.e., Resource Matrix Phase and Resource Allocation Phase) are repeated for each task in the Rank[T] list until all tasks are assigned to VMs.

As a result, the entire workflow makespan is 413 milliseconds in both the task priority and resource allocation phases. The deviation of the load balance is zero because the term of the Resource Matrix phase considers the number of tasks on each VM. The resource utilization rate is 94.67 %, and it appears that the Resource Matrix phase works better. (See Figure 8).

VI. THE PERFORMANCE EVALUATION

In this section, we discuss the performance matrices, experimental environment, and the benchmark used in evaluating the performance of the proposed TS-TD algorithm.
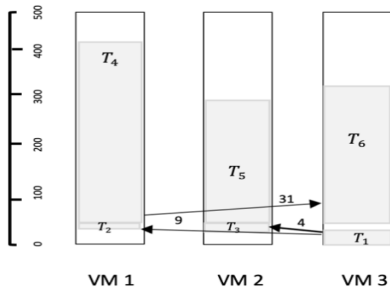
FIGURE 8. Using TS-DT algorithm, the makespan is 413 msec.

A. PERFORMANCE METRICS

The performance matrices which are used to measure the efficiency of the task scheduling algorithms on the cloud computing environment includes the following:

1) MAKESPAN

It displays the maximum completion time of the schedule. This parameter is defined by Equation (5) [34].

$$\text{Makespan} = \max \left(Ct_i\right)_{t_i \in T} \tag{5}$$

where Ct_i is the execution time of the longest task t_i, and T is the number of the tasks on the workflow of an application.

2) RESOURCE UTILIZATION RATE (RU)

We define RU as the ratio of the total consumed time by VMs to the makespan of the parallel application. It is calculated as a percentage using Equation (6) [35].

$$RU \left(VM_i\right) \% = \frac{\sum VM_i \text{BusyTime}}{\text{Makespan}} * 100 \tag{6}$$

3) LOAD BALANCING (LB)

This is the ratio of the total number of tasks to the number of VMs. We calculate LB using Equation (7) [36].

$$LB = \frac{\text{Number of Tasks}}{\text{Number of VMs}} \tag{7}$$

4) BUSY TIME

It represents the task length indicates the instructions length of a cloudlet to be processed on each VM_i. It is defined using Equation (8) [37].

$$T_{\text{busy}} \left(VM_i\right) = \frac{\sum t_{\text{length}}}{V M_i} \tag{8}$$

where $\sum$ length is the summation of each task length.

5) IDLE TIME

This is the difference between the total execution time and the busy time for each VM_i. It is determined using Equation (9) [38].

$$T_{\text{idle}} \left(VM_i\right) = T_{\text{total}} - T_{\text{busy}} \tag{9}$$

6) TOTAL EXECUTION TIME (TET)

It represents the summation between the busy time and idle time for each VMi. It is defined using Equation (10) [39].

$$\text{TET}_{\text{vm}} = T_{\text{busy}}(VM_i) + T_{\text{idle}}(VM_i) \tag{10}$$

7) POWER CONSUMPTION

The power consumption of **VM_i** consists of two parts; busy power consumption and idle power consumption. It is calculated using Equation (11) [40].

$$PC \left(VM_j\right) = \sum_0^n \left(P_{\text{busy}_{j}*} T_{\text{busy}}\right) + \left(P_{\text{idle}_{j}*} T_{\text{idle}}\right) \tag{11}$$

where P_{busyj} is the busy power of the machines with 220W, P_{idlej} is the idle power of the machines with 95W, T_{busy} is the busy time of the machine, and T_{idle} is the idle time of the machine.

8) IMPROVEMENT RATE (IR_x)

Makespan, resource utilization, and load balance are factors (x) that will be considered to determine the performance improvement of the proposed TS-DT algorithm relative to the current HEFT, TOPSIS-EWM, and QL-HEFT. We calculate IR_x using Equation (12) [41].

$$\begin{aligned} &\textbf{\textit{IR}}_x(\%) \\ &= -\left(\frac{x(\text{existing algorithm}) - x(\text{proposed algorithm})}{x(\text{existing algorithm})}\right) \\ &*100 \end{aligned} \tag{12}$$

9) AVERAGE DEVIATION

The ratio between the total summation of IR_x for each VM_i over their number to get the average deviation from the ideal rate in percentage is calculated using Equation (13) [42].

$$\text{Average}(\%) = \sum_{i=1}^n \left(\frac{IR_x \left(VM_i\right)}{n}\right) \tag{13}$$

B. THE EXPERIMENTAL ENVIRONMENT

The CloudSim 3.0.2 toolkit is an open-source simulator developed by WorkflowSim 1.0 on Windows 7 OS with a Core i7 2.70 GHz processor [43]. The CloudSim simulator is used to implement and evaluate the performance of the proposed TS-DT Algorithm. The Eclipse IDE 4.12.0 is used to run CloudSim 3.0.2. The used benchmarks are Montage_25, SIPHT_30, Cyber- Shake_30, and Epigenomics_24 workflows [44].

VII. PERFORMANCE ANALYSIS

To evaluate the performance of the proposed TS-DT algorithm, a comparative study was conducted among HEFT, TOPSIS-EWM, QL-HEFT algorithms, and the proposed TS-DT algorithm in terms of makespan, load balance, resource utilization, and power consumption. The tasks are considered dependent, and each task has different characteristics, such as length, id, start time, and finish time. The implementation was carried out with the consideration of 5,

10, 20, and 40 VMs using Montage_25, SIPHT_30, Cyber-Shake_30, and Epigenomics_24 workflows. We focus on the hypervolume indicator to measure the quality of a set of trade-off solutions [45].

A. MAKESPAN EVALUATION

The implementation results of the proposed TS-DT, HEFT, TOPSIS-EWM, and QL-HEFT algorithms with respect to makespan using Montage_25, SIPHT_30, Cyber-Shake_30, and Epigenomics_24 workflows by 5, 10, 20, and 40 VMs using Equation (5), as shown in Figure 9 from (a) to (d).

According to the implementation results in Figure 9, it is found that the proposed TS-DT algorithm outperforms the HEFT, TOPSIS-EWM, and QL-HEFT algorithms. This is because of the following reasons:

- During the Task Priority phase, the proposed TS-DT algorithm determines the proper task to be executed by increasing its priority while using the task length and number of childs.
- During the Resource Allocation phase, the decision tree and the summation of the features in the task's matrix are used to select the VM with the lowest value.
- Some features are used to enhance the makespan (i.e., computation cost, Earliest Finish Time (EFT), and parent location).

The average improvement rate of makespan, in percent, of the proposed TS-DT algorithm compared to existing HEFT, TOPSIS-EWM, and QL-HEFT algorithms were determined using Equation (12) and presented in Table 4.

TABLE 4. Improved average rate of makespan (in %) of the proposed TS-DT algorithm.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	-4.0900008	-0.2440566	-0.4378286
10	-13.106437	-8.647954	-9.9941048
20	-1.7497734	-0.7443627	-1.3288819
40	-1.9063134	-0.5330695	-1.5379343

According to the comparative results in Table 4, We found that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 5.21%, 2.54%, and 3.32%, respectively.

B. RESOURCE UTILIZATION

TS-DT, HEFT, TOPSIS-EWM, and QL-HEFT algorithms with respect to resource utilization using Montage_25, SIPHT_30, Cyber-Shake_30, and Epigenomics_24 workflows using 5, 10, 20, and 40 VMs are presented in Figure 10. Based on the comparison results in Figure 10 (a)-(d) (See Equation (6)), we confirmed that the proposed TS-DT algorithm outperforms the existing algorithms in terms of resource utilization for any number of VMs. This is possible because, during the Resource Matrix phase, the proposed TS-DT algorithm selects the minimum of the total length of tasks assigned to each VM. The total
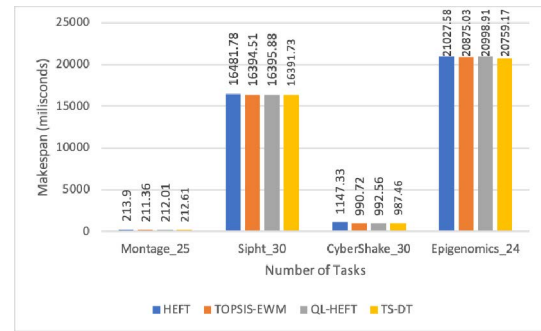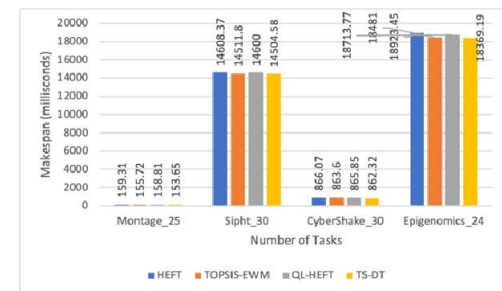

(a)   5 VMs
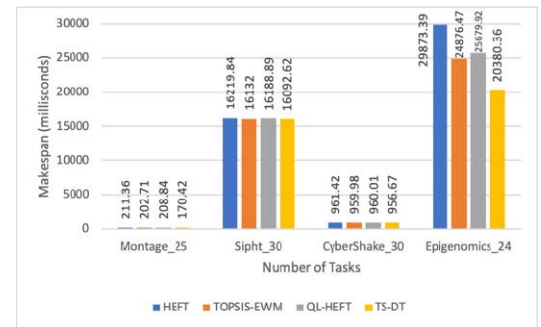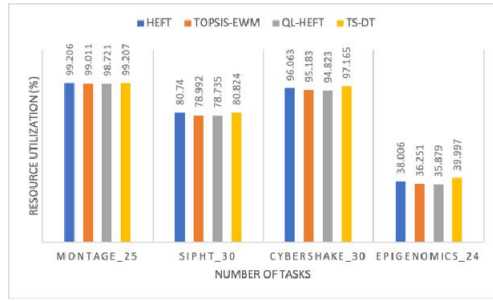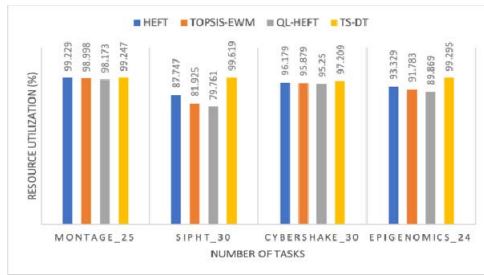

(b)   10 VMs
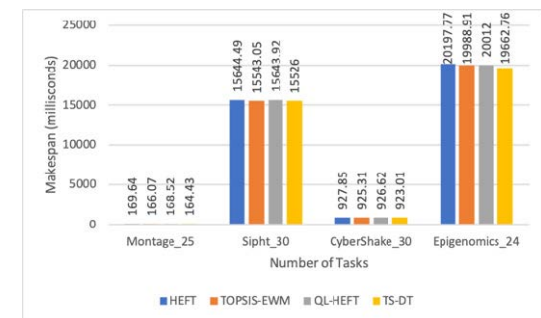
(c)   20 VMs

(d)   40 VMs

FIGURE 9. Comparative results for makespan.

instruction's length of tasks on VM considers the devices' consumption rate.
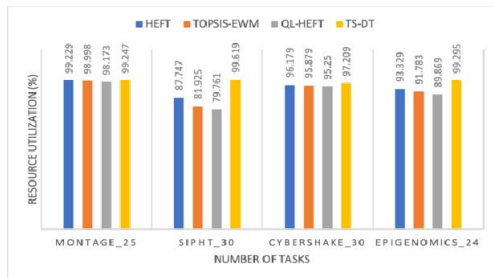
The average improvement of the proposed TS-DT algorithm in terms of resource utilization in percentage in relation to the existing HEFT, TOPSIS-EWM, and QL-HEFT algorithms were determined using Equation (12) and presented in Table 5.
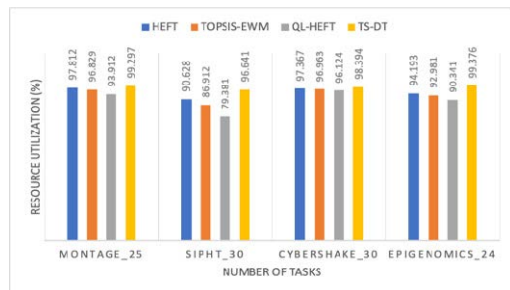
(a) 5 VMs

(b) 10 VMs

(c) 20 VMs

(d) 40 VMs

FIGURE 10. Comparative results for resource utilization.

According to our comparative results in Table 5 show that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 4.69%, 6.81%, and 8.27%, respectively.

C. LOAD BALANCING

The load balancing of each VM is calculated by the ratio of the number of tasks and the total execution time in VMs (See Equation (7)). The implementation results of our proposed TS-DT, HEFT, TOPSIS-EWM, and QL-HEFT algorithms with respect to load balancing using Montage_25,

TABLE 5. Improved average rate of resource utilization (in %) for the proposed TS-DT algorithm.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	1.62271401	3.73324816	4.27320777
10	8.20109244	10.1293424	9.2128822
20	5.25282666	7.85525259	9.63403989
40	3.67758441	5.52411861	9.95996476

SIPHT_30, Cyber-Shake_30, and Epigenomics_24 workflows using 5, 10, 20, and 40 VMs are presented in Figure 11 from (a) to (d).

According to the comparison results in Figure 11 is found that the proposed TS-DT algorithm outperforms the existing other algorithms in terms of load balancing. This is because that during the Resource Matrix phase (Feature 3), the proposed TS-DT algorithm considers the length of the total task on each VM when assigning a new task to VM. Finally, the TS-DT select the minimum value. So, the phase is influenced by the device load at similar rates.

The average load balance improvement in the percentage of the proposed TS-DT algorithm relative to the existing HEFT TOPSIS-EWM, and QL-HEFT algorithms has been determined using Equation (13) and illustrated in Table 6.

TABLE 6. Improved average rate of load balance (in %) of TS-DT algorithm.

| No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	30.0693209	43.158865	-24.383398
10	-51.539583	-32.022823	-69.803277
20	-56.820015	-44.696929	-74.803327
40	-55.15023	-45.207672	-67.241872

According to the comparative results in Table 6, it is found that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 33.36%, 19.69%, and 59.06%, respectively.

D. POWER CONSUMPTION

In the heterogeneous cloud platform, the power consumption consists of two components; busy power consumption and idle power consumption. During the power processing operations, the busy power consumption is the energy consumed, and the idle power consumption is the energy consumed during the VM in the idle state [40]. In order to accurately calculate the power consumption, the work in this paper estimates the busy power consumption and idle power consumption of the VM, respectively. Then, the total power consumption of the VM is obtained using Equations (8, 9, 10, 11, 12, and 13). According to the implementation results, it is found that our proposed TS-DT algorithm increases power consumption relative to the HEFT, TOPSIS-EWM, and QL-HEFT algorithms, this is because the following reason:
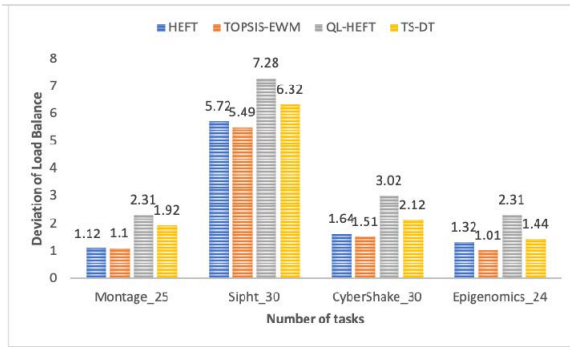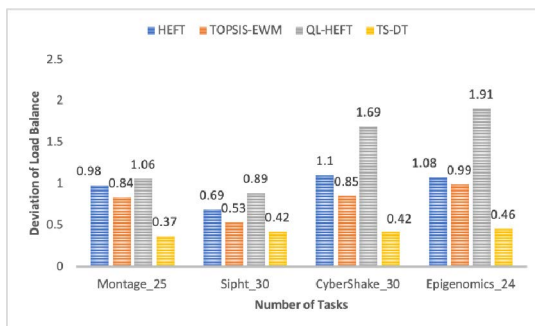
(a)   5 VMs
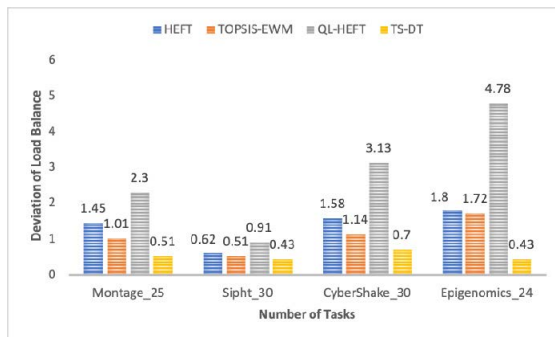
(b) 10 VMs
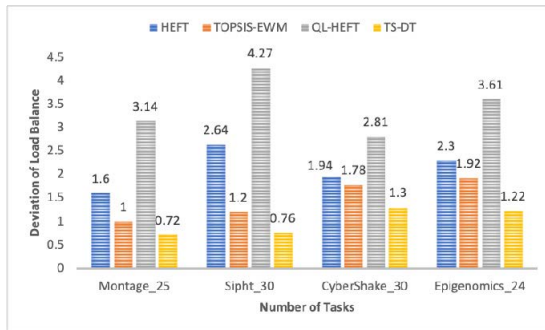
(c) 20 VMs

(d) 40 VMs

FIGURE 11. Comparative results for load balance.

- The proposed TS-DT algorithm selects the minimum of CP and VM with EFT for the assigned task to VM during the Resource Matrix phase. This means that the proposed algorithm consumes the high available

TABLE 7. Average in power consumption (in %) of the proposed TS-DT algorithm.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	0.752	22.811	18.312
10	12.576	36.823	22.276
20	6.752	51.726	31.553
40	44.468	62.714	41.398
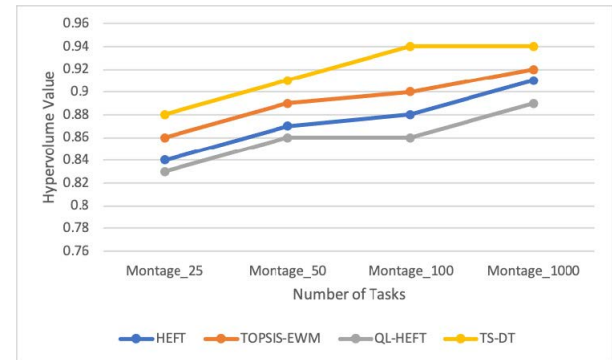
FIGURE 12. The hypervolume of the compared algorithms for Montage.
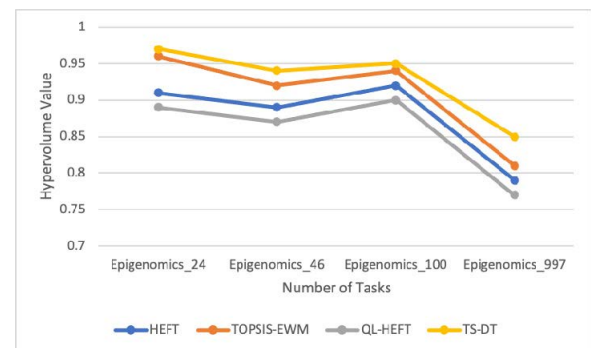
FIGURE 13. The hypervolume of the compared algorithms for Epigenomics.

VMs specifications, which increases the power consumption rate of the devices.

According to the implementation results in Table 7, it is found that the proposed TS-DT algorithm increases the power consumption relative to the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 12.89 %, 43.52 %, and 28.38 %, respectively. This is considered the main limitation of the proposed TS-DT.

Generally, the average Makespan, resource utilization, load balance, and power consumption of the proposed TS-DT, HEFT, TOPSIS-EWM, and QL-HEFT algorithms are shown in Tables (4,5, 6, and 7).

With respect to makespan, resource utilization, and load balance, the performance of the TS-DT algorithm is better than that of HEFT, TOPSIS-EWM, and QL-HEFT algorithms in most cases, which is also confirmed by the hypervolume as shown in Figures (12, 13) for Montage and Epigenomics workflow. This is because the HEFT algorithm is a

single-objective scheduling algorithm and does not consider other objectives such as load balance and resource utilization. At the same time, the TOPSIS-EWM algorithm is designed to select the best virtual machine for each task for multi-objective workflow scheduling. It doesn't taking into consideration any user preferences like deadlines, load balance, and resource utilization. In contrast, the QL-HEFT algorithm for solving large-scale task optimization issues has limitations, such as the Makespan task schedule.

VIII. CONCLUSION AND FUTURE WORK

In this paper, a new task scheduling algorithm using multi-objective based on a decision tree, called TS-DT algorithm, is proposed for a cloud computing environment. The proposed TS-DT algorithm targets minimizing the makespan, enhancing load balance, and maximizing resource utilization. A comparative study was conducted to evaluate the performance of the proposed TS-DT algorithm relative to the HEFT, TOPSIS-EWM, and QL-HEFT algorithms. According to the comparative results, the proposed TS-DT algorithm outperforms the HEFT, TOPSIS-EWM, and QL-HEFT algorithms by reducing the average makespan by 5.21%, 2.54%, and 3.32%, respectively, improving the average resource utilization by 4.69%, 6.81%, and 8.27%, respectively, and improving the average load balancing by 33.36%, 19.69%, and 59.06%. The main limitation of our proposed TS-DT algorithm is the increase in power consumption has been increased by around 12.89%, 43.52%, and 28.38%, respectively, relative to the existing HEFT TOPSIS-EWM, and QL-HEFT algorithms.

More performance parameters could be concerned in future work, such as power consumption, fault tolerance, and scalability.

REFERENCES

[1] J. Srinivas *et al.*, "Cloud computing basics," *Creating Smart Enterprises*, vol. 1, pp. 141–171, Jun. 2017, doi: 10.1201/9781315152455-6.

[2] D. Pooja, "Cloud computing—Overview and its challenges," *Int. J. Multidisciplinary*, vol. 3085, no. 3, pp. 499–501, 2019.

[3] M. Sajid and Z. Raza, "Cloud computing: Issues & challenges," in *Proc. Int. Conf. Cloud, Big Data Trust*, Jun. 2015.

[4] Q. Jiang, Y. C. Lee, M. Arenaz, L. M. Leslie, and A. Y. Zomaya, "Optimizing scientific workflows in the cloud: A montage example," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 517–522, doi: 10.1109/UCC.2014.77.

[5] E. Sarhan, A. Ghalwash, and M. Khafagy, "Queue weighting load-balancing technique for database replication in dynamic content web sites," *Proc. 9th WSEAS Int. Conf. Appl. Comput. Sci. (ACS)*, Jan. 2009, pp. 50–55.

[6] K. Kanagaraj and S. Swamynathan, "A realistic approach for representing and scheduling workflows in cloud computing environment," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 1615–1621, doi: 10.1109/ICACCI.2016.7732279.

[7] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "A comparative study of heterogenous task-based scheduling techniques in a cloud environment," in *Proc. Int. Conf. Innov. Trends Commun. Comput. Eng. (ITCE)*, Feb. 2020, pp. 1–6, doi: 10.1109/ITCE48509.2020.9047806.

[8] H. M. Alkhashai and F. A. Omara, "An enhanced task scheduling algorithm on cloud computing environment," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 7, pp. 91–100, Jul. 2016, doi: 10.14257/ijgdc.2016.9.7.10.

[9] M. Lavanya, B. Shanthi, and S. Saravanan, "Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment," *Comput. Commun.*, vol. 151, pp. 183–195, Feb. 2020, doi: 10.1016/j.comcom.2019.12.050.

[10] N. Kumar, "A review on machine learning algorithms, tasks and applications," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 6, no. 10, pp. 1–5, Oct. 2017.

[11] K. Naik, G. Gandhi, and S. Patil, "Multi-objective virtual machine selection for task scheduling in cloud computing: ICCI-2017," in *Advances in Intelligent Systems and Computing*. 2019, pp. 319–331.

[12] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146379–146389, 2019, doi: 10.1109/ACCESS.2019.2946216.

[13] M. S. Kumar, A. Tomar, and P. K. Jana, "Multi-objective workflow scheduling scheme: A multi-criteria decision making approach," *J. Ambient Intell. Hum. Comput.*, vol. 12, no. 12, pp. 10789–10808, Dec. 2021, doi: 10.1007/s12652-020-02833-y.

[14] J.-Q. Li and Y.-Q. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Comput.*, vol. 23, no. 4, pp. 2483–2499, Dec. 2020, doi: 10.1007/s10586-019-03022-z.

[15] A. Kaur, P. Singh, R. S. Batth, and C. P. Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," *Softw. Pract. Exp.*, vol. 52, pp. 1–21, Jan. 2020, doi: 10.1002/spe.2802.

[16] A. Al-maamari and F. A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 5, pp. 245–256, Oct. 2015, doi: 10.14257/ijgdc.2015.8.5.24.

[17] A. E. Keshk, "Cloud computing online scheduling," *IOSR J. Eng.*, vol. 4, no. 3, pp. 07–17, Mar. 2014, doi: 10.9790/3021-04360717.

[18] J.-M. Yu, H.-H. Doh, Y.-J. Kwon, J.-H. Shin, H.-W. Kim, S.-H. Nam, and D.-H. Lee, "Decision tree based scheduling for static and dynamic flexible job shops with multiple process plans," *J. Korean Soc. Precis. Eng.*, vol. 32, no. 1, pp. 25–37, Jan. 2015, doi: 10.7736/KSPE.2015.32.1.25.

[19] L. Yuan, Y. Dong, Y. Li, R. Zhang, and H. Xie, "A task parallel programming framework based on heterogeneous computing platforms," in *Intelligent Systems, Technologies and Applications*. Jan. 2020, pp. 169–184, doi: 10.1007/978-981-15-3914-5_13.

[20] S. K. Mishra, D. Puthal, B. Sahoo, S. K. Jena, and M. S. Obaidat, "An adaptive task allocation technique for green cloud computing," *J. Supercomput.*, vol. 74, no. 1, pp. 370–385, 2018, doi: 10.1007/s11227-017-2133-4.

[21] A. Dhari and K. I. Arif, "An efficient load balancing scheme for cloud computing," *Indian J. Sci. Technol.*, vol. 10, no. 11, pp. 1–8, Mar. 2017, doi: 10.17485/ijst/2017/v10i11/110107.

[22] M. S. Arif, Z. Iqbal, R. Tariq, F. Aadil, and M. Awais, "Parental prioritization-based task scheduling in heterogeneous systems," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3943–3952, Apr. 2019, doi: 10.1007/s13369-018-03698-2.

[23] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," *Proc. Comput. Sci.*, vol. 125, pp. 725–732, Jan. 2018, doi: 10.1016/j.procs.2017.12.093.

[24] B. Singh and P. Mehta, "A survey of scheduling algorithms for heterogeneous systems and comparative study of HEFT and CPOP algorithms," *Int. J. Eng. Res.*, vol. V5, no. 5, pp. 250–254, May 2016.

[25] A. Mazrekaj, A. Sheholli, D. Minarolli, and B. Freisleben, "The experiential heterogeneous earliest finish time algorithm for task scheduling in clouds," in *Proc. 9th Int. Conf. Cloud Comput. Services Sci.*, 2019, pp. 371–379, doi: 10.5220/0007722203710379.

[26] Z. Xie, X. Shao, and Y. Xin, "A scheduling algorithm for cloud computing system based on the driver of dynamic essential path," *PLoS One*, vol. 11, no. 8, pp. 1–19, 2016, doi: 10.1371/journal.pone.0159932.

[27] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," *IEEE Access*, vol. 7, pp. 160916–160926, 2019, doi: 10.1109/ACCESS.2019.2948704.

[28] Y. Samadi, M. Zbakh, and C. Tadonki, "E-HEFT: Enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2018, pp. 601–609, doi: 10.1109/HPCS.2018.00100.

[29] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "QL-HEFT: A novel machine learning scheduling scheme base on cloud computing environment," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5553–5570, May 2020, doi: 10.1007/s00521-019-04118-8.

[30] H. Alinejad-Rokny, "Divide and conquer classification," *Austral. J. Basic Appl. Sci.*, vol. 5, no. 12, 2014.

[31] K.-C. Jeong and Y.-D. Kim, "A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules," *Int. J. Prod. Res.*, vol. 36, no. 9, pp. 2609–2626, 1998.

[32] D. M. Abdelkader and F. Omara, "Dynamic task scheduling algorithm with load balancing for heterogeneous computing system," *Egyptian Informat. J.*, vol. 13, no. 2, pp. 135–145, Jul. 2012, doi: 10.1016/j.eij.2012.04.001.

[33] H.-S. Choi, J.-S. Kim, and D.-H. Lee, "Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3514–3521, Apr. 2011.

[34] N. Sasikaladevi, "Minimum makespan task scheduling algorithm in cloud computing," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 11, pp. 61–70, Nov. 2016, doi: 10.14257/ijgdc.2016.9.11.05.

[35] M. Pawlish, A. Varde, and S. Robila, "Analyze utilization rates in data centers to optimize energy management," in *Proc. Int. Green Comput. Conf. (IGCC)*, 2012, pp. 1–6.

[36] T. Rashid and M. T. Rashid, "Design and implementation of load balancing system for a smart home," *Proc. 3rd Int. Sci. Conf., Souther Tech. Univ.*, Mar. 2018, pp. 1–6.

[37] N. Quang-Hung and N. Thoai, "Minimizing total busy time with application to energy-efficient scheduling of virtual machines in IaaS clouds," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, Nov. 2016, pp. 141–148, doi: 10.1109/ACOMP.2016.029.

[38] J. J. Kanet and V. Sridharan, "Scheduling with inserted idle time: Problem taxonomy and literature review," *Oper. Res.*, vol. 48, no. 1, pp. 99–110, Feb. 2000, doi: 10.1287/opre.48.1.99.12447.

[39] A. M. Chirkin, A. S. Z. Belloum, S. V. Kovalchuk, M. X. Makkes, M. A. Melnik, A. A. Visheratin, and D. A. Nasonov, "Execution time estimation for workflow scheduling," *Future Gener. Comput. Syst.*, vol. 75, pp. 376–387, Oct. 2017, doi: 10.1016/j.future.2017.01.011.

[40] V. Legout, M. Jan, and L. Pautet, "Scheduling algorithms to reduce the static energy consumption of real-time systems," *Real-Time Syst.*, vol. 51, no. 2, pp. 153–191, Mar. 2015, doi: 10.1007/s11241-014-9207-7.

[41] H. Zhao, G. Qi, Q. Wang, J. Wang, P. Yang, and L. Qiao, "Energy-efficient task scheduling for heterogeneous cloud computing systems," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Communications; IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 952–959, doi: 10.1109/HPCC/SmartCity/DSS.2019.00137.

[42] N. R. Ortiz-Pimiento and F. J. Díaz-Serna, "Relative average deviation as measure of robustness in the stochastic project scheduling problem," *Revista Facultad Ingeniería*, vol. 28, no. 52, pp. 77–97, Jun. 2019, doi: 10.19053/01211129.v28.n52.2019.9756.

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

[44] *Caltech IPAC Software*. Accessed: May 2021. [Online]. Available: https://github.com/Caltech-IPAC/MontageMosaics

[45] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

MOSTAFA THABET received the Ph.D. degree in information systems, in 2019. He started M.B.A. studies at the Arab Academy for Science Technology & Maritime Transport (AASTMT). He is currently a Lecturer with the Faculty of Computers and Information, Fayoum University, Egypt. He also works as a Media Consultant at Fayoum University for four years. He is a member of the Big Data Research Group, Fayoum University. He has been working as the Vice President of the Fayoum Cancer Center (FOC), since July 2020. He worked as a project manager of many projects at Fayoum University. His research interests include the IoT, web, learning, database systems, high performance computing, cloud computing, and big data. He worked as the Editor-in-Chief of *Elfagr Egyptian Newspaper*. He is working as a Certified International Professional Trainer in a set of Egyptian organization.

MOHAMED H. KHAFAGY received the Ph.D. degree in computer science, in 2009. He worked as a Postdoctoral Researcher at the DIMA Group, Technique University Berlin, in 2012. He shares to establish the first Big Data Research Group in Egypt with Cairo University, in 2013. He is currently the Manager of the Electronic Exams Center, Supreme Council of Universities. He also works as a Consultant at Oracle Egypt. He is the Head of the Big Data Research Group, Fayoum University. He worked as a project manager of many projects at Fayoum University. He has many publications in the area of big data, cloud computing, and database systems.

HADEER MAHMOUD received the bachelor's degree in computer science from the Faculty of Information Systems and Computer Science, October 6 University, Egypt, in 2013, and the M.Sc. degree in computer science from the College of Computing and Information Technology, Arab Academy for Science Technology and Maritime Transport, in 2018. She is currently pursuing the Ph.D. degree with the Faculty of Computers and Information, Fayoum University, Egypt. She is currently a Staff Member with the Department of Computer Science, Faculty of Information Systems and Computer Science, October 6 University. She is also a Cloud Computing Instructor at Huawei Academy. Her research interests include high-performance computing, cloud computing, big data, and security.

FATMA A. OMARA (Member, IEEE) is currently a Professor with the Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University. She has supervised over 50 Ph.D. and M.Sc. theses. She has published over 100 research papers in prestigious international journals and conference proceedings. Her research interests include parallel and distributing computing, distributed operating systems, high performance computing, grid, cloud computing, and big data. She is a member of the IEEE Computer Society. She has served as the chairman and a member of Steering Committees and Program Committees of several national conferences.

● ● ●