The Journal of Supercomputing (2023) 79:1956–1979
https://doi.org/10.1007/s11227-022-04729-4

A novel technique to optimize quality of service for directed acyclic graph (DAG) scheduling in cloud computing environment using heuristic approach

Ranjit Rajak[1] · Shrawan Kumar[2] · Shiv Prakash[3] · Nidhi Rajak[1] · Pratibha Dixit[4]

Accepted: 16 July 2022 / Published online: 4 August 2022

Abstract

At present, the cloud computing environment (CCE) has emerged as one of the significant technologies in communication, computing, and the Internet. It facilitates on-demand services of different types based on pay-per-use access such as platforms, applications and infrastructure. Because of its growing reputation, the massive requests need to be served in an efficient way which gives the researcher a challenging problem known as task scheduling. These requests are handled by method of efficient allocation of resources. In the process of resource allocation, task scheduling is accomplished where there is a dependency between tasks, which is a Directed Acyclic Graph (DAG) scheduling. DAG is one of the most important scheduling due to wide range of its applicable in different areas such as environmental technology, resources, and energy optimization. NP-complete is a renowned concern, so various models deals with NP-complete that have been suggested in the literature. However, as the Quality of Service (QoS)-aware services in the CCEplatform have turned into an attractive and prevalent way to provide computing resources emerges as a novel critical issue. Therefore, the key aim of this manuscript is to develop a novel DAG scheduling model for optimizing the QoS parameters in the CCEplatform and validation of this can be done with the help of extensive simulation technique. Each simulated result is compared with the existing results, and it is found that newly developed algorithm performs better in comparison to the state-of-the-art algorithms.

Keywords Cloud computing · DAG scheduling · HEFT · QoS parameters · Heuristic approach

✉ Shiv Prakash
shivprakash@allduniv.ac.in

Extended author information available on the last page of the article

1 Introduction

A cloud computing environment (CCE) can be utilized as a prominent model for business and high-performance computing which got additional attention from academia and industry [1, 2]. Due to the focus on a very high number of shared resources [2] into a resource group to achieve large-scale, robust, and efficient utilization of resources via the Internet with negligible managing and maintenance costs. Further, it provides different services to the users via cloud virtualization methods and it also provides customers to focus on business activities without expenses on various resources. Alternatively, for infrastructure-as-a-service (IaaS) facilitators viz. Amazon EC2, Google Drive, Microsoft Azure [3, 4] provide pooled resources by using virtual machines (VMs) to users on-demand(UOD). It can accuse the user's charges which not only surgerate of resource utilization (RU) but also carry their significant benefit. There are three key groups of cloud service providers, such that public, private, and hybrid models [5]. The public model [6, 7] is most commonly used to organize the CCE platform, in which resources are possessed and functioned by the third-party service providers. The public cloud models send and impose services via Internet to end-users which are made for the service market adapted for organizations. On the other hand, private cloud models (such that IBM, Sun, and Cisco) are built for particular organizations. Precisely, reserved cloud models are accomplished and sustained by organizations and only organization members can access these models. However, to enhance the computing and storage capabilities, these private organizations strive to attain the public cloud resources also. Furthermore, hybrid cloud models [8–10] are evolved as encouraging computing prototypes that associate both cloud resources for better competence and scalability, also for attaining more plasticity and more disposition possibilities in hybrid cloud applications. Further, DAG scheduling [11] is frequently used in the scientific workflow model. A DAG scheduling contains numerous tasks/jobs and many dependencies between these tasks/jobs. In the CCE platform, users may buy various services facilitated by the CCE service provider to perform their acquiesced DAGs. Each DAG is typically linked with a deadline of execution to guarantee the quality of service (QoS) and low QoS typically enforces penalties; however, the provider of service may charge users based on makespan and QoS Parameters obtained during DAG Scheduling [12, 13]. Thus, to maximize profits along with guarantee QoS. Further, the applications of DAG scheduling in real-life problems [14, 15] are such as environmental technology; loop boundaries are identified during compile-time, data-flow computing problems, and various numerical methods, e.g. Gaussian elimination, Fourier transform, and its variants. In this paper, a novel heuristic-guided model with the QoS parameter optimization model is proposed to solve scheduling problems in DAGs in the CCE platform to get a near-optimal solution. So that the QoS parameters should be optimized to give better services to the user with optimal resource utilization.

After the introduction, the remaining parts of this manuscript are prearranged as given below. Section 2 contains a literature review and associated work. In

Sect. 3, the proposed model and its formulation are given. Section 4 explains our concern to reducing scheduling length and cost. QoS Parameters in the DAG scheduling model with an algorithm are illustrated in Sect. 5. Section 6 describes the proposed algorithm (NHGCPM) with experiments simulation setup with the experimental results and corresponding analysis. The concluding remarks with future directions are mentioned in Sect. 7.

2 Literature review

Scheduling is a process that is used to allocate resources such that processor, bandwidth, frequency, memory, etc. to perform many jobs/tasks. The main role of task scheduling in cloud platforms is to optimize the total execution time [16]. This guarantees that a system is capable of serving each request to attain the desired QoS parameters. Suppose there is a list having *n* jobs /tasks to be scheduled on a system. Each task takes a certain amount of time and gives when stated in this way, the Scheduling problem is NP-hard [17–20], and we do not know about of any efficient, that is, polynomial-time algorithm for it. Generally, scheduling is of two types first is static, and the second is dynamic. Static scheduling identifies the properties of a parallel program like processing-time, communication-cost, dependencies on data, and many synchronizations before executing the program. In the parallel program, node and edge's weighted DAG are indicated in which task/job processing times are represented by node's weights and data dependencies; moreover, the communication-time between jobs/tasks is represented by edge, respectively. A DAG is a directed graph with no cycles and has particular importance nowadays. The key concepts are captured and analyzed using task scheduling models. The DAG scheduling is a very difficult problem since it has been proved that it is NP-complete [13, 21, 22]. Its major objective is to optimize the QoS parameters such as the makespan of a different application by appropriately assigning the tasks/jobs to the systems. There is two scheduling problems such as dynamic and static but this entire paper is based on the static scheduling problem. There are various good approximation algorithms [23, 24] for NP-hard/complete optimization problems like DAG scheduling [25]. The scheduling problem is NP-hard [8, 14, 18], and not know any polynomial-time(P) algorithm which is effective and efficient for the same. As a significance, the general preemptive scheduling problem is also falling in NP-complete [22, 25, 26].

In static scheduling algorithms, every piece of information is required for scheduling, such as the parallel application structure, execution time associated with each task/job, and transmission time between tasks/jobs must be recognized in advance. There are many methods to provide an approximate solution for DAG Scheduling. Furthermore, a few examples of existing DAG scheduling algorithms available in the literature are as follows: Heterogeneous Earliest Execution Time (HEFT), Critical Path on a Processor (CPOP), Critical Path on a Cluster (CPOC), Modified Critical Path (MCP), and Mapping Heuristic (MH) [27]. Topcuoglu et al.[28] considered a comparative and critical analysis among the HEFT, CPOP, and MH algorithms for different values of different DAG inputs. The results obtained in this study reveal that the performance of the HEFT algorithm is better than the CPOP, and MH algorithms. In the HEFT Algorithm, the idle time slot is high, and utilization of the resource is small due to the data's excessive transmission delay between the jobs/tasks and the imbalanced structure of the DAG. Task scheduling performance may more improved by using prioritizing method and backfilling processes as suggested by Li et al. [29]. Some more task scheduling techniques [30–32] have been developed to optimize the scheduling length and other parameters of scheduling algorithms. To address the research gap a heuristic guided algorithm, i.e. a novel heuristic guided CPM (NHGCPM) is proposed to improve the performance.

The HEFT algorithm is typically used to obtain a scheduling length and also can be paralleled with different scheduling algorithms demonstrating a minimum time complexity. Moreover, it can be enhanced in two ways, firstly adding a new mechanism that can calculate the task priority, and secondly adding a novel mechanism that can improve the processor selection [33].Though this algorithm preschedules the residual task afore selected the scheduling tasks[34] and later calculated the relative duration of each DAG, moreover execution has become the most important part of the DAG. Therefore, each DAG can rearrange hybrid scheduling before the timeline, thus improving resource utilization. Nevertheless, various DAGs would be rejected when there is a resource shortage.

3 The Problem and the Proposed Model

The main objective of proposed model is to optimize the makespan-through appropriate allocation of the tasks to the nodes and arrangement of the execution sequence of the jobs/tasks. The performance of scheduling antecedence limits between jobs/tasks is conserved. The problem describes three key points related to jobs/task scheduling in CCE which can be applied in the DAG scheduling model, resource model, and the objective of scheduling [21, 25, 34].

3.1 DAG scheduling model

DAG model is the depiction of the application program using a directed graph (V, E) which is recognized as a DAG. It is denoted by F_w which is mathematically defined as $F_w = \{X_t, E, D_T\}$ where $X_t = \{x_1, x_2, \ldots, x_n\}$ is the finite set of jobs/tasks of a given DAG, $F_{w,}$, $E = \{e_{ij}\}, e_{ij}$ is $x_i \rightarrow x_j$ edges between the tasks x_i and x_j. This is also known as dependency between the jobs/tasks. And D_T is data transmission time which is linked with E and this time is taken between the jobs/tasks x_i and x_j.

A simple F_w is depicted in Fig. 1 which contains seven tasks. Here, x_1 is an entry or starting job/task which does not have any parent jobs/tasks. Also, x_7 is an end job/task or exit job/task and it also has not any children jobs/tasks.

The precedence constraint [21, 25, 33] should be kept between the jobs/tasks during the assignment of the jobs/tasks onto virtual machines (VMs) in the CCE platform.

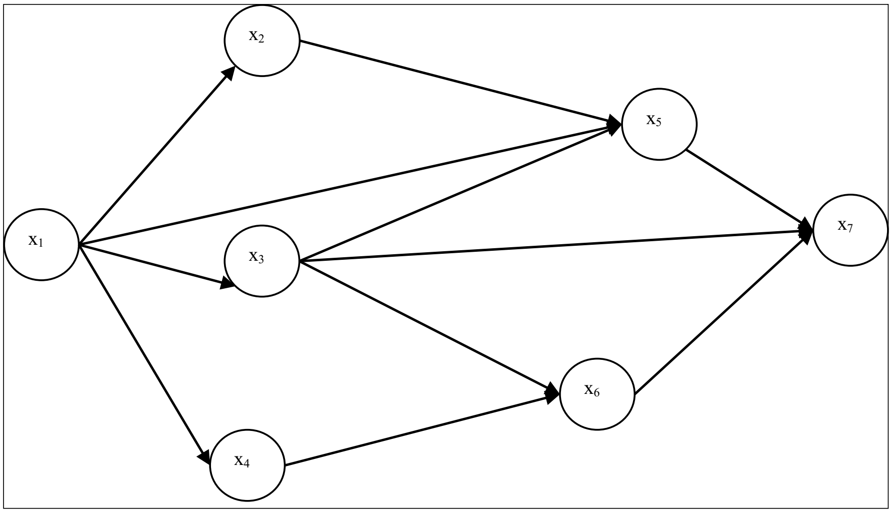


Fig. 1 F_w Simple Model with 7 jobs/tasks

3.2 Resource Allocation Model for CCE platform

This section is related with the jobs/tasks allocation model with VMs which can be categorized into two different types such as homogenous and heterogeneous resource allocation models for the CCE platform. The homogeneous model comprises multiple VMs which are interconnected in the half-duplex mode, whereas the heterogeneous model contains multiple VMs which are interconnected in the full-duplex mode. Formally, this model consists of cloud server $S=\{s_i| i=1$ to p^{th} servers$\}$ and each s_i having one or more VMs (M) [35].

The mapping of tasks of given F_w onto cloud server S is shown in Fig. 2 [20, 24, 34, 36–38]. If two jobs/tasks belong to the same cloud server, then their data transmission time should be null. i.e.

$$D_T(x_i, x_j) = \left\{ \begin{array}{ll} 0, & \text{if } x_i \text{ and } x_j \text{ onto same cloud server} \\ \text{Time}\left(D_T(x_i, x_j)\right), & \text{oterwise} \end{array} \right\} \quad (1)$$

where *Time* depicts the transmission delay between the jobs/tasks x_i and x_j.

3.3 Objective function for the DAG scheduling

The objective function of the proposed model is to optimize the overall execution time of the tasks onto VM, i.e. to minimize the makespan of the DAG scheduling job/task [11, 12, 32].

$$Makespan = Min.\left\{EFT\left(x_{exit}, M\right)\right\} \quad (2)$$

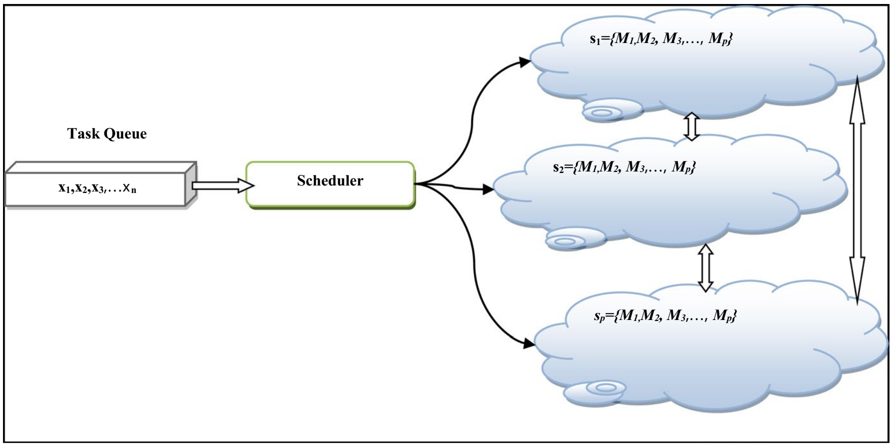


Fig. 2 Mapping of Tasks onto VMs [38]

where *EFT* is the earliest finished time of exit task x_{exit} *of given DAG* F_w onto M.

3.3.1 Scheduling Attributes

Task scheduling attributes are very essential for allocation of the jobs/tasks onto machines as well as for finding the performance metrics of scheduling algorithms. The brief details of the attributes are as follows [32]:

a. ***Estimated Computation Time(ECT)*** [38–41] is given as follows

$$ECT_{ij} = \begin{bmatrix} ECT_{11} & ECT_{12} \cdots & ECT_{1n} \\ ECT_{21} & ECT_{22} \ldots .. & ECT_{2n} \\ ECT_{m1} & ECT_{m2} & ECT_{mn} \end{bmatrix} \tag{3}$$

where ECT_{ij} is time of job/task $x_i$on machine $\mathrm{M_j}$.

b. **Average ECT (AVG)** [38, 42]of a job/task x_i can be computed as the ratio of summation of ECT of all M and the total number of M. i.e.

$$AVG_i = \frac{\sum_{j=1}^{TotalM} ECT_{i,j}}{TotalM} \tag{4}$$

c. **Critical Path(CP)** [38, 43, 44] of F_w is given by equation

$$CP = \max_{path \epsilon Fw} \{length(path)\} \tag{5}$$

$$length(path) = \sum_{x_i \epsilon X} AVG(x_i) + \sum_{e \epsilon E} D_T(x_i, x_j) \tag{6}$$

d. **Earliest Start Time (EST)**. [38, 45] is given as follows:

$$EST\left(x_i, M_j\right) = \left\{ \begin{array}{ll} 0 & \text{if } x_i \in x_{entry} \\ \max\limits_{x_i \in pred\left(x_i\right)} \left\{ EFT(x_j, M_j) + MET\left(x_i\right) + D_T\left(x_i, x_j\right) \right\} & \text{otherwise} \end{array} \right\} \tag{7}$$

e. **Minimum Execution Time (MET)** . [38, 46] is given as follows:

$$MET\left(x_i\right) = min.\left\{ECT\left(x_i, M_m\right)\right\} \tag{8}$$

f. **Earliest Finished Time (EFT)** . [38, 46] is given as follows:

$$EFT\left(x_i, M_j\right) = ECT_{ij} + EST\left(x_i, M_j\right) \tag{9}$$

4 QoS Parameters used in Performance Analysis

In order to differentiate between the proposed model and heuristic models, performance metrics play a very important role. It helps to analyze the performance of models in various F_w models. There are several scheduling metrics but here it has been taken only six metrics such as *Makespan, Scheduling-Length-Ratio (SLR), Speedup, Efficiency, Resource Utilization, and Cost*. The details of the metrics are as follows:

a. ***Makespan*** [35, 38, 42]: It is also called scheduling length and is defined as follows:

$$Makespan = Min.\{EFT(x_{exit}, M)\} \tag{10}$$

b. ***Scheduling Length Ratio (SLR)*** [38, 47, 48]:SLR can be computed as the ratio of Makespan and Critical Path tasks which takes minimum ECT value.

$$SLR = \frac{Makespan}{\sum_{x_i \in CP_{min}} Min(ECT_{i,j})} \tag{11}$$

c. ***Speedup*** [42, 43, 47] It is defined as follows:

where is the total number of M.

$$Speedup = \frac{Min.\left[\sum_{j=1}^{m} ECTi, j\right]}{Makespan} \tag{12}$$

d. ***Efficiency*** [42, 43, 47]: It can be computed as the ratio of *Speedup* and the total number of virtual machines. i.*e*.

$$Efficiency = \frac{Speedup}{m} \times 100 \tag{13}$$

e. ***Resource Utilization*** [36, 49]: It is measured by Average Resource Utilization(ARU) and the main goal of this metric is to increase the resource utilization.

$$ARU = \frac{\sum_{i=1}^{m} x(M_i)}{Makespan \times m} \times 100 \tag{14}$$

f. ***Cost*** [35, 38, 49]: The cost metric is very essential and it identifies the real cost of cloud resources used by end users. The general table for the cost metric is given in Table 1 [40]. It is defined as follows:

$$Cost = \sum E_{ij} \times C(VM_j) \tag{15}$$

where E_{ij} is the execution time of the task x_i on M_j and $C(VM_j)$ is the cost of M_j per unit time.

The existing scheduling algorithm considers specific characteristics in the CCE Platform for the proper allocation of resources. The complexity of the application, user need, and heterogeneity of the CCE platform prevent any DAG scheduling algorithm to achieve its optimal value of the QoS parameters to measure the relative performance.

5 Proposed algorithm Design and Analysis

The most widely used combinatorial optimization issues is the DAG scheduling issue, i.e. NP-Complete since it uses simple algorithms [13, 21, 22]. This can create problems that involves optimizing the QoS Parameters such as makespan and cost. To overcome this complex issue, there is an accurate method that indicates exponential time-complexity. Furthermore, such types of methods can only be used for small size issues. But for generalizion of these issues require an effective solution to optimize QoS parameters e.g. makespan, cost, throughput, etc. In the literature [27], many exact algorithms such as the HEFT and CPOP are designed to solve limited size DAG scheduling problems. The popular heuristic algorithms such as BFS and DFS methods [50] are the graph traversal methods. DFS is also known as an edge-oriented method that uses the stack technique for traversal in a graph whereas BFS is also known as a vertex-oriented method that uses the queue technique for traversal in a graph. These heuristics are easy to apply to simple search space problems but solving combinatorial issues is now become a major concern [50] such as DAG Scheduling using BFS and DFS. It motivates the authors to develop a novel algorithm referred to as a novel heuristic guided CPM (NHGCPM) to increase the performance of the existing system.

Table 1 Cost Rate for Virtual Machines (VM)

Virtual machine(M)	M_1	M_2	M_3	…	M_m
Cost/unit time	$Rate_1$	$Rate_2$	$Rate_3$	…	$Rate_r$

The NHGCPM combines the advantages of both DFS [44], HEFT, and CPM to find a higher extent of convergence to improve the application flexibility. Further, on the basis of jobs/tasks, priority of given F_w and the priority is computed using DFS [50]. Figure 3 represents the process of the proposed model. There are two key steps are used in the novel method of the DAG scheduling which are given as follows:

Step 1: This step is consisting of Task Stack (TS) and initially, it has only entry tasks of the given F_w model. The addition and deletion of the tasks/jobs as per the DFS method are performed. When the entry deletes for TS, it will be added to Ready Queue (RQ). This queue mostly comprises the priority of the tasks. All successors' tasks of the entry tasks will be inserted into TS. Similarly, all non-entry tasks are deleted from TS and added into RQ but here, the successor tasks of non-entry tasks whether it will be included in TS or not depending on the successor's tasks' present condition. The present condition means the successor task should be included in TS if and only if this task will not either TS or RQ. Otherwise, these tasks would be simple leave it, and not included as duplicates either in TS or RQ.

Step 2: This is the resource allocation step to the available virtual machine (VMs). In this step, the allocation of the tasks as per RQ, precedence constraint, Earliest start time first (EST), and Earliest finish time first (EFT). The priority of the task/job is in RQ but it should satisfy the precedence constraint before allocation to machines, if it is satisfied then allocation to VM as per EST and EFT of the task and machine. These attributes should be minimum. If any task/job does not satisfy the precedence constraint, in that case, the removed task will be added at the rear end of RQ. The complete description of the proposed model is described in Table 2.

6 Simulation environment and results discussion

The Simulation study has been conducted to test the performance of a novel developed model NHGCPM by simulating the benchmark the Cloud-Sim [6] which is used to generate based services for graph such as AWS Neptune., as a simulation platform using JAVA as a programming language. The components of the Cloud like data centers, brokers, hosts, and VM policies can be modeled by the Cloud-Sim toolkit. The analysis of the results is an essential and integral part of any research to find a good solution that helps to differentiate between the proposed model and the state of the art. This section presents the simulation result of the proposed model using three DAG Models as F_w *Model*^{1} [38] with 10 tasks $\{x_1,\ldots, x_{10}\}$, F_w *Model*^{2}[46] with 15 tasks $\{x_1,\ldots,x_{15}\}$ and F_w *Model*^{3} [38] with 26 tasks $\{x_1,\ldots,x_{26}\}$ as shown in Figures 4,7, and 10. Also, three models Expected time compute (ECT) value and cost value tables are shown in Tables 3, 4, 5, 6, 7, 8. The performance metrics such as makespan, speedup, efficiency, SLR, Average Resource Utilization (ARU), and Cost are used for comparison between the proposed model and the state-of-the-art models (Figs. 5, 6).

As per the proposed model, finding the TS and RQ of the given F_w *Model*^{1} and also computing the priority of the tasks/jobs are shown in Fig. 4.

The assignment of the tasks/jobs onto VM as PQ and Precedence Constraint is given in the algorithm. The tasks/jobs are removed from the front end of the PQ,

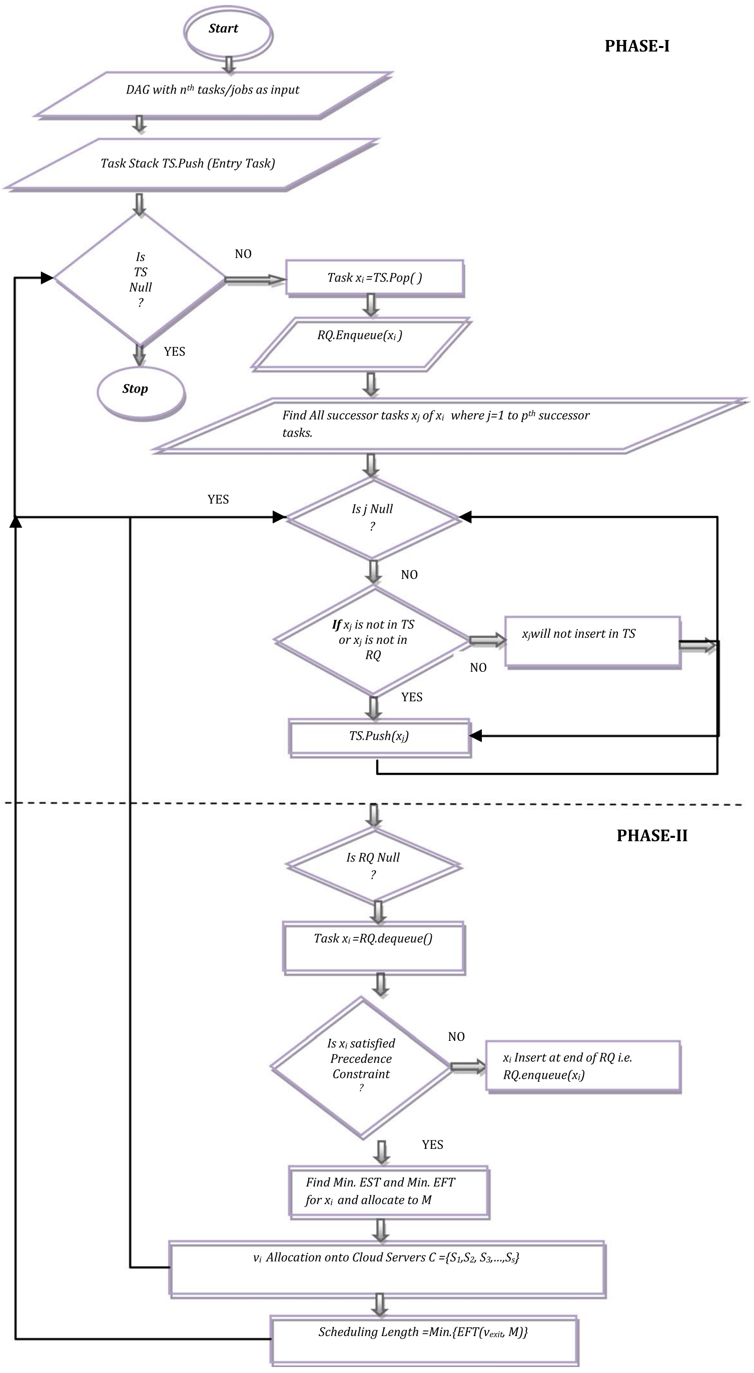

Start
PHASE-I
DAG with n^{th} tasks/jobs as input
Task Stack TS.Push (Entry Task)
Is TS Null ?
NO
YES
Stop
Task x_i =TS.Pop( )
RQ.Enqueue(x_i)
Find All successor tasks x_j of x_i where j=1 to p^{th} successor tasks.
YES
Is j Null ?
NO
If x_j is not in TS or x_j is not in RQ
x_j will not insert in TS
NO
YES
TS.Push(x_j)
PHASE-II
Is RQ Null ?
Task x_i =RQ.dequeue()
Is x_i satisfied Precedence Constraint ?
NO
x_i Insert at end of RQ i.e. RQ.enqueue(x_i)
YES
Find Min. EST and Min. EFT for x_i and allocate to M
v_i Allocation onto Cloud Servers C ={S_1,S_2, S_3,…,S_s}
Scheduling Length =Min.{EFT(v_{exit}, M)}

Fig. 3 Flowchart of NHGCPM Model

Table 2 Proposed algorithm: a novel heuristic guided CPM(NHGCPM)

```
Step 1.   Input :An Application Program DAG with nth Tasks/Jobs.
Step 2.   DFS_TaskScheduling(DAG, $x_{entry}$)      // $x_{entry}$ is an entry task of given DAG.
Step 3.       Take Task Stack TS
Step 4.   TS.push($x_{entry}$)
Step 5.   While  TS !=NULL then
Step 6.   Task$x_i$=TS.pop()
          RQ.enqueue($x_i$)    // Ready Queue RQ
Step 7.   Find All successor tasks $x_j$ of $x_i$  where j=1 to $p^{th}$  successor tasks.
          While j!=NULLthen
          If$x_j$ is not in TS or $x_j$ is not in RQ then
          TS.push($x_j$)
          Else
          $x_j$will not insert in TS
           End While.
Step 8.   End While
Step 9.   While RQ!=NULL then
Step 10.  Task $x_i$ =RQ.dequeue()
Step 11.  If $x_i$ is satisfied Precedence Constraint then
          Find Min. EST and Min. EFT for $x_i$
                        Allocate to M
Step 12.          Else
          $x_i$ Insert at end of RQ i.e. RQ.enqueue($x_i$)
Step 13.  End While
Step 14.  Find Makespan =Min.{EFT($x_{exit}$, M)}
Step 15.  Stop.
```

Table 3 ECT [38] Matrix for F_w model[1]

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
s_1 M_1	14	13	11	13	12	13	7	5	18	21
s_1 M_2	16	19	13	8	13	16	15	11	12	7
s_2 M_3	9	18	19	17	10	9	11	14	20	16

Table 4 VM rate for F_w model[1] [38]

Virtual machine(M)	M_1	M_2	M_3
Cost/unit time	1.71	1.63	1.21

Table 5 ECT [36] Matrix for F_w Model[2]

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
S_1	M_1	17	14	19	13	19	13	15	19	13	19	13	15	18	20	11
	M_2	14	17	17	20	20	18	15	20	17	15	22	21	17	18	18
S_2	M_3	13	14	16	13	21	13	13	13	13	16	14	22	16	13	21
	M_4	22	16	12	14	15	18	14	18	19	13	12	14	14	16	17

Table 6 VM rate for F_w model[2] [30]

Virtual machine s(M)	M_1	M_2	M_3	M_4
Cost/unit time	1.72	1.52	1.69	1.65

Table 7 ECT matrix for F_w model[3] [38]

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
s_1	M_1	16	13	19	12	18	13	14	20	12	18	13	15	17	21	11	12	18	19	14	16	16	12	12	10	14	11
	M_2	15	18	17	19	21	17	14	19	16	14	21	20	16	19	17	11	17	17	17	15	17	18	21	17	14	19
s_2	M_3	14	13	15	12	20	13	12	12	12	15	13	21	15	12	20	17	16	15	14	12	16	12	20	20	12	11
	M_4	21	15	13	15	14	17	17	18	18	12	11	13	13	15	16	12	14	13	15	21	20	17	13	18	13	13

Table 8 VM rate for F_w Model[3]

Virtual machine(VM)	M_1	M_2	M_3	M_4
Cost/unit time	1.82	1.67	1.79	1.77

check its precedence constraint, if it is satisfied then assign to virtual machine otherwise insert at the rear end of PQ.

Consider a cloud server $S=\{s_1,s_2\}$ having two servers $s_1=\{M_1,M_2\}$ consists of two virtual machines and $s_2=\{M_3\}$ contains of only one virtual machine. The assignment of the tasks/jobs as per the proposed algorithm, NHGCPM and computes makespan 65 units.

Case 1: F_w Model[1] with 10 Tasks

Case 2: *F_w Model[2] with 15 Tasks.*

Similarly, for F_wModel[2] finding the TS and RQ of the given F_w Model[2] and also computing the priority of the tasks in PQ as given in the proposed algorithm, NHGCPM, and process shown in Fig. 8.

The allocation of the tasks onto the virtual machine as PQ and Precedence Constraint is given in the algorithm. The tasks are removed from the front end of the PQ, check its precedence constraint, if it is satisfied then allocate to the virtual machine otherwise insert at the rear end of the PQ. Consider a cloud server $S=\{s_1,s_2\}$ having two servers $s_1=\{M_1,M_2\}$and $s_2=\{M_3,M_4\}$, both servers consists of two virtual machines. The allocation of the tasks as per the proposed algorithm, NHGCPM and gives makespan 136 units as shown in Fig. 9.

Case 3:F_w Model[3] with 26 Tasks

The allocation of the tasks onto the virtual machine as PQ and precedence constraint is given in the algorithm. The tasks are removed from the front end of the PQ, check its precedence constraint, if it is satisfied then allocate to virtual machine otherwise insert at the rear end of PQ (Figs. 10, 11). Consider a cloud server $S=\{CS_1, CS_2\}$ having two servers $CS_1=\{V_1,V_2\}$and $CS_2=\{V_3,V_4\}$, both servers consists of two virtual machines. In Fig. 12, Gantt chart depicted the allocation of the tasks as per the proposed algorithm (NHGCPM) and it gives makespan 181 units.

The details of the comparison between the proposed model and heuristic models using three DAG models are given in Table 9. Makespan is minimized in proposed model as compared to heuristic algorithms [38]. Other metrics results of heuristic algorithms are taken from [38]. The graphical representation of the scheduling algorithms is shown in Fig. 13, 14, 15, 16, 17, 18.

Firstly, we consider the effect of QoS Parameters in the input DAG over the makespan considering different parameters is shown in Table 9. Table 9 clearly specifies that NHGCPM performs better compared to the state of the art for all different scenarios.

Figure 13 represents the effect of makespan in the input DAG over the makespan considering different parameters like HEFT, CPOP, ALAP, PETS. It specifies that NHGCPM performs better compared to the state-of-the-art algorithms considered for three different scenarios.

Table 9 Performance analysis results for three DAGs

Performance metrics	Three DAG models														
	F_w model.[1]					F_w model.[2]					F_w model.[3]				
	Task scheduling algorithms					Task scheduling algorithms					Task scheduling algorithms				
	NHGCPM	HEFT	CPOP	ALAP	PETS	Proposed	HEFT	CPOP	ALAP	PETS	NHGCPM	HEFT	CPOP	ALAP	PETS
Scheduling Length	65	73	86	73	70	136	152	164	155	152	181	175	197	182	172
Speedup	1.95	1.74	1.48	1.74	1.81	1.70	1.52	1.41	1.49	1.52	2.12	2.19	1.95	2.11	2.23
Efficiency(%)	65.14	58.00	49.22	58.00	60.33	56.62	38.00	35.25	37.25	38.00	53.03	54.75	48.75	52.75	55.75
SLR	1.58	1.78	2.09	1.78	1.71	1.53	1.71	1.84	1.74	1.71	1.70	1.65	1.85	1.71	1.62
Resource Utilization(%)	73.85	77.12	77.51	77.16	80.47	79.60	69.73	64.93	72.74	65.13	73.89	90.71	89.84	91.07	90.98
Cost/Unit Time	227.04	258.65	301.12	258.65	265.39	711.16	702.75	694.92	751.53	666.43	939.09	931.76	1043.69	972.85	918.41

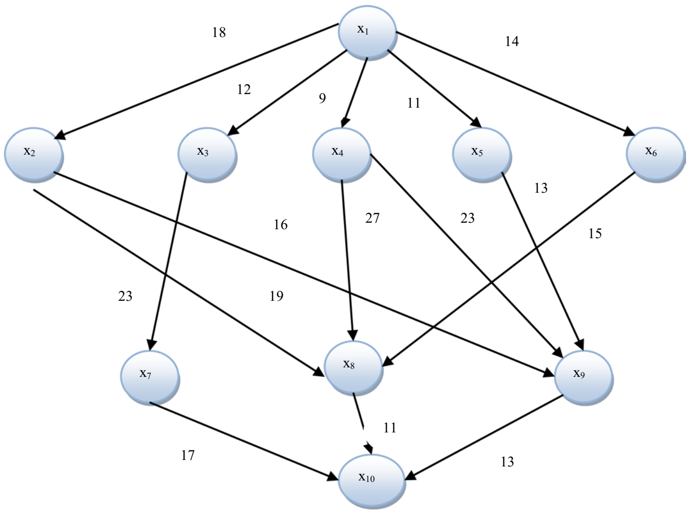
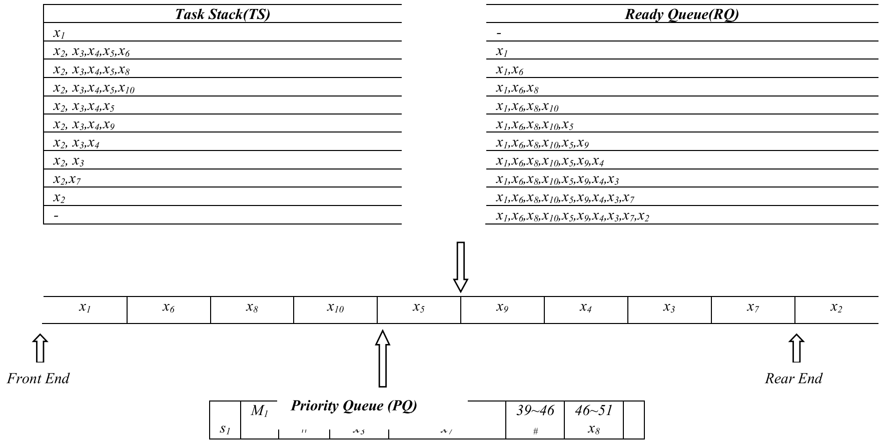


Fig. 4 F_w Model[1] with 10 tasks



Task Stack(TS)
x_1
x_2, x_3,x_4,x_5,x_6
x_2, x_3,x_4,x_5,x_8
x_2, x_3,x_4,x_5,x_{10}
x_2, x_3,x_4,x_5
x_2, x_3,x_4,x_9
x_2, x_3,x_4
x_2, x_3
x_2,x_7
x_2
-

Ready Queue(RQ)
-
x_1
x_1,x_6
x_1,x_6,x_8
x_1,x_6,x_8,x_{10}
x_1,x_6,x_8,x_{10},x_5
x_1,x_6,x_8,x_{10},x_5,x_9
x_1,x_6,x_8,x_{10},x_5,x_9,x_4
x_1,x_6,x_8,x_{10},x_5,x_9,x_4,x_3
x_1,x_6,x_8,x_{10},x_5,x_9,x_4,x_3,x_7
x_1,x_6,x_8,x_{10},x_5,x_9,x_4,x_3,x_7,x_2

x_1	x_6	x_8	x_{10}	x_5	x_9	x_4	x_3	x_7	x_2

Fig. 5 Tasks Transformation to Priority Queue

	M_2		18~26 x_4	26~27 #	27~46 x_2	46~58 x_9	**58~65 x_{10}**
s_2	M_3	0~9 x_1	9~18 x_5	19~28 x_6			

Fig. 6 Gantt. Chart for F_w Model.[1]

We consider the effect of *Speedup* in the input DAG over the makespan considering different parameters as shown in Fig. 14. It specifies that NHGCPM has highest values as compared to other parameters.

Case 2:*F_w Model^2 with 15 Tasks*

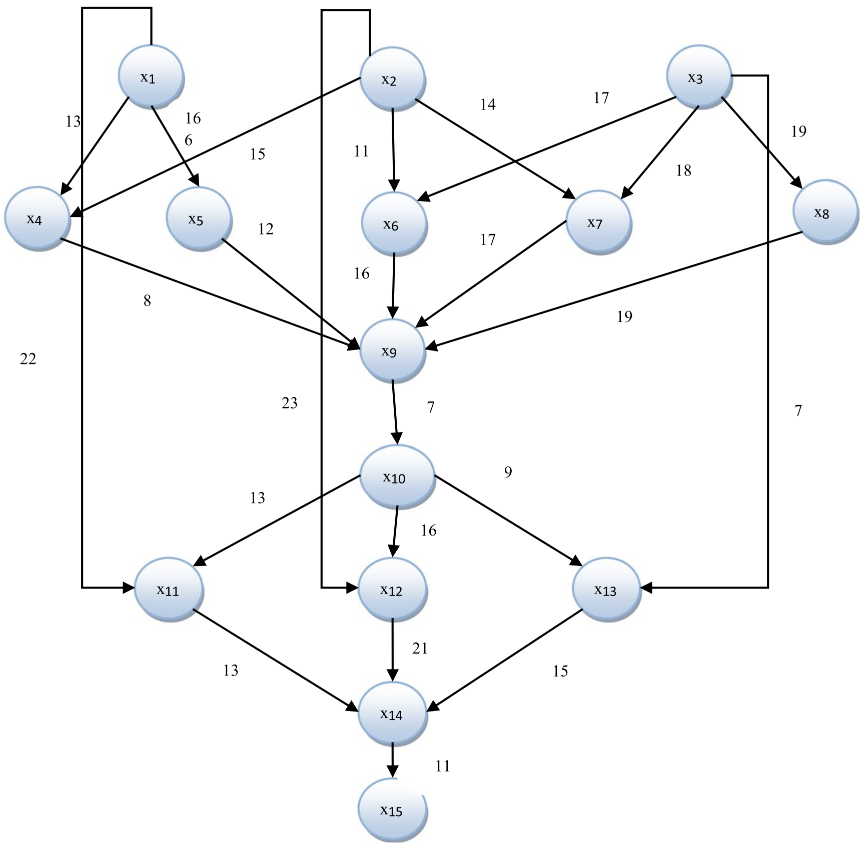


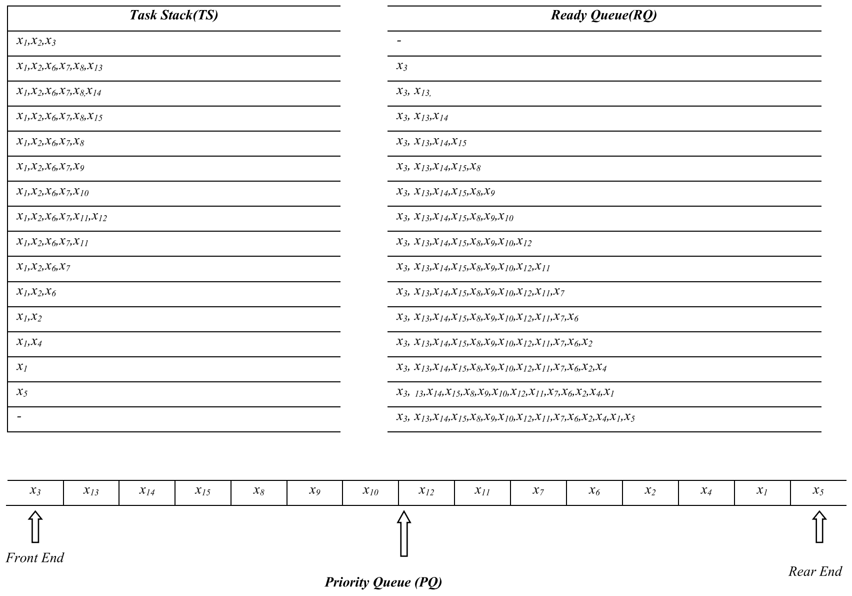
Fig. 7 F_w Model^2 with 15 tasks

Comparative study of *Efficiency* has been done in which the input DAG has been compared over the makespan considering different parameters in Fig. 15. It specifies that NHGCPM performs better compared to the state-of-the-art algorithms considered for three different scenarios.

We consider the effect of *SLR* in the input DAG over the makespan considering different parameters is shown in Fig. 16. NHGCPM performs best as compared to other algorithms.

Figure 17 represents the effect of *utilization* in the input DAG over the makespan considering different parameters. It specifies that NHGCPM performs best with respect to the state-of-the-art algorithms is considered for three scenarios.

We consider the effect of *cost* in the input DAG over the makespan considering different parameters is shown in Fig. 18 and it specifies that NHGCPM performs best with respect to other algorithms the state-of-the-art algorithms is considered for three scenarios.



Task Stack(TS)	***Ready Queue(RQ)***
x_1,x_2,x_3	-
$x_1,x_2,x_6,x_7,x_8,x_{13}$	x_3
$x_1,x_2,x_6,x_7,x_8,x_{14}$	x_3, x_{13},
$x_1,x_2,x_6,x_7,x_8,x_{15}$	x_3, x_{13},x_{14}
x_1,x_2,x_6,x_7,x_8	x_3, x_{13},x_{14},x_{15}
x_1,x_2,x_6,x_7,x_9	x_3, x_{13},x_{14},x_{15},x_8
x_1,x_2,x_6,x_7,x_{10}	x_3, $x_{13},x_{14},x_{15},x_8,x_9$
$x_1,x_2,x_6,x_7,x_{11},x_{12}$	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10}$
x_1,x_2,x_6,x_7,x_{11}	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12}$
x_1,x_2,x_6,x_7	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11}$
x_1,x_2,x_6	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7$
x_1,x_2	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7,x_6$
x_1,x_4	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7,x_6,x_2$
x_1	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7,x_6,x_2,x_4$
x_5	x_3, $_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7,x_6,x_2,x_4,x_1$
-	x_3, $x_{13},x_{14},x_{15},x_8,x_9,x_{10},x_{12},x_{11},x_7,x_6,x_2,x_4,x_1,x_5$

x_3	x_{13}	x_{14}	x_{15}	x_8	x_9	x_{10}	x_{12}	x_{11}	x_7	x_6	x_2	x_4	x_1	x_5

Fig. 8 Tasks Transformation to Priority Queue

s_1	M_1	0~14 x_2	14~27 x_4	27~125 #						**125~136** $\boldsymbol{x_{15}}$
	M_2	0~14 x_1	14~34 x_5	34~81 #			81~98 x_{13}	98~107 #	107~125 x_{14}	
s_2	M_3	0~12 #	12~25 x_8	25~38 x_6	38~46 #	46~59 x_9	59~72 #	72~86 x_{11}		
	M_4	0~12 x_3	12~26 x_7	26~59 #			59~72 x_{10}	72~86 x_{12}		

2

Fig. 9 Gantt. Chart F_w Model.[2]

7 Conclusion and future scope

The proposed model tried to analyze the cost optimization problem for DAG scheduling in a cloud environment. Several types of VM instances can be assigned, when diverse services are requested by users. From the experimental evaluation and comprehensive analysis of the proposed model demonstrate that the proposed algorithm, "A Novel heuristic guided CPM (NHGCPM)", outperforms the state-of-the-art algorithms with trade-offs between the several QoS Parameters.

However, it has been known that the most reliable and current platform is the QoS-driven public cloud services, and the QoS Parameters optimization emerges as a novel serious concern. Hence, we aimed to put forward a novel DAG scheduling model for this purpose to optimize the QoS parameters in a cloud environment, where task/job scheduling requires adopting resource provisioning to attain the near-optimal solution. This algorithm is widely replicated using a different benchmark, scientific and original DAGs used in environmental technology. All

***Case 3:**F_w Model^3 with 26 Tasks*

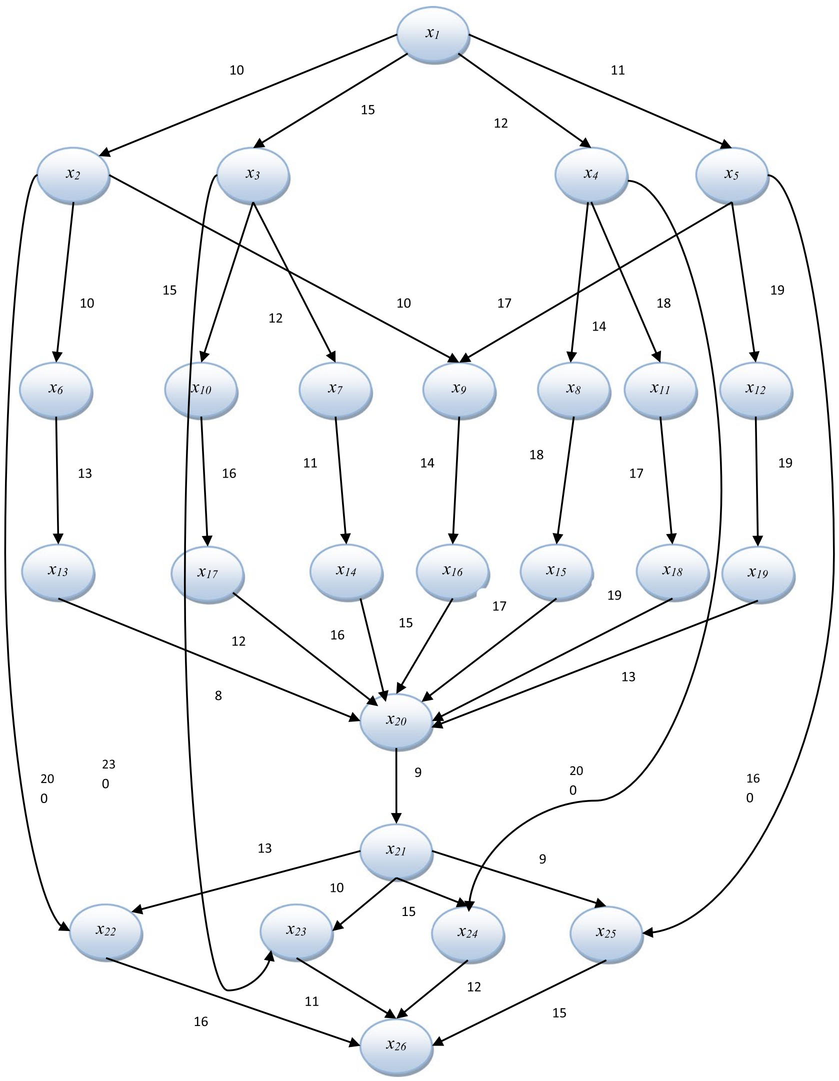
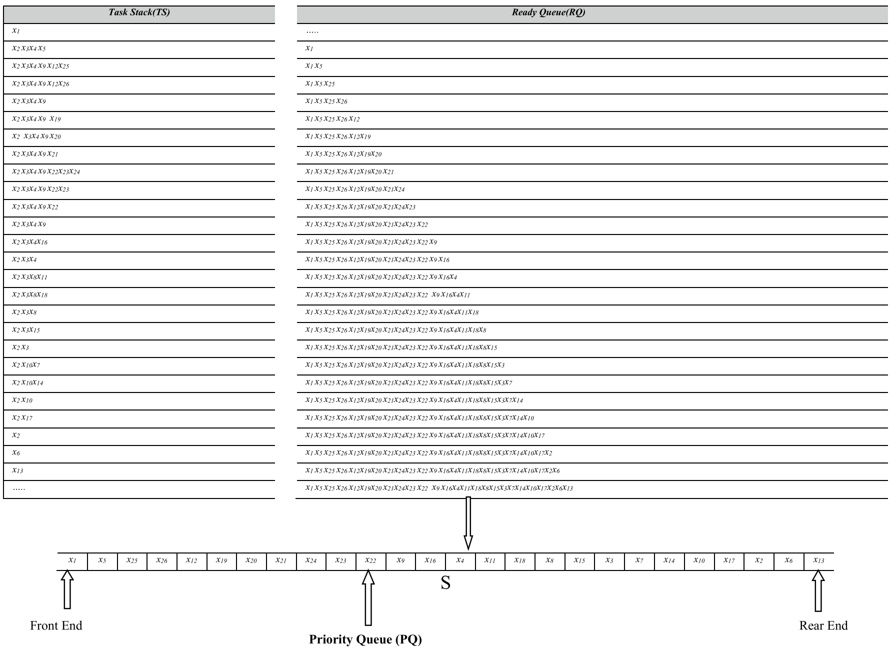


Fig. 10 F_w Model^3 with 26 tasks

the implemented results are paralleled with other state-of-the-art DAG scheduling algorithms, and it has been suggested that the given method facilitates users in the cloud computing environment. Outcomes are correspondingly endorsed



Task Stack(TS)	Ready Queue(RQ)
x_1	.....
$x_2 x_3 x_4 x_5$	x_1
$x_2 x_3 x_4 x_9 x_{12} x_{25}$	$x_1 x_5$
$x_2 x_3 x_4 x_9 x_{12} x_{26}$	$x_1 x_5 x_{25}$
$x_2 x_3 x_4 x_9$	$x_1 x_5 x_{25} x_{26}$
$x_2 x_3 x_4 x_9\ x_{19}$	$x_1 x_5 x_{25} x_{26} x_{12}$
$x_2\ x_3 x_4 x_9 x_{20}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19}$
$x_2 x_3 x_4 x_9 x_{21}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20}$
$x_2 x_3 x_4 x_9 x_{22} x_{23} x_{24}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21}$
$x_2 x_3 x_4 x_9 x_{22} x_{23}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24}$
$x_2 x_3 x_4 x_9 x_{22}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23}$
$x_2 x_3 x_4 x_9$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22}$
$x_2 x_3 x_4 x_{16}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9$
$x_2 x_3 x_4$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16}$
$x_2 x_3 x_8 x_{11}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4$
$x_2 x_3 x_8 x_{18}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22}\ x_9 x_{16} x_4 x_{11}$
$x_2 x_3 x_8$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18}$
$x_2 x_3 x_{15}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8$
$x_2 x_3$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15}$
$x_2 x_{10} x_7$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3$
$x_2 x_{10} x_{14}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7$
$x_2 x_{10}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14}$
$x_2 x_{17}$	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10}$
x_2	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10} x_{17}$
x_6	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10} x_{17} x_2$
x_{13}	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10} x_{17} x_2 x_6$
.....	$x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22}\ x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10} x_{17} x_2 x_6 x_{13}$

x_1 x_5 x_{25} x_{26} x_{12} x_{19} x_{20} x_{21} x_{24} x_{23} x_{22} x_9 x_{16} x_4 x_{11} x_{18} x_8 x_{15} x_3 x_7 x_{14} x_{10} x_{17} x_2 x_6 x_{13}
S
Front End
Priority Queue (PQ)
Rear End

Fig. 11 Tasks Transformation to Priority Queue (PQ)

S_1	M_1	0~29 \#	29~48 x_3	48~62 x_7	62~83 x_{14}	83~119 \#		119~135 x_{21}	135~145 x_{24}	145~157 x_{22}		157~170 \#	**170~181** $\mathbf{x_{26}}$
	M_2	0~40 \#		40~59 x_8	59~76 x_{15}	76~77 \#	77~93 x_9	93~104 x_{16}	104~119 x_{20}	119~135 \#	135~156 x_{23}	156~170 x_{25}	
S_2	M_3	0~14 x_1	14~26 x_4	26~39 x_{11}	39~54 x_{18}	54~67 x_2		67~80 x_6	80~95 x_{13}				
	M_4	0~14 \#	14~28 x_5	28~41 x_{12}	41~56 x_{19}	56~63 \#		63~75 x_{10}	75~89 x_{17}				

Fig. 12 Gantt. Chart for F_w Model.[3]

through the analysis of variance statistical tests. In future scope, we will focus on further optimizations of the proposed algorithm, NHGCPM, and extensions for the scientific workflow applications with various QoS parameters; moreover, it will be used in different real-time applications. Furthermore, future work is probable by learning the performance of current demand after making them security-guided and by incorporating different constraints.

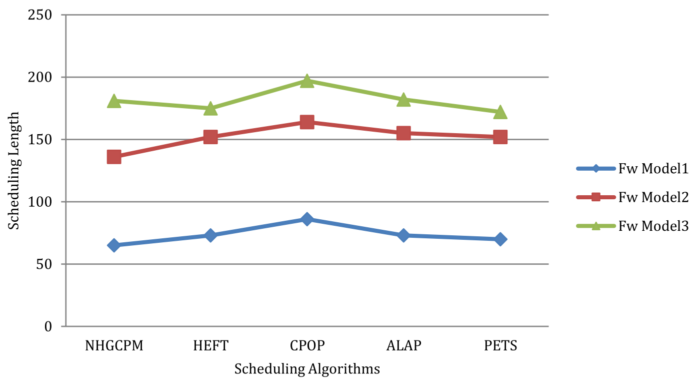


Fig. 13 Comparative Study of Scheduling Length

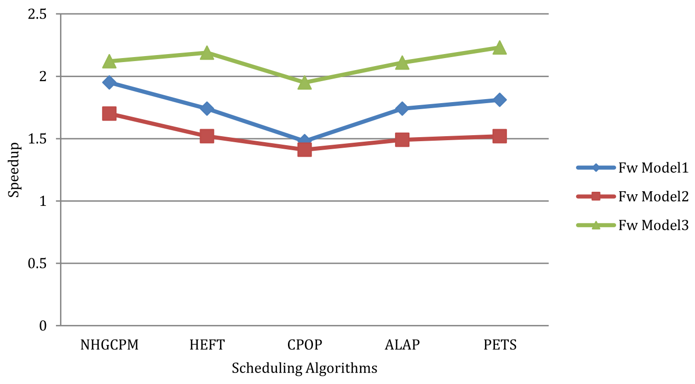


Fig. 14 Comparative Study of Speedup

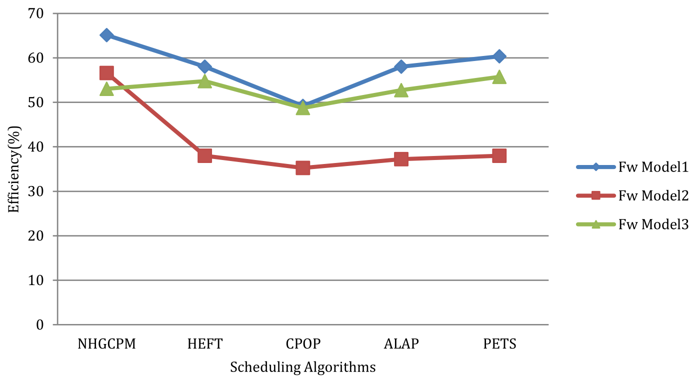


Fig. 15 Comparative Study of Efficiency

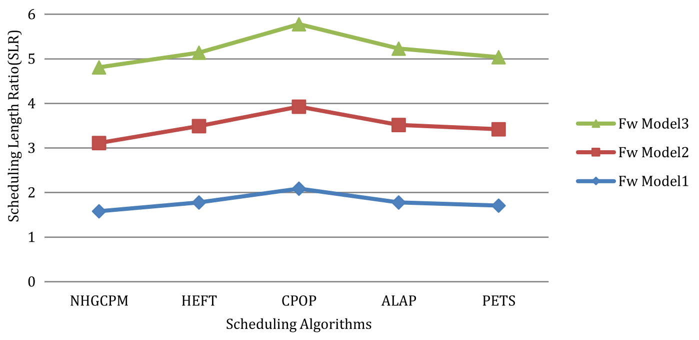


Fig. 16 Comparative Study of SLR

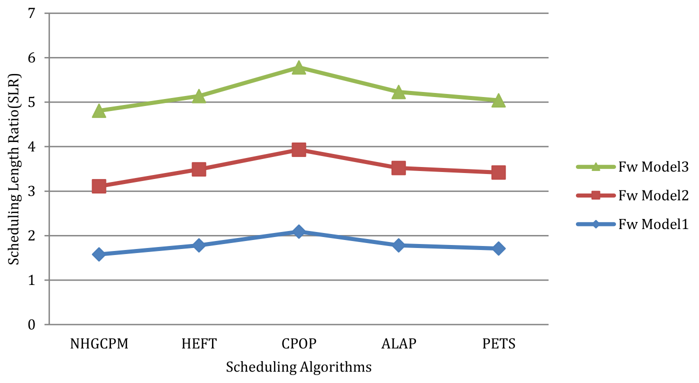


Fig. 17 Comparative Study of Resource Utilization

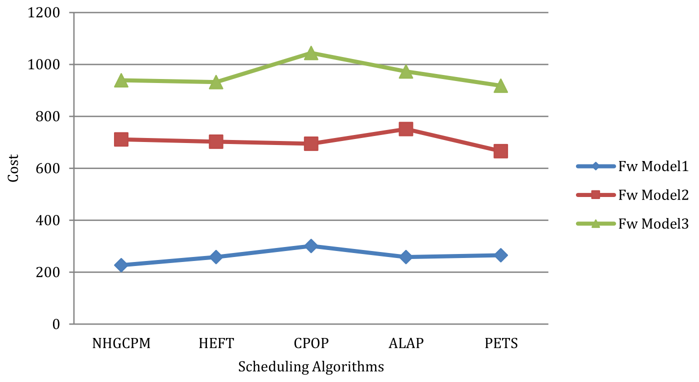


Fig. 18 Comparative Study of Cost

Funding None.

Availability of data and material Available on request.

Declarations

Conflict of interest Authors declares that they have mo conflict of interest.

References

1. Mutlag AA, Abd Ghani MK, Arunkumar N et al (2019) Enabling technologies for fog computing in healthcare IoT systems. Futur Gener Comput Syst 90:62–78. https://doi.org/10.1016/j.future.2018.07.049
2. Gai K, Guo J, Zhu L, Yu S (2020) Blockchain Meets Cloud Computing: A Survey. IEEE Commun Surv Tutorials 22:2009–2030. https://doi.org/10.1109/COMST.2020.2989392
3. Malla S, Christensen K (2020) HPC in the cloud: Performance comparison of function as a service (FaaS) vs infrastructure as a service (IaaS). Internet Technol Lett 3:e137. https://doi.org/10.1002/itl2.137
4. Scheuner J, Leitner P (2020) Function-as-a-Service performance evaluation: A multivocal literature review. J Syst Softw 170:110708. https://doi.org/10.1016/j.jss.2020.110708
5. Sharma S, Sajid M (2021) Integrated fog and cloud computing: issues and challenges. Int J Cloud Appl Comput (IGI) 11(4), Article 10
6. Buyya R, Pandey S, Vecchiola C (2009) Cloudbus toolkit for market-oriented cloud computing. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 24–44
7. Marozzo F (2018) Infrastructures for high-performance computing: Cloud infrastructures. Encycl Bioinforma Comput Biol ABC Bioinforma 1–3:240–246. https://doi.org/10.1016/B978-0-12-809633-8.20374-9
8. Hammed SS, Arunkumar B (2020) A cost effective- secure algorithm for work-flow scheduling in cloud computing. Internet Technol Lett e233. Doi: https://doi.org/10.1002/itl2.233
9. Zhou J, Wang T, Cong P et al (2019) Cost and makespan-aware workflow scheduling in hybrid clouds. J Syst Archit 100:101631. https://doi.org/10.1016/j.sysarc.2019.08.004
10. Sahitya A (2021) Importance of Fog Computing in. Integr Cloud Comput with Internet Things Found Anal Appl, p 211
11. Song A, Chen W-N, Luo X-N, et al (2020) Scheduling Workflows with Composite Tasks: A Nested Particle Swarm Optimization Approach. IEEE Trans Serv Comput
12. Jain R, Sharma N (2021) A QoS Aware Binary Salp Swarm Algorithm for Effective Task Scheduling in Cloud Computing. In: Progress in Advanced Computing and Intelligent Engineering. Springer, pp 462–473
13. Farid M, Latip R, Hussin M, Abdul Hamid NAW (2020) A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing. Symmetry (Basel) 12:551
14. da Silva EC, Gabriel PHR (2020) A Comprehensive Review of Evolutionary Algorithms for Multiprocessor DAG Scheduling. Computation 8:26
15. Hosseinzadeh M, Ghafour MY, Hama HK, et al (2020) Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J Grid Comput, pp 1–30
16. .Li J, Zhang X, Han L et al. (2021) OKCM: improving parallel task scheduling in high-performance computing systems using online learning. J Supercomput 77:5960–5983
17. Woeginger GJ (2003) Exact algorithms for NP-hard problems: A survey. In: Combinatorial optimization—eureka, you shrink! Springer, pp 185–207
18. Hanen C (1994) Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. Eur J Oper Res 72:82–101
19. Tong Z, Chen H, Deng X et al (2020) A scheduling scheme in the cloud computing environment using deep Q-learning. Inf Sci (Ny) 512:1170–1191

20. Du J, Leung JY-T (1989) Complexity of scheduling parallel task systems. SIAM J Discret Math 2:473–487
21. Pop F, Dobre C, Cristea V (2008) Performance analysis of grid DAG scheduling algorithms using MONARC simulation tool. In: 2008 International Symposium on Parallel and Distributed Computing, pp 131–138
22. Bozdag D, Ozguner F, Catalyurek UV (2008) Compaction of schedules and a two-stage approach for duplication-based DAG scheduling. IEEE Trans Parallel Distrib Syst 20:857–871
23. Kannan R, Karpinski M (2005) Approximation algorithms for NP-hard problems. Oberwolfach Reports 1:1461–1540
24. Hochba DS (1997) Approximation algorithms for NP-hard problems. ACM SIGACT News 28:40–52
25. Demirci G, Marincic I, Hoffmann H (2018) A divide and conquer algorithm for dag scheduling under power constraints. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp 466–477
26. Epstein L, Tassa T (2006) Optimal preemptive scheduling for general target functions. J Comput Syst Sci 72:132–162
27. Sulaiman M, Halim Z, Waqas M et al (2021) A hybrid list-based task scheduling scheme for heterogeneous computing. J Supercomput 77:10252–10288
28. Topcuoglu H, Hariri S, Wu M-Y (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans parallel Distrib Syst 13:260–274
29. Li J, Zhang X, Han L et al (2021) OKCM: improving parallel task scheduling in high-performance computing systems using online learning. J Supercomput 77:5960–5983
30. Ramezani R (2021) Dynamic scheduling of task graphs in multi-FPGA systems using the critical path. J Supercomput 77:597–618
31. Chowdhary SK, Rao ALN (2021) QoS Enhancement in Cloud-IoT Framework for Educational Institution with Task Allocation and Scheduling with Task-VM Matching Approach. Wireless PersCommun 121:267–286
32. Medara R, Singh RS (2022) A Review on Energy-Aware Scheduling Techniques for Workflows in IaaS Clouds. Wireless PersCommun. https://doi.org/10.1007/s11277-022-09621-1
33. Xu Y, Li K, Hu J, Li K (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci (Ny) 270:255–287
34. Xu X-J, Xiao C-B, Tian G-Z, Sun T (2016) Hybrid scheduling deadline-constrained multi-DAGs based on reverse HEFT. In: 2016 International Conference on Information System and Artificial Intelligence (ISAI), pp 196–202
35. Samimi P, Teimouri Y, Mukhtar M (2016) A combinatorial double auction resource allocation model in cloud computing. Inf Sci (Ny) 357:201–216
36. Rajak R, Shukla D, Alim A (2018) Modified critical path and top-level attributes (MCPTL)-based task scheduling algorithm in parallel computing. In: Soft Computing: Theories and Applications. Springer, pp 1–13
37. Rajak R (2018) Deterministic task scheduling method in multiprocessor environment. In: International Conference on Advances in Computing and Data Sciences, pp 331–341
38. Rajak N, Shukla D, (2019) Performance analysis of workflow scheduling algorithm in cloud computing environment using priority attribute. Int J Adv Sci Technol Australia 28(16):1810 – 1831
39. Braun TD, Siegel HJ, Beck N et al (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Comput 61:810–837
40. Pop F, Dobre C, Cristea V (2009) Genetic algorithm for DAG scheduling in grid environments. In: 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, pp 299–305
41. Canon L-C, Jeannot E (2009) Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. IEEE Trans Parallel Distrib Syst 21:532–546
42. Raza Abbas Haidri (2020) ChittaranjanPadmanabhKatti, Prem Chandra Saxena, Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. J King Saud Univ Comput Inf Sci 32(6):666–683
43. Darbha S, Aggarwal DP (1994) SDBS: A task duplication based optimal scheduling algorithm. In Proceedings of IEEE scalable high performance computing conference, Knoxville, TN, pp 756_61.
44. Sinnen O Task scheduling for parallel systems. Wiley-Interscience Publication (2007)

45. Kumar MS, Gupta I (2017) Jana PK Delay-based workflow scheduling for cost optimization in heterogeneous cloud system. In: 2017 Tenth International Conference on Contemporary Computing (IC3), Noida, pp. 1–6
46. Gupta I, Kumar MS, Jana PK (2018) Efficient workflow scheduling algorithm for cloud computing system: a dynamic priority-based approach. Arab J Sci Eng 43(12):7945–7960
47. Hwang K (2005) Advanced computer architecture: parallelism,scalability, programmability, 5th reprint. New Delhi:TMH Publishing Company, pp 51_104
48. Akbar MF, Munir EU, Rafique M M, Malik, Khan SU, Yang LT (2016)zs List-Based Task Scheduling for Cloud Computing. In: IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical And Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, pp 652–659
49. Kalra M, Singh S (2015) A review of metaheuristic scheduling techniques in cloud computing. Egypt informatics J 16:275–295
50. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms. MIT press

Authors and Affiliations

Ranjit Rajak[1] · Shrawan Kumar[2] · Shiv Prakash[3] · Nidhi Rajak[1] · Pratibha Dixit[4]

[1] Department of Computer Science and Applications, Dr. HarisinghGour Central University, Sagar, MP, India

[2] Department of Computer Science, Regional Campus Manipur, Indira Gandhi National Tribal University, Manipur, India

[3] Department of Electronics and Communication, University of Allahabad, Prayagraj, UP, India

[4] King Georges Medical University, Lucknow, UP, India