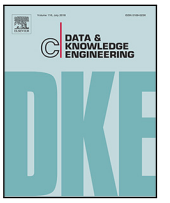
Data & Knowledge Engineering 145 (2023) 102138

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak



A novel approach for Credit-Based Resource Aware Load Balancing algorithm (CB-RALB-SA) for scheduling jobs in cloud computing

Abhikriti Narwal *, Sunita Dhingra

Department of Computer Science & Engineering, University Institute of Engineering & Technology, Maharshi Dayanand University, Rohtak, India

ARTICLE INFO

Keywords:
Cloud computing
Virtual machines
Task scheduling
Load balancing
Credit based scheduling algorithm
Enhanced multiobjective scheduling algorithm
Resource aware load balancing

ABSTRACT

In recent years, cloud computing has gained popularity, mainly because of its utility and relevance to current technological trends. It is an arrangement that is highly customizable and encapsulated for providing better computational services to its clients worldwide. In the cloud, computing scheduling plays a pivotal role in optimizing resources. A better scheduling algorithm should be efficient and impartial, reducing the makespan time with proper resource utilization. However, most scheduling algorithms customarily lead to less resource utilization, termed load imbalance. The analysis of the existing papers exhibits better Makespan time but cannot guarantee the load-balanced mapping of jobs with proper resource utilization. Therefore, to eliminate the shortcomings of the prevalent/existing algorithms and enhance the performance, CB-RALB-SA, Credit-based Resource Aware Load Balancing scheduling algorithm has been rendered. The proposed work ensures a balanced distribution of tasks based on the capabilities of the resources, which eventually proves sustainable improvement against the existing scheduling algorithms. Therefore, a novel Credit Based Resource Aware Load Balancing Scheduling algorithm (CB-RALB-SA) is proposed. The tasks weighted by the credit-based scheduling algorithm are then mapped to the resources considering each resource's load and computing capability using FILL and SPILL functions of Resource Aware and Load using Honey bee optimization heuristic algorithm. With the experimental evaluations and results, it has been proved that the proposed approach provides 48.5% better in Processing Time and 16.90 % better results in makespan time than the Existing CBSA-LB algorithm. Thus, it improves the processor's efficiency while uplifting the whole system's performance and has saved memory allocated to tasks and RAM.

1. Introduction

As the internet has become ubiquitous in this contemporary era and data is increasing daily in terms of volume, the need for cloud computing has proliferated quickly in academia, society, and industry [1]. Cloud computing has spread contagiously around the four directions for research and business institutions in the past few years. Cloud catches our attention whenever we open any website, television channel, or information technology magazine. It is the fastest means to deliver services to its users on-demand over the internet, and thus, it can be explained as internet-centric software. It has become an intriguing, indispensable, and practical means of modifying the whole of computing [2]. Cloud computing models are increasing day by day. A key feature of cloud computing is elasticity, which allows the provisioning and de-provisioning of computing resources on-demand via auto-scaling. Auto-scaling techniques are diverse and involve various components at the infrastructure, platform, and software levels. Auto-scaling also overlaps with other quality attributes, thereby contributing to service level agreements, and often applies modeling and control techniques

* Corresponding author.
E-mail address: abhikritiin@gmail.com (A. Narwal).

https://doi.org/10.1016/j.datak.2022.102138
Received 29 January 2021; Received in revised form 6 January 2022; Accepted 21 December 2022
Available online 26 December 2022

to make the auto-scaling process adaptive. A study of auto-scaling architectures, existing techniques, and open issues provides a comprehensive understanding of future research solutions. In the cloud/customer design, the customer needs a rich application running on an Internet-associated gadget [3].

Schedulers are required for cloud computing to decide which job of a workflow is suitable for which processing resource. The scheduling theory for cloud computing is becoming more popular with users' demand for cloud computing. Service providers of cloud computing ensure that the income is utilized effectively and to the best capacity to satisfy that resource power is not under-used [2].

With the increasing number of users accessing the resources of cloud environments, job scheduling is clear and crucial to lessen the complexity of the data center's load [4]. Therefore, task scheduling is an essential issue in the cloud environment. To utilize resources effectively and efficiently, task scheduling is required by distributing particular tasks to specific resources at a certain time. The prominent role of the task scheduling algorithm is to optimize the working and quality of service, preserve the efficiency between the various tasks, and finally mitigate the expenses [5]. Undoubtedly, task scheduling of cloud computing is challenging [6]. Improper distribution of workload on virtual machines increases the time of execution and increases the consumption of energy. Most scheduling procedures are not effective enough to connect the whole measurements of the available cloud possessions. A survey demonstrates that around 1.6 million tons of extra carbon dioxide are emitted only because of idle computing resources inside the cloud data centers. Moreover, the expense rose to $19 billion in the energy cost and disbursed possessions. To resolve this problem of load unevenness, umpteen scholars have proposed various apparatuses for scheduling to map jobs in a load balancing way to attain optimized system throughput and resource utilization [7].

An effective algorithm for scheduling should consider the total time for execution of the available resources and load balancing of the whole system. It is better not to under-utilize the resources having high capacity and the jobs having low execution time. Therefore, the scheduler is obliged to assign the jobs to resources based on the length of the job and resource capacity [8].

There are some challenges associated with job scheduling in cloud computing [9]:

- **Fluctuating and Dynamic workloads** - the main challenge to elasticity in cloud computing is the unpredictable nature of workloads. The fluctuation of workload may occur in an unplanned or planned manner. However, for fluctuations that are planned, it is possible to predict the situation in advance so that allocation of the resource can be done smoothly and timely.
- **To ensure efficient resource use** - whenever needed, the resources should be allocated instantly, although demand should be unplanned. This is known as Auto-scaling in cloud computing. The incoming workload must be allocated to resources known as virtual machines so that the cloud service provider has to make sure that resource utilization is efficient. This needs techniques for optimal scheduling to allocate the tasks to current machines.
- Various physical nodes are scheduled across the available nodes, widely distributed at various locations, and differ in architecture, computational power, memory, and network performance. Thus, at different nodes, different tasks execute differently.

This research work proposes job scheduling in cloud computing and load balancing and resource awareness to alleviate the load disproportion problem in Cloud computing. This is a new concept of batch-dynamic scheduling procedures; scheduling of cloud trades would be done in a load balancing way.In addition, cloud-hosted trading systems allow traders to have real-time access to data anytime, from anywhere, across multiple devices via the internet. This comes handy with the trendy Bring Your Own Device (BYOD) working system.

This work will schedule a batch of nonpre-emptive, compute-intensive, and self-regulating trades. This work aims to minimize makespan time or execution and maximize resource utilization and throughput. The execution would be performed in two phases; firstly, the workload is scheduled based on the capability of computing virtual machines and necessities of computing of the cloud trades. After this, the remaining jobs would be scheduled to virtual machines in the second phase, producing the earliest finish time.

Improvement in terms of resource utilization and makespan for the skewed workload; i.e., high quantity of shorter cloudlets; can be achieved by combining job scheduling, load balancing, and resource awareness, whereas intermediate improvement in terms of makespan and relatively higher improvement in resource utilization can be achieved for the non-skewed workload (high quantity of larger cloudlets). Throughput would also increase by using this combined approach to job scheduling in cloud computing.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 explains the existing work. Section 4 describes the proposed work. Section 5 provides the experimental results and discussion details, and Section 6 concludes the paper.

2. Background

Recently, researchers have paid much attention to umpteen smart techniques for job scheduling in cloud computing [8]. Many researchers in the literature tried to solve cloud job scheduling using different techniques. Here, in this section, we discuss some work of various authors in job scheduling in cloud computing.

Non-dominating sorting algorithm has been used in various works by many researchers in past years; such as Narwal et al. proposed a multiobjective algorithm of task scheduling which considered a wide variety of attributes in the cloud and utilized non-dominate sorting to prioritize the tasks. This work aimed to enhance the performance of existing algorithms [10]. Furthermore, the multiobjective task scheduling technique was proposed by Lakra et al. to map tasks to a virtual machine so that the data center's throughput t would improve, thus reducing the expenses with no violation of service level agreement. This turns out to be an optimal algorithm for scheduling [11].

Further, a naïve technique for job scheduling and genetic algorithm and game theory concepts. This work not only focused on maximizing utilization of resources but also provided a better solution with the help of a non-domination sorting genetic

algorithm [12]. A lot of work has been done on credit-based scheduling algorithms in previous years. One such enhanced algorithm for scheduling was introduced after analysis of the traditional algorithms by Thomas et al. which were based on the length of the task and priority of the user. This task with higher priority did not get more importance on their arrival than the current tasks [13].

One more parameter known as deadline constraints was considered by Raja et al. along with the parameters in the previous work, i.e., task length and user priority. Finally, results demonstrated that the makespan of tasks was reduced [13]. Narwal et al. proposed a credit-based scheduling algorithm and load balancing to balance the load and schedule the tasks independently [14]. Srichandan et al. explored the algorithm for task scheduling by utilizing a hybrid technique; combined beneficial features of two mainly utilized heuristic algorithm, which was inspired biologically, bacterial foraging and the genetic algorithm in cloud computing. The prominent contribution of this work was double: firstly, it reduced the makespan and secondly, the energy consumed; thus provided both ecological and economic benefits [15].

In another research work, Maqableh et al. implemented an algorithm using an artificial neural network to enhance the results of job scheduling by finding a novel set of classification, rather than searching only within the current set [16]. A naïve bee swarm algorithm for optimization is known as BLA. i.e., the Bees Life Algorithm was presented by Bitam et al. so that computational jobs can be effectively scheduled among the processing resources. This algorithm was inspired by the life of bees, their nature, which they represent through their behavior while searching for food sources and reproduction [17]. Kimpan et al. proposed an artificial bee colony algorithm utilizing heuristic task scheduling; basically, this was the naïve algorithm for load balancing and task scheduling. Using this technique, even for large number of task and varied data types; makespan time was reduced, giving an effective outcome [18].

Many authors used resource-based scheduling algorithms to obtain adequate and efficient results. For example, a novel algorithm known as Resource-Aware Load Balancing Algorithm (RALBA) is presented by Hussain et al. to balance the workload distribution among virtual machines based on their computation capability. This framework consists of two modes: schedule based on the capability of virtual machines and virtual machine having short finish time was chosen for jobs mapping [7]. Another work by Loganathan et al. provided the effective structure of data for resource scheduling and management method in a cloud environment, which is private and also discussed a cloud model. In addition, this work considered the availability of resources and kinds of jobs while deciding on scheduling [19].

3. Existing work

The existing algorithms first include the MOSA algorithm, i.e. Multiobjective Scheduling algorithm, which uses sort to prioritize the tasks [10]. The comparison algorithms used in this work are First come first serve (FCFS) and Shortest Job first (SJF). The MOSA algorithm is further improved to minimize tasks' processing time, select the task from the task list, and map that task with the best optimized virtual machine based on million instructions per second (MIPS) [20]. The problem analyzed in the MOSA is that the sort is based only on task length. If some tasks have the same length, it would be not easy to perform the sort between them as the tasks with same length have the same priority. Therefore, Enhanced multi-objective task scheduling algorithm(EMOSA) is proposed, in which three parameters are taken to sort the tasks: task length, task file size, and task output size. Also, virtual machines are optimized in this algorithm based on MIPS and granularity factor i.e. granularity size. The EMOSA algorithm is compared with Credit based scheduling algorithm (CBSA) [21], and experimental results show that the CBSA algorithm performs better than the EMOSA algorithm as examined from the results that EMOSA provides the same priority to many tasks as the prioritized concept is based on sort, but the CBSA algorithm sort the task by providing credits to each task.

The process of providing credits uses task length difference (TLD), priority, deadline, and cost to assign the credits to tasks. Prioritizing tasks using this process would not provide the same priority to a task in any case. Further Literature survey study inspects that task scheduling contains load balancing problem as in scheduling algorithm, the number of tasks are mapped to the virtual machine without taking into consideration the capabilities of a virtual machine, i.e., how much load of tasks the machine can take. Therefore, the concept of load balancing is introduced, and various load balancing algorithms are surveyed [22] and examined that the Honey Bee Load Balancing is applied for searching the optimal path towards the best suitable resources in the cloud network. Load Balancing is a key issue in the cloud system that must dispense the workload in an efficient and scalable manner. It also assures that each computational resource is dispensed evenly and fairly. All the existing algorithms that have been studied mainly considers the reduction of overhead, reducing the migration time and enhancing the performance etc, whereas response time is an important challenge for every engineer to produce the application that can maximize the overall throughput in the cloud scenario. Many techniques lack prominent scheduling and load balancing, leading to increased processing costs. However, the proposed algorithm thrives on balancing the workload of the cloud infrastructure while reducing the response time for the given number of tasks.

Honey bee load-balancing algorithm is best suited for the load balancing in credit-based scheduling algorithm. The Credit-based Honey bee load balancing scheduling algorithm (CBSA-LB) [23] is implemented, and experimental results show that CBSA-LB is better than MOSA, EMOSA, and CBSA. Again, in the concept of load balancing, proper resource utilization is a must, which is missing in the case of CBSA-LB as with the load balancing, resource awareness needs to be considered to optimize the results. Therefore, the proposed algorithm uses the concept of resource-aware with load balancing in a credit-based scheduling algorithm.

4. Proposed algorithm

From the existing work, synthesis has derived that most of the scheduling algorithm customarily leads to less resource utilization which is eventually termed load imbalance. Load Balancing in the cloud reduces the cost associated with document management

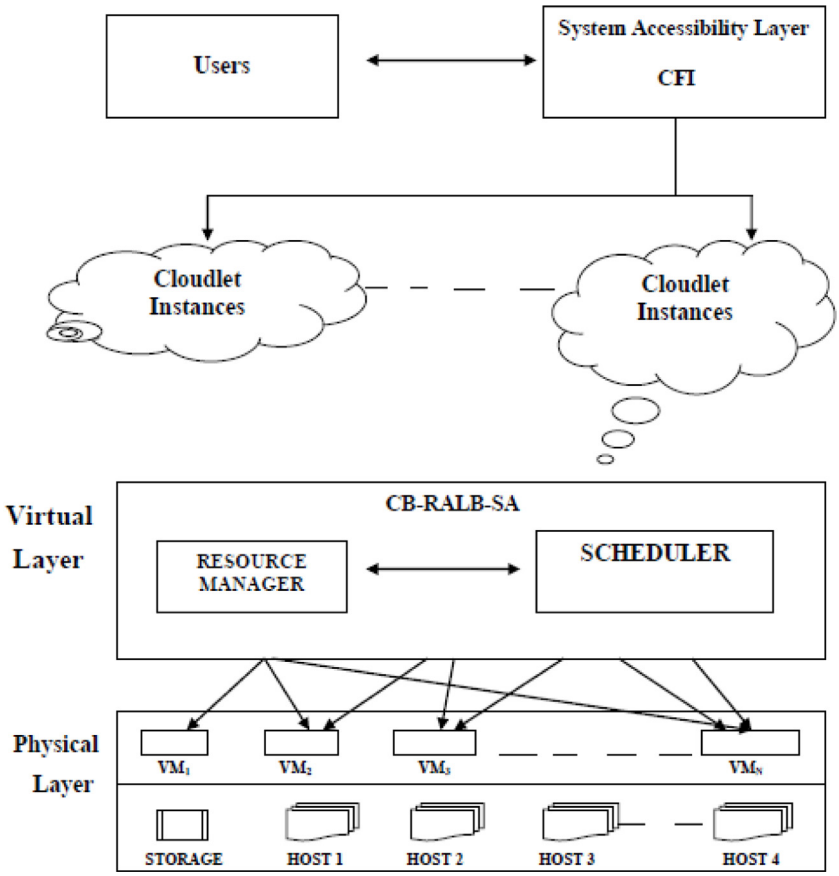


Fig. 1. CB-RALB-SA Cloud Architecture.

systems and maximizes the availability of resources. The load balancer system allows creating an infrastructure able to distribute the workload balancing it between two or more cloud servers. Load balancing must take into account two major tasks, one is resource provisioning or resource allocation, and the other is task scheduling in the cloud environment. Resource provisioning is possible in the cloud environment by using various functions. Proper load balancing in the cloud environment maximizes throughput, minimizes response time, and avoids overload of any single resource. Instead of using a single component, load balancing takes advantage of multiple resources; by increasing architecture availability and reliability. The analysis of the existing papers exhibits better Makespan time but cannot assure the mapping of load-balanced jobs with proper resource utilization.

The proposed work ensures a distribution of balanced tasks based on the capabilities of the resources, which eventually proves sustainable improvement against the existing scheduling algorithms. Furthermore, a new Credit Based Resource Aware Load Balancing Scheduling algorithm (CB-RALB-SA) is proposed in this research. In the proposed CB-RALB-SA algorithm, the tasks weighted by the credit-based scheduling algorithm are then mapped to the resources considering each resource's load and computing capability using FILL and SPILL functions of Resource Aware and Load using Honey bee optimization heuristic algorithm.

Fig. 1 represents the CB-RALB-SA based cloud architecture. Cloudlets are used as the task in the cloudsim simulator. It represents the user's requests. The physical and virtual layer comprises a combination of hosts with storage servers and virtual machines. The physical layer is considered a cloud data center which acts as computing power for the server. Resource manager blocks above the virtual layer each virtual machine's records and status of its availability in the layer, which is also responsible for the formation, migration, and shutting down of the virtual machine. The proposed CB-RALB-SA is placed on the top of the virtual layer, takes the information of available VMs with their computing capabilities and current load from the resource manager, and ensures the balanced distribution of cloudlets among virtual machines improves resource utilization.

According to the factors, the proposed algorithm CB-RALB-SA uses the credit-based approach to provide the weightage to the tasks [23]. Resource aware technique by using spill and fill function ensures balancing of load by calculating system load and computational capabilities on each machine. It works in 2 phases, which comprises scheduling based on VMs, computing capability, and (2) the Earliest less loaded VM selection for the job, i.e., migrating superfluous jobs based on the demand and supply values from an overloaded VM to the underloaded VM. In the proposed load balancing algorithm, resource awareness is used to manage the resources with load balancing using the optimization honey bee technique.

RALBA comprises two sub-schedulers, i.e., Fill and Spill schedulers. Fill Scheduler performs Cloudlet to VM allocation via considering the computing share of VMs. First, fill scheduler selects VMj with Largest VM Share and determines max PCloudletVMj for VMj. Then, the VM sharing on candidate cloudlet is performed, and after allocation of the Cloudlet, VMSharej of VMj is modified. Finally, the progression of mapping cloudlets is repeated by Fill Scheduler until VMj with non-empty RPCloudletj does not exist, or the CLS turn out to be empty.

Mapping VMs allocation to Cloudlets is performed based on the Earliest First Time (EFT) of aspirant Cloudlet by Spill scheduler. Afterwards Fill scheduler, RALB scheme allot the left-over Cloudlets of CLS using Spill scheduler. The maxCloudlet is carefully chosen and yields EFT for the maxCloudlet when assigned to the particular VM. On candidate Task–VM sharing, the makespan time of the VM is restructured. Repetition of Task to VM allocation is done until CLS turn out to be empty.

Further based on this vm allocation list results are optimized using honey bee foraging behavior and scheduled the tasks.

The general outline of CB-RALB-SA algorithm is :

Input: VM schedule and Cloudlets schedule from CB algorithm

STEP 1: Input Cloudletlist and set of sorted VMs list with its computing ratio.

STEP 2: CloudletVMMap = Null

STEP 3: Call FillScheduler (VMschedule,Cloudletlist)

STEP 4: vmlist = get vmlist(VMschedule)

STEP 5: if cloudletList.Size() > 1**then**

STEP 6:Cloudlet VMMap = **CALL** SpillScheduler(vmlist,cloudletlist,CloudletVMMap)

STEP 7: return CloudletVMMap

STEP 8: CALL Procedure 1

STEP 9: endif

FillScheduler

Input: VM schedule and Cloudlets schedule from Credit Based algorithm

STEP 1: Iniliase newVmShare, CloudletVMMap to null;

STEP 2: for all cloudlet in cloudletlist **do**
$totallength = totallength + cloudlet.getCloudletLength()$**;**

STEP 3: end for

STEP 4: **for** all v in vmlist **do**
$vShareMap = totallength * VMMap$**;**

STEP 5: end for

STEP 5: while VMShare > Smallest Cloudlet **do**

STEP 6: Find the maxPcloudletVM

STEP 7: Assign maxPcloudletVM and remove it from cloudletlist

STEP 8: Update VMshare of VM

STEP 9: end while

STEP 10: **return** cloudletVMMap

SpillScheduler Function

Input: VM schedule and Cloudlets schedule from Credit Based algorithm

STEP 1: **while** cloudlist.size ≥ 1 **do**

STEP 2: Get the cloudlet with maximum size

STEP 3: Get the Vm with Earliest finish time for Cloudlet

STEP 4 : Assign cloudlet to VM and remove it from cloudletlist

STEP 5: Update ready time of VM

STEP 6: end While

STEP 7: return cloudletVMMap

Table 1
Cloudsim specification for experimental analysis.

CloudSim objects	Input specifications	Value
Tasks	Len_Cloudlet	1000–10000
	Number._of_Cloudlets	50–2000
Virtual Machine	Count_of_Vm	5–100
	MIPS_Vm	500–5000
	VM_Memory	256–2048
	VM_BW	500–2000
	Pes_Count	1–4
Server (Datacenter)	Number_of._Datacenters	1–15
	Hosts_Count	2–8

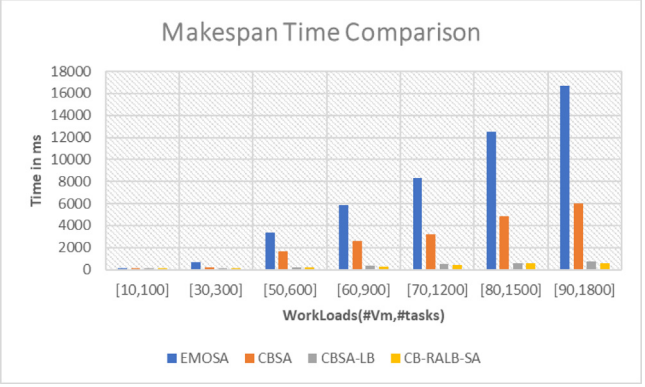


Fig. 2. Comparison of Makespan time by varying workloads.

5. Experimental evaluation and discussion

The simulation of propounded work has been done in cloudsim simulator. The consequences have been analyzed on various workloads and performance parameters. The proposed work is evaluated using Makespan time, total processing time, response time, total processing cost and execution time. Table 1 explains the cloudsim specifications for the experimental analysis in which length of cloudlet value varies from 1000 to 10,000, number of cloudlets varies from 50–2000. The VM count varies from 5 to 100, processing element counts varies from 1 to 4.

5.1. Experimental analysis of results

The propounded CB-RALB-SA evaluation with the existing CBSA-LB [23] and other algorithms including EMOSA [20] and CBSA [21] algorithm is analyzed on various input parameters, the performance is compared based on five metrics comprises of Total Processing Time, Processing Cost, Makespan Time, Execution Time and Response Time. Furthermore, the performance evaluation is done by experimenting seven workloads of cloudlets and comparing the proposed CB-RALB-SA Resource Aware Honey Bee Load Balancing based credit scheduling algorithm with the other existing algorithms.

5.1.1. Analysis on the basis makespan time

This segment includes the analysis based on changing the numeral of cloudlets for each technique. Mentioned below table provides the demonstration of comparison of each procedure on the basis of makespan time.

Makespan is the completion time of all the jobs in the sequence i.e the finish time of the last job in the sequence of execution. It can be denoted as

$$MS = \max\{CT_{kl} | k \in T.k = 1, 2 \ldots n \, and \, l \in VM.l = 1, 2 \ldots m\} \tag{1}$$

At this point, the examination between the various algorithms is taken into consideration using seven workloads, the make span time of the profounded CB-RALB-SA procedure is 610.13 ms for 1800 tasks which is 734.28 ms in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and CBSA [21]. The Makespan time for EMOSA algorithm is 16,690.55 ms and CBSA algorithm is 5980.98 ms. The Fig. 2 illustrates that the proposed CB-RALB-SA algorithm offers a improved makespan time as compared to other heuristics (see Table 2).

After this, the examination between the various algorithms is taken into consideration by keeping the virtual machines stable to 80. Using four different workloads, the make span time of the profounded CB-RALB-SA procedure is 949.94 ms for 2000 tasks which is 1038.42 ms in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and

Table 2
Makespan time estimation of different loads for various algorithms.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[10,100]	146.91	143.44	108.47	106.31
[30,300]	698.34	224.84	138.71	113.81
[50,600]	3372.57	1684.49	193.35	186.86
[60,900]	5817.06	2578.59	316.55	297.88
[70,1200]	8360.06	3252.17	500.98	394.64
[80,1500]	12476.09	4872.29	600.15	560.96
[90,1800]	16690.55	5980.98	734.28	610.13

Table 3
Makespan time estimation of different loads for various algorithms for fixed 80 virtual machines.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[2000]	2650.07	2385.78	1038.42	949.94
[2400]	3780.07	3482.58	1680.65	1088.75
[2800]	5110.07	4729.83	2487.86	1524.24
[3200]	6640.06	6122.23	3043.1	2061.04

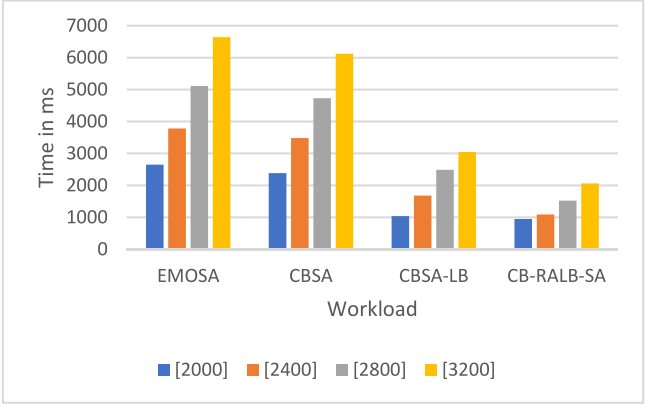


Fig. 3. Comparison of Makespan time by fix virtual machines..

CBSA [21]. The Makespan time for EMOSA algorithm is 2650.07 ms and CBSA algorithm is 2385.78 ms. The Fig. 3 illustrates that the proposed CB-RALB-SA algorithm offers a improved makespan time as compared to other heuristics in the fix virtal machines scenario.This improved value of CB-RALB-SA is achieved due to resource-aware mapping mechanism, and load-balance factor for scheduling . Analysis reveals that CB-RALB-SA is a propounded algorithm (see Table 3).

5.1.2. Analysis on the basis processing time

This section includes the analysis based on changing the numeral of cloudlets for each technique. In addition, mentioned below table provides the demonstration of comparison of each procedure based on processing time.

***Processing Time*:** To execute the task, the time taken by cloudsim clock. It is designed by using resulting formulation:

$$Processing_{time} = cloudlet_{length}/(vm_{MIPS} * no_{of\,PES}) \tag{2}$$

At this point, the examination between the various algorithms is taken into consideration using seven workloads, the processing time of the profounded CB-RALB-SA procedure is 11,280.1 ms in case of 1800 tasks which is 21,903.41 ms in CBSA-LB algorithm [23]. Likewise, the evaluation has been finished with other procedures including EMOSA and CBSA.

The Processing time for EMOSA [20] and CBSA [21] algorithm following the same input parameters are 237,596.81 and 43,689.12 ms. The Fig. 3 illustrates that the proposed CB-RALB-SA algorithm offers a less processing time as compared to other heuristics (see Fig. 4).

After this, the examination between the various algorithms is taken into consideration by keeping the virtual machines stable to 80. Using four different workloads, the total processing time of the profounded CB-RALB-SA procedure is 13,113.11 ms for 2000 tasks which is 18,492.48 ms in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and CBSA [21]. The Total Processing time for EMOSA algorithm is 121,730.78 ms and CBSA algorithm is 26,155.38 ms. The Fig. 5 illustrates that the proposed CB-RALB-SA algorithm offers a improved total processing time as compared to other heuristics in the fix virtal machines scenario.This improved value of CB-RALB-SA is achieved due to resource-aware mapping mechanism, and load-balance factor for scheduling . Analysis reveals that CB-RALB-SA is a propounded algorithm.

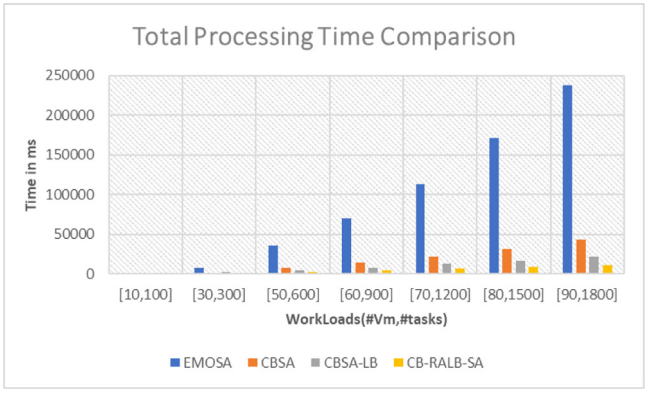


Fig. 4. Comparison of Total Processing time by varying workloads.

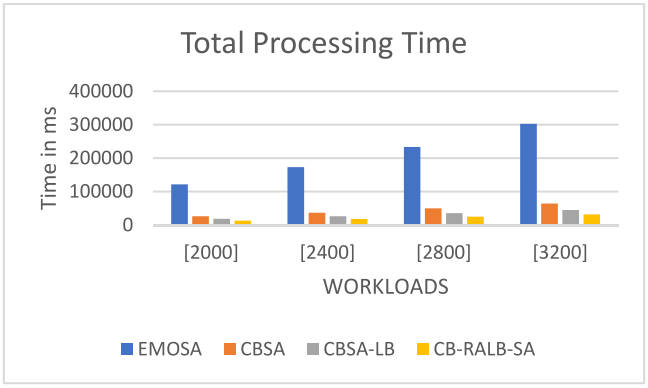


Fig. 5. Comparison of Total Processing time by fix virtual machines..

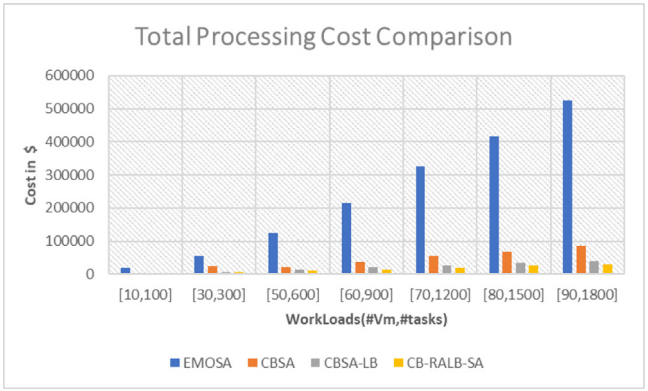


Fig. 6. Comparison of Total Processing Cost by varying workloads.

5.1.3. Analysis on the basis processing cost

This section includes the analysis of Total Processing Cost of each algorithm by varying the number of cloudlets. Table 4 below provides the demonstration of Total Processing Cost comparison of algorithms.

***Processing Cost*:** To accomplish specified set of jobs by procedure, the cost required is Processing Cost.

$$Processing_{cost} = DataCenter_{costpermemory} * VM_{RAM} \tag{3}$$

At this point, the examination between the various algorithms is taken into consideration using seven workloads, the processing cost of the profounded CB-RALB-SA procedure is 30,025.65 in case of 1800 tasks which is 40,619.3 in CBSA-LB algorithm [23]. Likewise, the evaluation has been finished with other procedures. The Fig. 6 illustrates that the proposed CB-RALB-SA algorithm offers a less total processing cost as compared to other heuristics.

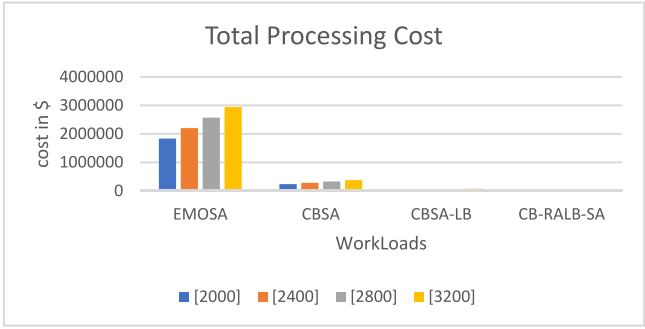


Fig. 7. Comparison of Total Processing Cost by fix virtual machines..

Table 4
Total processing time estimation of different loads for various algorithms.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[10,100]	1826.13	434.144	489.73	408.697
[30,300]	7619.82	790.77	2228.45	1132.34
[50,600]	35885.05	7469.92	4539.82	2486.8
[60,900]	70356.08	13912.11	8046.94	4380.69
[70,1200]	113463.12	21516.76	12859.82	6573.28
[80,1500]	171265.65	32042.97	16749.69	8668.03
[90,1800]	237596.81	43689.12	21903.41	11280.1

Table 5
Total processing time estimation of different loads for various algorithms for fixed 80 virtual machines.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[2000]	121730.78	26155.38	18492.48	13113.11
[2400]	172986.84	36876.97	26175.04	18187.14
[2800]	233212.86	49752.03	35498.58	24761.69
[3200]	302408.84	64360.6	44838.6	31840.84

Table 6
Total processing cost estimation of different loads for various algorithms.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[10,100]	18664.15	3685.88	3223.21	1382.39
[30,300]	55992.44	24786.58	5630.45	5082.22
[50,600]	124529.1	22728.58	13117.71	10086.26
[60,900]	215018.1	37214.98	20301.85	14531.85
[70,1200]	326706.79	54008.12	26606.377	20272.81
[80,1500]	417321.23	68585.12	33866.04	25987.09
[90,1800]	524117.69	84929.92	40619.3	30025.65

After this, the examination between the various algorithms is taken into consideration by keeping the virtual machines stable to 80. Using four different workloads, the total processing time of the profounded CB-RALB-SA procedure is 33,526.41 $ ms for 2000 tasks which is 46,394.61 $ in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and CBSA [21]. The Total Processing cost for EMOSA algorithm is 1,836,773 $ ms and CBSA algorithm is 234,237 $. The Fig. 7 illustrates that the proposed CB-RALB-SA algorithm offers a improved total processing cost as compared to other heuristics in the fix virtal machines scenario.This improved value of CB-RALB-SA is achieved due to resource-aware mapping mechanism, and load-balance factor for scheduling . Analysis reveals that CB-RALB-SA is a propounded algorithm.

5.1.4. Analysis on the basis execution time

This section includes the analysis based on changing the numeral of cloudlets for each technique. In addition, mentioned below table provides the demonstration of comparison of each procedure based on execution time (see Tables 5–11).

Execution Time is the time taken to execute a task. It is the extent of variance between finish time and the execution start time of the task.

$$ExecutionTime = FinishTime\,of\,Task - ExecStartTime\,of\,Task \tag{4}$$

Table 7
Total processing cost of different loads for fix virtual machines.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[2000]	1836773	234237	46394.61	33526.41
[2400]	2204128	281084.4	55409.18	39538.11
[2800]	2571482	327931.8	63891.78	44720.05
[3200]	2938837	374779.2	76934.87	52238.72

Table 8
Total execution time estimation of different loads for various algorithms.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[10,100]	146.81	143.34	108.37	106.21
[30,300]	698.24	224.74	138.61	113.71
[50,600]	3372.47	1684.39	193.25	186.76
[60,900]	5816.96	2578.49	316.45	297.78
[70,1200]	8358.86	3250.97	499.78	393.44
[80,1500]	12475.99	4872.19	600.05	560.86
[90,1800]	16690.45	5980.88	734.18	610.03

Table 9
Total execution time of different loads for fix virtual machines.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[2000]	60865.39	52310.77	33414.84	24898.01
[2400]	86493.42	73753.95	47721.62	34762.07
[2800]	116606.4	99504.06	64275.12	47570.4
[3200]	151204.4	128721.2	81157.13	61336.39

Table 10
Total response time estimation of different loads for various algorithms.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[10,100]	146.71	143.24	108.27	106.11
[30,300]	698.14	224.64	138.51	113.61
[50,600]	3372.37	1684.29	193.15	186.66
[60,900]	5816.86	2578.39	316.35	297.68
[70,1200]	8357.66	3249.77	498.58	392.24
[80,1500]	12475.89	4872.09	599.95	560.76
[90,1800]	16690.35	5980.77	734.08	609.93

Table 11
Response time of different loads for fix virtual machines.

Workloads	EMOSA	CBSA	CBSA-LB	CB-RALB-SA
[2000]	527.59	462.84	304.8	225.53
[2400]	745.16	648.11	432.88	310.27
[2800]	1000.1	870	579.48	422.21
[3200]	1292.42	1121.34	728.19	539.24

At this point, the examination between the various algorithms is taken into consideration using seven workloads, the execution time of the profounded CB-RALB-SA procedure is 610.03 ms in case of 1800 tasks which is 734.18 ms in CBSA-LB [23] algorithm. Likewise, the evaluation has been finished with other procedures. The Fig. 8 illustrates that the proposed CB-RALB-SA algorithm offers less execution time as compared to other heuristics.

After this, the examination between the various algorithms is taken into consideration by keeping the virtual machines stable to 80. Using four different workloads, the total execution time of the profounded CB-RALB-SA procedure is 60,865.39 ms for 2000 tasks which is 52,310.77 ms in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and CBSA [21]. The Total Execution time for EMOSA algorithm is 60,865.39 ms and CBSA algorithm is 52,310.77 ms. The Fig. 9 illustrates that the proposed CB-RALB-SA algorithm offers a improved total processing time as compared to other heuristics in the fix virtal machines scenario.This improved value of CB-RALB-SA is achieved due to resource-aware mapping mechanism, and load-balance factor for scheduling . Analysis reveals that CB-RALB-SA is a propounded algorithm.

5.1.5. Analysis on the basis of response time

This section includes the analysis based on changing the numeral of cloudlets for each technique. In addition, mentioned below table provides the demonstration of comparison of each procedure based on response time.

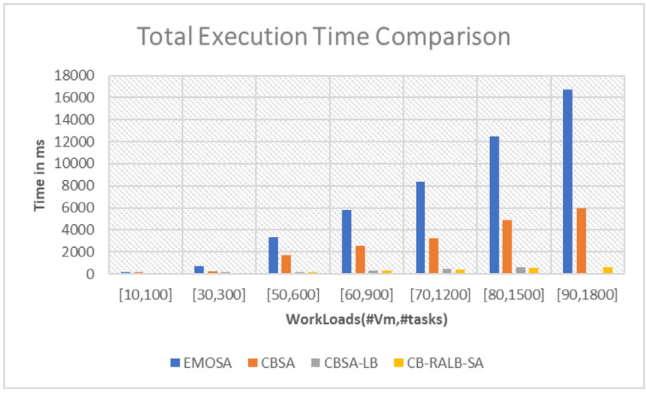


Fig. 8. Comparison of Execution time by varying workloads.

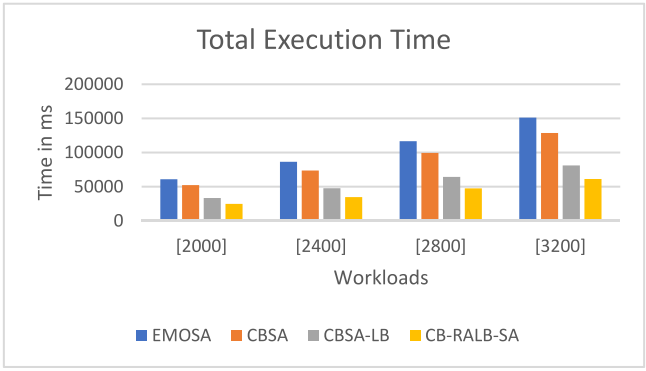


Fig. 9. Comparison of Total Execution Time by fix virtual machines..

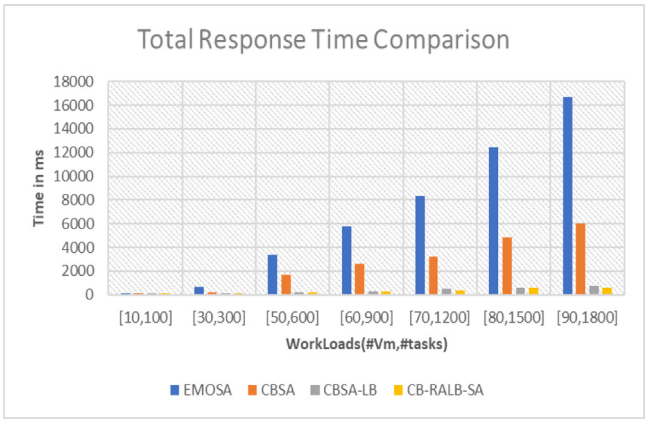


Fig. 10. Comparison of Response time by varying workloads.

Response Time : The time a framework requires for responding to begin the specific task. It is the proportion of contrast among the beginning time of execution of the undertaking and the actual cpu time of job when job arrived.

$$ResponseTime = ActualCPUTime\,of\,Task - ExecStartTime\,of\,Task \tag{5}$$

The examination has been done using seven different workloads, the response time of the propounded CB-RALB-SA algorithm is 609.93 ms in case of 1800 tasks which is 734.08 ms in CBSA-LB [23] algorithm. Likewise, the evaluation has been finished with other procedures. The Fig. 6 illustrates that the proposed CB-RALB-SA algorithm offers less response time as compared to other heuristics (see Fig. 10).

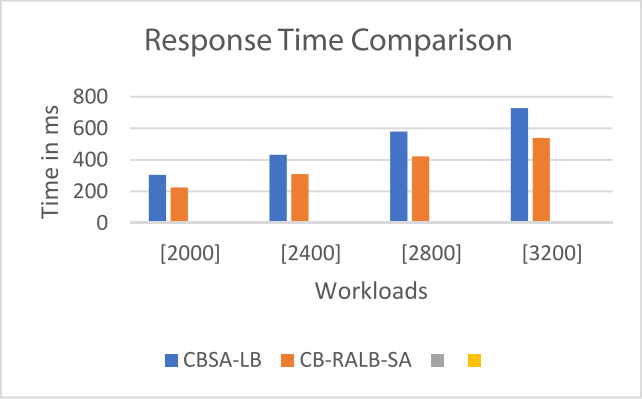


Fig. 11. Comparison Of Response Time By Fix Virtual Machines.

After this, the examination between the various algorithms is taken into consideration by keeping the virtual machines stable to 80. Using four different workloads, the response time of the profounded CB-RALB-SA procedure is 225.53 ms for 2000 tasks which is 304.8 ms in CBSA-LB [23]. Likewise, the evaluation has been finished with other procedures including EMOSA [20] and CBSA [21]. The response time for EMOSA algorithm is 527.59 ms and CBSA algorithm is 462.84 ms. The Fig. 11 illustrates that the proposed CB-RALB-SA algorithm offers a improved total processing time as compared to other heuristics in the fix virtal machines scenario.This improved value of CB-RALB-SA is achieved due to resource-aware mapping mechanism, and load-balance factor for scheduling . Analysis reveals that CB-RALB-SA is a propounded algorithm.

With the experimental evaluations and consequences, it has been substantiated that the proposed methodology provides 48.5% better in Processing Time and 16.90% better results in make span time than the Existing CBSA-LB [23] algorithm. Thus, it improves the efficiency of the processor while uplifting the whole system's performance and has saved memory allocated to tasks and RAM.

6. Conclusion

Cloud computing is a promising development which is a model for giving the assets that are situated by customer benefits. However, scheduling still one of the progressing explorations organized area in distributed computing context. In this paper, different existing MOSA, EMOSA, CBSA, CBSA-LB are scrutinized in associated work. CB-RALB-SA (credit-based resource aware load balancing scheduling) is used in propounded algorithm. The algorithm offers superior scheduling elucidations with efficiently less completion time. Proposed algorithm is compared with other existing techniques. Results demonstrate that the proposed algorithm is good exclusive of increasing processing cost, execution time, and response time. With the improved Makespan Time, Processing Time, Processing Cost, Execution Time and Response Time, the proposed system is saving time and improving the efficiency of the processor while uplifting the performance of the whole system by 48.5% in processing time and 16.90% in Makespan time Also, has saved memory allocated to tasks and RAM by abandoning the employed memory on cloud resources, which is now free and available to other tasks for use.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature and acknowledgment of collaborative research and discussions. The work was done under the guidance of Dr Sunita Dhingra, at the University Institute of Engineering and Technology, MDU, Rohtak.

References

[1] S. Srichandan, T.A. Kumar, S. Bibhudatta, Task scheduling for cloud computing using multiobjective hybrid bacteria foraging algorithm science direct, Future Comput. Inf. J. (2018) 210–230.
[2] M. Vijayalakshmi, V.V. Kumar, Investigations on job scheduling algorithms in cloud computing, Int. J. Adv. Res. Comput. Sci. Technol. (2014) 157–161.
[3] A. Narwal, S. Dhingra, Scheduling techniques in cloud computing framework: A systematic review, Int. J. Adv. Stud. Comput. Sci. Eng. 5 (7) (2016) 1–9.
[4] K. Sutha, G.M.K. Dr Nawaz, Research perspective of job scheduling in cloud computing, in: International Conference on Advanced Computing, IEEE, 2016, pp. 61–66.
[5] A. Sharma, S. Tyagi, Task scheduling in cloud computing, Int. J. Sci. Eng. Res. (2016) 1–5.

[6] S. Sindhu, Task scheduling in cloud computing, Int. J. Adv. Res. Comput. Eng. Technol. (2015) 3019–3023.
[7] A. Hussain, M. Aleem, A. Khan, M.A. Iqbal, M.A. Islam, RALBA: A Computation-Aware Load Balancing Scheduler for Cloud Computing, Springer, 2018.
[8] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, A. Abraham, Hybrid job scheduling algorithm for cloud computing environment, in: Proceedings of the Fifth Intern. Conf. on Innov. in Bio-Inspired Computing. and Applied Science, 2014, pp. 43–52.
[9] A. Kaur, B. Kaur, D. Singh, Challenges to task and workflow scheduling in cloud environment, Int. J. Adv. Res. Comput. Sci. (2017) 412–415.
[10] A. Narwal, S. Dhingra, Task scheduling algorithm using multiobjective functions for cloud computing environment, Int. J. Control Theory Appl. (2017) 227–238.
[11] A.V. Lakra, D.K. Yadav, Multiobjective tasks scheduling algorithm for cloud computing throughput optimization, in: International Conference on Intelligent Computing, Communication and Convergence, Elsevier, 2015, pp. 107–113.
[12] A. Ananth, K. Chandrasekaran, Cooperative game theoretic approach for job scheduling in cloud computing, in: Conference on Computing and Network Communications, IEEE, 2015, pp. 147–156.
[13] A. Thomas, G. Krishnalal, V.P.J. Raj, Credit based scheduling algorithm in cloud computing environment, in: International Conference on Information and Communication Technologies, Elsevier, 2015, pp. 913–920.
[14] K. Raja, G. Sekar, An algorithm for credit based scheduling in cloud computing environment depending upon deadline strategy, Int. J. Curr. Trends Eng. Res. (2016) 70–76.
[15] S. Srichandan, T.A. Kumar, S. Bibhudatta, Task scheduling for cloud computing using multiobjective hybrid bacteria foraging algorithm science direct, Future Comput. Inform. J. (2018) 210–230.
[16] M. Maqableh, H. Karajeh, Job scheduling for cloud computing using neural networks, Commun. Netw. (2014) 191–200.
[17] S. Bitam, Bees life algorithm for job scheduling in cloud computing, ICCIT 18 (2012) 6–191.
[18] W. Kimpan, B. Kruekaew, Heuristic task scheduling with artificial bee colony algorithm for virtual machines, in: International Conference on Soft Computing and Intelligent Systems, IEEE, 2016, pp. 281–286.
[19] S. Loganathan, S. Mukherjee, Job Scheduling with Efficient Resource Monitoring in Cloud Datacenter, Hindawi Publishing Corporation, 2015, pp. 1–11.
[20] A. Narwal, S. Dhingra, Enhanced task scheduling algorithm using multiobjective function for cloud computing framework, in: NGCT 2017, vol. 827, Springer CCIS, 2018, pp. 110–121.
[21] A. Narwal, S. Dhingra, Performance analysis of multi objective algorithms for cloud computing framework, J. Adv. Res. Dyn. Control Syst. 11 (Special Issue-05) (2019) 2093–2099.
[22] A. Narwal, S. Dhingra, Analytical review of load balancing techniques in cloud computing, Int. J. Adv. Res. Comput. Sci. 9 (2) (2018) 00.
[23] A. Narwal, S. Dhingra, Credit based scheduling with load balancing in cloud environment, Int. J. Adv. Trends Comput. Sci. Eng. 9 (2) (2020) 1121–1127.

Abhikriti Narwal completed her UG and PG in the Department of Computer Science Engineering and completed her Ph.D. in the domain of Department of Computer Science. Currently she pursuing Ph.D. in the Department of Computer Science at Maharshi Dayanand University. Her area of interest's is data mining, data security and cloud computing.

Sunita Dhingra completed his UG and PG in the Computer Science Engineering and completed his Ph.D. in the domain of Department of Computer Science. Currently she is working as an Assistant Professor in the Department of Computer Science at Maharshi Dayanand University. Her area of interest's is Artificial Intelligence, cloud computing and data mining.