Computing and Informatics, Vol. 41, 2022, 1157–1185, doi: 10.31577/cai_2022_5_1157

PSO-CALBA: PARTICLE SWARM OPTIMIZATION BASED CONTENT-AWARE LOAD BALANCING ALGORITHM IN CLOUD COMPUTING ENVIRONMENT

Muhammad Adil, Said Nabi, Summair Raza

Department of Computer Science and Information Technology
Virtual University of Pakistan
Islamabad, Pakistan
e-mail: fazal.adil@gmail.com, {said.nabi, sraza}@vu.edu.pk

Abstract. Cloud computing provides hosted services (i.e., servers, storage, bandwidth, and software) over the internet. The key benefits of cloud computing are scalability, efficiency, and cost reduction. The key challenge in cloud computing is the even distribution of workload across numerous heterogeneous servers. Several Cloud scheduling and load-balancing techniques have been proposed in the literature. These techniques include heuristic-based, meta-heuristics-based, and hybrid algorithms. However, most of the current cloud scheduling and load balancing schemes are not content-aware (i.e., they are not considering the content-type of user tasks). The literature studies show that the content type of tasks can significantly improve the balanced distribution of workload. In this paper, a novel hybrid approach named Particle Swarm Optimization based Content-Aware Load Balancing Algorithm (PSO-CALBA) is proposed. PSO-CALBA scheduling scheme combines machine learning and meta-heuristic algorithm that performs classification utilizing file content type. The SVM classifier is used to classify users' tasks into different content types like video, audio, image, and text. Particle Swarm Optimization (PSO) based meta-heuristic algorithm is used to map user's tasks on Cloud. The proposed approach has been implemented and evaluated using a renowned Cloudsim simulation kit and compared with ACOFTF and DFTF. The proposed study shows significant improvement in terms of makespan, degree of imbalance (DI).

Keywords: Cloud, load-balance, content-aware, classification, SVM, PSO, workload scheduling, optimization

1 INTRODUCTION

Cloud Computing is providing computing services over the internet called "The Cloud". Cloud comprises servers, networking, storage, analytics, and software services. The popular public cloud services are Amazon Web Services [1], Microsoft Azure [2], and Google Cloud Platform [3]. Cloud computing makes your operations more reliable, provides agility to your software deployment process, and enables better collaboration amongst teams. Moreover, client data is available over the internet, and users can access data from anywhere. This lowers the operational and infrastructure cost. Moreover, this makes client applications more reliable, scalable, and secure [4].

To achieve the full benefit of cloud computing, the users' jobs should be scheduled in a balanced manner. The process of effective distribution of workload across multiple cloud resources is called Load Balancing. This reduces computing costs and maximizes resource availability. Cloud load balancing also plays a vital role in scalability [5]. The load balancing process adjusts itself as the number of Virtual Machines (VM) increases to scale the application. Numerous cloud load-balancing and scheduling methods have been proposed by different researchers, i.e., heuristic-based, meta-heuristic-based, and hybrid.

Heuristic-based algorithms like round-robin, max-min, and min-min are problem dependent [6]. These are fast and feasible solutions to cloud scheduling problems. A number of heuristic-based techniques are proposed in the literature including [7] that is based on task transfer time onto the network, Resource Aware Dynamic Load Balancing Algorithm (DRALBA) [8], and heuristic-based load balancing technique [9] is dependent on task size among others. These studies presented improvement in the makespan Quality of Service (QoS) metric. However, the limitation of the heuristic-based model is that it is sometimes unable to deliver an optimal solution [10], especially for conflicting parameters for instance task execution time and execution cost among others.

Meta-heuristic-based algorithms are problem-independent design patterns for solving a wide range of problems [11]. Meta-heuristic methods overcome the limitation of heuristic-based methods and provide the same performance in searching for optimal solutions. A number of meta-heuristic-based procedures have been suggested in the literature that includes PSO-based deadline and resource aware load-balancing scheduler PSORDAL [12], dynamic [6] and elasticity approach (D-ACOELB) [13], Simulated Annealing [14] and PSO-based meta-heuristic [15]. However, PSO-based meta-heuristic algorithms are effective in the matter of both speed and memory [16, 17].

A hybrid meta-heuristic is a concept to combine multiple algorithms for optimization problems. Firefly-Genetic [18] and Cuckoo-Firefly [19] are hybrid meta-heuristic approaches to address load-balancing challenges. Furthermore, different surveys [20, 21] present the application of different meta-heuristic-based methods for task scheduling problems in cloud computing.

1.1 Motivations

This paper's motivations are as per followings:

- The existing load-balancing approaches are dependent on different parameters of the tasks; i.e., length, priority, etc. Most of the state-of-the-art cloud load-balancing methods are not content-aware. However, the literature study reveals that the content type of tasks can significantly improve the balanced distribution of workload.
- Moreover, with the growing usage of cloud computing, the data in the clouds is massively increasing as well. There are several studies to handle both these challenges with machine learning algorithms [22]. Machine learning is a concept of using computing algorithms that takes and analyzes data to build intelligence. Machine learning algorithms can be classified into the following categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning [23]. Classification is a subset of supervised learning algorithms [24] in machine learning. The classification algorithms learn from the data given to them and make new classifications [11, 25].
- To enhance classification precision and to resolve scheduling problems, hybrid approaches of machine learning and meta-heuristic-based algorithms were proposed by some researchers. These hybrid methodologies have uncovered great outcomes in numerous studies [26, 27]. Most of the existing classification techniques that have incorporated the content types in cloud computing are by utilizing PostgreSQL and Amazon Web Services (AWS) [28]. However, these techniques do not support content classification for a content type like video, audio, image, and text. The content-type classification can be achieved with supervised learning algorithms like Support Vector Machine (SVM) [29] among others.
- The existing content-aware load balancing techniques like Ant Colony Optimization File Type Format (ACOFTF) [30] and Data Files Type Formatting (DFTF) [31] are using the polynomial kernel method for content classification. However, these techniques need to be improved with more refined datasets, refined kernel methods, and a simplified scheduling algorithm.
- The above discussion shows that the performance of load balancing algorithms can be improved if equipped with content awareness. The content awareness will be achieved by using ML based classification algorithms. These algorithms classify tasks into different categories based on their content type. To overcome limitation of state-of-the-art, a content-aware load-balancing algorithm is presented in the next sections.

1.2 Contributions

This paper presents a PSO based Content Aware Load Balancing Algorithm (PSO-CALBA) in cloud computing which uses a content-aware model that classifies the

users' tasks based on their content type. PSO-CALBA uses SVM for the classification of the tasks. SVM is considered one of the best classification techniques for cloud-based workload [32]. The proposed PSO-CALBA technique uses PSO for mapping the tasks onto the appropriate VM. The purpose of using the PSO for task scheduling is that PSO-based algorithms are effective with regard to both speed and memory. The proposed method classifies users' tasks based on file fragment type and classified these tasks into video, audio, image, and text tasks. It creates appropriate virtual machine (VM) groups for each task type. Further, it distributes and schedules the workload using PSO amongst these VM groups. The major contributions are summarized as:

- This research proposes a PSO-based content-aware load balancing model (PSO-CALBA) that is based on SVM and PSO algorithms. PSO-CALBA uses the state-of-the-art dataset based on File Fragment Type (FFT) [33]. The proposed model not just emphasizes accomplishing the best classification precision yet also performs balanced scheduling.
- The classification of the workload based on their content type (i.e., video, audio, image, and text) would be performed using SVM in a cloud environment.
- The proposed PSO-CALBA scheduling technique performs optimum load balancing of the classified tasks using PSO based optimization algorithm.
- Comprehensive evaluation and comparison have been performed in terms of makespan and degree of imbalance.
- Simpler and easy to implement the solution as compared to existing content-aware approaches.
- The proposed model improves makespan by 17 % and degree of imbalance by 52 % as compared to existing models ACOFTF [30] and DFTF [31].

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 presents the proposed framework that includes PSO-CALBA algorithm, classification, load balancing, fitness evaluation, and performance model. Section 4 discusses experimental setup, analysis, performance metric, and evaluation analysis. Section 5 concludes the paper.

2 RELATED WORK

Resource-Aware Load Balancing Algorithm (RALBA) has been proposed in [34] that distributes tasks according to the computing capabilities of VMs. RALBA performs task execution in two steps: the tasks are distributed based on the processing power of VMs and obtain the computation requirements of cloud tasks. RALBA comprises two nested schedulers named Fill and Spill sub-schedulers. The RALBA has improved makespan, execution time, and resource utilization. However, tasks are not classified based on task types, and experiment results show it needs improvement in the fault-tolerant scheduling mechanism.

DRALBA is a resource aware and dynamic load balancing scheduler is presented in [8]. DRALBA scheduler keeps track of the workload and computation power of virtual machines (VM). It performs load balancing by allocating a group of independent tasks to the pre-defined number of VMs. It estimates the processing capabilities of all VMs for a group of tasks, selects the best VM with maximum computation share, and assigns task with maximum computation requirements less than are equal to the VM computation share. DRALBA shows improvements in response time and resource allocation. However, it does not consider task classification based on task content type.

OG-RADL [35] is a dynamic scheduling method that improves compliance with tasks' deadlines, provides better resource utilization, and offers heightened performance. The proposed method also computes and evaluates the combined effect of multiple evaluation parameters like makespan, task deadline, task response time, and resource utilization ratio. The proposed method has achieved improvements in the overall gain. However, the proposed method is not classifying tasks based on the content type. It needs further work for sequence-based tasks.

Authors in [36] propose a cluster-based task scheduling framework (CBTS) using K-Means clustering by considering task length and VM capacity. In CBTS, the tasks are allocated depending on the length, and the VMs are clustered with regard to computing power. After clustering, the individual task in each cluster is scheduled to the appropriate VM in the VM groups. The purposed method presented improvement in execution time and makespan. However, it shows no task classification based on task content type.

PSO-RDAL is a PSO-based deadline and resource aware dynamic load balancer (PSO-RDAL) proposed in [12]. The PSO-RDAL scheduling algorithm presents improvements with regard to cost and time for heavy processing and independent tasks. It shows improvements in terms of makespan, resource consumption, compliance with the task's deadline, response time, overall execution cost, and penalty cost. Though, it does not support any sort of task classification.

Authors in [37] have proposed a Starvation Threshold-based scheduling scheme. Each VM keeps its own state of workload and performs load-balancing without considering other VMs' states. Performance of the purposed scheme is evaluated using up to 100 VMs and up to 800 tasks. The purposed study presented a better performance in makespan and task response time. However, it is tested with a smaller dataset. Moreover, task content-based classification is not supported.

Authors in [38] have presented a mutation-based PSO task scheduling algorithm. The fitness value of each particle is updated for every iteration. Makespan is used for experiment result analysis. The purposed method used up to 200 tasks and 20 data centers for experiments. The purposed method presented improvements in makespan. However, the study does not explain the pseudo implementation. Also, it shows no proper dataset implementation. Further, it lacks any sort of content-based task classification.

Artificial Bee Colony (ABC) based task scheduling algorithm is proposed in [39]. The ABC algorithm finds best source of food by using honey bee's searching technique. This scheme considers makespan and task execution time as scheduling objectives. The proposed technique is not using state-of-the-art datasets, and is not content-aware.

Firefly Algorithm (FA) and particle swarm-based optimization technique have been proposed in [40]. The proposed approach allocates the jobs with shorter jobs to the quickest processor and applies to the shortest job next (SJN to PSO). The proposed method presented the improvements in makespan and task migration. However, it experiments within a limited environment and lacks state-of-the-art datasets. Furthermore, it does not support any task classification based on task nature.

Honey-Bee Optimization (HBO) and PSO-based scheduling scheme are proposed in [41]. The proposed approach distributes workload across VMs and presents improvement in makespan, DI, and response time. However, experiments are conducted in a limited setup with limited jobs. Statistics can vary with the larger dataset. Moreover, the proposed method lacks any kind of task classification based on task type.

The Firefly Algorithm (FA) and Dragonfly Algorithm (DA) based adaptive scheduling technique has been presented in [42]. The proposed approach's primary objective is to map tasks on VM by using Adaptive DA. The presented method has shown enhancements in execution time. Though, the number of VMs considered is high compared to the total number of tasks. Moreover, it is not a content-aware approach.

A research proposed in [43] is based on the Best Worst Method (BWM) and the ranking method (VIKOR). The VIKOR algorithm acts as an administrator to indicate the task computation requirements. The proposed scheduling scheme considers and improves makespan, throughput, and utilization of virtual resources. However, it lacks classification of tasks and scalability is difficult to be determined.

Authors have presented Simulated Annealing and Harris Hawks Optimization in [44]. The SA algorithm provides a better local search that improves the performance of HHO technique. This technique improves makespan, however, the proposed scheme lacks the content based classification of tasks.

In [45], a Harmony-Inspired Genetic Algorithm (HIGA) hybrid meta-heuristic approach has been proposed. It uses exploration features and exploitation features of respectively genetic algorithm and harmony search. It finds local and global optimal and gives a speedy combination. The proposed research does not classify tasks based on task types.

Data Files Type Formatting (DFTF) that is based on Cat Swarm Optimization (CSO) and SVM [31]. It classifies the cloud tasks into different types, i.e., text, images, video, and audio with SVM by using a polynomial kernel. The classified tasks input CSO to perform load balancing. However, it does not use a state-of-the-art dataset. Further, classifying videos and audio into further categories is excessive.

Ref	Approach	Pros	Cons	CA
[8]	Heuristic	minimized response time and higher resource utilization	no classification and DI	✗
[12]	Meta-heuristic	reduced makespan, execution cost, and reduced resource consumption	lacks of tasks classification and DI	✗
[31]	Hybrid	lower energy consumption, higher throughput, and minimized overhead time	lack painless classification, unified dataset, and need kernel method optimization	✓
[35]	Heuristic	higher utilization of resources and improved makespan	lacks DI, sequence-based support, and tasks' classification	✗
[36]	Heuristic	improved makespan and execution time	DI not evaluated	✗
[37]	Meta-heuristic	improved makespan and reduced response time	no substantial enhancement in DI	✗
[38]	Meta-heuristic	lower makespan	lacks DI, pseudo implementation not given	✗
[39]	Meta-heuristic	lower makespan and improved execution time	scalability not checked	✗
[40]	Hybrid	higher resource utilization and lower makespan	no task classification, limited dataset and test environment	✗
[41]	Hybrid	lower makespan and response time	lacks proper test environment, dataset, and categorization	✗
[42]	Hybrid	lower cost and execution time	dataset and testing environment not upto the mark	✗
[43]	Hybrid	improved makespan, throughput, and waiting time	results can change with more refine test environment and proper dataset	✗
[44]	Hybrid	lower makespan	dataset used is not up to the mark	✗
[45]	Hybrid	better energy consumption and reduced makespan	more QoS metrics, smaller dataset	✗
[46]	Hybrid	enhanced throughput, makespan	lacks proper dataset and environment	✗

Table 1. Summary of studied literature

The Radial Basis Function (RBF) kernel provides more accuracy for the fine-tuned dataset as compared to the polynomial kernel.

The QMPSO is a hybrid approach of modified Q-learning and particle swarm optimization [46]. Three objective functions have been formulated; the first one, the difference of load between each host and average load on the Cloud network; the second one, with regard to the total energy consumption and the third one, with regard to numerous tasks submitted. The purposed method presented improvements

in throughput, energy utilization, and makespan. However, it lacks a state-of-the-art, refined experiment environment, and content-aware classification.

The purposed method is a mixture of Firefly and PSO techniques (FIMPSO) [47]. The FIMPSO algorithm improves load balancing based on cloud tasks' resource utilization. The purposed model presented improvements in makespan and throughput. However, experiments are not conducted in a state-of-the-art simulation environment. Analysis of results may vary with a proper testing environment. Moreover, it lacks the proper dataset since it creates tasks randomly. The purposed model categorizes tasks based on size but it is not addressing the classification of tasks based on task types.

The study proposed the Honey Bee Behavior-based Load Balancing method [48] that tries to minimize load redundancy by mapping the tasks to fitting VMs. After task allocation, it calculates the state of the VM. The proposed method presented improvement in following QoS performance matrices makespan, degree of load balancing. However, as compared to other honey bee methods it presents no such improvements in response time. Further, the purposed study lacks any classification approach to categorize tasks based on content type.

The comparison of related work is available in Table 1 in the summarized form.

3 PROPOSED FRAMEWORK

Load balancing is a key challenge in the cloud computing setup. To get maximum output from cloud computing the workload needs to be distributed in a balanced way across numerous VMs. This aids in the best consumption of resources and henceforth in improving the system performance. The load balancers intercept every inward request that is diverted to an appropriate customer. In light of pre-defined parameters, like accessibility or existing workload, the load balancing methods use different scheduling algorithms to figure out the most fitted VM and advance the request onto the particular VM.

The proposed model is a hybrid approach of the SVM machine learning algorithm with PSO called PSO-CALBA for load balancing in the cloud environment. The system diagram of the proposed framework is presented in Figure 1. In a cloud computing environment, the physical instance layer [49] contains physical servers which are called hosts. Hosts contain processing, storage, and transfer capabilities. The physical instance layer also contains storage devices that make data available from anywhere. The virtual instance layer contains virtual machines. Whereas, VM is a virtual computer system that has storage, processing element(s), network interface, and operating system. Virtual machines share defined resources of the host machine. The users submit tasks as cloudlets. These tasks are processed by the load balancing component. That component is responsible for the optimal distribution of cloudlets amongst the VMs.

The architecture diagram of the proposed model is shown in Figure 2. The proposed model is divided into two phases:

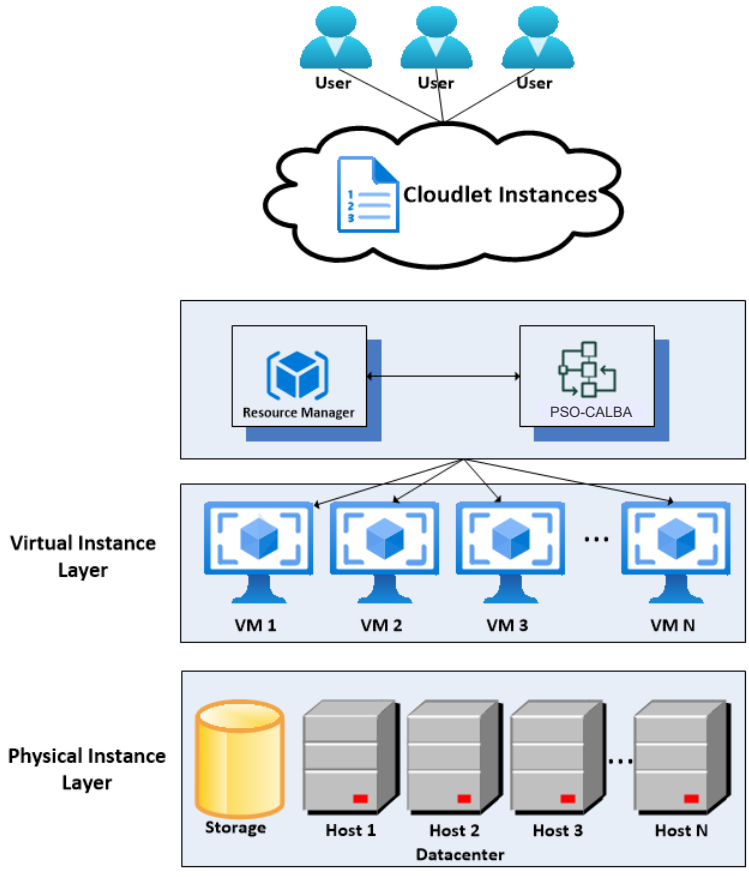

Figure 1. System diagram of PSO-CALBA

1. the Classification phase and
2. the Load Balancing phase.

These phases are explained in Section 3.1. The process starts by taking input tasks and continues with the classification process. The output of the classification process further proceeds to the load balancing process.

3.1 Description of PSO-CALBA Algorithm

PSO-CALBA Algorithm 1 starts with taking file fragments-based tasks which include various file fragments of content types such as text, image, audio, and video.

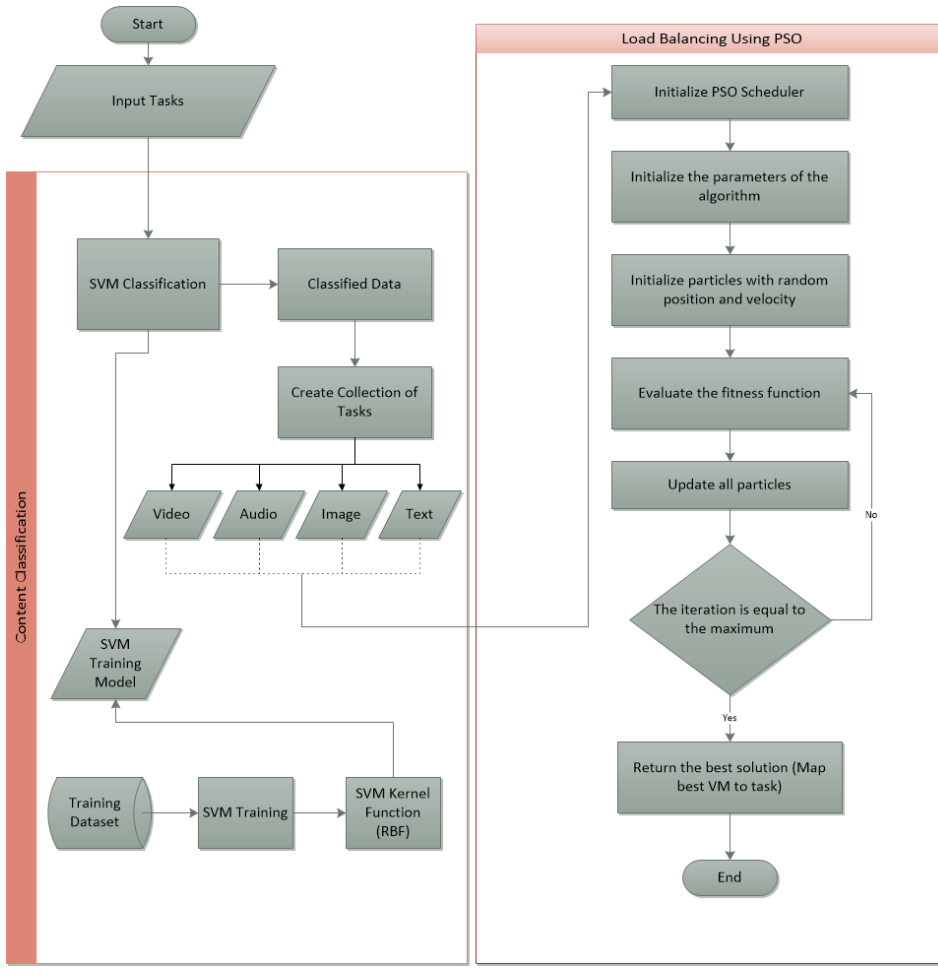

Figure 2. Architecture of PSO-CALBA

The first phase is classification, which classifies tasks using SVM and produces classified tasks. It processes high dimension data using the Radial Basis Function (RBF) kernel method. This is presented in Algorithm 1 from line number 1 to line number 8. Further, the second phase is to perform load balancing of classified tasks. This starts from line number 9, it takes classified tasks collection and groups of VMs. It uses PSO based algorithm to perform the task scheduling which results in scheduled data as explained from line number 9 to line number 35. The PSO scheduler initializes its parameters from line number 11 to 12. Then it initializes particles with random position and velocity, as explained in lines number 13 to 18.

Then it evaluates the fitness function in each iteration to update the local best and global best. It performs iterations until exit criteria are reached, as explained in lines 19 to 32. PSOSchedule method selects the appropriate VM for each task and returns the scheduled tasks data, as explained in line 33. The key components of the proposed model are described below.

3.1.1 Classification Phase

The purpose model's process initiates with a sparse matrix collection of file fragments of videos, audios, images, and texts. The SVM classification algorithm is used to prepare a training model from the training dataset. A training model is a process of machine learning that uses a training dataset to learn and examine the training examples to prepare a database of values and labels of all classes. Whereas, the training dataset contains class labels, feature sets, and values. The SVM classification algorithm process uses a training model to process the input data which is a testing dataset for cross-referencing and predicting the class of each input task.

In the purposed model we produced the SVM Training model with help of the kernel function feature. SVM algorithms utilize a bunch of mathematical functions that are characterized as the kernel. The capacity of the kernel is to accept data as input and change it into the necessary structure. That feature helps to convert lower dimension space data to higher dimension space. For the training model, Gaussian Radial Basis Function (RBF) is used as a kernel function for classification which is one of the most robust and commonly used kernel functions in SVM. RBF kernel equation is presented below:

$$f(x1, x2) = \exp(-\gamma * \|X1 - X2\|^2). \tag{1}$$

In the above equation, gamma (γ) states other points around a single training point. $X1 - X2$ is the product among features.

The FFT dataset [33] used in the purposed model includes random collection of file fragments from 75 content types. It is a fine tuned dataset for classification. The dataset includes class identifications and labels, and is arranged for SVM training models. The dataset is in NumPy [50] format which is further transformed using Scikit-learn [51] to LibSVM [52] dataset format.

The classification of tasks based on their contents like text, image, audio, and video classes reduces the need to pre-process feature learning, extraction of feature sets, and classification of a task in the workload in the balancing process. These classified tasks collect inputs to the load balancing process. According to the assumptions of the proposed technique, load balancing has been achieved and is based on three factors that are task type like audio, video, text, and image, task computation requirements (i.e., task length in Million Instructions), and virtual machine computation capacity in Million Instructions Per Second (MIPS). The task classification lowers the load balancer processing efforts by pre-processing learning, extraction of

feature sets, and classification. Tasks are converted into CSV (Comma-Separated Values) form after classification which is commonly used format and is easy to be read by a programming language. The row in CSV file represents the type of content that a task has, task size (in MI) and file size (in MI). Table 2 presents the sample of the dataset.

Type of Task	Task Size (in MI)	Size of File
VIDEO	4 825	1 632
AUDIO	3 112	945
IMAGE	1 730	502
TEXT	850	273
VIDEO	8 276	2 255
AUDIO	3 020	899
TEXT	720	235
TEXT	898	300
IMAGE	2 253	415
VIDEO	696	1 988
AUDIO	2 963	857
VIDEO	6 895	2 085
IMAGE	996	289
VIDEO	8 215	2 532

Table 2. Dataset (sample) used

The proposed model has partitioned the virtual machines (VMs) into four kinds of sets for VMs for text, audio, video, and image type cloud tasks. Every VM type has distinctive computing and storage capabilities. All the more definitely, each VM has mapped a cloud task dependent on cloud task content type. For instance, video tasks require 1 000 MIPS, required memory is 16 GB, and required storage is 360 GB. Similarly, audio tasks require 900 MIPS, required memory is 12 GB, and required storage is 250 GB, image tasks require 700 processing power (in MIPS), 8 GB and 200 GB memory and storage, respectively, and text tasks require 500 computation capacity (in MIPS), 120 GB and 4 GB storage and memory, respectively.

3.1.2 Load Balancing Phase

In the load balancing phase, we have collections of classified tasks and sets of VMs. Each set of VM is in control to perform the appropriate type of task. The task scheduling to the appropriate VM is done with PSO based on task length in the form of time. PSO is a swarm-based optimization algorithm inspired by birds and their food searching approach. It initiates the population randomly and finds the optimal solution by updating its position. Each possible solution in the problem domain is called a particle. PSO provides simpler and more efficient solutions with regard to memory and speed requirements. For implementation of PSO in CloudSim [53] environment, we have used JSwarm-PSO [54].

Algorithm 1 PSO-CALBA

procedure PSO-CALBA(tasks T, vms V)

  procedure ClassifyTasks(*tasks*)
    classifiedTasks : *null*
    for $i \leftarrow 1 \ldots \text{size}(T)$ **do**
      $classifiedTask \leftarrow predict(T[i])$
      $classifiedTasks.add(classifiedTask)$
    end for
    return *classifiedTasks* ▷ return classified tasks
  end procedure
  procedure PSOSchedule($classifiedTasks, V$)
    $itr \leftarrow 900$ ▷ *itr*: No. of iterations
    $p \leftarrow 80$ ▷ p represents population
    for particle $\leftarrow 1 \ldots p$ **do** ▷ particles, initialization
      for $t \leftarrow 1 \ldots size(classifiedTasks)$ **do**
        $position_{pt} \leftarrow random(Pos_{min}, Pos_{max})$
        $velocity_{pt} \leftarrow random(Vel_{min}, Vel_{max})$
      end for
    end for
    while $k \leq itr$ **do**
      for each particle p **do**
        $FV \leftarrow CalculateFitnessValue(\text{task type}, \text{task length}$
        if $FV \geq pbest_{pt}$ **then**
          $pbest_{pt} \leftarrow FV$
        end if
        $gbest_t \leftarrow pbest_{pt}$
      end for
      for each particle p **do**
        for task t in T **do**
          $v_{pt} \leftarrow W * v_{pt} + C * r() * (pbest_{pt} - p_{pt}) + C * r() * (gbest_{pt} - p_{pt})$
        end for
      end for
    end while
    return tasks to VMs mapping
  end procedure
  return Mapped Tasks
end procedure

Fitness Evaluation. The fitness evaluation is a crucial part of meta-heuristic algorithms. This helps to implement your required optimization and apply the appropriate algorithm. Using the fitness function we can optimize the performance of the load balancing with regard to optimization targets. Fitness evaluation is performed at each iteration of the meta-heuristic algorithm. It also checks if the termination condition is satisfied and ends the algorithm execution. The key objective of the proposed model is to optimize the workload distribution by minimizing the makespan and degree of imbalance.

Makespan. Minimizing the makespan is essential for the load-balancing algorithms in the cloud computing environment [55]. Makespan is the completion time of all the scheduled tasks with the available computing and storage resources [56]. The makespan T_k of VM_l is defined in the following equation CT_{kl}.

$$Makespan = CT_{Max[k,l]}, \quad k \in T,\ k = 1, \ldots, n,\ l \in VM,\ l = 1, \ldots, m \tag{2}$$

where CT_{max} is the maximum completion time for the k^{th} task on the l^{th} VM. Where n represent number of tasks and VM count is represented by m. To minimize the completion time, the processing time of every task for each virtual machine should be calculated by the load balancing process. So, the makespan can be calculated with Equation (3).

$$CT_{max} = \sum_{k=1}^{m} \sum_{l=1}^{n} (t_{kl}.X_{kl}). \tag{3}$$

Thus, makespan is the total completion time of all the tasks that can be calculated with help of the objective function [57, 58] presented in Equation (4).

$$F2(t) = \min\{Makespan(t)\}. \tag{4}$$

Degree of Imbalance. The degree of imbalance is a measure to figure out the balance of the tasks amongst VMs [59]. We can measure [60, 61] DI with the help of Equations (5) and (6).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}}, \tag{5}$$

$$T_j = \frac{LI}{PC_{n_j} \times PC_{MIP_j}}. \tag{6}$$

In Equation (5), T_{max} is the maximum finish time of T_j tasks amongst all the VMs, and T_{min} is the minimum finish time of T_j tasks for all VMs. Moreover, T_{avg} is the average of T_j tasks on VMs. In Equation (6), LI is the length of instructions. PC_{nj} is the amount of processing cores in the j^{th} VM. Further, PC_{MIPj} is the computation power (the million instructions per second) of the j^{th} VM.

4 EXPERIMENTAL EVALUATION

Hosts	PEs	RAM (GB)	Storage (TB)	Speed	Bandwidth
1–100	2	64	3	10 000	10 240
1–100	4	64	6	10 000	10 240
1–100	8	128	8	10 000	10 240

Table 3. Parameters of host machine

4.1 Experimental Setup

CloudSim [53] is used for experiments which is an extensible simulation framework that supports the simulation [62], modeling, and experimentation of cloud computing infrastructures and application services. It helps users to focus on specific system design without getting low-level details of cloud infrastructure and services. It supports the modeling and simulation of large-scale cloud computing data centers, virtualized server hosts, and customizable policies for provisioning host resources to virtual machines. In this study, we have divided VMs into four categories; Text VM, Image VM, Audio VM, and Video VM. Each category of VMs has same number of VMs which ranges from 2 to 1 000, speed, and bandwidth and different number of processing elements (i.e., number of cores), memory, and storage configurations. The host settings of the test environment are explained in Table 3. Moreover, the VM settings of the testing environment are explained in Table 4

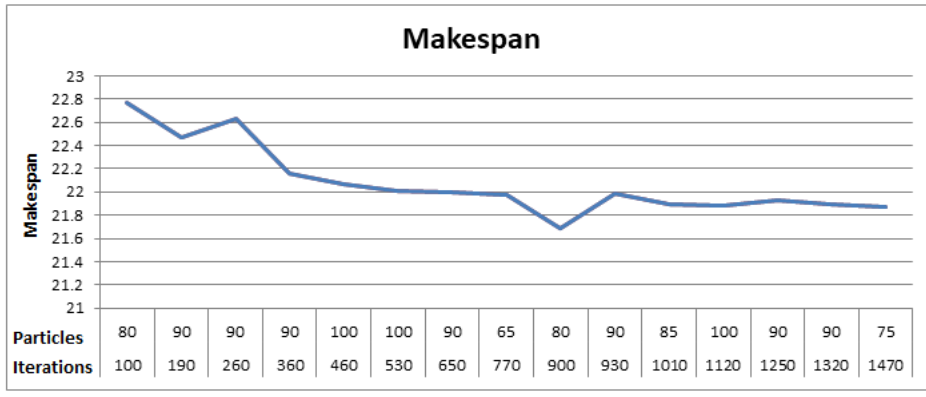

Figure 3. Makespan comparison by iterations and population

For experimental evaluation, state-of-the-art File Fragment Type dataset [33] from IEEE Dataport is used. This dataset contains file fragments of different file

VMs	PEs	Speed	RAM (GB)	Bandwidth	Storage (TB)
2–1 000	2	1 000	16	1 024	512
2–1 000	1	1 000	4	1 024	128
2–1 000	2	1 000	8	1 024	256
2–1 000	4	1 000	32	1 024	1 024

Table 4. Parameters of VMs

types, i.e., Text, Image, Audio, and Video. The dataset of each category is presented in Table 7. There is a total of 64 000 datasets which are further divided equally, as given in Table 8.

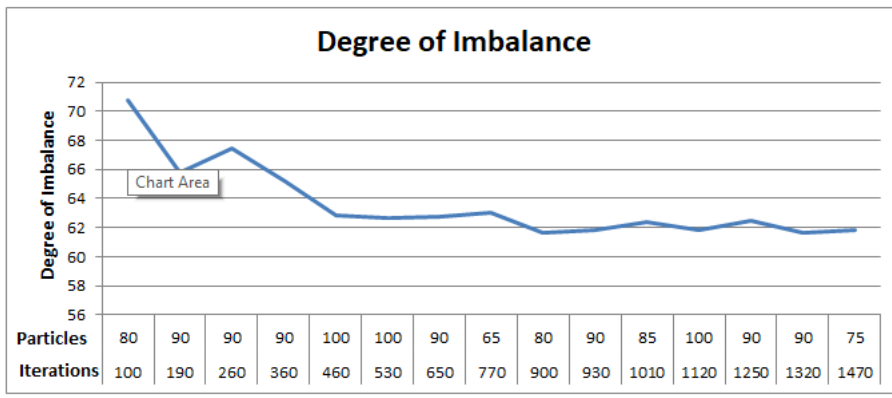

Figure 4. DI comparison by iterations and population

To select the best amount of iterations and population, we performed rigorous testing with 15 000 different combinations ranging iterations from 10 to 1 500 and populations from 1 to 100. Average statistics of each 100 iterations are compiled to get the best combination of the respective iteration range. The statistics of these best results are presented in Table 9. Based on these analytics, best combination with regard to makespan, degree of imbalance, and fitness value is presented in Figures 3, 4, and 5, respectively. Therefore, after thorough experiments with a wide range of combinations of iterations and populations, these statistics are prepared, and based on these analyses, we have selected the best iteration (900) and population (80) that produced the best makespan and DI.

Tables 5 and 6 show configuration and settings of parameters used for comparison of state-of-the-art, i.e., ACOFTF and DFTF, respectively. These approaches have been evaluated using different values and selected the values that give best results.

Param. Name	**Value**
Size of Population	500
Size of Cats	100
CDC	80 %
SMP	5
SRD	20 %

Table 5. Configurations of parameter used in DFTF

Parameter Name	**Value**
Initial Pheromone	0.1
RHO	12
Alpha	3
Beta	1

Table 6. Configurations of parameter used in ACOFTF

4.2 Experimental Analysis

The simulation of the proposed model has been compared with two state-of-the-art models. The following models are selected because both of these models are using machine learning technique for task classification and load balancing using meta-heuristic algorithm. Moreover, these strategies are considering task contents like video, audio, image, and text for load balancing. Therefore these strategies are the ultimate choices for comparison, as compared to others.

1. DFTF [31] that utilizes an updated CSO and SVM classifier. In the first step, DFTF receives data from different sources and classifies them into video, audio, image, and text using SVM classifier. At the classification stage, data is accepted in random way and using polynomial SVM to classify the input data. In the second step, the classified data is mapped on VMs using enhanced CSO algorithm. The CSO efficiently maps data on different resources.
2. Hybrid algorithm of File Type Format based ACO (ACOFTF) and SVM have been proposed in [30]. To perform classification of tasks into various types like video, audio, image, and text using polynomial SVM classifier. This technique distributes the classified tasks on different heterogeneous resources by using enhanced CSO algorithm.

Type of Dataset	**Size**
Text based data	16 000
Image based data	16 000
Audio based data	16 000
Video based data	16 000

Table 7. Description of dataset

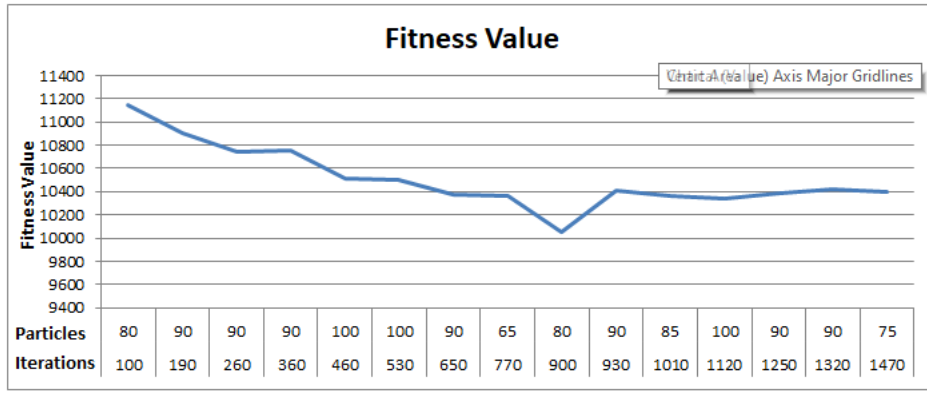

Figure 5. Comparison of fitness value by iterations and population

Samples	Size
Text File-Training	15 000
Text File-Testing	1 000
Image File-Training	15 000
Image File-Testing	1 000
Audio File-Training	15 000
Audio File-Testing	1 000
Video File-Training	15 000
Video File-Testing	1 000

Table 8. Training and test datasets statistics

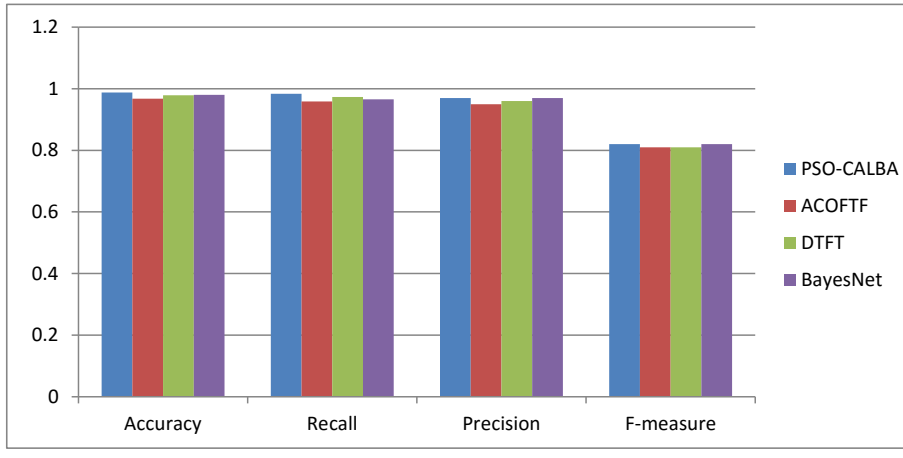

Figure 6. Classification comparison of all algorithms

Iterations	Particles	Makespan	DI	Fitness Value
100	80	22.7652	70.75877999	11 141
190	90	22.4708	65.77331986	10 901.8
260	90	22.6359	67.47649331	10 740
360	90	22.1577	65.25152384	10 758.2
460	100	22.0689	62.8479112	10 516.4
530	100	22.0109	62.69606111	10 506.2
650	90	21.9963	62.72235321	10 378.8
770	65	21.9741	63.00233047	10 364
900	80	21.689	61.63622496	10 056.6
930	90	21.99155	61.86157393	10 409.4
1 010	85	21.8958	62.38507693	10 368.4
1 120	100	21.8874	61.85361503	10 345
1 250	90	21.9318	62.50222742	10 384.4
1 320	90	21.8924	61.6423634	10 422.4
1 470	75	21.8731	61.84628536	10 394.6

Table 9. Iterations and population's best statistics

Measure	DTFT	ACOFTF	Bayes Net	PSO-CALBA
Precision	0.96	0.973	0.81	0.979
Recall	0.95	0.959	0.81	0.968
Accuracy	0.97	0.984	0.82	0.988
F-measure	0.97	0.966	0.82	0.98

Table 10. Results comparison of various classifiers

4.3 Metrics Used for Performance Evaluation

To evaluate the performance the proposed PSC-CALBA, two state-of-the-art techniques has been used, as discussed in Section 4.2. The makespan and degree of imbalance QoS parameter have been used for comparison and evaluation.

1. **Makespan:** Makespan is the maximum time taken by a resource to finish the execution of allocated tasks in a cloud datacenter. We have already explained the makespan with Equation (3).

	PSO-CALBA		ACOFTF		DFTF	
Tasks	**Makespan**	**DI**	**Makespan**	**DI**	**Makespan**	**DI**
500	23.59	58.92	31.29	93.83	36.854	95.18
1 000	47.595	69.03	77.842	105.18	82.859	123.17
2 000	92.11	80.20	127.322	118.76	134.595	125.28
4 000	177.18	81.83	209.332	128.80	210.526	141.82

Table 11. Makespan and DI based comparison results

2. **Degree of Imbalance:** The degree of imbalance is a measure to figure out the balance of the tasks amongst VMs. We have already explained how to measure DI with help of Equation (5).

4.4 Statistical Analysis and Evaluation

The proposed PSO-CALBA technique is evaluated in two steps:

1. Classification phase,
2. Load Balancing phase.

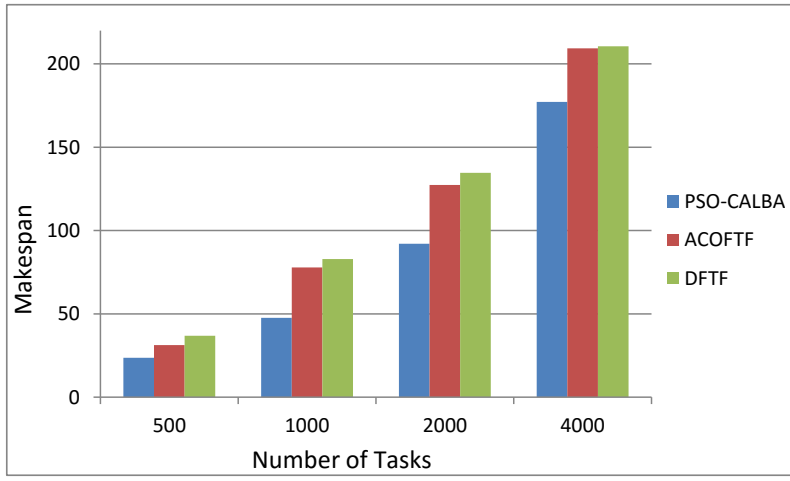

Figure 7. Makespan comparison of all algorithms

To find the significance of experimental results of the proposed technique, ANOVA test has been performed. Table 12 shows details of ANOVA test results.

Table 13 shows Degree of Imbalance results for the ANOVA test.

4.4.1 Classification Evaluations

Validation of the classification method is done based on the QoS measures – precision, recall, accuracy, and F-measure – to verify the accuracy of the PSO-CALBA. The classification methods such as DTFT [31], ACOFTF [30], and Bayes Net [63] are used to compare results. The results are presented in Table 10. The comparison of these results is presented in Figure 6. In this figure, the x-axis presents accuracy, recall, precision, and F-measure. Whereas, the y-axis presents the value of each classification measure. The evaluation reveals that the PSO-CALBA has shown better performance in all classification evaluation measures.

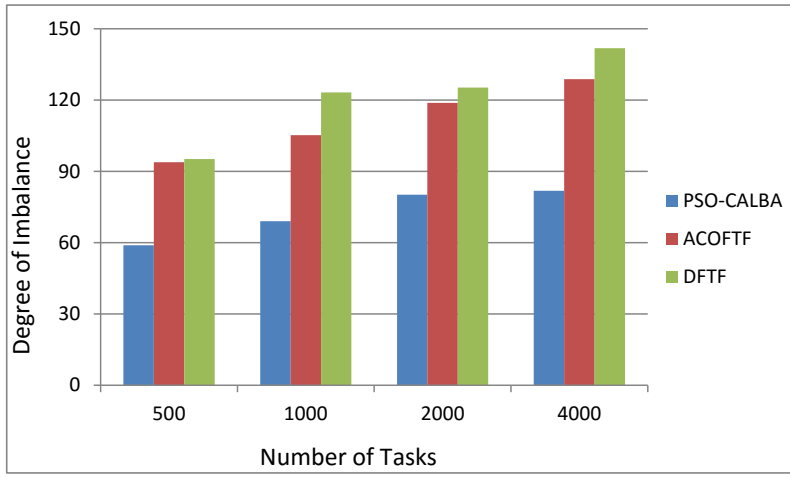

Figure 8. DI comparison of all algorithms

Groups	Count	Sum	Average	Variance		
PSO-CALBA	15	347	23.13	1.84		
ACOFTF	15	501	33.4	4.67		
DFTF	15	566	37.73	2.49		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1 587.64	2	793.82	264.048	1.63421E−24	3.22
Within Groups	126.26	42	3.006			
Total	1 713.91	44				

Table 12. ANOVA test for Makespan

Groups	Count	Sum	Average	Variance		
PSO-CALBA	15	914	60.93	7.64		
ACOFTF	15	1 470	98	22.42		
DFTF	15	1 539	102.6	24.54		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	17 463.11	2	8 731.56	479.671	1.1912E−29	3.22
Within Groups	764.53	42	18.20			
Total	18 227.64	44				

Table 13. ANOVA test for Degree of Imblance

4.4.2 Evaluation of Load Balancing Technique

1. **Makespan based evaluation:** The evaluation of the PSO-CALBA on makespan was performed and compared with other algorithms. This evaluation was performed on a varying number of VMs and tasks. The results of these evaluations are presented in Table 11. Based on these results, the comparison graph is presented in Figure 7 where the x-axis presents the number of tasks, and the y-axis presents the makespan value. Whereas, each column presents the respective algorithm. The experimental results reveal that PSO-CALBA reduced the makespan by 16 % and 17 %, as compared to ACOFTF and DFTF, respectively.
2. **Evaluation based on DI:** The evaluation of the PSO-CALBA on the degree of imbalance was performed and compared with other algorithms. This evaluation was performed on a varying number of VMs and tasks. The results of these evaluations are presented in Table 11. Based on these results, the comparison graph is presented in Figure 8 where the x-axis presents the number of tasks and the y-axis presents the DI value. Whereas, each column presents the respective algorithm. The experimental results reveal that PSO-CALBA improves the DI by 44 % and 52 %, as compared to ACOFTF and DFTF, respectively.

5 CONCLUSIONS AND FUTURE WORK

The content type is a significant part of the user's tasks in the cloud computing environment. Several studies show that tasks classification using machine learning algorithms can improve the load balancing process in cloud computing. In this research, we have proposed a content-aware load balancing model PSO-CALBA that produces improved results in workload distribution amongst virtual machines. The proposed model is designed in two steps,

1. classification step and
2. load-balancing step.

In the classification step, SVM is used to classify the users' tasks into different content types, e.g. images, videos, text, and audio. The classification phase takes these file fragments and produces classified tasks list based on content type. In the load-balancing phase, varying numbers of VMs are set up whereas, VMs are divided into four categories as regards content type. Each VM category has distinct storage, processing, and network configurations. The classified tasks list and VM sets are input to a PSO-based load-balancing algorithm that performs scheduling and maps tasks to the appropriate VM. The evaluation of the proposed model is performed with the comparison of DFTF and ACOFTF. The evaluation results show that PSO-CALBA presented significant improvements with regard to QoS measures such as makespan and DI, as compared to DFTF and ACOFTF. The proposed model also

is simpler and easy to implement, as compared to existing load balancing models. In the future, we will improve the load balancing model for other QoS measures, such as cost consumption, migration time, overhead time, energy consumption, and optimization time.

REFERENCES

[1] Services, A. W.: Amazon EC2. http://aws.amazon.com/ec2/.

[2] Chappell, D. et al.: Introducing the Windows Azure Platform. 2010.

[3] Krishnan, S. P. T.—Gonzalez, J. L. U.: Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects. Apress Berkeley, CA, 2015, doi: 10.1007/978-1-4842-1004-8.

[4] Sunyaev, A.: Cloud Computing. Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies, Springer, Cham, 2020, pp. 195–236, doi: 10.1007/978-3-030-34957-8_7.

[5] Mishra, S. K.—Sahoo, B.—Parida, P. P.: Load Balancing in Cloud Computing: A Big Picture. Journal of King Saud University – Computer and Information Sciences, Vol. 32, 2020, No. 2, pp. 149–158, doi: 10.1016/j.jksuci.2018.01.003.

[6] Nabi, S.—Aleem, M.—Ahmed, M.—Islam, M. A.—Iqbal, M. A.: RADL: A Resource and Deadline-Aware Dynamic Load-Balancer for Cloud Tasks. The Journal of Supercomputing, Vol. 78, 2022, No. 12, pp. 14231–14265, doi: 10.1007/s11227-022-04426-2.

[7] Kong, L.—Mapetu, J. P. B.—Chen, Z.: Heuristic Load Balancing Based Zero Imbalance Mechanism in Cloud Computing. Journal of Grid Computing, Vol. 18, 2020, No. 1, pp. 123–148, doi: 10.1007/s10723-019-09486-y.

[8] Nabi, S.—Ibrahim, M.—Jimenez, J. M.: DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. IEEE Access, Vol. 9, 2021, pp. 61283–61297, doi: 10.1109/ACCESS.2021.3074145.

[9] Adhikari, M.—Amgoth, T.: Heuristic-Based Load-Balancing Algorithm for IaaS Cloud. Future Generation Computer Systems, Vol. 81, 2018, pp. 156–165, doi: 10.1016/j.future.2017.10.035.

[10] Kaur, A.—Kaur, B.: Load Balancing Optimization Based on Hybrid Heuristic-Metaheuristic Techniques in Cloud Environment. Journal of King Saud University – Computer and Information Sciences, Vol. 34, 2022, No. 3, pp. 813–824, doi: 10.1016/j.jksuci.2019.02.010.

[11] Abdel-Basset, M.—Abdel-Fatah, L.—Sangaiah, A. K.: Chapter 10 – Metaheuristic Algorithms: A Comprehensive Review. In: Sangaiah, A. K., Sheng, M., Zhang, Z. (Eds.): Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications. Academic Press, Intelligent Data-Centric Systems, 2018, pp. 185–231, doi: 10.1016/B978-0-12-813314-9.00010-4.

[12] Nabi, S.—Ahmed, M.: PSO-RDAL: Particle Swarm Optimization-Based Resource- and Deadline-Aware Dynamic Load Balancer for Deadline Constrained Cloud

Tasks. The Journal of Supercomputing, Vol. 78, 2021, No. 4, pp. 4624–4654, doi: 10.1007/s11227-021-04062-2.

[13] Naik, K. J.: A Dynamic ACO-Based Elastic Load Balancer for Cloud Computing (D-ACOELB). In: Raju, K. S., Senkerik, R., Lanka, S. P., Rajagopal, V. (Eds.): Data Engineering and Communication Technology. Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 1079, 2020, pp. 11–20, doi: 10.1007/978-981-15-1097-7_2.

[14] Hanine, M.—Benlahmar, E. H.: A Load-Balancing Approach Using an Improved Simulated Annealing Algorithm. Journal of Information Processing Systems, Vol. 16, 2020, No. 1, pp. 132–144, doi: 10.3745/JIPS.01.0050.

[15] Megharaj, G.—Kabadi, M. G.: Metaheuristic-Based Virtual Machine Task Migration Technique for Load Balancing in the Cloud. In: Krishna, A. N., Srikantaiah, K. C., Naveena, C. (Eds.): Integrated Intelligent Computing, Communication and Security. Springer, Singapore, Studies in Computational Intelligence, Vol. 771, 2019, pp. 435–446, doi: 10.1007/978-981-10-8797-4_45.

[16] Li, X.—Wu, D.—He, J.—Bashir, M.—Liping, M.: An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. Journal of Control Science and Engineering, Vol. 2020, 2020, Art. No. 3857894, doi: 10.1155/2020/3857894.

[17] Nabi, S.—Ahmad, M.—Ibrahim, M.—Hamam, H.: AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing. Sensors, Vol. 22, 2022, No. 3, Art. No. 920, doi: 10.3390/s22030920.

[18] Rajagopalan, A.—Modale, D. R.—Senthilkumar, R.: Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm. In: Satapathy, S., Raju, K., Shyamala, K., Krishna, D., Favorskaya, M. (Eds.): Advances in Decision Sciences, Image Processing, Security and Computer Vision. Springer, Cham, Learning and Analytics in Intelligent Systems, Vol. 4, 2020, pp. 678–687, doi: 10.1007/978-3-030-24318-0_77.

[19] Kumar, K. P.—Ragunathan, T.—Vasumathi, D.—Prasad, P. K.: An Efficient Load Balancing Technique Based on Cuckoo Search and Firefly Algorithm in Cloud. International Journal of Intelligent Engineering and Systems, Vol. 13, 2020, No. 3, pp. 422–432, doi: 10.22266/ijies2020.0630.38.

[20] Pradhan, A.—Bisoy, S. K.—Das, A.: A Survey on PSO Based Meta-Heuristic Scheduling Mechanism in Cloud Computing Environment. Journal of King Saud University – Computer and Information Sciences, Vol. 34, 2021, No. 8, pp. 4888–4901, doi: 10.1016/j.jksuci.2021.01.003.

[21] Abrol, P.—Guupta, S.—Singh, S.: Nature-Inspired Metaheuristics in Cloud: A Review. In: Tuba, M., Akashe, S., Joshi, A. (Eds.): ICT Systems and Sustainability. Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 1077, 2020, pp. 13–34, doi: 10.1007/978-981-15-0936-0_2.

[22] Salkuti, S. R.: A Survey of Big Data and Machine Learning. International Journal of Electrical and Computer Engineering (IJECE), Vol. 10, 2020, No. 1, pp. 575–580, doi: 10.11591/ijece.v10i1.pp575-580.

[23] Mirjalili, S.—Faris, H.—Aljarah, I.: Introduction to Evolutionary Machine Learning Techniques. In: Mirjalili, S., Faris, H., Aljarah, I. (Eds.): Evolutionary

Machine Learning Techniques: Algorithms and Applications. Springer, Singapore, Algorithms for Intelligent Systems, 2020, pp. 1–7, doi: 10.1007/978-981-32-9990-0_1.

[24] Reddy, Y. C. A. P.—Varma, N. M. K.: Review on Supervised Learning Techniques. In: Venkata Krishna, P., Obaidat, M. S. (Eds.): Emerging Research in Data Engineering Systems and Computer Communications. Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 1054, 2020, pp. 577–587, doi: 10.1007/978-981-15-0135-7_53.

[25] Sen, P. C.—Hajra, M.—Ghosh, M.: Supervised Classification Algorithms in Machine Learning: A Survey and Review. In: Mandal, J. K., Bhattacharya, D. (Eds.): Emerging Technology in Modelling and Graphics. Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 937, 2020, pp. 99–111, doi: 10.1007/978-981-13-7403-6_11.

[26] Rabbani, M.—Wang, Y. L.—Khoshkangini, R.—Jelodar, H.—Zhao, R.—Hu, P.: A Hybrid Machine Learning Approach for Malicious Behaviour Detection and Recognition in Cloud Computing. Journal of Network and Computer Applications, Vol. 151, 2020, Art. No. 102507, doi: 10.1016/j.jnca.2019.102507.

[27] de Pinho Pinheiro, C. A.—Nedjah, N.—de Macedo Mourelle, L.: Detection and Classification of Pulmonary Nodules Using Deep Learning and Swarm Intelligence. Multimedia Tools and Applications, Vol. 79, 2020, No. 21, pp. 15437–15465, doi: 10.1007/s11042-019-7473-z.

[28] Amazon Web Services (AWS) – Cloud Computing Services: Using a PostgreSQL Database as an AWS DMS Source. `https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.PostgreSQL.html` [accessed 13.08.2021].

[29] Eswaran, S.—Rajakannu, M.: Multiservice Load Balancing with Hybrid Particle Swarm Optimization in Cloud-Based Multimedia Storage System with QoS Provision. Mobile Networks and Applications, Vol. 22, 2017, No. 4, pp. 760–770, doi: 10.1007/s11036-017-0840-y.

[30] Junaid, M.—Sohail, A.—Ahmed, A.—Baz, A.—Khan, I. A.—Alhakami, H.: A Hybrid Model for Load Balancing in Cloud Using File Type Formatting. IEEE Access, Vol. 8, 2020, pp. 118135–118155, doi: 10.1109/ACCESS.2020.3003825.

[31] Junaid, M.—Sohail, A.—Rais, R. N. B.—Ahmed, A.—Khalid, O.—Khan, I. A.—Hussain, S. S.—Ejaz, N.: Modeling an Optimized Approach for Load Balancing in Cloud. IEEE Access, Vol. 8, 2020, pp. 173208–173226, doi: 10.1109/ACCESS.2020.3024113.

[32] Cervantes, J.—Garcia-Lamont, F.—Rodríguez-Mazahua, L.—Lopez, A.: A Comprehensive Survey on Support Vector Machine Classification: Applications, Challenges and Trends. Neurocomputing, Vol. 408, 2020, pp. 189–215, doi: 10.1016/j.neucom.2019.10.118.

[33] Mittal, G.—Korus, P.—Memon, N.: File Fragment Type (FFT) - 75 Dataset. IEEE Dataport, 2019, doi: 10.21227/kfxw-8084.

[34] Hussain, A.—Aleem, M.—Khan, A.—Iqbal, M. A.—Islam, M. A.: RALBA: A Computation-Aware Load Balancing Scheduler for Cloud Computing. Cluster Computing, Vol. 21, 2018, No. 3, pp. 1667–1680, doi: 10.1007/s10586-018-2414-6.

[35] Nabi, S.—Ahmed, M.: OG-RADL: Overall Performance-Based Resource-Aware

Dynamic Load-Balancer for Deadline Constrained Cloud Tasks. The Journal of Supercomputing, Vol. 77, 2021, No. 7, pp. 7476–7508, doi: 10.1007/s11227-020-03544-z.

[36] Muthusamy, G.—Chandran, S. R.: Cluster-Based Task Scheduling Using K-Means Clustering for Load Balancing in Cloud Datacenters. Journal of Internet Technology, Vol. 22, 2021, No. 1, pp. 121–130.

[37] Semmoud, A.—Hakem, M.—Benmammar, B.—Charr, J. C.: Load Balancing in Cloud Computing Environments Based on Adaptive Starvation Threshold. Concurrency and Computation: Practice and Experience, Vol. 32, 2020, No. 11, Art. No. e5652, doi: 10.1002/cpe.5652.

[38] Agarwal, R.—Baghel, N.—Khan, M. A.: Load Balancing in Cloud Computing Using Mutation Based Particle Swarm Optimization. 2020 International Conference on Contemporary Computing and Applications (IC3A), IEEE, 2020, pp. 191–195, doi: 10.1109/IC3A48958.2020.233295.

[39] Muthsamy, G.—Ravi Chandran, S.: Task Scheduling Using Artificial Bee Foraging Optimization for Load Balancing in Cloud Data Centers. Computer Applications in Engineering Education, Vol. 28, 2020, No. 4, pp. 769–778, doi: 10.1002/cae.22236.

[40] Lilhore, U. K.—Simaiya, S.—Maheshwari, S.—Manhar, A.—Kumar, S.: Cloud Performance Evaluation: Hybrid Load Balancing Model Based on Modified Particle Swarm Optimization and Improved Metaheuristic Firefly Algorithms. International Journal of Advanced Science and Technology, Vol. 29, 2020, No. 5, pp. 12315–12331.

[41] Mishra, K.—Majhi, S. K.: A Binary Bird Swarm Optimization Based Load Balancing Algorithm for Cloud Computing Environment. Open Computer Science, Vol. 11, 2021, No. 1, pp. 146–160, doi: 10.1515/comp-2020-0215.

[42] Neelima, P.—Reddy, A. R. M.: An Efficient Load Balancing System Using Adaptive Dragonfly Algorithm in Cloud Computing. Cluster Computing, Vol. 23, 2020, No. 4, pp. 2891–2899, doi: 10.1007/s10586-020-03054-w.

[43] Rafieyan, E.—Khorsand, R.—Ramezanpour, M.: An Adaptive Scheduling Approach Based on Integrated Best-Worst and VIKOR for Cloud Computing. Computers and Industrial Engineering, Vol. 140, 2020, Art. No. 106272, doi: 10.1016/j.cie.2020.106272.

[44] Attiya, I.—Abd Elaziz, M.—Xiong, S.: Job Scheduling in Cloud Computing Using a Modified Harris Hawks Optimization and Simulated Annealing Algorithm. Computational Intelligence and Neuroscience, Vol. 2020, 2020, Art. No. 3504642, doi: 10.1155/2020/3504642.

[45] Sharma, M.—Garg, R.: HIGA: Harmony-Inspired Genetic Algorithm for Rack-Aware Energy-Efficient Task Scheduling in Cloud Data Centers. Engineering Science and Technology, an International Journal, Vol. 23, 2020, No. 1, pp. 211–224, doi: 10.1016/j.jestch.2019.03.009.

[46] Jena, U. K.—Das, P. K.—Kabat, M. R.: Hybridization of Meta-Heuristic Algorithm for Load Balancing in Cloud Computing Environment. Journal of King Saud University – Computer and Information Sciences, Vol. 34, 2020, No. 6, pp. 2332–2342, doi: 10.1016/j.jksuci.2020.01.012.

[47] Devaraj, A. F. S.—Elhoseny, M.—Dhanasekaran, S.—Lydia, E. L.—

Shankar, K.: Hybridization of Firefly and Improved Multi-Objective Particle Swarm Optimization Algorithm for Energy Efficient Load Balancing in Cloud Computing Environments. Journal of Parallel and Distributed Computing, Vol. 142, 2020, pp. 36–45, doi: 10.1016/j.jpdc.2020.03.022.

[48] Ebadifard, F.—Babamir, S. M.—Barani, S.: A Dynamic Task Scheduling Algorithm Improved by Load Balancing in Cloud Computing. 2020 6th International Conference on Web Research (ICWR), IEEE, 2020, pp. 177–183, doi: 10.1109/ICWR49608.2020.9122287.

[49] Kaur, G.: Framework for Resource Management in Cloud Computing. In: Senjyu, T., Mahalle, P. N., Perumal, T., Joshi, A. (Eds.): Information and Communication Technology for Intelligent Systems (ICTIS 2020). Springer, Singapore, Smart Innovation, Systems and Technologies, Vol. 196, 2020, pp. 25–32, doi: 10.1007/978-981-15-7062-9_3.

[50] Harris, C. R.—Millman, K. J.—van der Walt, S. J.—Gommers, R.—Virtanen, P. et al.: Array Programming with NumPy. Nature, Vol. 585, 2020, No. 7825, pp. 357–362, doi: 10.1038/s41586-020-2649-2.

[51] Pedregosa, F.—Varoquaux, G.—Gramfort, A.—Michel, V.—Thirion, B. et al.: Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research, Vol. 12, 2011, Art. No. 85.

[52] Chang, C. C.—Lin, C. J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology (TIST), Vol. 2, 2011, No. 3, Art. No. 27, doi: 10.1145/1961189.1961199.

[53] Calheiros, R. N.—Ranjan, R.—Beloglazov, A.—De Rose, C. A. F.—Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience, Vol. 41, 2011, No. 1, pp. 23–50, doi: 10.1002/spe.995.

[54] Cingolani, P.: JSwarm-PSO. `http://jswarm-pso.sourceforge.net`.

[55] Ibrahim, M.—Nabi, S.—Baz, A.—Naveed, N.—Alhakami, H.: Towards a Task and Resource Aware Task Scheduling in Cloud Computing: An Experimental Comparative Evaluation. International Journal of Networked and Distributed Computing, Vol. 8, 2020, No. 3, pp. 131–138, doi: 10.2991/ijndc.k.200515.003.

[56] Zheng, Z.—Xie, K.—He, S.—Deng, J.: A Multi-Objective Optimization Scheduling Method Based on the Improved Differential Evolution Algorithm in Cloud Computing. In: Sun, X., Chao, H. C., You, X., Bertino, E. (Eds.): Cloud Computing and Security (ICCCS 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10602, 2017, pp. 226–238, doi: 10.1007/978-3-319-68505-2_20.

[57] Khorsand, R.—Ghobaei-Arani, M.—Ramezanpour, M.: A Self-Learning Fuzzy Approach for Proactive Resource Provisioning in Cloud Environment. Software: Practice and Experience, Vol. 49, 2019, No. 11, pp. 1618–1642, doi: 10.1002/spe.2737.

[58] Saeedi, S.—Khorsand, R.—Bidgoli, S. G.—Ramezanpour, M.: Improved Many-Objective Particle Swarm Optimization Algorithm for Scientific Workflow Scheduling in Cloud Computing. Computers and Industrial Engineering, Vol. 147, 2020, Art. No. 106649, doi: 10.1016/j.cie.2020.106649.

[59] Li, K.—Xu, G.—Zhao, G.—Dong, Y.—Wang, D.: Cloud Task Scheduling

Based on Load Balancing Ant Colony Optimization. 2011 Sixth Annual Chinagrid Conference, IEEE, 2011, pp. 3–9, doi: 10.1109/ChinaGrid.2011.17.

[60] Milan, S. T.—Rajabion, L.—Ranjbar, H.—Navimipour, N. J.: Nature Inspired Meta-Heuristic Algorithms for Solving the Load-Balancing Problem in Cloud Environments. Computers and Operations Research, Vol. 110, 2019, pp. 159–187, doi: 10.1016/j.cor.2019.05.022.

[61] Mapetu, J. P. B.—Chen, Z.—Kong, L.: Low-Time Complexity and Low-Cost Binary Particle Swarm Optimization Algorithm for Task Scheduling and Load Balancing in Cloud Computing. Applied Intelligence, Vol. 49, 2019, No. 9, pp. 3308–3330, doi: 10.1007/s10489-019-01448-x.

[62] Lin, C. C.—Chin, H. H.—Deng, D. J.: Dynamic Multiservice Load Balancing in Cloud-Based Multimedia System. IEEE Systems Journal, Vol. 8, 2014, No. 1, pp. 225–234, doi: 10.1109/JSYST.2013.2256320.

[63] Çığşar, B.—Ünal, D.: Comparison of Data Mining Classification Algorithms Determining the Default Risk. Scientific Programming, Vol. 2019, 2019, Art. No. 8706505, doi: 10.1155/2019/8706505.

Muhammad Adil received his Master degree in computer science from the Virtual University of Pakistan, in 2007. He currently works as Product Manager at the Aspose Pty Ltd., a development software company offering numerous award-winning Cloud APIs.

Said Nabi is currently serving as Instructor of IT and Computer Science at the Department of Computer Science and Information Technology, Virtual University of Pakistan, Rawalpindi Campus. He has completed his Ph.D. in computer science at the Capital University of Science and Technology (CUST). He served as Software Engineer and Developer at Esided Solutions (US Based Company) and also worked as Free Lance Software Developer in PHP and MySQL.

Summair Raza has completed his Ph.D. from the National University of Science and Technology (NUST) in 2018. Recently he is working as Assistant Professor in Computer Science Department in Virtual University of Pakistan. He has published various research articles in national/international level journals and conferences. His research interests include feature selection, rough set theory, software architecture and non-functional requirements.