IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2020, accepted September 8, 2020, date of publication September 14, 2020, date of current version October 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024113

Modeling an Optimized Approach for Load Balancing in Cloud

MUHAMMAD JUNAID[1], ADNAN SOHAIL[1], RAO NAVEED BIN RAIS[2], ADEEL AHMED[3], (Graduate Student Member, IEEE), OSMAN KHALID[4], IMRAN ALI KHAN[4], SYED SAJID HUSSAIN[4], AND NAVEED EJAZ[1]

[1]Department of Computing, Iqra University, Islamabad 46000, Pakistan
[2]Department of Electrical and Computer Engineering, College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates
[3]Department of Computer Science, Quaid-i-Azam University, Islamabad 45320, Pakistan
[4]Department of Computer Science, COMSATS University Islamabad, Abbottabad 22060, Pakistan

Corresponding author: Rao Naveed Bin Rais (r.rais@ajman.ac.ae)

This work was supported in part by Ajman University, United Arab Emirates, through the Deanship of Graduate Studies and Research (DGSR).

ABSTRACT Despite significant infrastructure improvements, cloud computing still faces numerous challenges in terms of load balancing. Several techniques have been applied in the literature to improve load balancing efficiency. Recent research manifested that load balancing techniques based on metaheuristics provide better solutions for proper scheduling and allocation of resources in the cloud. However, most of the existing approaches consider only a single or few QoS metrics and ignore many important factors. The performance efficiency of these approaches is further enhanced by merging with machine learning techniques. These approaches combine the relative benefits of load balancing algorithm backed up by powerful machine learning models such as Support Vector Machines (SVM). In the cloud, data exists in huge volume and variety that requires extensive computations for its accessibility, and hence performance efficiency is a major concern. To address such concerns, we propose a load balancing algorithm, namely, Data Files Type Formatting (DFTF) that utilizes a modified version of Cat Swarm Optimization (CSO) along with SVM. First, the proposed system classifies data in the cloud from diverse sources into various types, such as text, images, video, and audio using one to many types of SVM classifiers. Then, the data is input to the modified load balancing algorithm CSO that efficiently distributes the load on VMs. Simulation results compared to existing approaches showed an improved performance in terms of throughput (7%), the response time (8.2%), migration time (13%), energy consumption (8.5%), optimization time (9.7%), overhead time (6.2%), SLA violation (8.9%), and average execution time (9%). These results outperformed some of the existing baselines used in this research such as CBSMKC, FSALB, PSO-BOOST, IACSO-SVM, CSO-DA, and GA-ACO.

INDEX TERMS Classification, cloud, SVM, load balancing, metaheuristics, virtual machine.

I. INTRODUCTION

Over the years, an increase in online applications has resulted in huge volumes of data accumulated daily. Generally, the data is classified into different types, such as audio, video, image, and text. Despite the significant evolution of clouding computing to handle such diverse data, still it faces numerous challenges in real-time processing and load balancing of resources employed to process mega volumes of data.

In the past few years, several load balancing approaches have been developed for cloud computing, such as [1]–[5]. For instance, in [1], the authors have applied the Bin-packing algorithm for multi capacity Bin-packing to achieve task waiting time and degree of imbalance on cloud resources. In a similar work [2], the authors used the Bin-packing algorithm for cost-aware and fragmentation enabled consolidation of tasks to achieve minimum energy consumption. In a work by [3], the authors used a dynamic clustering algorithm to achieve throughput and execution time. A study by [4] applied a dynamic real clustering algorithm for achieving geographical load balancing in the cloud that results in better throughput and response time. In [5], the authors applied adaptive load balancing to achieve optimal resource provisioning resulting in better resource utilization and throughput.

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Shahid.

However, most of the traditional load balancing approaches suffer from high computational cost, energy consumption, several overheads, scalability, and deadline constraints.

In recent years, the research trend has shifted towards metaheuristics-based approaches for load balancing, as these techniques are better in addressing flexibility, multimodal optimization, efficient randomization, discontinuous problems through intensification (exploitation), and diversification (exploration) respectively. The authors in [6] presented a metaheuristics approach for load balancing using modified PSO in which they minimized tasks overhead and maximize resource utilization over varying VMs and tasks. In [7], the authors combined ACO and PSO in a hybrid metaheuristics load balancer ACOPS which uses the historical information to predict future workload of the VMs in the cloud. This approach helps in reducing computational time while keeping optimum load balancing among VMs and tasks. This metaheuristic approach helps in finding a local and global best position in the solutions with fast convergence and hence performing better than many heuristic approaches. Similarly, In [8], the authors employed SVM in cloud load balancing metaheuristics ACO to achieve better throughput, SLA, migration, overhead, and optimization but lacks other critical factors such as energy consumption, response time and execution time. Similarly, most of the existing metaheuristic techniques are covering either one or a few optimization parameters but ignoring other critical factors that can play a pivotal role in achieving multi-factor optimization [9]. Moreover, the issues faced in the cloud due to load balancing can be further minimized with the combination of metaheuristics and data mining techniques to solve complex optimization problems more efficiently [10]–[13]. Nowadays, Cloud Data Mining (CDM) is gaining popularity in which machine learning models, such as supervised machine learning are integrated with cloud load balancing approaches that result in new efficient algorithms [14], [15]. Similarly, the classification of multiple file types in the cloud can achieve an improved load balancing with increased accuracy due to the pre-assignment and categorization of data for virtual machines with different resources. For instance, the audio classification exists in various forms such as noise, speech, silence, and music, etc, and can achieve performance efficiency using deep learning algorithms such as Convolutional Neural Network (CNN) [16]. Similarly, video datasets need proper categorization and automatic classification for quick retrieval and indexing. This helps in understanding the semantic gap to minimize computational complexity. Integrated metaheuristic algorithms such as ACO, ABC, PSO, etc. are used in several ways to attain more accuracy in video classification using classifiers such as SVM, KNN, and NN [17]–[19]. It has been observed that a huge increase in text documents is making the extraction process quite complex. Text clustering is used for text mining to categorize the text documents but cannot perform text feature selection [20]. The metaheuristic algorithms and classifiers such as GA, HS, PSO, NN, and SVM are widely used for text classifications in high dimension space providing high accuracies [21]–[24]. Image classification involves the selection of image feature subsets from large feature space. Selecting optimum features is a complex process in image classification for load balancing that is solved by several hybrid metaheuristic techniques such as CSO, GA, ACO, PSO, with SVM, NN, K-NN [25]–[28]. Despite several advantages, the aforementioned approaches have certain deficiencies and therefore, there is still a need for multi-factor optimal solutions for load balancing.

Our proposed work focuses on the development of a new load balancing algorithm named Data Files Type Formatting (DFTF) that combines SVM (a machine learning classifier) with modified Cat Swarm Optimization (CSO) algorithm (a scheduling algorithm). The proposed DFTF algorithm considers multi-factor QoS metrics, such as energy consumption, response time, SLA violations, migration time, optimization time, execution time, throughput time, and overhead time as performance evaluation measures. In this work, SVM is applied using one-to-many classifications for generating the data class over a set of file formats, such as audio, video, text, and images in the cloud environment. The classification process can easily reduce such complexities while performing offline preprocessing and make the data available in the processed form [8]. This refined form of data when applied to load balancing can significantly improve scheduling using QoS parameters [29].

Original CSO is more suitable for small population size with a minimum number of iterations and hence not providing good solutions in a situation where the processing involves a large number of complex tasks [30], [31]. This drawback eventually leads CSO to fall into local optimum which takes more iterations in finding solutions space and hence make CSO computationally complex [32]. Therefore, we have modified the original CSO by introducing a new grouping phase process that takes the data files into four groups: audio, video, image, and text taking from SVM keeping in view the properties associated with each group. In doing so, the population of cats in the sub-groups is sorted and later in the stage, the best fitness value of the cat in local best solution is selected. Thereby, integrating the two approaches SVM and CSO into a merged one addresses their individual limitations and reinforces their combined benefits into a single combined model. Earlier data type approaches such as AWS and PostgreSQL focus only on data types classification but not on file format types [33]. However, the proposed approach is using file format types for classification in the cloud environment and then uses the resultant data class into load balancing algorithm modified CSO for load balancing. This combination has outperformed some sate of the art metaheuristic load balancing algorithms used in this study such as CBS-MKC [34], FSALB [35], PSO-BOOST [36], IACSO-SVM [37], CSO-DA [38] and GA-ACO [39].

The main objective of this research is to propose a new optimized metaheuristic algorithm DFTF in a cloud that performs classification and load balancing effectively.

This optimization model addresses the limitations of the earlier load balancing approaches by its multi-factor approach. Further, the contributions of this paper include:

- A new algorithm DFTF is developed based on SVM and modified CSO that provides better-optimized load balancing in the cloud environment.
- The classification of data file formats into audio, video, image, and text is performed in a cloud environment that shows an improved classification accuracy in confusion matrix such as accuracy, precision, recall, and F-measure over state-of-the-art classifiers helping in decreasing computational complexities later in the scheduling phase.
- The proposed DFTF model has provided improved results for energy consumption, response time, SLA violations, migration time, optimization time, execution time, throughput time, and overhead time as performance evaluation measures.

The rest of this paper is organized as follows. A literature review is presented in Section 2, the proposed methodology is discussed in Section 3, the experimental setup is described in Section 4, results and analysis are discussed in Section 5 and conclusions are presented in Section 6.

II. LITERATURE REVIEW

The load balancing algorithms are classified as dynamic, static, or hybrid and it depends on the machine state. They are also known as allocation and scheduling algorithms based on the features used during load balancing. Further, they are categorized as Cloud Data mining load balancing, VM load balancing, CPU based load balancing, Task-based load balancing, Server-based load balancing, Network-based load balancing, and Standard Cloud load balancing based on their combination. Numerous studies discussed the limitations of the load balancing algorithms for proposing more effective methods. The study lacks discussion of some essential QoS metrics, such as migration duration, migration expense, service quality breach, task failure rate, algorithmic efficiency, percentage of load balancing measures, and level of balance. The algorithm for load balancing must improve responsiveness, implementation cost, implementation time, throughput, fault sensitivity, migration duration, makespan, resource throughput, and usability. At the same time, energy consumption, carbon pollution, relocation costs, energy efficiency, and SLA needs more consideration [40]–[42]. It has been observed that in reducing the efficiency of the load balancing, algorithm complexity is not given much consideration. The studies also concluded that several issues remain a huge challenge in the load balancing that can be traversed in the future by implementing an adequate, effective, and robust load balancing algorithm. The decline of these dimensions' leads to poor QoS at Cloud Service Centers and a decreased economy for CSP. However, keeping QoS and economics into consideration, delivering optimized multi-factor QoS based solutions is becoming a major challenge for CSPs. Some of these challenges are being discussed which mainly rotates around our developed QoS metrics in the below sections.

Cloud computing is making significant contributions in extracting useful information when combined with data mining techniques. This combination called cloud data mining (CDN) has made easy information retrieval from huge volume and variety of data with the help of load balancing approaches. Similarly, several real-time cloud data mining frameworks, algorithms, and services are available that provide information through a number of applications [43]. These real-time cloud data mining and load balancing applications are VM tasks classification for load balancing, feature extraction, anomaly, and intrusion detection, open shop scheduling, attribute importance, spatial classifications, data analysis and satellite imagery [44], spectral and statistical data analysis [45], gene expression data mining and bioinformatics [46], geo-spatial analysis and geo-informatics, large-scale mining in big data and web mining [47], machine-learning applications [48], high-dimensional data mining [49], highly diversified and dense data mining in rule mining [46], the security of data in the cloud, clustering, datacenters resources optimization in the cloud, noise removal, reactive power problem, face recognition, biomedical image processing, teaching based learning, manufacturing design, water resource problem and routing optimization.

This research is mainly focused on classification, specifically the combination of classifier with a load balancer in the cloud. So, most of the presented information rotates around classification and load balancing. In classification, data is assigned to appropriate classes using supervised machine learning techniques. The classification consists of many techniques such as SVM, decision tree, Bayesian classifier, NN, belief networks, and Rule-based classifiers. Cloud data mining classification techniques including K-NN, Point data, NB, GA, and their hybrids such as NB with SVM, GA with SVM, ACO with SVM, ACO with NN, GA with K-NN, etc. These combinations help in getting the highest accuracies in classification and further reducing computation complexities in load balancing over the number of applications.

In [50], the authors proposed a hybrid metaheuristic algorithm called WOA-AEFS. They solved the resource scheduling problem in cloud computing. This study has two scheduling approaches that consider makespan and cost. The algorithm outperformed other metaheuristic algorithms such as original BAT and PSO but did not consider factors such as execution time and performance efficiency. Extended BAT Algorithm (EBA) is suggested by [51] that modifies three benchmark functions such as Ackley, Hyperellipsoid, and Rosenbrock resulting in better performance of searching optimum solutions, fitness function, and convergence rate. The algorithm outperformed other metaheuristics, such as MMBO, and MBO-FS in the same cloud but computational complexity is a major concern. A study by [52] suggested Gravitational Search Algorithm (GSA) for load balancing in the cloud where it proved strong convergence

over a set of iterations. The drawback of this algorithm is computation-intensive that effects its scalability. A study by [53] proposed hybrid IRRO-CSO which is inspired by the perception of information flow in raven social behavior among members in searching food while CSO is based on chicken behavior during a search of food. The algorithm has been validated against CEC 2017 benchmark functions resulting in showing better performance over BAT, PSO, and CSO. However, the performance needs to be checked on real-time large datasets that improve the execution time.

In [82], modified Heterogeneous Earliest Finish Time (MHEFT) is proposed for dynamic load balancing in the cloud resulting. The algorithm works well under a smaller number of tasks and did not consider other QoS metrics for performance evaluations. CEGA is a genetic inspired balancing algorithm designed to meet deadline constraints while reducing the execution time of the tasks [83]. Results of the CEGA have shown better performance on the same workflows but the algorithm suffers exponential time complexity. A variation of CSO called Improved Cat Swarm Optimization (ICSO) is proposed by [79]. Here, the first modification is improving the tracing mode with changing position and velocity equations. Similarly, the other modification is to make changes in such a way that local optima are prevented. However, the algorithm is only tested on benchmark functions with fewer tasks whereas, the number of QoS metrics is not considered. In a study by [84], authors developed a hybrid metaheuristic algorithm HBMMO for workflow scheduling in cloud computing for improving throughput in the cloud. The algorithm takes a multi-objective approach such as quantization of execution cost, throughput, and makespan. Despite many factors discussed in the study; energy consumption is not considered. In [85], Weighted Wavelet SVM (WW-SVM) is suggested for estimating load sequences in a data center using a cloud computing environment. For parameter selection and optimization, PSO is used to make a final prediction. The proposed algorithm outperforms other baselines in terms of execution time, throughput, and error prediction. The algorithm considers only prediction and accuracy while simple multi objectives are not discussed.

An improved SLA violation for load balancing in the cloud with the objective of minimum resource wastages is discussed by [86]. The algorithm used optimum resources in which there is less failure rate of fulfilling tasks and maintaining low energy usage and least SLA violations up to 17%. The algorithm did not consider execution time and numerous scientific evaluations. Sharma *et al.* proposed SLA agile-based VM to reduce response time [87]. This research used ghost VM to reduce the VM creation time by approximately 12%. Static workloads are used, and performance metrics are not discussed. In [88], comprehensive comparisons of SLA violations in a cloud environment are performed for five metaheuristics algorithms in which QoS constraints and penalty costs are considered. Experiments on benchmark functions have shown the better performance of each metaheuristic in best, worst, average, and SD scenarios. However, multi-objective QoS metrics are not considered.

A dynamic VM migration algorithm MMA is suggested by [89] suitable for High-Performance Computing (HPC). MMA algorithm minimizes the load on overloaded machines and reduced communication costs. However, saturation can cause performance degradation and computational complexity. A study by [90] discussed a VM migration strategy that provides better scalability over other strategies. In this approach, a composite scoring function is used and find the host that has workload handling capability. At this stage, migration takes place and load is transferred to it. However, the high computation cost, overhead, and multi-objective approach are lacking. An improvement in reduced overhead using a metaheuristic automatic power-aware algorithm in VANETs is presented by [91]. This research uses PSO that achieves energy-efficient communication. However, performance degradation of 8% is observed and computation cost is high. Biological Inspired Self Organized Autonomous Routing Protocol (BIOSARP) is proposed by [92] for the earliest searching of a neighbor using an optimal decision by ACO. This helps in reducing overload to a significant extent but the algorithm has a relatively higher overhead cost.

A study by [93] discussed load balancing in multi-core clusters using frequent data mining in the cloud. In their, work SDFEM is proposed that provides high mining performance in the large complex data analytic real-time applications. They used a hybrid approach of OpenMP and MPI and tested their implementation on 12 core shared memory nodes. The results have shown a remarkable increase in performance that is much faster and reliable. However, this combination has some complexities especially computational and memory complexities. In [49], the authors suggested a pattern mining load balancing technique for high dimension data PaMPa-HD based on Map-reduce. The algorithm performed well in terms of robustness and execution time due to its inherent properties of better mining patterns and the least amount of transactions. However, there are a large number of items per transaction. The authors in [94] presented metaheuristic EELBF Firefly load balancing algorithms in the cloud in which throughput and response time are focused. This algorithm besides finding relational mining models provides better energy consumption by balancing workload in multiple VMs (considering less loaded and high loaded VMs). The algorithm has been implemented in CloudSim 4.0 and compared with ACO, HBB, and WRRLB, and overall better performance is observed. A study by [95] presented an energy-efficient load balancing algorithm that uses the combination of BWM and TOPSIS methodology for a multi-objective mining approach. The selection of most appropriate cloud scheduling solutions is performed in two steps in which initially a decision criterion is defined followed by BWM for weights assigning and then TOPSIS is applied

TABLE 1. Cloud data mining using load balancing algorithms.

Technique	Year	Algorithm	Pros	Cons
Association Rule Mining (Hadoop+Java) [54]	2016	EFP	Parallel implementation and accuracy, High scalability	Noise handling needs improvement
Scheduling [55]	2018	Prediction based GA	Optimal VM placement, Low energy consumption, Increased CPU utilization, Better accuracy	Fewer resources considered, Few VMs and PMs considered
Association Rule Mining (Cloud Storage) [56]	2016	FPMA	Efficient storage use in the cloud	Low scalability
Scheduling [57]	2019	Modified PSO	Effective Bandwidth and memory utilization	Few tasks considered, Lacking multi-objective
Association Rule Mining (Cloud data & Parallel Compute) [58]	2017	HUDS	Excellent noise handling and accurate classification	Low scalability
Scheduling [59]	2019	CSO for OSSP	Better performance and execution time	Less number of tasks are used
Classification (image) [62]	2017	Proposed LBA	Improved influential parameters, Good exploration and exploitation rate, High convergence rate.	Reduces makespan time, did not consider task priority, Deadline constraint
Association Rule Mining (Hadoop + HDFS) [61]	2014	PLFPG	Scalability, Extensibility, Efficiency, and Fault-tolerant	Medium scalability
Scheduling [62]	2018	CSO+PSO	Minimized makespan and Execution time, Achieving deadline constraints	Real-world applications are missing
Classification (Cloud storage & Computer) [47]	2013	Cloud SVM	Excellent concurrency, High level of parallelism, High scalability	Noise handling needs improvement
Scheduling [63]	2017	Modified CSO	Better performance and Summary generation	Computational time and controlling parameters
Classification (Audio) [64]	2017	Deep Learning Methods (DLM)	Deep belief networks are used for evaluating audio data classification, Learned features have shown better performance in classification	This approach did not consider other factors like cost and time of convergence
Classification (Hadoop Map reduce) [48]	2014	Hybrid of NB+SVM	High accuracy, Efficiency, and throughput	Computation complexity
Scheduling [65]	2020	DMOOTC	Minimum execution time and cost, Higher significant reliability	Smaller workloads are used, and scalability is a concern
Classification (Cloud parallel model) [66]	2015	K-NN based	High security and efficiency	Very low noise handling and scalability
Scheduling [71]	2016	OTB-CSO	Better makespan and execution time, Improve computational performance	Multi-objective approach is required, Scalability issue
Classification (Apache Spark) [68]	2015	PDCT in cloud	High scalability and good noise handling	Computation costs
Scheduling [69]	2015	MOACO	Minimize makespan, Budgetary cost, and execution time	Trapping in local minima
Classification (Text) [70]	2016	ICSO	Effective in text classification of documents, Low convergence rate	Need to improve the tracing mode so that high throughput may be achieved
Classification (Cloud environment) [71]	2017	EVFDT	High streaming data classification and performance	Low scalability
Scheduling [72]	2019	CSO-LDIW	Fast convergence, Minimum makespan, and efficient task mapping	Network QoS not discussed, Small number of tasks, and scalability issue
Classification (Audio) [73]	2014	Simplified Swarm Optimization	Improved simple metaheuristic algorithm, Only a few particles resulted in an improved accuracy of classification that is 91%.	Search strategy needs to be improved, Only fewer audio features are tested.
Clustering (CloudSim environment) [34]	2013	Dynamic VM Allocation using K-Means	Efficient CPU Utilization and load balancing with High scalability	Lower noise handling and non-parallel approach
Scheduling [75]	2018	CSM-CSOSA	Minimum execution time, Execution cost and better Scalability	Performance on complex datasets and workload needs to be measured
Classification (Video) [76]	2017	Micro Learning video Classification	Efficient and effective to classify videos, Reduced computational time, Good performance in Precession recall analysis	Lack of subjective analysis and quantification impact, Reduced better fit the interest of the students
Clustering (MongoDB Platform and Google Cloud) [77]-[78]	2015	DenStream and D-Stream	High distribution, Parallelism, and scalability	Higher implementation costs
Scheduling [79]	2018	ICSO	Better CEC function results, Excellent clustering	Many QoS metrics are not discussed especially execution and throughput
Clustering (Cloud Map-reduce) [80]	2016	Genetic K-Means	Accuracy of clustering big data	Lower scalability
Clustering (Google Cloud) [81]	2018	CluSTree	Efficiency and accuracy	Medium scalability and low noise handling

to measure the performance of each alternative. Experiments have shown that the proposed algorithm has attained better results in terms of makespan, energy consumption, and VM utilization over other baselines. However, this study did

not consider large scale datacenters which provide scalability and reliability of the proposed solution over a larger number of tasks. In [96], the authors discussed the PSO based load balancing algorithm used for resource allocations in cloud computing. The algorithm finds task initiation overload on VMs by optimized migration transfers to other VMs. As a result, the algorithm achieved reduced execution time and transfer time. However, this algorithm is only considering a few tasks based on a single factor thereby not addressing scalability issues. A study by [97] presented a new adaptive integrated approach based on best-worst decision making and the ranking method called VIKOR which is used to define tasks' priorities. This algorithm uses a compromised approach in which group benefits are maximized over individual losses. The algorithm provides better reliability by keeping all VMs in the process during runtime. Further, the algorithm achieves better throughput, reduced makespan, improved waiting time, more virtual machine (VM) utilization, and less VM usage cost when compared with other baselines. However, a maximum of 1000 tasks is considered for various QoS performance metrics which means that scalability may be the issue when tasks are significantly increased along with VMs.

It is observed from the number of studies presented here that no comprehensive multi-factor approach is adopted that optimizes the QoS metrics without effecting the quality solution.

III. PROPOSED METHODOLOGY

We combined our approach using SVM and CSO to make a hybrid model called DFTF with the objective of improving the load balancing and performance in the cloud environment. The architecture of the proposed DFTF approach is shown in Figure 1. This architecture is divided into two main modules: 'Data Classification based on SVM' and 'Load Balancing using CSO'. The input to the data classification module is the collection of diverse data in the form of video, text, audio, and images, which are stored in the cloud environment. The classification module takes the input data randomly and then performs the classification on these data using polynomial SVM. The output of this algorithm is in the form of the partitioned data class. The second module performs load balancing using Cat Swarm Optimization (CSO). The performance analysis of the proposed model is then performed to achieve an efficient load balancing by considering the parameters such as execution time, number of migrations, optimization time, throughput time, and overhead time. The various tools used in this research are CloudSim 4.0 and Java environment.

Algorithm 1 describes the process of the proposed model called DFTF. In this algorithm, Lines 1 to 11 performed data categorization that first classifies the type of data and then classifies the type of VMs using SVM and assigned it to the particular class. Lines 12 to 32 performed load balancing using CSO and then output the schedule data.

A. DATA CLASSIFICATION BASED ON SUPPORT VECTOR MACHINE

We collected the data from different cloud sources and then preprocessed the data to transform it as per our model requirements. The data format of the collected data from the cloud is comprised of video, audio, text, and images. These data sets are diverse and are of different sizes. In the proposed model, at first, the SVM classifier determines the type of data (audio, video, text, image) based on features and then classify the data by assigning it to a particular class.

We have divided the VMs into four types of sets, such as AudioVM, VideoVM, TextVM, and ImageVM based on input data. Each set of VM has different processing and storage resources in a cloud environment. More precisely, each machine (VM) is assigned a task based on task requirements. For example, video tasks require 1000 floating-point operations and 16GB memory, audio tasks require 800 floating-point operations and 12GB memory, image tasks require 800 floating-point operations, and 8GB memory, textual tasks require 400 floating-point operations and 4GB memory. After that, the SVM classifier identifies the set of VM types such as VideoVM, AudioVM, TextVM, ImageVM, based on the requirements, size, and features of the tasks. Here the respective VM is assigned concerning each task. Hence, SVM intends to classify data and match it to the most suitable class type and VM type.

For video data classification, we extracted feature vectors of sequences of 40 frames extracted from four different video classes, where we have a 40×4096 matrix, where each row refers to features of one frame (one frame per row), so we classified videos between these four different classes. We preprocess a new video to limit its number of frames and then extract features from this video to classify it. Assume that we have four video classes (ci, $i = 1, .., 4$). Each video has 40 ($n = 1, .., 40$) frames and from each frame we extracted 4096 features ($[1 \times 4096]$). Since each frame has enough information to predict the video class (ci) so we used 40 frames from each video as training/test samples, which creates an input matrix of $[160 \times 4096]$ dimensions, with 160 samples and each sample have 4096 features. Additionally, we have created an output vector $[160 \times 1]$ that contains the label of each class $ci = i$, where $i = 1 \ldots, 4$.

For audio data classification, four feature sets of audios are evaluated for identifying five kinds of audio classes: classical music, popular music, crowd noise, speech, and simple noise. The feature sets include low-level signal properties, mel-frequency spectral coefficients [98], and two new sets based on perceptual models of hearing. For image classification, we have considered 256×256 pixels (total 65,536 pixels). We used each pixel as a feature in the SVM classifier.

For text classification, there are text documents of about 6GB which are extracted in the form of unstructured text. We performed stemming and stop word removal and extracted the words in the form of features. We then used these features for text data classification using SVM.

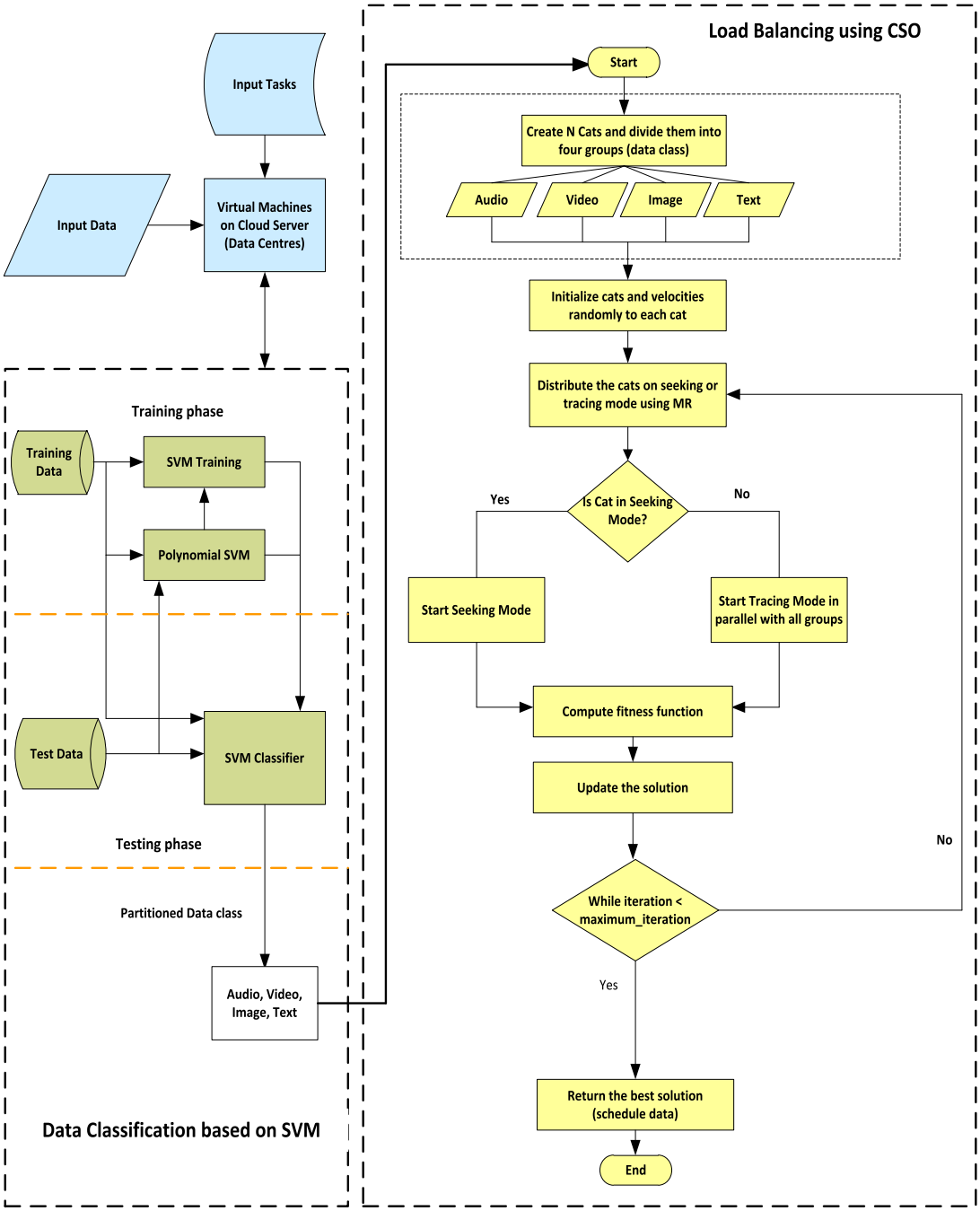

FIGURE 1. Proposed architecture of DFTF.

SVM works on the principle of linear classification with a special type of rule that generates classes with effective performance and is based on the quality of classification. Kernel trick can be used in the construction of a special kind of non-linear method using SVM. There are two types of classic kernel functions that are used in SVMs, one of them is the radial basis function kernel and the other is a polynomial kernel. where, u_i is used for support vector, $\propto_i$ is represented as Lagrange multiplier and u_j is known as the label of membership class $(+1, -1)$ where $n = 1, 2, 3 \ldots .. N$.

Equation (1) shows the polynomial function,

$$POLY\ (u, v) = \left(\left(u^k v + 1\right)\right)^s, \tag{1}$$

where 's' is the polynomial degree.

The polynomial kernel function is used with SVMs and other kernel models representing the similarity between features over the polynomials of the original variables. A polynomial kernel is defined as:

$$K\ (x, x_i) = 1 + \sum (xxx_i)^d\ . \tag{2}$$

Here, d=1, this confirms to the linear kernel.

Algorithm 1 DFTF

Input: video, text, audio, image, N, number of virtual machines (VM), number of cats, max_iterations, SMP (seeking memory pool) = 5 to 10, MR is the Mixture ratio, Sz: Size of the population, maximum_iter
Output: Data class, Scheduled data
1:**for** data classification **do**
2: **for each** $P(u, v)$ **do**
3: Evaluate = SVM
4: **for each** Classification accuracy $\neq$ 100 **do**
5: Evaluate data accuracy
6: **if** max_ iterations $\neq$ N **then**
7: perform data categorization and VM categorization
8: **end if**
9: **end for**
10: **end for**
11: **end for**
12: **for** load balancing **do**
13. Create N cats and divide them into four groups G, that is G={audio, video, text, image}.
14. Randomly initialize velocities to each cat belongs to group G.
15. Evaluate initial fitness function F_i
16. C_{sz} = Create C_{pop} (sz, F_i) //Create cat population
// Distribute the cats in seeking or tracing mode
17. **while** k $\leq$ maximum_iter **do**
18. **for each** i = 1 to Sz do
19. **if** C[i] = $Seek_m$ **then**
20. Sol = Apply $Seek_m$ (C_j)
21. **else**
22. Sol = Apply $Trace_m$ (C_j)
23: **end if**
24. F_{best} = Sol $_{best}$
25. **if** C(F,k) detected **then**
26. C_{sz} = create C_{pop} [sz, F]
27. **else**
28. C_{sz} = reset C_{pop} [C_{sz}]
29: **end if**
30. **end for**
31. **end while**
32. **return** F (schedule data) //return best solution
33. Exit.

The output of the classification phase is in the form of classified tasks, thus reducing the computational cost such as to avoid preprocessing of features learning, features extraction, data conversion, data transformation, and data classification at the scheduling phase.

B. LOAD BALANCING USING CAT SWARM OPTIMIZATION

We developed the network of VMs in the form of an undirected weighted graph as shown in Figure 2. The VMs' network can be represented as an undirected graph G = (V, E)

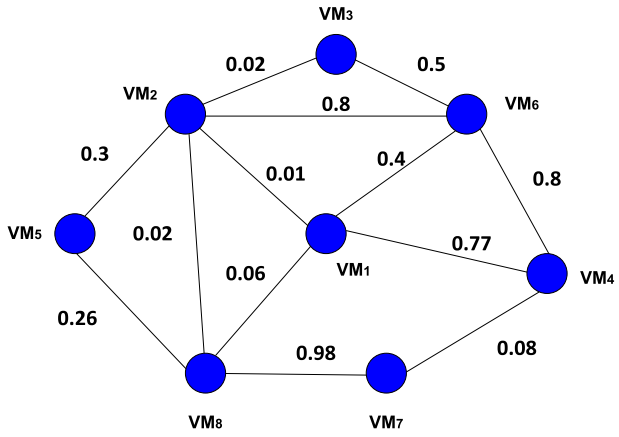

FIGURE 2. The network of Virtual Machines (VMs).

where V represents the virtual machine (VM) or node and E represents the undirected edge having a probability a weight that shows the overload and underload intensity between two nodes. After the data classification, load balancing is performed using CSO. In the load balancing phase, these data are called tasks. Let us assume that:

$$\begin{aligned} VideoVM &= \{VM_1, VM_2, \ldots\ldots, VM_n\}, \\ AudioVM &= \{VM_1, VM_2, \ldots\ldots, VM_n, \\ TextVM &= \{VM_1, VM_2, \ldots\ldots, VM_n\}, \\ ImageVM &= \{VM_1, VM_2, \ldots\ldots, VM_n\} \end{aligned}$$

be the set of virtual machines for video, audio, text, image, respectively. Each set of machines is responsible to execute one task. Each task is executed for a period of maximum iterations and is evaluated using computational cost in the form of time. The mapping of tasks on virtual machines is computed using the SVM, where each machine is assigned a task based on requirements, size, and features of the tasks.

Once the number of tasks and VMs are selected, the scheduling process will be initiated. Initially, N instances are created and split into G groups. CSO takes into consideration the behavior of the cats into two modes that are seeking mode and tracing mode. Swarm algorithms are widely accepted as they adapted the best-obtained solutions for searching the most similar neighbors (nodes). So, in this method, the cat behavior is considered for searching a solution space. Every cat has its position having d-dimensions with different velocities used for each dimension. Every cat is evaluated using fitness function, if the fitness is not equal then compute the probability using equation (3), and by default, the probability value is set to 1. We used the Boolean flag variable to identify whether the cat is in seeking mode or tracing mode. The tracing mode is considered in terms of its fitness function where the position of the cat is changed according to the fitness function. The fitness function of CSO can be obtained with the help of equation (3).

$$P_i = \frac{FS_i - FS_{max}}{FS_{max} - FS_{min}}, \quad where\ 0 < i < j, \tag{3}$$

where P_i shows the probability, value associated with the position of i^{th} Cat. FS_i is the fitness of i^{th} cat, FS_{max} represents the maximum fitness value and FS_{min} represents the minimum fitness value achieved so far.

The tracing mode of the CSO method is described in terms of the movement of cats that is based on the outstanding hunting skills of cats. In tracing mode, the movement of cats is according to their velocities in each dimension and then updating their positions accordingly. The updated positions of cats and velocities are calculated using equations (4) and (5). These equations are:

$$V_{i,d}(t) = V_{i,d} + r_i c_i \left(X_{best,d} - X_{i,d}\right), \quad d = 1, 2, \ldots M, \quad (4)$$

$$X_{i,d(t)} = X_{i,d} + V_{i,d}, \quad (5)$$

Here, various terms are used for the position of the Cats like $X_{best,d}$ is the best position of a cat in d-dimensional space, $X_{i,d}$ is the position of Cat_i, $V_{i,d}$ is the velocity of C_i, r_i is a random value between [0, 1], C_i is the acceleration coefficient that extends the Cats velocity to move into solution space, which is set to 2.0, and t is the iteration number. Mixture Ratio (MR) is used to combine the two modes in the algorithm that is seeking mode and tracing mode and to determine the ratio of Cats in the modes. We have set the control variable MR to 1%-3% that determines the position a Cat which is either seeking or tracing mode. It means that at any instance, 10% to 30% of the Cats are in tracing mode, and the rest of the Cats are in seeking mode. Here local search refers to tracing mode and global search refers to seeking mode. Cats spend most of the time in resting mode (seeking mode), so, the MR value should be a tiny value to show their behavior in the real world. There is a need to put the control check on the velocity of Cat for every dimension value so that velocities must be in the range and have not crossed maximum. This control check is added through inertia weight (w) for which the optimum value must be between [0.4-0.9]. In our case, 0.7 is giving the best results. For the first iteration, it needs to be started at 0.9 and gradually decreased to 0.4. The optimum solution will be available somewhere between the values. The seeking mode of the CSO technique is composed of the following parameters: Seeking Memory Pool (SMP), Seeking Range of selected Dimension (SRD), Counts of Dimension Change (CDC), and Self-Position Consideration (SPC).

We have computed the evaluation of the DFTF model based on time complexity. As our model follows a combined approach that is the combination of SVM and CSO, so we have used different parameters that are specified in evolutionary algorithms as well as classification algorithms. For CSO, these parameters are given in Table 2. For CloudSim, we used the parameter settings as reported in Table 3 These parameter settings are chosen based on the convergence of the DFTF algorithm after conducting several experiments.

From Algorithm 1, the computational time for Lines 1 to 8 is $O(N^3(n))$ and Lines 9 to 13 take constant time O (1). Lines 14 and 15 take $O(N^2)$. From experiments, it is observed that the seeking mode takes less time as compared to tracing mode, therefore, computational complexity from Lines 16 to 26, is $O(N(ks))$ and Line 27 takes time, $O(l)$. Thus, the overall time complexity of the proposed DFTF is $\boldsymbol{O(N^3 + N^3(n + ks + l)}$ that simplifies to $\boldsymbol{O\left(N^3 + N^3.n + N^3.ks + N^3.l\right)}$ after further simplification gives $\boldsymbol{O(N^3\,(1 + 1.n + 1.ks + 1.l)\,)}$ and finally its $O(\boldsymbol{N^3})$.

TABLE 2. Parameter settings used for DFTF.

Parameter	Value
Cat size	100
Population size	30 to 500
SMP	5-10
CDC	80%
SRD	20%
SPC (Boolean value)	0 or 1
Self-recognition coefficients (c)	2.0
W	0.9-0.4 (0.7)
Uniform random number (r)	[0,1]
Maximum iteration	1000
Mixed Ratio (MR)	1%-3%

TABLE 3. Parameter settings used for DFTF in CloudSim.

Datacenter	Various Parameters	Parameter Values
CLOUDLETS	Total Number of Datacenters	2
	Total Number of Hosts in Datacenters	2
	Size of RAM for each Host	2GB
	Total RAM	16 GB
	Capacity Storage	1TB
	Total Storage	
	Data Bandwidth	10GB/s
	Processing Elements Power	1000000 MIPS
	Total Number of Tasks	100000
	Length of each Task	1MB-1GB
	Total File Size as Tasks	48.4 GB
VIRTUAL MACHINE	ID of the Virtual Machine	[1-20]
	VM Monitor	Xen
	VM Policy	Time-shared

TABLE 4. Description about datasets.

Category	Size in quantity
Audio Files datasets	15,000
Video Files datasets	15,000
Images Files datasets	15,000
Text Files datasets	15,000

IV. EXPERIMENTAL SETUP

In this section, various files in the form of datasets are presented including audio, video, text, and images which are taken from UCI repository [99] and other sources. There is a total of 60,000 datasets which are further divided equally as given in Table 4.

TABLE 5. Statistics about training and test sets.

Samples	Size in quantity
Training Audio Files	10500
Testing Audio Files	4500
Training Video Files	10500
Testing Video Files	4500
Training Images Files	10500
Testing Images Files	4500
Training Text Files	10500
Testing Text Files	4500

Further, various dataset files are placed into training and testing mode with a ratio of 70:30, where 70% of data files are training datasets and 30% are testing datasets as mentioned in Table 5.

CloudSim 4.0 [100] as compared to other simulators is widely used in conducting and implementing cloud-related research work. The simulator is providing on-demand resources in virtualization form and has several advantages such as flexibility, performance, and ease of use. A data center is configured with the region, architecture, operating system, VM, memory, storage data transfer cost, and the number of physical hardware units. In our case, we have set 500 and 1000 VMs, respectively, during experiments along with 4096, 8192 MB of RAM and 2 TB of memory. All simulations are performed on Desktop PC comprising of MS Windows 10 Operating System, Intel Quad-Core i7 with 2.6 GHz processor, 12 GB RAM, and 1 TB of HDD.

The algorithms used in this research include CBS-MKC, FSALB, PSO-BOOST, IACSO-SVM, CSO-DA, GA-ACO, and proposed DFTF. There are eight metrics on which DFTF is compared, such as energy consumption, response time, SLA violations, number of migrations, execution time, overhead time, throughput time, and optimization time. There is a total of 60,000 tasks on which evaluations are performed. All algorithms are implemented in CloudSim 4.0 taken from their respective research papers with the same configuration and environmental setting to make the results reliable. Further, the results are statistically verified through analysis of student t-test to check their reliability that eliminates the fact that the values are not by chance.

V. RESULTS AND ANALYSIS

In this section, the proposed algorithm has been divided into two major parts. In the first part, the classification of the file's datasets is done using the SVM classifier in the cloud. In the second part, the output of the SVM is fed into ICSO for load balancing in the cloud environment. To obtain better, fast, and accurate results, we have used the One-vs-All classification approach that initially classifies the files datasets by comparing it with all classes [101]. Overall, the output of the SVM falls into one of the four major data classes such as audio, video, image, and text. The baselines used in this research include CBS-MKC, FSALB, PSO-BOOST, IACSO-SVM, CSO-DA, and GA-ACO.

CBS-MKC used credit-based scheduling with an emphasis on task categorization but lacks a multi-factor approach. FSALB largely focused on reducing communication delays experienced by the machine learning users and hence improved response time but lacks a multi-factor approach, PSO-BOOST considered deadline constraint within a limited number of tasks, VMs and has shown improvements on few parameters. IACSO-SVM worked on the classification accuracy of the limited number of tasks and datasets. CSO-DA emphasized response time, number of migrations, and execution time on fewer tasks and VMs. GA-ACO improved completion time, response time, and throughput under limited resources. However, the proposed algorithm DFTF not only addresses these limitations but also adopted a multi-factor approach to solving.

Similarly, there are deep learning approaches which are producing better results than traditional algorithms, but they take more time in training with a large number of datasets. Therefore, as per the requirement of our proposed work, we have selected One-vs-Many SVM that has outperformed other classifiers in the first stage in terms of accuracy.

A. ACCURACY OF DFTF

Validation methods such as accuracy, precision, recall, and F-measure are used to check the accuracy of the DFTF. The classifiers such as ACO-SVM [102], Bayes Net [103], J48 [103], and Multiclass [104] are used for comparative analysis as shown in Table 5. The results of DTFT are presented as comparative analysis over other algorithms in which DFTF has shown better performance in all validation methods.

The results of classification algorithms are validated using classification validation accuracy measures concerning Accuracy, Precision, Recall, and F-Measure [105] reported in Table 6. The results of these classifiers are ranged between [0-1] with 1 being accurate classification. The more the value closer to 1, the higher the accuracy of the classifier is achieved. From Figure 3, DFTF has attained better accuracy than other classifiers.

TABLE 6. Comparative analysis of DFTF with classification techniques.

Metric	DFTF	Multiclass	J48	Bayes Net	ACO-SVM
Accuracy	0.974	0.96	0.824	0.825	0.812
Precision	0.963	0.96	0.818	0.824	0.87
Recall	0.951	0.96	0.825	0.824	0.751
F-Measure	0.977	0.97	0.821	0.824	0.817

B. EVALUATION OF DFTF ON ENERGY CONSUMPTION

Energy Consumption is calculated using the following proposed equation:

$$E_C = \sum_{k=1}^{N} \sum_{N=1}^{n} \left(\frac{T_E(\mathrm{VM}_N)}{E(T_k)} \right), \tag{6}$$

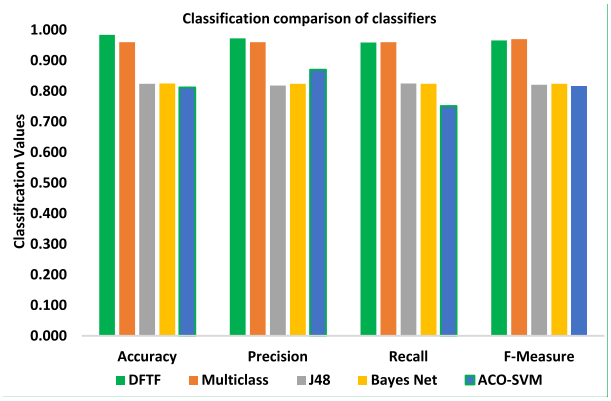

FIGURE 3. Comparison of DTFT with other classifiers.

where, $T_E(VM_N)$: Total Energy consumed by VM_N, where, $N = 1, .., n$

$E\ (T_k)$: Energy consumed for a particular task T_k

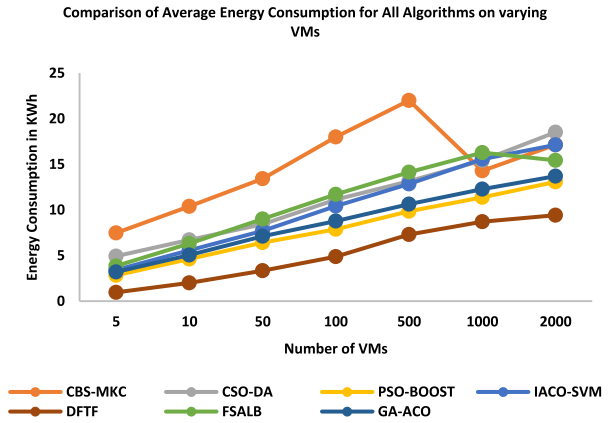

FIGURE 4. Performance of DFTF and Baselines on Energy Consumption on VMs (5-2000).

Energy consumption is depicted in Figure 4 in which comparative analysis is performed for all baselines. From the very start, a significant difference is seen which continues till 2000 VMs. Since the number of tasks and other configuration parameters is set equal so that reliable results may be obtained. After a gradual increase in the number of tasks from 1000 to 2000, few of the baselines such as CBS-MKC and CSO-DA start to consume slightly more energy which is further followed by a few other baselines. Likewise, the increase in VMs from 5 to 10, 50, 100, 500, 1000, and 2000 also results in an increase in energy consumption which keeps on getting increased with every additional VMs. In 2000 VMs, most of the baselines suffer from huge energy consumption leaving only DFTF with comparatively least consumption of energy.

The progress of DFTF in this scenario remains quite smooth even with the addition of more VMs which shows that DFTF is consistent. In other words, CBS-MKC has consumed 21% energy followed by CSO-DA with 16%, FSALB with 16%, IACO-SVM with 15%, GA-ACO with 12%, PSO-Boost with 11% and DFTF only consumed 9% energy. Preprocessing using a classification in cloud reduced the computational complexity resulted in the least energy consumption and further faster convergence of CSO achieved the best solution in the minimum number of iterations which also preserved the energy.

C. EVALUATION OF DFTF ON RESPONSE TIME

Response time is computed using the following proposed equation

$$\left[R_T = \sum_{t=1}^{N} C_{Tt} - (Ts_{Tt} - Te_{Tt}) \right], \tag{7}$$

where, t:Task

R_T : Response time

C_{Tt} : Computational time for task t

Ts_{Tt} : Task start time of t^{th} task

Te_{Tt} : Task end time of t^{th} task

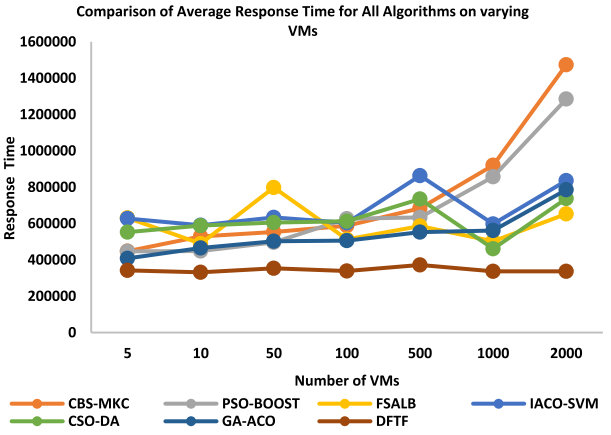

FIGURE 5. Performance of DFTF and Baselines on Response Time on VMs (5-2000).

Figure 5 represents a response time against a varying number of tasks and VMs for all algorithms. Comparative analysis of algorithms has not revealed much difference in response time initially as these algorithms are producing output in an almost equal time. Further, with the increase in VMs and tasks, a surge is seen especially for the two baselines such as CBS-MKC and PSO-Boost taking more response time after 1000 VMs. The other baselines such as IACO-SVM, GA-ACO, CSO-DA, and FSALB are also showing more response time whereas the response time of DFTF remains stable throughout.

This shows that DFTF is scalable in a dynamic environment where the number of tasks and VMs are getting increased. In other words, CBS-MKC has delivered its response in 18% of total response time followed by PSO-Boost with 16%, IACO-SVM with 16%, CSODA with 15%, FSALB with 14% and GA-ACO with 13%. However, DFTF has only taken 8% response time and has outperformed all baselines. The total response time is the sum of

all response times taken by all baselines. This is because of CSO's stronger convergence to a solution and achieving global minima in fewer iterations.

D. EVALUATION OF DFTF ON NUMBER OF SLA VIOLATIONS

SLA violation is calculated using the proposed equation (8):

$$\left[S_V = \sum_{i=1}^{N} \frac{T_{\text{total}}}{T_{\text{vm}_i}}\right]. \tag{8}$$

where,

T_{total} : Total CPU time

T_{vm_i} : Time is taken by i^{th}VM.

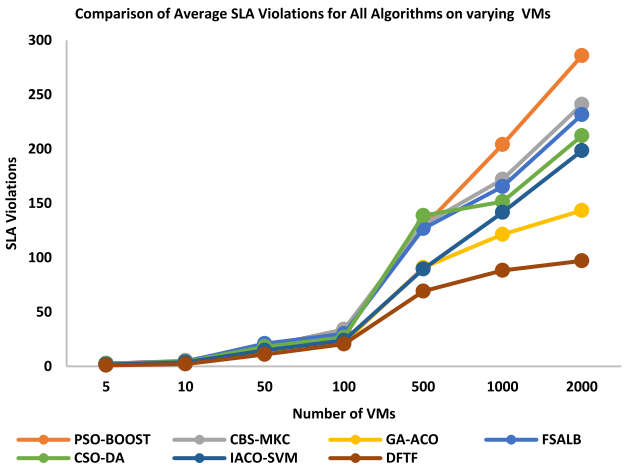

FIGURE 6. Performance of DFTF and Baselines on SLA Violations on VMs (5-2000).

In this case, SLA violation occurs if VM takes more time than allotted CPU time. SLA violations against varying number of tasks and VMs for all baselines are shown in Figure 6. It is observed that till 100 VMs, no major change in SLA violations is seen in the graph which becomes more obvious after 500 and above VMs. The algorithms such as PSO-Boost, CBS-MKC, and FSALB have violated more SLAs as soon as tasks and VMs are increased. DFTF, in this case, has done least violations that are only 8% as compared to PSO-Boost with 19% violations followed by CBS-MKC with 17%, FSALB with 16%, CSO-DA with 16%, IACO-SVM with 11% and GA-ACO with 11% SLA violations. The least number of SLA violations confirmed that DFTF has performed fewer migrations and avoiding complexity.

E. EVALUATION OF DFTF ON MIGRATION TIME

Migration time is calculated using the following function:

$$M_t = \sum_{k=1}^{n} T_{S_k}(VM_i, VM_{i+1}). \tag{9}$$

where,

$T_{S_k}(VM_i, VM_{i+1}) = T_{S_k}(VM_i \rightarrow VM_{i+1})$, scheduling time is taken for allocating k^{th} data from i^{th} VM to $(i+1)^{th}$ VM is based on availability.

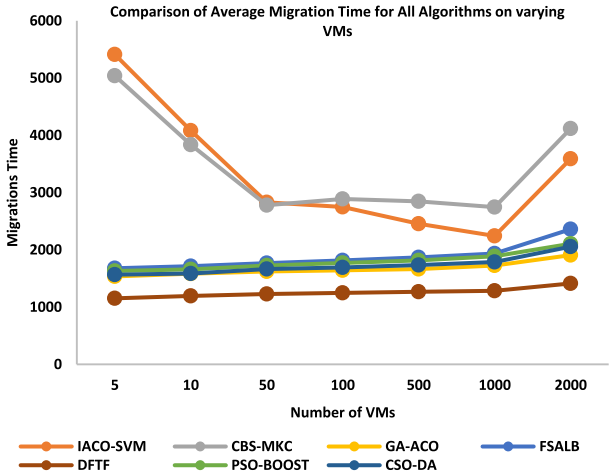

FIGURE 7. Performance of DFTF and Baselines on Migration time on VMs (5-2000).

Figure 7 shows the migration time taken by all baselines with varying tasks and VMs. More migrations take more time as compared to fewer migrations. Optimization algorithms make the least number of migrations because a single migration involves several processes, their communication, interrupts, addresses, and other factors. VM migration policy is administered by the administrator in which a rule is defined about when to trigger the VM migration from one host to another host. Generally, a threshold is defined which enables VM migrations while considering the computing capabilities of the host machines to minimize the number of SLA violations and migrations.

Both heuristic and metaheuristic algorithms as a hybrid approach work well for finding the solution in VM migration. For initial VM placements, the heuristic algorithm is suitable whereas, for optimization during migration, the metaheuristic algorithm performs well. This helps in reducing cost and solution space. Figure 7 initially shows IACO-SVM and CBS-MKC started by taking more migration time due to many migrations which gets better after 50 VMs till 1000 VMs and further increased afterward. This shows the unstable behavior of these algorithms with varying VMs and tasks. In other words, CBS-MKC has taken 23% migration time followed by IACO-SVM with 22%, FSLAB with 12%, PSO-Boost with 12%, CSO-DA with 12%, GA-ACO with 11% and DFTF with only 8% migration time due to least number of migrations and further supported by the least SLA violations.

F. EVALUATION OF DFTF ON OPTIMIZATION TIME

Optimization time is calculated using the following proposed equation:

$$O_T = \sum_{i=0}^{Imax} T_i. \tag{10}$$

where,

T_i : Time taken for i^{th} iteration

$\sum_{i=0}^{Imax} T_i$: Total Time for complete iteration.

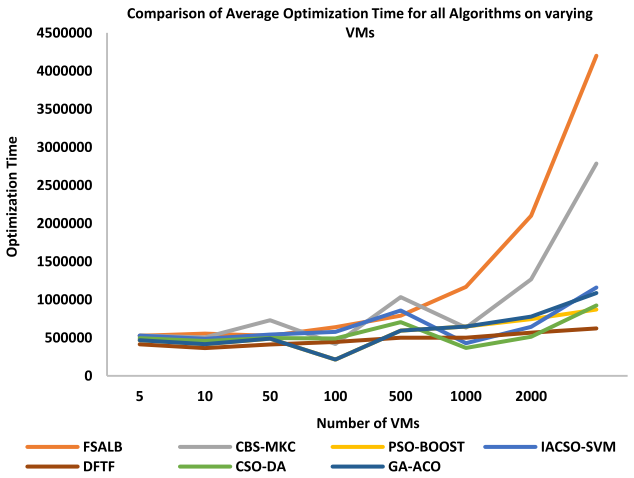

FIGURE 8. Performance of DFTF and Baselines on Optimization Time on VMs (5-2000).

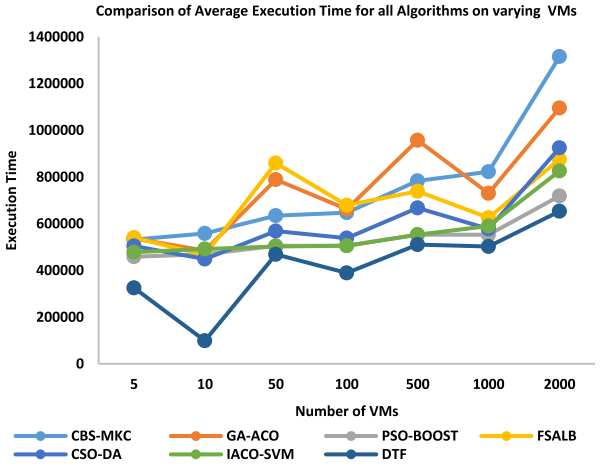

FIGURE 9. Performance of DFTF and Baselines on Execution Time on VMs (5-2000).

The optimization time also known as convergence time of all baselines is plotted in Figure 8. Only two algorithms FSALB and CBS-MKC are showing an exponential increase in optimization time resulting in comparatively higher unstable behavior. The algorithms such as IACSO-SVM, GA-ACO, and CSO-DA deviate a little bit in optimizing the tasks because these algorithms get trapped into local optimum, whereas as PSO-Boost and DFTF optimize quite fast and produce better results in the presence of other baselines. Overall, FSALB has taken much time in getting optimized which is 22% followed by CBBS-MKC with 17%, IACSO-SVM with 14%, CSO-DA with 12%, GA-ACO with 12%, PSO-Boost 12% and only 11% optimization time taken by DFTF. Cats move on a global scale to find the global best position that prevents them to fall into global optima so, they tend to optimize the solution quite fast.

G. EVALUATION OF DFTF ON EXECUTION TIME

Execution time is calculated using the following proposed equation:

$$E_t = \sum_{k=1}^{N} T_t(T_k). \tag{11}$$

where,

$T_t(T_k)$: Total time for executing k^{th} task

The execution time of all baselines is shown in Figure 9. Here, DFTF initially performed extremely well and took the least time and then started to rise when VMs gets 50 in size because DFTF initially converges slowly. However, not huge improvement is observed in DFTF but overall, comparatively better performance can be seen. The algorithms like CBS-MKC and GA-ACO from the very start deviate a lot and therefore take more execution time in almost all runs. FSALB remains quite better with every increase in VMs whereas, PSO-Boost and DFTF execute quite fast and produced better results in the presence of other baselines. Overall, CBS-MKC has taken much execution time that is 19% followed by GA-ACO with 17%, FSALB with 16%, CSO-DA with 14%, IACSO-SVM with 13%, PSO-Boost with 12% and only 9% execution time taken by DFTF.

It shows that the classification method using SVM plays an effective role in shortening the task execution time of the DFTF and further establishes the stronger scheduling ability of the algorithm.

H. EVALUATION OF DFTF ON THROUGHPUT TIME

Throughput time is calculated using the following proposed equation:

$$T_{TP} = \sum_{k=1}^{N} . \frac{T_k}{T_{p_k}}. \tag{12}$$

where,

T_k : k^{th} task

T_{p_k} : Time period for completing k^{th} task

The throughput time of all baselines is shown in Figure 10. Two algorithms CBS-MKC and IACSO-SVM have initially taken more time in providing throughput which gets further increased to 100 VMs because these algorithms could not quickly optimize. CSO-DA started better but surged after the addition of 500 VMs because more tasks are adding complexity in it.

However, FSALB, GA-ACO, and DFTF have shown good throughput performance. Overall, IACSO-SVM has taken much throughput time that is 19% followed by CSO-DA with 18%, CBS-MKC with 16%, PSO-Boost with 13%, FSALB and GA-ACO with 12% each, and only 10% throughput time taken by DFTF. Stronger robustness by DFTF has resulted in generating solutions in minimum throughput time.

I. EVALUATION OF DFTF ON OVERHEAD TIME

Overhead time is calculated using the following proposed equation:

$$OH_T = \sum_{i=1}^{N} (Tot_t(T_i) - E_t(T_i)). \tag{13}$$

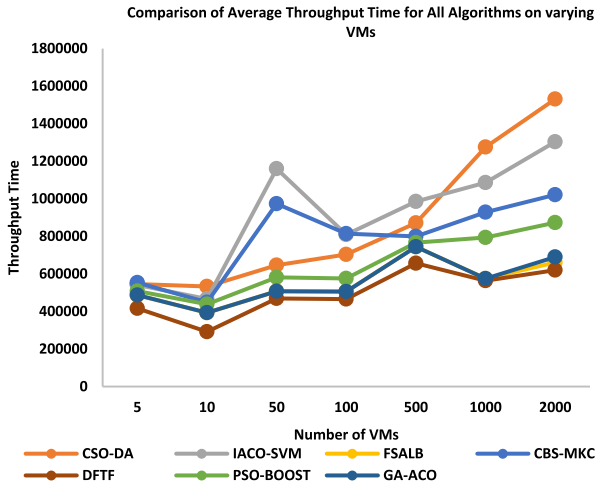

FIGURE 10. Performance of DFTF and Baselines on Throughput Time on VMs (5-2000).

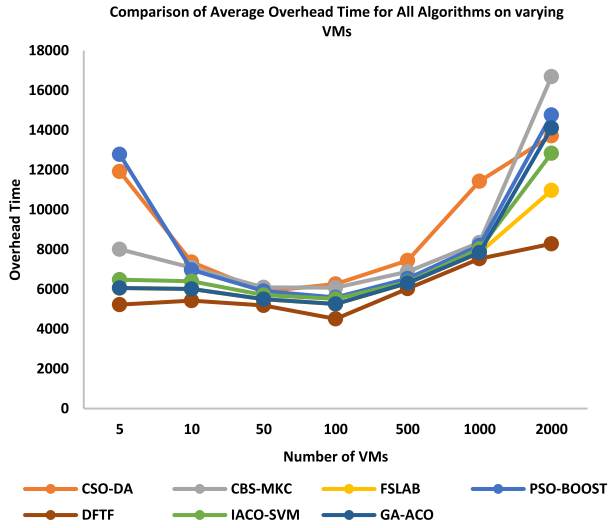

FIGURE 11. Performance of DFTF and Baselines on Overhead Time on VMs (5-2000).

where,

$Tot_t(T_i)$: Total time required for executing i^{th} task

$E_t(T_i)$: Execution time of i^{th} task

The overhead time of all baselines is shown in Figure 11. Two algorithms CSO-DA and PSO-BOOST started with huge overhead time which gets stable at 50 VMs but again instability is observed after 500 VMs which gets increased after every run. This is mainly because of their computational complexity. Overall, on average, CSO-DA has taken more overhead time that is 18% followed by CBS-MKC and PSO-BOOST with 17% each, IACSO-SVM with 15%, PSO-Boost with 13%, FSALB and GA-ACO with 13% each, and only 7% throughput time taken by DFTF.

The minimal computational complexity, fewer iterations in finding global optima, low communication cost, low overhead, and better convergence has made DFTF a better choice over other baselines.

J. STATISTICAL ANALYSIS

We have checked the resulting values of all parameters and found their distribution is normal. In that case, there is a need for a parametric test that involves 2 variables because we have taken one baseline at a time and compare it with DFTF. In statistics, the suitable test for 2 variables with normal distribution is student t-test. Similarly, we can see in Table 7 the values such as mean, standard deviation (SD), p-value, and t-value. Meanwhile, the significance level is set to $p < 0.05$ [106]. At this stage, we need to define the hypothesis in the following manner:

H0: DFTF and other baselines have no difference.

H1: A significant difference exists between DFTF and other baselines.

We can see that p-values in all cases are less than the significance level that is <0.05 which means that the significant difference exists among the values of DFTF and other baselines. So, we are right to reject the null hypothesis and accept the alternate hypothesis. Similarly, we can say that a significant difference exists in terms of energy consumption, response time, SLA violations, migration time, optimization time, execution time, throughput time, and overhead time.

K. RANKING BASELINES

Table 8 shows eight Quality of Service (QoS) metrics used in this study against seven baselines.

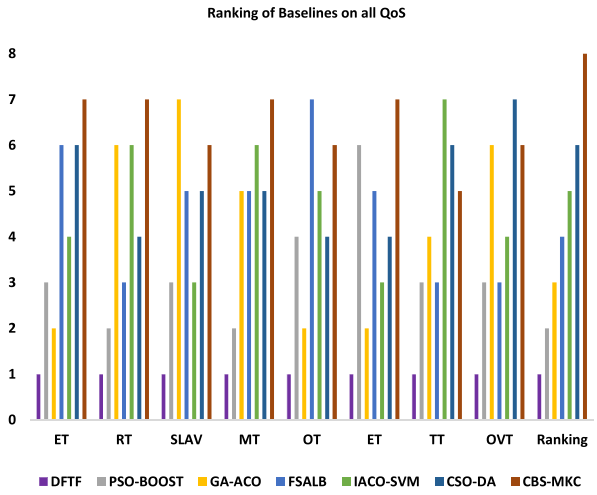

FIGURE 12. Comparative analysis of all baselines on various parameters.

It can be observed that certain baselines perform better in one scenario and average or worst in another scenario but proposed DFTF performed better among them followed by PSO-BOOST, GA-ACO, FSALB, IACO-SVM, CSO-DA, and CBS-MKC, respectively. Figure 12 shows the averaged performance of all baselines in terms of energy efficiency, response time, SLA violations, migration time, and optimization time over varying tasks and VMs. It is shown that overall DFTF has outperformed in all five-performance metrics in the presence of other baselines.

TABLE 7. Statistical comparison of DFTF with other baselines.

Parameters	CBS-MKC	CSO-DA	PSO-BOOST	IACO-SVM	FSALB	GA-ACO	DFTF
Energy Consumption							
Mean	15	11	8	10	11	9	6
SD	4.51332	4.5085	3.4046	4.73865	4.3878	3.54017	3.4489
t-value	0.00373	0.05924	0.4199	0.118051	0.065995	0.27118	-
p-value	0.0023	0.0072	5E-05	4.7E-05	6.4E-06	5.3E-05	-
Response Time							
Mean	742536	685020	596168	678927	613221	540195	344752
SD	329834.8	278827.9	101683.8	109320.1	90816.62	111245	13130.29
t-value	0.012104	0.01136	6.15E-05	7.9E-06	1.14E-05	0.001081	-
p-value	0.00069	0.00053	6E-05	4.1E-05	5.6E-05	7.5E-05	-
SLA Violations							
Mean	97	86	57	83	79	68	41
SD	104.01	88.1174	55.6901	84.582	79.708	71.929	38.9197
t-value	0.2444	0.27727	0.58768	0.29277	0.31653	0.4453	-
p-value	0.0037	0.0036	0.0004	0.0036	0.0035	0.0037	-
Migration Time							
Mean	3338	3465	1670	1877	1799	1727	1256
SD	1035.20	823.93	112.06	213.05	149.24	152.509	76.5265
t-value	0.00035	2.7E-05	7.5E-06	2.1E-05	4.1E-06	2E-05	-
p-value	0.0031	0.0018	5E-07	3E-09	2.7E-08	7.1E-13	-
Optimization Time							
Mean	900261	730955	510383	580855	503450	515006	457501
SD	533247.5	286857.7	159877.7	129402.8	94082.1	166891.3	63622.1
t-value	0.06632	0.04171	0.4660	0.05801	0.34124	0.44561	-
p-value	0.0134	0.027	0.0006	0.0157	0.0255	0.0075	-
Execution Time							
Mean	756178	750764	537757	683537	604282	563737	421116
SD	249029.3	202597.6	81261.5	142556	145026	112865	162419.7
t-value	0.01764	0.0096	0.14169	0.01165	0.06176	0.10275	-
p-value	0.003	0.011	0.045	0.033	0.042	0.012	-
Throughput Time							
Mean	872105	907447	553461	791252	648363	557567	497563
SD	358026.107	292284.2	108703.2	199630.7	150541.5	113139.7	116556.3
t-value	0.031351	0.0077	0.40712	0.00898	0.07622	0.38336	-
p-value	0.011	0.031	0.0016	0.009	0.007	0.001	-
Overhead Time							
Mean	9146	8454	6851	8679	7332	7298	6028
SD	2897.594	3456.208	1849.99	3353.83	2369.41	2884.37	1275.4
t-value	0.03279	0.1327	0.38773	0.09547	0.25840	0.34337	-
p-value	0.0013	0.039	0.0051	0.027	0.016	0.022	-

TABLE 8. Ranking of all baselines.

Baselines	Energy Consumption	Response Time	SLA Violations	Migration Time	Optimization Time	Execution Time	Throughput Time	Overhead Time	Ranking
DFTF	1	1	1	1	1	1	1	1	1
PSO-BOOST	3	2	3	2	4	6	3	3	2
GA-ACO	2	6	7	5	2	2	4	6	3
FSALB	6	3	5	5	7	5	3	3	4
IACO-SVM	4	6	3	6	5	3	7	4	5
CSO-DA	6	4	5	5	4	4	6	7	6
CBS-MKC	7	7	6	7	6	7	5	6	7

VI. CONCLUSION

The impact of file type format classification has made significant contributions to cloud computing. We have proposed a DFTF approach that achieves better results in load balancing. In the conducted study, DFTF is developed in two steps. In the first step, file type classification is done in various formats such as video, audio, text, and images in a cloud environment resulting in an appropriate data class. In our case, we have used four data classes in which appropriate file format falls. A total of 60,000 datasets/data files are collected from different sources and placed in the cloud for classification. Classification is performed using SVM one to many classification approaches providing the best accuracy among other classifiers such as Multiclass, J48, Bayes Net, and ACO-SVM. In the second step, the resultant data class is fed into a CSO which performs load balancing in an efficient manner. In CSO, we have introduced the grouping phase which divides the data files into four groups' audio, video, image, and text. The offline preprocessing in the cloud for classification helps in reducing the computational complexity and increases the efficiency in load balancing. Furthermore, the validation of DFTF is established through QoS evaluation metrics in terms of energy consumption, response time, SLA violations, migration time, execution time, throughput time, overhead time, and optimization time. DFTF due to its hybrid nature has taken the relative advantages of SVM and ICSO which helps in achieving better performance in the presence of baselines such as CBS-MKC, FSALB, PSO-BOOST, IACSO-SVM, CSO-DA, and GA-ACO.

The proposed approach is a multi-factor approach that ultimately saves time, cost, and valuable resources. It also improves the scalability, and robustness in the cloud environment. In the future, we will perform load balancing in the cloud by considering other sensitive parameters like deadline constraints, priority-based scheduling, and task immigrations using deep learning approaches.

REFERENCES

[1] M. Sheikhalishahi, R. M. Wallace, L. Grandinetti, J. L. Vazquez-Poletti, and F. Guerriero, "A multi-dimensional job scheduling," *Future Gener. Comput. Syst.*, vol. 54, pp. 123–131, Jan. 2016.

[2] T. Carli, S. Henriot, J. Cohen, and J. Tomasik, "A packing problem approach to energy-aware load distribution in Clouds," *Sustain. Comput., Inform. Syst.*, vol. 9, pp. 30–32, Mar. 2016.

[3] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 305–316, Feb. 2016.

[4] A. Nadjaran Toosi, C. Qu, M. D. de Assunção, and R. Buyya, "Renewable-aware geographical load balancing of Web applications for sustainable data centers," *J. Netw. Comput. Appl.*, vol. 83, pp. 155–168, Apr. 2017.

[5] S. S. Patil and A. N. Gopal, "Dynamic load balancing using periodically load collection with past experience policy on linux cluster system," *Amer. J. Math. Comput.*, vol. 2, no. 2, pp. 60–75, 2017.

[6] S. Mohanty, P. K. Patra, M. Ray, and S. Mohapatra, "A novel meta-heuristic approach for load balancing in cloud computing," *Int. J. Knowl.-Based Organizations*, vol. 8, no. 1, pp. 29–49, Jan. 2018.

[7] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1297–1309, Aug. 2015.

[8] M. Junaid, A. Sohail, A. Ahmed, A. Baz, I. A. Khan, and H. Alhakami, "A hybrid model for load balancing in cloud using file type formatting," *IEEE Access*, vol. 8, pp. 118135–118155, 2020.

[9] L. Heilig, R. Buyya, and S. Voß, "Location-aware brokering for consumers in multi-cloud computing environments," *J. Netw. Comput. Appl.*, vol. 95, pp. 79–93, Oct. 2017.

[10] A. Kaur, B. Kaur, and D. Singh, "Comparative analysis of metaheuristics based load balancing optimization in cloud environment," in *Smart and Innovative Trends in Next Generation Computing Technologies* (Communications in Computer and Information Science), vol. 827. Singapore: Springer, 2018, pp. 30–46.

[11] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, p. 120, 2019.

[12] W. Gai, C. Qu, J. Liu, and J. Zhang, "A novel hybrid meta-heuristic algorithm for optimization problems," *Syst. Sci. Control Eng.*, vol. 6, no. 3, pp. 64–73, Sep. 2018.

[13] P. Kaur and P. D. Kaur, "Efficient and enhanced load balancing algorithms in cloud computing," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 2, pp. 9–14, Apr. 2015.

[14] C. Gomez, A. Shami, and X. Wang, "Machine learning aided scheme for load balancing in dense IoT networks," *Sensors*, vol. 18, no. 11, p. 3779, Nov. 2018.

[15] X. Sui, D. Liu, L. Li, H. Wang, and H. Yang, "Virtual machine scheduling strategy based on machine learning algorithms for load balancing," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 160, Dec. 2019.

[16] B. Tang, Y. Li, X. Li, L. Xu, Y. Yan, and Q. Yang, "Deep CNN framework for environmental sound classification using weighting filters," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Tianjin, China, Aug. 2019, pp. 2297–2302.

[17] A. Zakaria, R. Rizal, and O. Dwi, "Particle swarm optimization and support vector machine for vehicle type classification in video stream," *Int. J. Comput. Appl.*, vol. 182, no. 18, pp. 9–13, Sep. 2018.

[18] Y. F. Huang and S. H. Wang, "Movie genre classification using SVM with audio and video features," in *Active Media Technology* (Lecture Notes in Computer Science), vol. 7669, R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, and B. Jin, Eds. Berlin, Germany: Springer, 2012.

[19] K. M. Salama and A. M. Abdelbar, "Learning neural network structures with ant colony algorithms," *Swarm Intell.*, vol. 9, no. 4, pp. 229–265, Dec. 2015.
[20] L. Jiao and L. Feng, "Text classification based on ant colony optimization," in *Proc. 3rd Int. Conf. Inf. Comput.*, Jun. 2010, pp. 229–232.
[21] Q. Wang, R. Peng, J. Wang, Y. Xie, and Y. Zhou, "Research on text classification method of LDA- SVM based on PSO optimization," in *Proc. Chin. Autom. Congr. (CAC)*, Hangzhou, China, Nov. 2019, pp. 1974–1978.
[22] P. Adriana, L. Veronica, P. R. Pasquale, and I. Sidhu, "A genetic algorithm for text classification rule induction," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2008, pp. 188–203.
[23] H. Hasanpour, R. Ghavamizadeh Meibodi, and K. Navi, "Improving rule-based classification using harmony search," *PeerJ Comput. Sci.*, vol. 5, p. e188, Nov. 2019.
[24] F. Yigit and O. K. Baykan, "A new feature selection method for text categorization based on information gain and particle swarm optimization," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Intell. Syst.*, Nov. 2014, pp. 523–529.
[25] H. Peng, C. Ying, S. Tan, B. Hu, and Z. Sun, "An improved feature selection algorithm based on ant colony optimization," *IEEE Access*, vol. 6, pp. 69203–69209, 2018.
[26] C. López-Franco, L. Villavicencio, N. Arana-Daniel, and A. Y. Alanis, "Image classification using PSO-SVM and an RGB-D sensor," *Math. Problems Eng.*, vol. 2014, Jul. 2014, Art. no. 695910.
[27] C. Sukawattanavijit, J. Chen, and H. Zhang, "GA-SVM algorithm for improving land-cover classification using SAR and optical remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 3, pp. 284–288, Mar. 2017.
[28] V. Pallavi and V. Vaithiyanathan, "Combined artificial neural network and genetic algorithm for cloud classification," *Int. J. Eng. Res. Technol.*, vol. 5, pp. 787–794, Apr. 2013.
[29] *Data Preprocessing for Machine Learning: Options and Recommendations*. Accessed: May 17, 2020. [Online]. Available: https://cloud.google.com/solu tions/machine-learning/data-preprocessing-for-ml-with-tf-transform-pt2
[30] S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimization," in *PRICAI 2006: Trends in Artificial Intelligence* (Lecture Notes in Computer Science), vol. 4099, Q. Yang and G. Webb, Eds. Berlin, Germany: Springer, 2006.
[31] A. M. Ahmed, T. A. Rashid, and S. A. M. Saeed, "Cat swarm optimization algorithm: A survey and performance evaluation," *Comput. Intell. Neurosci.*, vol. 2020, Jan. 2020, 4854895.
[32] C. E. Klein, L. dos Santos Coelho, Â. M. O. Sant'Anna R. Z. Freire, and V. C. Mariani, "Improved cat swarm optimization approach applied to reliability-redundancy problem," in *Proc. 22nd Eur. Symp. Artif. Neural Netw. (ESANN)*, Bruges, Belgium, Apr. 2014, pp. 1–6.
[33] (Jan. 1, 2016). *Using a PostgreSQL Database as a Source for AWS DMS*. Accessed: May 9, 2020. [Online]. Available:https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.PostgreSQL.html
[34] S. Vrajesh and M. Bala, "An improved task allocation strategy in cloud using modified K-means clustering technique," *Egyptian Inform. J.*, vol. 4, pp. 1–8, 2020.
[35] K. Sekaran, M. S. Khan, R. Patan, A. H. Gandomi, P. V. Krishna, and S. Kallam, "Improving the response time of M-Learning and cloud computing environments using a dominant firefly approach," *IEEE Access*, vol. 7, pp. 30203–30212, 2019.
[36] M. Kumar and S. C. Sharma, "PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing," *Neural Comput. Appl.*, vol. 194, pp. 1–24, Jun. 2019.
[37] S. Rongali and R. Yalavarthi, "An improved ant colony optimization for parameter optimization using support vector machine," *Int. J. Eng. Adv. Technol. (IJEAT)*, vol. 6, no. 3, pp. 198–204, 2017.
[38] A. Pourghaffari and M. Barar, "Workflow scheduling in cloud computing environment using hybrid CSO-DA," *Int. J. Nonlinear Anal. Appl.*, vol. 10, no. 2, pp. 177–188, 2019.
[39] A. M. Senthil Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Pers. Commun.*, vol. 107, no. 4, pp. 1835–1848, Aug. 2019.
[40] A. Thakur and M. S. Goraya, "A taxonomic survey on load balancing in cloud," *J. Netw. Comput. Appl.*, vol. 98, pp. 43–57, Nov. 2017.
[41] S. Kumar Mishra, S. Bibhudatta, and P. P. Parida, "Load balancing in cloud computing: A big picture," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, 2020.
[42] R. Shaikh and M. Sasikumar, "Data classification for achieving security in cloud computing," *Procedia Comput. Sci.*, vol. 45, pp. 493–498, 2015.
[43] H. B. Barua and K. C. Mondal, "A comprehensive survey on cloud data mining (CDM) frameworks and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–62, Oct. 2019.
[44] H. Song and J. G. Lee, "RP-DBSCAN: A superfast parallel DBSCAN algorithm based on random partitioning," in *Proc. Int. Conf. Manage. Data*, Houston, TX, 2018, pp. 1173–1187.
[45] R. Jin, C. Kou, R. Liu, and Y. Li, "Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 2, no. 1, p. 18, 2013.
[46] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, Apr. 2017.
[47] F. Ozgur Catak and M. Erdal Balaban, "CloudSVM: Training an SVM classifier in cloud computing systems," in *Proc. Joint Int. Conf. Pervasive Comput. Netw. World*. Berlin, Germany: Springer, 2012, pp. 57–68.
[48] B. Apexa Kamdar and M. Jay Jagani, "A survey: Classification of huge cloud datasets with efficient map-reduce policy," *Int. J. Eng. Trends Technol. (IJETT)*, vol. 18, no. 2, pp. 103–107, 2014.
[49] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, P. Michiardi, and F. Pulvirenti, "PaMPa-HD: A parallel MapReduce-based frequent pattern miner for high-dimensional data," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Atlantic City, NJ, USA, Nov. 2015, pp. 839–846.
[50] I. Strumberger, N. Bacanin, M. Tuba, and E. Tuba, "Resource scheduling in cloud computing based on a hybridized whale optimization algorithm," *Appl. Sci.*, vol. 9, no. 22, p. 4893, Nov. 2019.
[51] D. Pebrianti, A. Nurnajmin, B. Luhur, A. Nor Rul Hasma, Z. Zainah, and I. Riyanto, "Extended bat algorithm (EBA) as an improved searching optimization algorithm," in *Proc. 10th Nat. Tech. Seminar Underwater Syst. Technol. (NUSYS)*. Singapore: Springer, 2018.
[52] D. Chaudhary and B. Kumar, "Cloudy GSA for load scheduling in cloud computing," *Appl. Soft Comput.*, vol. 71, pp. 861–871, Oct. 2018.
[53] S. Torabi and F. Safi-Esfahani, "A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing," *J. Supercomput.*, vol. 74, no. 6, pp. 2581–2626, Jun. 2018.
[54] A. Al-Hamodi, S. Lu, and Y. Al-Salhi, "An enhanced frequent pattern growth based on MapReduce for mining association rules," *Int. J. Data Mining Knowl. Manage. Process*, vol. 6, no. 2, pp. 19–28, 2016.
[55] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1688–1699, Jun. 2018.
[56] D. Soni, A. Mishra, and H. Gupta, "An efficient cloud data mining (CDM) algorithm for frequent pattern mining in cloud computing environment," *Lect. Notes Softw. Eng*, vol. 4, no. 3, pp. 234–237, 2016.
[57] M. S. Sudheer and M. Dr Vamsi Krishna 2019, "Dynamic PSO for task scheduling optimization in cloud computing," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2S11, pp. 3559–3589, 2019.
[58] K. Mangayarkkarasi and M. Chidambaram, "An intelligent service recommendation model for service usage pattern discovery in secure cloud computing environment," *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 12, pp. 3500–3512, 2017.
[59] A. Bouzidi, M. E. Riffi, and M. Barkatou, "Cat swarm optimization for solving the open shop scheduling problem," *J. Ind. Eng. Int.*, vol. 15, no. 2, pp. 367–378, Jun. 2019.
[60] M. Kumar, SC. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," in *Proc. 7th Int. Conf. Adv. Comput. Commun. (ICACC)*, Cochin, India, Aug. 2017, pp. 322–329.
[61] L. Zhou and X. Wang, "Research of the FP-growth algorithm based on cloud environments," J. Software, vol. 9, no. 3, pp. 676–683, 2014.

[62] A. K. Maurya and A. K. Tripathi, "Deadline-constrained algorithms for scheduling of bag-of-tasks and workflows in cloud computing environments," in *Proc. 2nd Int. Conf. High Perform. Compilation, Comput. Commun. (HP3C)*, Hong Kong, Mar. 2018, pp. 6–10.

[63] R. Rautray and R. C. Balabantaray, "Cat swarm optimization based evolutionary framework for multi document summarization," *Phys. A, Stat. Mech. Appl.*, vol. 477, pp. 174–186, Jul. 2017.

[64] M. Meyer, J. Beutel, and L. Thiele, "Unsupervised feature learning for audio analysis," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR), Workshop Track*, Toulon, France, 2017, pp. 1–4.

[65] G. Danlami, A. S. Ismail, A. Zainal, Z. Zakaria, A. Abraham, and N. M. Dankolo, "Cloud customers service selection scheme based on improved conventional cat swarm optimization," *Neural Comput. Appl.*, vol. 6, pp. 1–22, 2020.

[66] P. Bajare, M. Bhoyate, Y. Bhujbal, E. Monika, and V. Shinde, "k-nearest neighbor classification over encrypted cloud data," *IOSR J. Comput. Eng. (IOSR-JCE)*, pp. 45–48, 2015.

[67] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Abraham, "Orthogonal taguchi-based cat algorithm for solving task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 30, no. 6, pp. 1845–1863, Sep. 2018.

[68] K. Liu and J. Boehm, "Classification of big point cloud data using cloud computing," *ISPRS-Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 40, no. 3, p. 553, 2015.

[69] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.

[70] K.-C. Lin, K.-Y. Zhang, Y.-H. Huang, J. C. Hung, and N. Yen, "Feature selection based on an improved cat swarm optimization algorithm for big data classification," *J. Supercomput.*, vol. 72, no. 8, pp. 3210–3221, Aug. 2016.

[71] R. Latif, H. Abbas, S. Latif, and A. Masood, "EVFDT: An enhanced very fast decision tree algorithm for detecting distributed denial of service attack in cloud-assisted wireless body area network," *Mobile Inf. Syst.*, vol. 2015, pp. 1–13, 2015, 260594.

[72] D. Gabi, A. S. Ismail, and N. M. Dankolo, "Minimized makespan based improved cat swarm optimization for efficient task scheduling in cloud datacenter," in *Proc. 3rd High Perform. Comput. Cluster Technol. Conf. (HPCCT)*, New York, NY, USA, Jun. 2019, pp. 16–20.

[73] C. Bae, N. Wahid, Y. Y. Chung, and W. C. Yeh, "Effective audio classification algorithm swarm-based optimization," *Int. J. Innov. Comput., Inf. Control*, vol. 10, no. 1, pp. 151–167, 2014.

[74] B. Panchal and R. K. Kapoor, "Performance enhancement of cloud computing with clustering," *Int. J. Eng. Adv. Technol.*, vol. 14, no. 6, pp. 37–40, 2014.

[75] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Al-Khasawneh, "Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment," *J. Inf. Commun. Technol.*, vol. 17, no. 3, pp. 435–467, Jun. 2018.

[76] D. Danilo, G. Fenu, M. Marras, and D. R. Recupero, "Bridging learning analytics and cognitive computing for big data classification in micro-learning video collections," *Comput. Hum. Behav.*, vol. 92, pp. 468–477, 2019.

[77] C. Sauvanaud, G. Silvestre, M. Kaaniche, and K. Kanoun, "Data stream clustering for online anomaly detection in cloud applications," in *Proc. 11th Eur. Dependable Comput. Conf. (EDCC)*, Sep. 2015, pp. 120–131.

[78] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 3, pp. 1–27, Jul. 2009.

[79] Y. Kumar and P. K. Singh, "Improved cat swarm optimization algorithm for solving global optimization problems and its application to clustering," *Int. J. Speech Technol.*, vol. 48, no. 9, pp. 2681–2697, Sep. 2018.

[80] P. Bisht and K. Singh, "Big data mining: Analysis of genetic K-means algorithm for big data clustering," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 6, no. 7, pp. 223–228, 2016.

[81] J. Zgraja and M. Woniak, "Drifted data stream clustering based on ClusTree algorithm," in *Proc. Int. Conf. Hybrid Artif. Intell. Syst.* Cham, Switzerland: Springer, 2018, pp. 338–349.

[82] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," *Procedia Comput. Sci.*, vol. 125, pp. 725–732, 2018.

[83] J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for work?ow scheduling in cloud under deadline constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016.

[84] A. Nazia and D. Huifang, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment," *Appl. Sci.*, vol. 8, no. 4, p. 538, 2018.

[85] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine," *Int. J. Speech Technol.*, vol. 48, no. 11, pp. 4072–4083, Nov. 2018.

[86] M. Ashouraei, S. N. Khezr, R. Benlamri, and N. J. Navimipour, "A new SLA-aware load balancing method in the cloud using an improved parallel task scheduling algorithm," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Barcelona, Spain, Aug. 2018, pp. 71–76.

[87] N. Sharma and S. Maurya, "SLA-based agile VM management in cloud & datacenter," in *Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon))*, Faridabad, India, Feb. 2019, pp. 252–257.

[88] A. Kumar and S. Bawa, "A comparative review of meta-heuristic approaches to optimize the SLA violation costs for dynamic execution of cloud services," *Soft Comput.*, vol. 24, no. 6, pp. 3909–3922, Mar. 2020.

[89] X. Song, Y. Ma, and D. Teng, "A load balancing scheme using federate migration based on virtual machines for cloud simulations," *Math. Problems Eng.*, vol. 2015, Mar. 2015, Art. no. 506432.

[90] J. Rouzaud-Cornabas, "A distributed and collaborative dynamic load balancer for virtual machine," in *Proc. Eur. Conf. Parallel Process. (ECPP)*, Ischia, Italy, 2010, pp. 641–648.

[91] T. Jamal and A. Enrique, "Metaheuristics for energy-efficient data routing in vehicular networks," *Int. J. Metaheuristics*, vol. 4, no. 1, pp. 27–56, 2015.

[92] K. Saleem and N. Fisal, "Enhanced ant colony algorithm for self-optimized data assured routing in wireless sensor networks," in *Proc. 18th IEEE Int. Conf. Netw. (ICON)*, Dec. 2012, pp. 422–427.

[93] L. Vu and G. Alaghband, "A load balancing parallel method for frequent pattern mining on multi-core cluster," in *Proc. Symp. High Perform. Comput.*, San Diego, CA, USA, 2015, pp. 49–58.

[94] N. Susila, "An efficient load balancing approach for energy aware cloud environment," Ph.D. dissertation, Dept. Inf. Commun. Eng., Anna Univ., Chennai, India, 2017.

[95] R. Khorsand and M. Ramezanpour, "An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing," *Int. J. Commun. Syst.*, vol. 33, no. 9, p. e4379, 2020.

[96] R. M. Alguliyev, Y. N. Imamverdiyev, and F. J. Abdullayeva, "PSO-based load balancing method in cloud computing," *Autom. Control Comput. Sci.*, vol. 53, no. 1, pp. 45–55, Jan. 2019.

[97] E. Rafieyan, R. Khorsand, and M. Ramezanpour, "An adaptive scheduling approach based on integrated best-worst and VIKOR for cloud computing," *Comput. Ind. Eng.*, vol. 140, Feb. 2020, Art. no. 106272.

[98] I. Mierswa and K. Morik, "Automatic feature extraction for classifying audio data," *Mach. Learn.*, vol. 58, nos. 2–3, pp. 127–149, Feb. 2005.

[99] (2010). *UCI Machine Learning Repository*. Accessed: May 20, 2020. [Online]. Available: http://archive.ics.uci.edu/ml

[100] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services," 2009, *arXiv:0903.2525*. [Online]. Available: https://arxiv.org/abs/0903.2525

[101] D. Kai-Bo, C. R. Jagath, and M. N. Nguyen, "One-versus-one and one-versus-all multiclass SVM-RFE for gene selection in cancer classification," in *Proc. Eur. Conf. Evol. Comput., Mach. Learn. Data Mining Bioinf.* Berlin, Germany: Springer, 2007, pp. 47–56.

[102] R. Karthika and P. Visalakshi, "A hybrid ACO based feature selection method for email spam classification," *WSEAS Trans. Comput.*, vol. 14, pp. 171–177, 2015.

[103] B. Çiğşar, D. Ünal, "Comparison of data mining classification algorithms determining the default risk," *Sci. Program.*, vol. 2019, Feb. 2019, Art. no. 8706505.

[104] P. Kanu, J. Vala, and P. Jaymit, "Comparison of various classification algorithms on iris datasets using WEKA," *Int. J. Advance Eng. Res. Develop.*, vol. 1, pp. 1–7, Feb. 2014.

[105] B. Desgraupes, "Clustering indices," Ouest-Lab Modal'X, Univ. Paris, Paris, France, Tech. Rep., 2013, pp. 1–34.

[106] M. Ghobaei-Arani, A. A. Rahmanian, A. Rahmanian, A. Souri, and A. M. Rahmani, "A moth-flame optimization algorithm for Web service composition in cloud computing: Simulation and verification," *Softw., Pract. Exper.*, vol. 48, no. 10, pp. 1865–1892, 2018.

MUHAMMAD JUNAID is currently pursuing the Ph.D. degree with Iqra University, Islamabad, Pakistan. His research interests include cloud computing, blended learning, machine learning, swarm intelligence, and information security.

ADNAN SOHAIL received the master's degree in computer science from Bahria University, Islamabad, Pakistan, and the Ph.D. degree in electrical engineering and information technology from the Institute of Telecommunications, Vienna University of Technology, Vienna, Austria. He was an Assistant Professor with CU and IIUI, Pakistan. He is currently an Assistant Professor with the Computing and Technology Department, Iqra University, Islamabad. His main research interests include performance modeling and analysis of communication networks, cloud computing, optimization techniques, artificial intelligence, and agent-based computing.

RAO NAVEED BIN RAIS received the M.S. and Ph.D. degrees in computer engineering (networks and distributed systems) from the University of Nice Sophia Antipolis, France, in 2007 and 2011, respectively. He has experience of more than 15 years in teaching, research, and industrial development. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, College of Engineering and Information Technology, Ajman University, United Arab Emirates. His research interests include network protocols and architectures, information-centric and software-defined networks, network virtualization, machine learning, internet naming, and addressing issues.

ADEEL AHMED (Graduate Student Member, IEEE) received the M.Phil. degree in computer science from Quaid-i-Azam University, Islamabad, Pakistan, in 2011. He was with software industry for few years. He is currently a Ph.D. Scholar with the Department of Computer Science, Quaid-i-Azam University. He is also with the Faculty of Information Technology, The University of Haripur, Pakistan. His research interests include social network analysis, recommender systems, machine learning, and swarm intelligence.

OSMAN KHALID received the master's degree from the Center for Advanced Studies in Engineering and the Ph.D. degree from North Dakota State University, USA. He is currently an Assistant Professor with COMSATS University Islamabad, Abbottabad. His research interests include recommender systems, network routing protocols, the Internet of Things, and fog computing.

IMRAN ALI KHAN received the master's degree from Gomal University, Dera Ismail Khan, Pakistan, and the Ph.D. degree from the Graduate University Chinese Academy of Sciences, China. He is currently an Associate Professor with the Department of Computer Science, COMSATS University Islamabad, Abbottabad, Pakistan. He has produced over 50 publications in journal of the international repute and presented papers in the international conferences. His research interests include wired and wireless networks and distributed systems.

SYED SAJID HUSSAIN received the Master of Science degree in computer science from COMSATS University Islamabad (CUI), Pakistan, in 2007, and the Ph.D. degree from the Fern Universität in Hagen, Germany, in 2013. He is currently an Assistant Professor with CUI, Abbottabad. His research interests include collaborative computing, human-computer interaction, and distributed systems.

NAVEED EJAZ received the B.S. degree in computer sciences from the National University of Computer and Emerging Sciences, Islamabad, Pakistan, the M.S. degree in computer sciences from the National University of Sciences and Technology, Islamabad, and the Ph.D. degree in digital contents engineering from Sejong University, South Korea, in August 2013. He was a Faculty Member with reputed Universities in Pakistan and Saudi Arabia. His research interests include video summarization, video tagging, object detection in videos, and applications of deep learning in other image and video domains.