IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received May 29, 2020, accepted June 14, 2020, date of publication June 19, 2020, date of current version July 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3003825

A Hybrid Model for Load Balancing in Cloud Using File Type Formatting

MUHAMMAD JUNAID[1], ADNAN SOHAIL[1], ADEEL AHMED[2], (Graduate Student Member, IEEE), ABDULLAH BAZ[3], (Senior Member, IEEE), IMRAN ALI KHAN[4], AND HOSAM ALHAKAMI[5]

[1]Department of Computing, Iqra University, Islamabad 46000, Pakistan
[2]Department of Computer Science, Quaid-i-Azam University, Islamabad 45320, Pakistan
[3]Department of Computer Engineering, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[4]Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22060, Pakistan
[5]Department of Computer Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia

Corresponding author: Muhammad Junaid (mjunaid@uoh.edu.pk)

This work was supported by the Deanship of Scientific Research at Umm Al-Qura University, Makkah, Saudi Arabia, under Grant 19-COM-1-01-0015.

ABSTRACT Maintaining accuracy in load balancing using metaheuristics is a difficult task even with the help of recent hybrid approaches. In the existing literature, various optimized metaheuristic approaches are being used to achieve their combined benefits for proper load balancing in the cloud. These approaches often adopt multi-objective QoS metrics, such as reduced SLA violations, reduced makespan, high throughput, low overload, low energy consumption, high optimization, minimum migrations, and higher response time. The cloud applications are generally computation-intensive and can grow exponentially in memory with the increase in size if no proper effective and efficient load balancing technique is adopted resulting in poor quality solutions. To provide a better load balancing solution in cloud computing, with extensive data, a new hybrid model is being proposed that performs classification on the number of files present in the cloud using file type formatting. The classification is performed using Support Vector Machine (SVM) considering various file formats such as audio, video, text maps, and images in the cloud. The resultant data class provides high classification accuracy which is further fed into a metaheuristic algorithm namely Ant Colony Optimization (ACO) using File Type Formatting FTF for better load balancing in the cloud. Frequently used QoS metrics, such as SLA violations, migration time, throughput time, overhead time, and optimization time are evaluated in the cloud environment and comparative analysis is performed with recent metaheuristics, such as Ant Colony Optimization-Particle Swarm Optimization (ACOPS), Chaotic Particle Swarm Optimization (CPSO), Q- learning Modified Particle Swarm Optimization (QMPSO), Cat Swarm Optimization (CSO) and D-ACOELB. The proposed algorithm outperforms them and provides good performance with scalability and robustness.

INDEX TERMS ACO, classification, hybrid metaheuristics, load balancing, machine learning, SVM, virtual machine.

I. INTRODUCTION

Nowadays, cloud computing is playing a significant role by providing on-demand services on a pay as you go basis. The service models like SaaS, PaaS, and IaaS are being exploited by the vendors for the provision of quality services which has shown huge growth (21.5 % approx.) in public cloud computing markets during the last five years [1]. This QoS provision also involves other internal and external factors such as environmental issues, economy, sustainability, performance, energy consumption, development of new policies and techniques [2]. This means that cloud computing success is highly dependent on efficient supported polices and intelligent decisions by the vendors and consumers. Similarly, other features of cloud including load balancing, scalability, throughput, SLAs, energy consumption,

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed A. Zaki Diab.

execution time, deadline constraint, optimization, migration, makespan, and response time are considered by consumers and vendors to maintain QoS. Metaheuristic algorithms are used to optimize such QoS and solve combinatorial problems in a cloud environment where traditional algorithms fail to provide optimum solutions effectively and efficiently [3]. Similarly, these algorithms provide rapid decision making and fast convergence [4]. Therefore, the development of heuristic, metaheuristic, hybrid metaheuristic, and integrated machine learning approaches are the main areas of cloud research to explore their potential benefits. Metaheuristic algorithms are being integrated into machine learning models to improve classification accuracy and load balancing issues. The integration of such recent approaches has revealed good results in the number of studies [5].

There are numerous challenges jointly faced by the cloud service providers and clients regarding faster access to the cloud services. Due to huge volume and variety of data placed in the cloud, extraction of only relevant information is a difficult job that requires more resources. The situation becomes more daunting when it is required to process large scaled, computationally complex, and resource demanding applications. In such scenarios, data preprocessing can play an important role where an offline classification of data with machine learning models may significantly reduce execution time and memory requirements during online processing phase. Moreover, the task assignment to VMs also needs to be carefully performed to ensure optimal load balancing. Therefore, to achieve better classification results and efficient task assignment, we proposed a new hybrid model based on SVM and ACO that achieves optimal load balancing performance as compared to existing models. By integrating these two models into a hybrid one with multi-objective approach addresses their individual limitations and reinforces their combined benefits. Further, earlier studies focused on various factors such as cost, response time, SLA violation, and energy consumption by developing appropriate single or multi-objective QoS metrics [6], [7]. The development of metaheuristics and hybrid metaheuristics are emerging ways to solve such multi-objective metrics in cloud computing.

Hybrid metaheuristics are used for several purposes such as classification, load balancing, fault tolerance, cost analysis, and energy conservation. However, classification of cloud data into various file formats is a new contribution to the body of knowledge. The classification approaches already exist for datatype formats as initially used by PostgreSQL and AWS [8]. However, classification concerning data files such as audio, video, text, images, maps, in cloud computing requires some extra effort to achieve accurate classifications and perform load balancing. This problem can be solved in two steps. In the first step, there is a need to develop classification algorithm that performs accurate classifications over cloud datasets resulting inappropriate data classes. In the second step, resultant data class is fed into some load balancing algorithm like metaheuristics. Overall, this can be achieved through proposed model. SVM is a robust algorithm that can handle both classification and regression assignments making it more advantageous for classification. It signifies the data set components each of which has an ''n'' dimensional space separated by maximum margin known as a hyper-plane. Similarly, ACO generates better results in load balancing problems and is one of the most widely used algorithm with a number of variants. ACO provides strong robustness and can search for the solutions faster [104]. Moreover, as ants search concurrently, this helps in achieving good performance quickly and it can be easily integrated with other metaheuristics [9]. Because of its diversity, ACO has been applied in a wide variety of studies [10], [11], [100]–[102] including supervised learning models, such as classification rules [12]–[17]. In this paper, we focus on the ACO algorithm with SVM [18]–[20], and they both together have been applied to several optimization problems [21], [22]. This research aims at the development of a new hybrid algorithm ACOFTF which considers important QoS metrics such as SLA violations, migration time, throughput time, overhead time, and optimization time.

The proposed model has proved to avoid premature convergence which is one of the objectives of hybrid metaheuristics even in the presence of diverse datasets. Similarly, low diversity promotes exploration whereas, high diversity often but not necessarily results in exploitation. Proposed model can search the space efficiently and effectively resulting in intelligent exploration and gathering of desired features through exploitation which may help in getting quality solutions. Such features are also being exploited by the deep learning approaches but the fact that these approaches take a lot of time in training makes them sometimes infeasible for time-bound problems. Further, hybridized features are being combined in a way to take maximum advantage from the proposed algorithm. In the same way, machine learning algorithms are being hybridized with load balancing algorithms for accuracy purposes such as SVM and PSO for audio file classifications [23], K-Nearest Neighbor (K-NN) and ACO for datasets classification [24], SVM with PSO used for video classifications [25], Decision Trees (DT) and SVM for text classifications [26], Naïve Bayes and SVM for image classifications [27].

The proposed approach not only focuses on achieving the best classification accuracy among baselines but also efficiently performs scheduling over competitor baselines such as ACOPS [28], CPSO [29], QMPSO [30], CSO [31] and D-ACOELB [64]. All these algorithms are used for achieving load balancing and have performed reasonably well in many approaches. On contrary, each of them has some issues such as: In the case of ACOPS, it has not considered a multi-objective optimization approach. CPSO focused only on cost and lacks a multi-objective approach. QMPSO is applied only on a limited number of tasks and VMs' resulting in scalability concern. CSO has shown a high chance of premature convergence and a multi-objective approach is not present. To address these problems, a multiobjective metaheuristic is developed and implemented that has shown comparatively better results over them in the experimental setups. However, the contributions of this research include:

- A new hybrid multi-objective metaheuristic approach called ACOFTF is developed based on SVM and ACO that performs load balancing in a cloud environment using file formats.
- The classification of file formats into audio, video, image, and text is performed in a cloud environment that shows improved results over baselines.
- Proposed model is evaluated using multi-objective QoS metrics such as SLA violations, migration time, throughput time, overhead time, and optimization time, which shows better significance of the proposed approach in multiple scenarios.
- For evaluation, we have also compared our approach with some baselines and obtained improved results.

The rest of the paper is organized as follows. Section 2 presents existing work, Section 3 and Section 4 covers research methodology and experimental setup, respectively. Section 5 provides results analysis, and Section 6 presents the conclusion.

II. EXISTING WORK

We categorized the existing work into two types of approaches, first we discussed metaheuristics-based approaches used for load balancing, secondly, we discussed classification-based approaches used for load balancing.

A. METAHEURISTICS BASED APPROACHES USED FOR LOAD BALANCING

One of the NP-hard problems in cloud computing is resource scheduling which is performed using load balancing. Researchers have performed a lot of work to find out the optimum solution to this problem but still, there is a need for improvement that further enhances the optimization of resources. Table 1 describe the summary of scheduling algorithms that are most commonly used in cloud computing. In cloud computing, tasks are submitted to the available VMs' that use heuristic or metaheuristic algorithms to provide the optimum solutions which are generally not solvable within time by the classical deterministic algorithms. Metaheuristic algorithms can solve NP-hard problems within constraints such as time, space, robustness, and providing many feasible solutions.

A dynamic elastic load balancer D-ACOLB is proposed by [64] which is based on ACO that aims at reducing the throughput and makespan of the system. The algorithm has been tested using CloudSim on up-to 1500 tasks gradually increased from 300 tasks. It has been shown that the proposed algorithm performed better than ACO, MACO, and FCFS. However, this algorithm is the only metaheuristic considering few parameters with fewer tasks. Further, due to a smaller number of tasks, the scalability of the algorithm is not discussed along with time complexity measures. Similarly, in the real scenario of computation using cloud, complex and huge diversified tasks are present for which other factors like SLA violations, overhead, optimization, migration time, and scalability needs to be considered. Since (ACOFTF) is a hybrid model, it not only considers these parameters with a number of complex tasks but also provides better efficiency and scalability in a virtualized cloud environment.

B. CLASSIFICATION BASED APPROACHES USED FOR LOAD BALANCING

SVM classifiers are extensively used in cloud computing in combination with metaheuristics. One of the studies discussed intensification and diversification using scheduling in cloud computing showing that there is a need to maintain a balance between them so that quality solutions are achieved [7]. It has been observed that a careful combination of various metaheuristics results in a more efficient performance, accuracy, and strong convergence because the best features of metaheuristics are combined into a single metaheuristic. As a drawback of this study, only response time is considered and it further lacks a multi-objective approach. In one of the studies by [65], SVM is combined with cloud scheduling algorithm to obtain better performance efficiency with good accuracies in classification. Studies have shown that in metaheuristics, modifications are required in operators used, fitness function, and their hybrid with proper optimizations. This work considered only a few metaheuristics that are discussed for intensification and diversification. A study by [66] discussed the firefly algorithm adjusted using fine-tuning with SVM for error rate classification. The approach did not consider energy efficiency, throughput, and response time to provide overall effectiveness of the solution. Hybrid of SVM and Firefly taking spot size radius as optimization is proposed by [67]. Experiments have shown that this algorithm has outperformed GP, ANN, and SVM. However, classification errors are not properly minimized resulting in misclassification in certain cases. In research [68], authors focused on resource prediction using SVM for estimation of their distribution. In their study, a fitness function is exploited for the VM having maximum resource utilization capabilities. However, this research lacks better coefficient of error estimation along with number of QoS metrics. A study by [69] discussed network anomaly detection intrusion system NIDS used for monitoring and analyzing the network in cloud computing. In this study, SVM served as a classifier whereas, Binary Particle Swarm Optimization (BPSO) chooses the respective features. The study considered only fitness function leaving the best solution set questionable and there is also a lack of scientific evaluations. Optimized parameter determination in SVM is discussed by [70] that focused on Feature Selection (FS) in an image. In their study, results were evaluated using McNemar's test showing 12% overall accuracy with the help of SVM. In this approach, only a small number of features are used which needs to be extended to determine the scalability of the system. Classification technique for the detection of beverages using tongue is suggested by [71] that aimed to produce the best classification accuracies as compared to various classifiers but,

TABLE 1. Summary of scheduling algorithms in cloud computing.

Reference	Technique Utilized	QoS Metrics	Evaluation Tool	Advantages	Limitations
Arabinda et. al [32]	Scheduling (Load balancing)	Survey	CloudSim	Emerging areas identified	Only basic survey
Mostafa et. al [33]	Metaheuristic (CSA+WSC)	Cost Response Time	CloudSim	Reduced cost Reduced response time Availability Reliability	Limited number of services in an experimental setup, Increased Computation cost with an increased number of requests
Monire et. al [34]	Scheduling (Energy aware)	Energy Execution time SLA violations	CloudSim	Better energy efficiency Better Resource utilization Minimum Execution time Low SLA violation	Limited number of tasks, Limited process deadlines
Ashmeet et. al [35]	Scheduling (Load balancing)	Response Time Cost	CloudSim	Reduced Response time Better Datacentre processing time Low Cost	Number of tasks are not mentioned, Complexity not discussed
Mostafa et. al [36]	Resource provisioning framework (ANFIS + Fuzzy decision tree)	Cost Response Time	CloudSim	Reduction in cost Accuracy Reduced response time High Correlation between ANFIS and experimental data	The parameters like energy efficiency, throughput and few others are not discussed
Mostafa et. al [37]	Controlling Elasticity Framework (ControCity)	Elasticity Response Time Resource utilization	CloudSim	Highly elasticity value, Low response time High resources utilization	Scalability issue
Mostafa et. al [38]	Scheduling (MFO based)	Makespan Execution Time Transfer Time	iFogSim	Reduced makespan Reduced execution time Reduced transfer time Faster fitness convergence	Only 20 nodes are used requiring modes nodes to establish the scalability
Elham et. al [39]	Scheduling (BWM+VIKOR)	Throughput time Makesoan VM Utilization	CloudSim	Better throughput, reduced makespan, Less waiting time More VM utilization and less VM usage cost	Tasks and VMs' needs to be increased
Reihaneh et. al [40]	Scheduling (BWM+TOPSIS)	Energy Consumption Makespan Resource Utilization	CloudSim	Reduced makespan, Better Energy consumption High VM utilization	Large scale datacentres need to be considered Reliability is an issue
Monireh et. al [41]	Scheduling (PL+DVFS)	Execution Time SLA violation Energy Consumption	CloudSim	Reduced execution time Minimum SLA violations Maximizing energy saving (38%) and resource utilization (11%)	Increased number of VMs' needs to be considered along with other QoS metrics and diversified user requirements must be used
Reihaneh at. Al [42]	Resource provisioning (Self-Learning Fuzzy approach)	Cost Response Time Resource Utilization	CloudSim	Correct prediction of workload overall increased performance Reduced resources cost Reduced response time	Varying diverse user requirements are not considered
Strumberger et. al [43] Adhikari et. al [44]	Scheduling (MBO) Scheduling (LBRC+BAT)	Robustness Efficiency Accuracy Decision making Merging clusters	CloudSim CloudSim	Improved convergence Improved efficiency Improved accuracy Better decision making Merging similar clusters	Throughput and response time are not addressed Scalability issue Datasets diversity
Amanpreet et. al [45]	Optimization HDD-PLB framework (HPA+HHA)	Makespan Execution cost	CWS	Better Makespan Lower execution cost	Fewer VMs' (20) are considered, Scalability issue

TABLE 1. ***(Continued.)*** **Summary of scheduling algorithms in cloud computing.**

Strumberger et. al [46] Torabi et. al [47]	Scheduling (WOA-AEFS) Scheduling (IRRO-CSO)	Execution time Convergence Execution time Response time Throughput time	CloudSim CloudSim	Better execution time Faster convergence Faster convergence Better execution, response, and throughput time	Fewer datasets are used, Scalability issue Decision making capability needs improvement
Strumberger et. al [48]	Scheduling EHO+ TGA	Accuracy Location search Consistency	CloudSim	Better accuracy Faster location search Consistently well	Fewer nodes are considered, Few localization errors
Attiya et. al [49]	Scheduling HHOSA (HHO+SA)	Makespan Job scheduling	CloudSim	Reduced makespan Better scheduling	Fewer tasks are considered, More QoS metrics required
Susila [50]	Scheduling (EELBF) (EFCFS)	Energy consumption Response time Computation time	Eucalyptus	Improved energy consumption, response and reduced computation time	No machine learning technique is used, Scalability issue
Kumar et. al [51]	Scheduling (ICSO)	F-Measure CEC functions Clustering problems	MATLAB	Better CEC function results on 12 benchmarks Better clustering	QoS parameters such as response time, energy consumption and throughput are not discussed
Nazia et. al [52]	Hybrid metaheuristic algorithm HBMMO	Execution time Makespan Throughput	CloudSim	Better execution time Low makespan time High throughput time	Energy consumption is not considered
Zhong et. al [53]	Scheduling (PSO+ WWSVMLP)	Accuracy prediction Execution time	Google Cloud	Better efficiency	Low prediction accuracy
Ashouraei et. al [54]	Scheduling (SLA aware load balancing)	Energy consumption SLA violations Migration time	MATLAB	Better energy consumption Few SLA violations Less migration time	Throughput and execution time are not considered
Sharma et. al [55]	Scheduling (SLA agile-based VM)	SLA violations	CloudSim	Reduced SLA violations	Dynamic workloads are not used, QoS metrics are missing
Ajay et. al [6]	Scheduling (Optimizing SLA violation cost)	SLA violations Penalty cost	CloudSim	Fewer SLA violations No penalty cost	Only SLA is evaluated leaving other QoS metrics unattended
Song et. al [56]	Scheduling VM migration using MMA	Communication cost Overload	CloudSim	Low communication cost Low overhead	Saturation caused performance degradation, Computational complexity
Jonathan et. al [57]	Scheduling (P2P)	VM Migration	JXTA	Few VM migrations	Overhead cost, More computational cost
Jamal et. al [58]	Metaheuristics VANET Optimization	Energy efficiency Network overhead	NS2	Better energy consumption Reduced overhead	Performance degradation More computational cost
Saleem et. al [59]	Scheduling ACO using BIOSARP	Energy efficiency Overhead	NS2	Higher energy efficiency Reduced overhead	Performance efficiency
Mohanty et. al [60]	Task Scheduling MPSO	Task overhead Resource utilization	CloudSim	Minimize task overhead Maximize resource utilization	Few VMs' (10) and tasks (20) are considered leaving scalability issue unattended
Hajimirzaei et. al [61]	Scheduling MLP+ABC	Accuracy	CloudSim NSL-KDD	Improved MAE Improved RMSE Better kappa value	The only accuracy compared QoS metrics are not evaluated
Mohan et. al [62]	Scheduling HIGA (HS+GA)	Task overhead Energy consumption	CloudSim MATLAB	Low overhead Low energy consumption	Experiments conducted in a limited environment
Wendong et. al [63]	Optimization HMGCG GWO+SOS+GA	Response time Accuracy	MATLAB	Quick response time High accuracy	Computationally complex

the presented approach is computationally intensive. Convolutional Neural Network (CNN) for classification detection of frauds in credit/debit cards is proposed by [72]. This technique provided good accuracy for detecting frauds during transactions. However, the technique has performance constraints. Further, comparative accuracy and performance need to be checked on more datasets.

In [73], authors proposed CNN for the classification of animals using large datasets of animal images. This study used a hybrid approach of CNN with SVM in which training of the images is performed using CNN, and multiclass is obtained as an output. Here, classification accuracy is only performed using F1-score and there is a need to extend the confusion matrix along with K-fold CV testing. In [74], N-SVM is proposed by training all layers of deep learning using SVM. Using the standard datasets, the approach proves better than SVM. As a drawback, few datasets are used with a large number of features and performance comparison needs to be determined using more confusion matrix and K-fold CV. SVM classification using a Decision tree is presented by [75] in which a huge number of classes are established. Experiments are performed by comparing NN with SVM linear Kernel and SVM polynomial Kernel and it is proved that SVM with polynomial kernel outperformed NN. However, NSVM takes a lot of time due to large number of complex parameters. Image classification using RBF neural network for the optimization of GA is presented by [76]. This research is used to train RBF-NN classifier and GA is applied for the optimization of parameters. An overall 11% enhanced accuracy is achieved in remote sensing image classification. In this case, bidirectional misclassification and error analysis are not considered. Further, only accuracy is discussed but not proved through confusion matrix. Decision Trees (DT) is a predictive classifier used for pattern recognition. DT is an indicator of a logic-based supervised learning approach primarily utilized to classify the data, that produces a collection of ranges to take decisions to determine the category of unidentifiable information. DTF algorithm operates quickly with enough precision. SVM is a commonly used algorithm that works effectively on relatively smaller datasets. NN-based classifier is an identification system in which the classification of a dataset item is performed based on neighboring votes. In the cloud environment, several classification techniques based on K-NN are implemented. K-NN neighbor preserves all accessible information and calculates the class of a specific instance. Naive Bayes classifier depends on Bayes theorem that possesses strong independent assumptions for features used in various tasks-based classifications. A study by [77] discussed anomaly detection that is solved by a hybrid of SVM and NN classifiers. Experiments have shown that by using weights elements, the error metric is minimized in a way that higher classifier gets a higher weight and low accuracy gets lower weight. However, this research considered the only accuracy and has higher computation time. An improved voice pathology classifier as a diagnostic tool is proposed by [78]. Classification is performed using SVM for classifying standard and pathology speech using the extracted features. Experiments have shown that the adopted system comprising of SVM and Naïve Bayes classifier produced 98% and 94% accuracy, respectively. However, details regarding the number and type of datasets are missing and only a single classification factor is discussed. In [79], the authors proposed audio segments classification data applied to call centers using Naïve Bayes and SVM. Experiments have shown that SVM achieved 83% accuracy when MFCC is used with first and second derivatives and further 87% is achieved when Naïve Bayes is used. However, classification using balanced datasets is important and needs to take into consideration. Text classification of documents is suggested by [80] in which testing and training of documents is performed using SVM and Naïve Bayes classifiers on ten categories. Results have shown that SVM text classification with 85% accuracy has attained better results as compared to Naïve Bayes. However, performance needs to be checked on large scale datasets using evaluations of confusion matrix and K-fold CV. A study by [27] discussed the classification of oral images of tooth diseases. Overall, 72 X-ray images are used with five classes comprising of 5 dental impairments. SVM has attained better accuracy (an average of 100% with no false alarm rate) as compared to Naïve Bayes (62% with 17% false alarm rate). Fewer images (72 only) are classified with very small features set considering it only suitable for small one dimension datasets. Personality trait classification is discussed by [81] in which Naïve Bayes produced better results with 63% accuracy than both SVM and K-NN classifier. However, this classification approach attained less than 70% accuracy. A Random Forest (RF) is another type of classifier consists of many sub decisions trees which are used for classification of data. Several votes are used from each subtree to make the final classification. In [82], five common tree species classifications are proposed in which SVM has attained a classification accuracy of 77%. Experiments are conducted on small datasets and overall accuracy in the competitors is less than 80%. A study by [83] suggested the classification of three expansive plant species using SVM and RF for their accurate identification of hyperspectral images. However, in this study, a low F1 score is achieved even when a small number of training sets are used. Classification of sentiment analysis is proposed by [84] in which text is analyzed using a hybrid RF-SVM classifier which has shown better accuracy of 83.4%. In this study, only 1000 reviews are analyzed that makes the results only suitable for small scale classification. Random Multimodal Deep Learning Algorithm (RMDLA) is proposed by [85] that can solve complex classification problems while maintaining 95% accuracy. However, the approach is complex and computation-intensive. A study by [86] presented an improved and modified RF algorithm "WRTF" for classifying text in higher dimension space by categorizing hundreds of documents. Experiments have shown that WRTF has improved performance classification over SVM, NB, KNN, BRF, and TRF on given six datasets. However, the study lacks scientific evaluations. The above metaheuristic and classifiers studies have taken either single or few multi-objective parameters but most of the time they lack accuracy, slow convergence, lacking global optimum, smaller or less diversified datasets and proper load balancing in the presence of large diverse datasets.

III. PROPOSED ACOFTF APPROACH

Load balancing faces one of the challenges to distribute a large amount of data and allocate a suitable resource at the time of task allocation. One of the challenges of task scheduling in cloud is to assign the tasks to different VMs so that the load balancing is achieved with minimum resources. The advantage is to better utilize the resources on cloud and fulfill the demands of users in a timely manner.

We have developed a hybrid approach called ACOFTF for efficient load balancing in cloud computing, in which we combined SVM process with Ant colony metaheuristics. The architecture of proposed approach is shown in Fig. 1 and is initiated with the input data and proceeds with the classification process which is then followed by load balancing. The process starts with the collection of data inputs in the form of videos, texts, audios, and images which are stored in the cloud environment. Data classification is then performed using SVM, which gives the data class in the form of output. Then load balancing of data is carried out using ACO.

We have selected ACO proposed by Dorigo *et al.* in 2006 [87] by considering its capabilities such as: ACO is dominant over genetic algorithms and simulated annealing approaches, as the convergence time of ACO is faster than genetic algorithms and simulated annealing approaches [88]–[90]. ACO has the ability of adapting changes continuously when the requirements change dynamically [91], [92]. Ants can find a high-quality solution in a search space and they share their knowledge in the form of pheromone update strategy and solve the problem efficiently. Further, ACO has been utilized in solving the load balancing problem that results in reduced computational time and has an efficient global search that does not fall into local optima [93]–[95], [103].

A. EXPLANATION OF ACOFTF ALGORITHM

Algorithm 1 takes various formats as inputs such as audio, video, image, and text from the cloud and performs classification using one to many classification techniques. The algorithm iterates 100 times before it assigns the data to the proper class. To entertain high dimension complex data, POLYSVM kernel is used. The output of this SVM classifier is a data class. The same is shown from Line 1 to 8. In the second part of ACOFTF, load balancing is performed over the classified data that is described from Line 9 to 27. Here ACO is used that performs task scheduling and as a result, it returns the complete scheduled data. The main modules of proposed architecture are discussed as below:

1) INPUT DATA

The input data is collected from the cloud source to feed into the system. The collected data has a type format of video, audio, text, and images.

Algorithm 1 ACOFTF

Input: video, text, audio, image, number of virtual machines (VM), ants: number of ants, i: iteration, α: pheromone influence factor, β: local node influence factor, ρ: pheromone evaporation coefficient
Output: Data class, Scheduled data
1: **for** data classification **do**
2: **for each** $P(u, v)$ **do**
3: Evaluate ← Kernel SVM
4: **for each** Classification accuracy$\neq$ 100 **do**
5: Evaluate data accuracy
6: **if** Number of iterations $\neq N$ **then**
7: perform data categorization
8: **return** data class
9: **for** load balancing **do**
10. m ← num of ants.
11. Set iteration ← 0
12. **while iteration** < maximum_Iteration **do**
13. Place m ants on cloud (a network of virtual machines).
14. Generate random order of VMs' in the trail for each ant k using Fisher-Yates shuffle.
15. Initialize pheromones to each trail t_i by 0.1.
16. **for** each task 1 *to* n, do:
17. **for** each virtual machine 1 *to* m, do:
18. Construct a trail t_i and Initializing k ant for each trail.
19. Place the node VM in U_k where $k = 1$ to m //U_k is a list that keeps the record of each visited node by ant k.
20. Compute the probability of ant k from active node $VM1$ to select node VMj, using equation (5).
21. **end for**
22. Allocate task to VM that has a high probability
23. Update the time with respect to equation (10).
24. **end for**
25. Update pheromones and increase the value of pheromones using equation (6) and (7).
26. **end while**
27. **return** schedule data
28. Exit.

2) CLASSIFICATION OF DATA USING SVM

The data collected are then classified with the help of SVM. For these types of cases SVM introduces Kernel function to change the original data space into a higher dimension space having a function that includes the transformation function with dot product. Now the hyperfunction is given as:

$$K\left(u_i, u_j\right) = \varphi\left(u_i\right)\varphi\left(v_j\right), \tag{1}$$

$$f\left(u_j\right) = \sum_{i=1}^{N} \propto_i u_i K\left(u_i u_j \,|\, u_i\right) + c, \tag{2}$$

where, u_i is used for support vector, $\propto_i$ is represented as Lagrange multiplier and u_j is known as label of membership class $(+1, -1)$ where $n = 1, 2, 3 \ldots .N$.

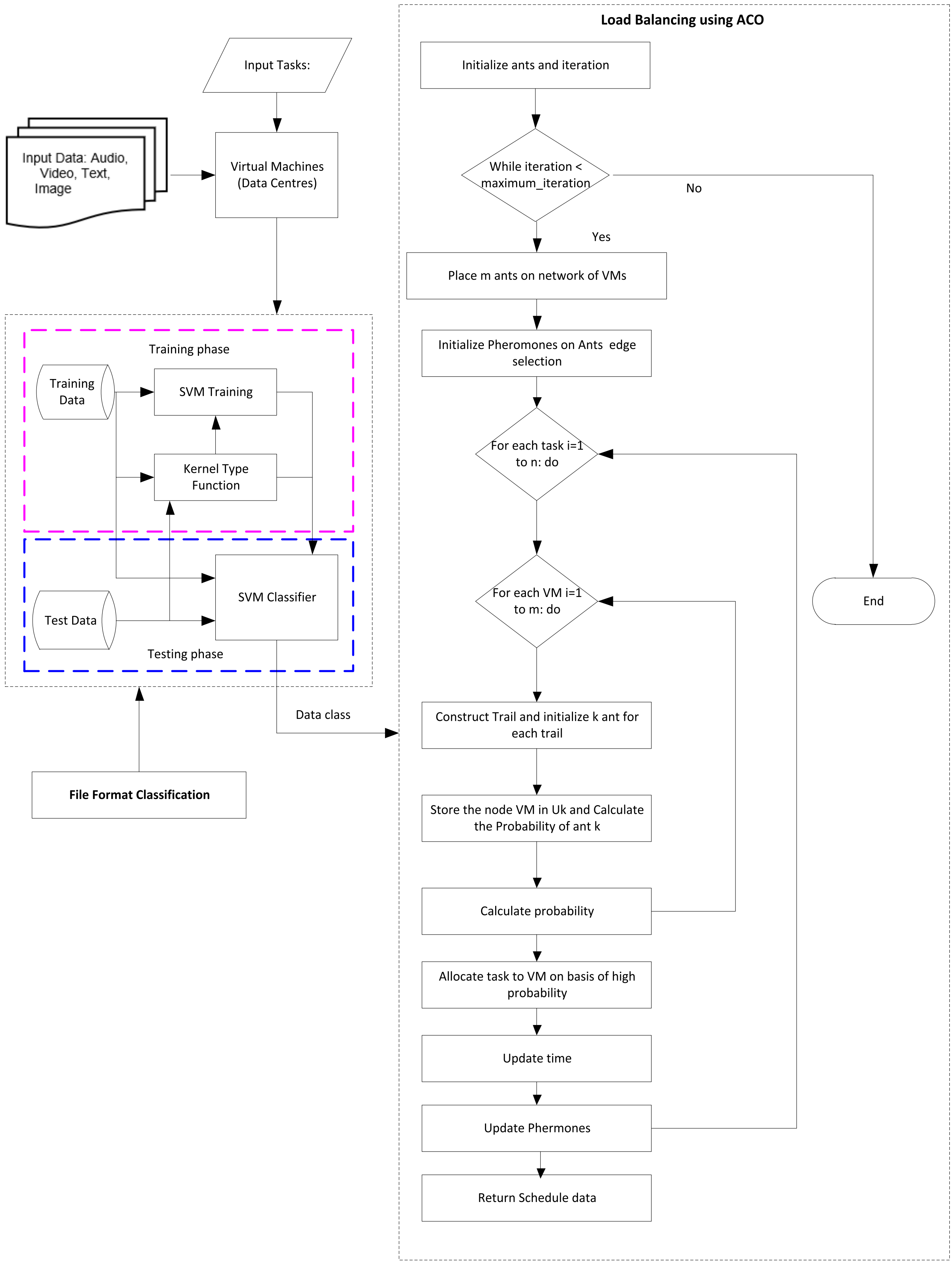


FIGURE 1. **Architecture of ACOFTF approach.**

As our inputs consist of various forms of data, so in order to make the data linearly separable with a given non-separable data, we choose POLY function that is

$$POLY\ (u, v) = \left(\left(u^k v + 1\right)\right)^s. \tag{3}$$

where 's' is the polynomial degree. A polynomial kernel is defined as,

$$K\ (x, x_i) = 1 + \sum (x * x_i)^d\ . \tag{4}$$

where polynomial degree must be selected as per learning algorithm. When $d = 1$, this confirms to linear kernel. The polynomial kernel is suitable for curved lines in the input space.

3) LOAD BALANCING USING ANT COLONY OPTIMIZATION

Let us assume that $VM_1, VM_{2,} \ldots . . VM_n$ is the set of virtual machines and each machine is responsible to execute one task. Each task is executed for a period of 100 iterations and is evaluated using computational cost in the form of time. The mapping of tasks on virtual machines is computed using a metaheuristic algorithm called ACO where each machine is assigned a task based on available resources in cloud environment.

We present the network of VMs (virtual machines) in the form of undirected weighted graph as shown in Fig. 2. The VMs network can be represented as an undirected graph G = (V, E) where V represents the virtual machine (VM) or node and E represents the undirected edge having pheromone weight that shows the overload and underload intensity between two nodes and is updated in the form of pheromone.

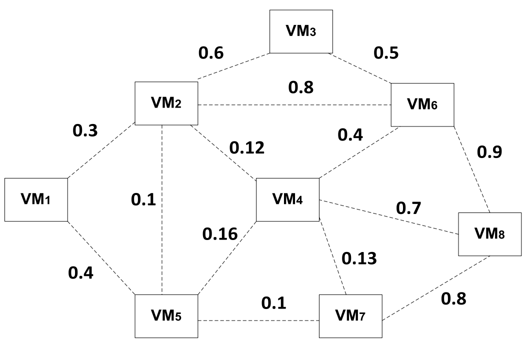


FIGURE 2. VMs network.

a: INITIALIZING PHEROMONE

In our proposed approach, we set the initial pheromone level as 0.1. Initial pheromone value lies between two nodes that is VM_i and VM_j. After first iteration, this pheromone level is globally updated.

b: COMPUTING PROBABILITY

Each ant 'k' moves from current node i(VM) to next node j(VM) by calculating the probability of ρ_{ij}^k of crossing the edge using the following equation

$$\rho_{ij}^k = \frac{(\tau_{ij})^\alpha (n_{ij})^\beta}{\sum_1^n (\tau_{ij})^\alpha (n_{ij})^\beta}. \tag{5}$$

where N_i^k are the neighbors of ant k; The probability ρ_{ij}^k from node i to node j depends upon two parameters that are pheromone level τ_{ij} and desirability of moves from node i to j, which is denoted by $\eta_{ij}\alpha$ and β are used to control the influence of τ_{ij} and η_{ij}.

c: UPDATING PHEROMONE

The amount of pheromone shows the type of nodes (VM), the ant is searching. A greater amount of pheromone along with trailing path shows that the target node is overloaded so the ant will try to find another path with less amount of pheromones, that is after encountering an overloaded node it will find the underloaded node and assign a task to that node.

In ant colony optimization, a pheromone is updated locally and globally. Local pheromone is updated by using equation (6).

$$\tau_{ij} = (1 - \rho)\ \tau_{ij} + \rho\tau_{ij.}^0 \tag{6}$$

where τ_{ij} is the pheromone level from node i to node j, when each ant traverses an edge ij, ρ is constant pheromone evaporation coefficient and τ_{ij} is the initial pheromone level on edge ij. Second level of pheromone is global pheromone which takes place at the end of each iteration when all k ants have constructed the paths. The global pheromone level is computed using equation (7).

$$\tau_{ij} = (1 - \rho)\ \tau_{ij} + \frac{\Delta\tau_{ij}}{L^k}. \tag{7}$$

where 'm' is number of ants and $\Delta\tau_{ij}^k$ is the amount of pheromone deposited by ant k at edge ij in one iteration. L^k is the length of the trail t_i that k ant constructed. Large value of $\Delta\tau_{ij}$ increases the amount of pheromone level on each edge of the constructed paths as the time passes and is computed as:

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k \tag{8}$$

$$\text{where } \Delta\tau_{ij}^k = \frac{1}{CompletionTime} \tag{9}$$

$$CompletionTime_{(t_i,m_j)} = StartTime_i - EstTime_{(t_i,m_j)} \tag{10}$$

The completion time is computed using equation (10), as start time of the task is depending upon the completion time of task that is previously assigned to the respective machine. This time is helpful in load balancing. Here, $StartTime_i$ is assigned randomly to the task i, when the machine is available, and $EstTime_{(t_i,m_j)}$ is the time estimate to complete the task i at machine j. As the proposed model follows a hybrid approach, we conducted several experiments with different parameter settings with ants $k = 4$, iterations $i = 100$, $\alpha = 3$, $\beta = 2$ and $\rho = 0.1$ in our ACOFTF algorithm. These parameter settings are chosen based on convergence of ACOFTF algorithm after conducting several experiments.

B. COMPLEXITY OF ACOFTF

We have computed the evaluation of ACOFTF model based on time complexity. As our model follows a hybrid approach that is the combination of SVM and ACO, so we have used parameters which are specified in evolutionary algorithms as well as classification algorithms. From Algorithm 1, the computational time for Steps 1 to 8 is O(N^3). A data class is then passed to ACO Algorithm so Line 9 to 11 with N iterations, so it takes O(N). The size of VMs is $l = |SM|$. To apply the ACO, a fully connected undirected graph with l virtual machines is created. Then, the initial weight is assigned and then probability is computed so Line 11 to 20 takes $O(l + N^2)$. After that the task is allocated to machine and then time and pheromone are updated, from Lines 22 to 25, it takes $O(1)$. The overall time complexity of proposed ACOFTF is $O(N^3 + N + l + N^2 + 1)$. Finally, the overall time complexity of proposed ACOFTF algorithm is $O(N^3)$.

IV. EXPERIMENTAL SETUP

In this study, datasets are used in equal proportion for each category of audio file type, video, image, and text. A total of 100,000 datasets are used in which 70% of the files are used for training and remaining 30% are used for testing. Each data class has varying file sizes, such as audio datasets are 9 GB, video datasets are 19 GB, image datasets are 14 GB and text data sets are 6 GB. Datasets/File types are used interchangeably throughout the study. Datasets are public and manually developed collected from UCI, YouTube and own sources [95], [96]. In all cases, same quantity of training and testing is maintained. Implementation of this study is conducted on a desktop computer with specifications such as Core i7 processor, 12 GB RAM, 1 TB HDD and Windows 10 enterprise edition. Simulations are conducted in CloudSim 4.0 and MS Excel 2016. Number of configuration settings are done in CloudSim 4.0 for running simulations with resources like data centers (2-16), hosts (2-32), VMs' (5-1000), tasks (1000-14000) and task size (1MB-1GB). Table 2 shows a summary of the datasets used in this study with proportions.

TABLE 2. Data sampling.

Sr.	File Type	Files Sampling		Size (GB)
		Training	Testing	
1	Audio	17500	7500	9
2	Video	17500	7500	19
3	Image	17500	7500	14
4	Text	17500	7500	6
Total Files & Size		70000	30000	48

A. ACCURACY OF FTFSVM

In order to check the accuracy of the developed algorithm FTFSVM (File Type Formatting using Support Vector Machine), performance metrics are used given in Table 3 which shows its average performance taken together from audio, video, image and text. Performance metrics comprising of accuracy, sensitivity, specificity, precision, recall, F-Measure, G-Mean, Area Under Curve (AUC), Kappa, and Mathews Correlation, are used to provide the classifier performance. On average, highest classification performance is observed in the developed classification model. This shows that FTFSVM is classifying files quite accurately which will have a huge impact when used in scheduling.

TABLE 3. Combined results of FTFSVM on performance metrics.

Metric	Audio	Video	Image	Text	Average
Accuracy	0.98022	0.98177	0.97499	0.99925	0.984057
Sensitivity	0.92788	0.93622	0.97199	0.99891	0.958749
Specification	0.99095	0.99108	0.94478	0.99929	0.981524
Precision	0.95424	0.95508	0.98224	0.99851	0.972518
Recall	0.92788	0.93622	0.97199	0.99891	0.958749
F-Measure	0.94085	0.94548	0.97708	0.99871	0.96553
G-Mean	0.95876	0.96306	0.95797	0.9991	0.969724
AUC	0.99043	0.97308	0.96351	0.9989	0.98148
Kappa	0.92897	0.93454	0.90837	0.9981	0.942496
Matthews CR	0.92912	0.93466	0.90864	0.9981	0.942632

In Table 4 some of the well-known classifiers are taken from literature such as Random Forest (RF) [97], Naïve Bayes (NB) [97], K-Nearest Neighbor (K-NN) [98] and Convolutional Neural Network (CNN) [98] in which their classification performance is compared on the same set of datasets. The overall performance of FTFSVM is better as compared to other classifiers. The same effect can be seen in Fig. 3 that shows the comparative performance of FTFSVM with others. The values are between [0,10] with 0 being nil accuracy and 1 being highest accuracy. Mathematically, on an average, 96.60% accuracy is shown by FTFSVM, followed by CNN with 95.80%, NB with 94.40%, RF with % 93.60 and K-NN with 93.20% accuracy.

TABLE 4. Comparison of classifiers.

Metric	FTFSVM	RF	NB	KNN	CNN
Accuracy	0.984	0.912	0.923	0.901	0.943
Sensitivity	0.959	0.921	0.931	0.911	0.951
Specificity	0.982	0.951	0.962	0.942	0.972
Precision	0.973	0.946	0.959	0.947	0.971
Recall	0.959	0.933	0.945	0.932	0.949
F-Measure	0.966	0.949	0.944	0.939	0.961
G-Mean	0.970	0.952	0.958	0.951	0.969
AUC	0.981	0.954	0.961	0.959	0.979
Kappa	0.942	0.925	0.929	0.919	0.941
Matthews CR	0.943	0.913	0.927	0.92	0.939

V. RESULTS AND ANALYSIS

The evaluation of the proposed model is performed by considering the parameters such as execution time, number of migrations, optimization time, throughput time and overhead time. Beside this we also compared our approach with the following baselines:

ACOPS: ACOPS is a metaheuristic hybrid algorithm combining the best features of both ACO and PSO for solving VM scheduling. ACO-PS algorithm adopts a dynamic

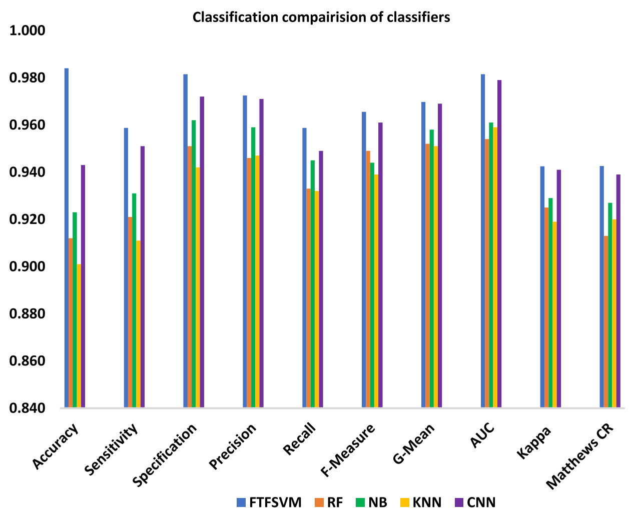


FIGURE 3. Performance comparison of classifiers.

scheduling policy to predict the workload of VM in cloud computing [28].

CPSO: CPSO improves the diversity of solutions and achieves good global convergence while focusing only on cost. The algorithm lacks a multiobjective approach using only one factor into account [29].

QMPSO: QMPSO is a new hybrid metaheuristic algorithm combining modified PSO and improved Q-learning algorithms used for load balancing in a cloud environment [30].

CSO: CSO is a metaheuristic algorithm that belongs to a swarm intelligence family and is based on the natural behavior of cats [31].

D-ACOELB: D-ACOELB is a metaheuristic algorithm based on ACO algorithm used for load balancing in cloud [64].

A. AVERAGE NUMBER OF SLA VIOLATIONS ON VARYING TASKS FROM 1000 TO 14000

Fig. 4 shows average number of SLA violations by baselines over 14000 tasks taken randomly. Further, results are generated over varying VMs' such as 5, 10, 50, 100, 500 and 1000.

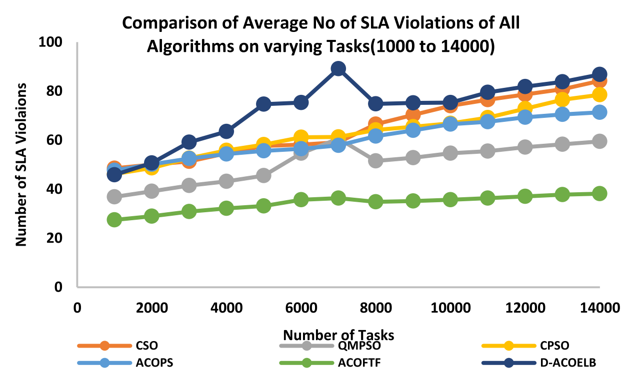


FIGURE 4. Performance comparison of average number of SLA violations of baselines over varying tasks.

Similarly, tasks are chosen as 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 11000, 12000, 13000, and 14000 randomly. It is observed that ACOFTF has done fewer SLA violations over varying tasks. The stability of ACOFTF can easily be seen in Fig. 4 with every run of tasks over baselines showing a larger number of violations.

The fact supported by Fig. 5 shows SLA violations in percentage terms with respect to ACOFTF which is taken as benchmark. Other algorithms such as QMPSO, ACOPS, CPSO, CSO, D-ACOELB have violated 12%, 18%, 20% 23% and 27% respectively with respect to ACOFTF (Since ACOFTF is set as a benchmark in the first pie chart of Fig. 5, so the value of ACOFTF is "0" which means that other baselines are relatively measured with ACOFTF in this case and rest of all subsequent cases in such figures. By considering SLA parameters such as performance, memory, and CPU cycle usage, the careful utilization of overutilized and underutilized VMs' effectively results in lowest possible violations. The same effects in consuming more energy and greater time to optimize the solution increasing computational complexity. So, keeping SLAs as low as possible is one of right things to achieve efficiency.

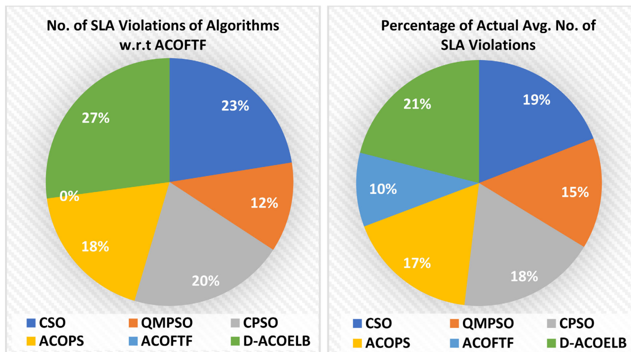


FIGURE 5. Percentage of SLA violations w.r.t ACOFTF and actual percentage of ACOFTF with baselines.

Similarly, actual SLA violations by all algorithms are also shown. ACOFTF has made 10% violations as compared to QMPSO with 15%, ACOPS 17%, CPSO 18%, CSO 19% and D-ACOELB 21%. Another important observation is that with the increase in tasks and VMs' in ACOFTF, SLA violations remain stable whereas for all other baselines, SLA violations increased greatly showing their unstable behavior. Overall, it helps in decreasing the corresponding SLA penalties and increasing customers' satisfaction.

B. AVERAGE NUMBER OF MIGRATION TIME ON VARYING TASKS FROM 1000 TO 14000

Fig. 6 shows average number of migration time by baselines over 14,000 tasks taken randomly from 100,000 datasets. It is observed that ACOFTF has done smallest migration time over varying tasks. Stability of ACOFTF can easily be seen in Fig. 6 with every run of tasks over baselines which are taking a greater migration time. The impact of

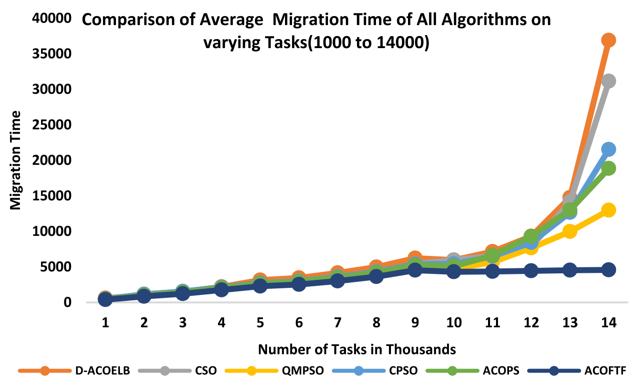


FIGURE 6. Performance comparison of average migration time of baselines over varying tasks.

VM migration time is reflected in the data center performance with varying memory sizes with every run. Soon after starting the simulation, in the very first run, few migrations to other physical hosts get activated which keeps on getting larger after every run. When a simulation is finished, the migration time is calculated for every baseline and the target performance is observed. It is important to say that the VMs' are increased with every run in which one VM migration takes the least time as compared to 2,3,4,5.... VM migrations taking more time. Other baselines have consumed severely more migration time with impact on a large number of resources consumed because, during execution, resource requirements of the VMs' may exceed the acquired resources resulting in imbalanced migrations and non-scalability.

Fig. 7 shows migration time in percentage with respect to ACOFTF which is taken as a benchmark. Other algorithms such as QMSO, ACOPS, CPSO, CSO and D-ACOELB have migrated 15%, 17%, 23%, 25% and 20% respectively. Similarly, it also shows actual VM migration time by all algorithms. ACOFTF has made 6% migrations as compared to QMPSO with 15%, ACOPS 17%, CPSO 21%, CSO 22% and D-ACOELB with 19%. An important observation is that with the increase in tasks and VMs' subsequently, migration time by VMs' is not much affected in case of ACOFTF whereas for all other baselines, a greater number of migrations are observed showing their unstable behavior.

C. AVERAGE NUMBER OF OPTIMIZATION TIME ON VARYING TASKS FROM 1000 TO 14000

Fig. 8 shows average optimization time by baselines over 14000 tasks taken randomly from 100000 datasets. It is observed that ACOFTF has optimized itself in earliest possible time over varying tasks. Stability of ACOFTF can easily be seen in Fig. 8 with every run of tasks over baselines which are taking greater time to optimize. Because of the advantage that ACO converges quickly can help in finding global optima in earliest possible time over other baselines. The same faster convergence fact is demonstrated during simulation resulting in better earliest optimization than other baselines. Further, the inherent property of quick optimization helps ACO to solve even complex problems in less computational time.

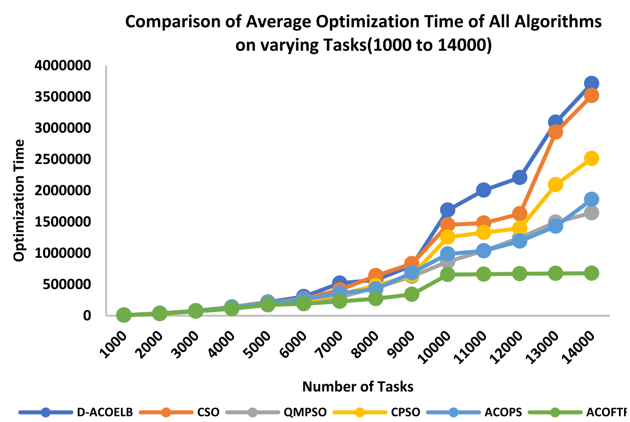


FIGURE 8. Performance comparison of average optimization time of baselines over varying tasks.

Fig. 9 shows optimization time in percentage with respect to ACOFTF which is taken as a benchmark. Other algorithms such as QMSO, ACOPS, CPSO, CSO and D-ACOELB have taken more time to optimize such as 10%, 11%, 18%, 27% and 34% respectively. Similarly, actual optimization time in

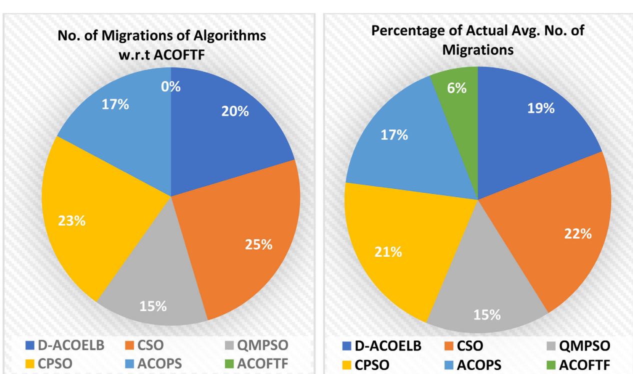


FIGURE 7. Percentage of migration Time w.r.t ACOFTF and actual percentage of ACOFTF with baselines.

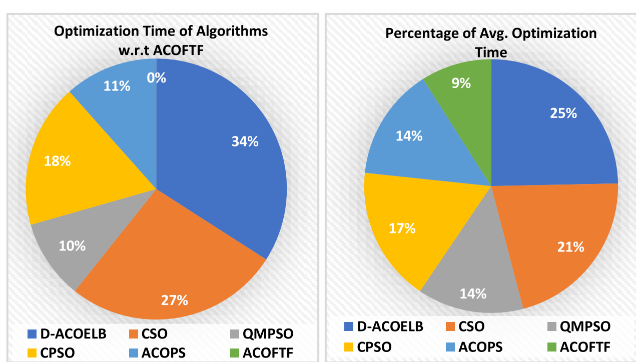


FIGURE 9. Percentage of optimization time w.r.t ACOFTF and actual percentage of ACOFTF with baselines.

percent taken by the baselines is also shown. On average, ACOFTF has optimized in 9% of total time as compared to QMPSO with 14%, ACOPS 14%, CPSO 17%, CSO 21% and D-ACOELB with 25%. This optimization behavior supports that with the increase in tasks and VMs', algorithm remains stable and scalable whereas for other baselines, time to optimize the solution jumped rapidly showing their unstable behavior.

D. AVERAGING THROUGHPUT TIME ON VARYING TASKS FROM 1000 TO 14000

Fig. 10 shows average throughput time by baselines over 14000 tasks taken randomly from 100000 datasets. It is observed that ACOFTF has shown maximum throughput time over varying tasks. This is because the earliest response time by ACOFTF helps in getting faster throughput whereas, response time in other baselines is higher resulting in low throughput (higher values). Further, the stability provided by a higher throughput of ACOFTF is shown in Fig. 10 with every run of tasks over baselines. Fig. 11 shows throughput time in percentage with respect to ACOFTF taken as benchmark.

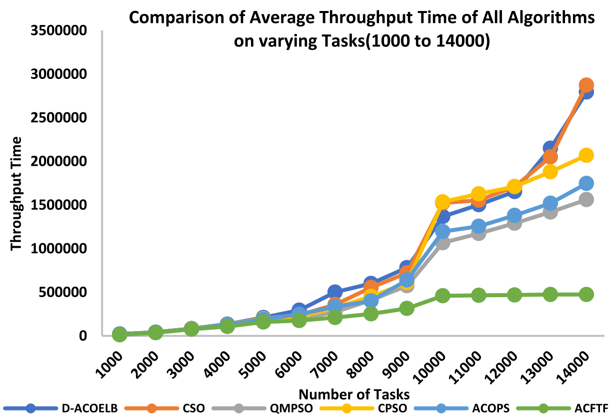


FIGURE 10. Performance comparison of average throughput time of baselines over varying tasks.

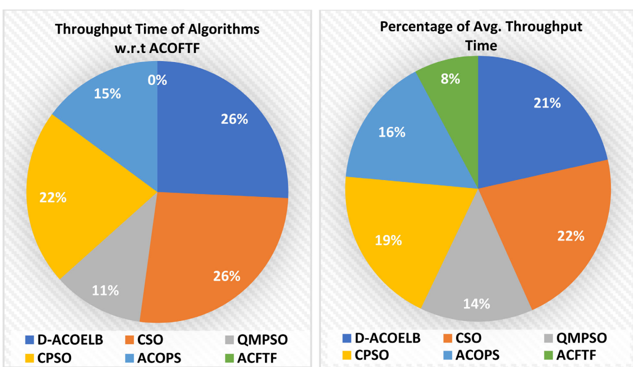


FIGURE 11. Percentage of throughput time w.r.t ACOFTF and actual percentage of ACOFTF with baselines.

Other algorithms such as QMSO, ACOPS, CPSO, CSO and D-ACOELB have shown their throughput as 11%, 15%, 22%, 26% and 26% respectively. Similarly, it also shows actual throughput in percent by all algorithms. ACOFTF provided throughput in 8% of total time as compared to QMPSO with 14%, ACOPS 16%, CPSO 19%, CSO 22% and D-ACOELB 22%. Throughput time is dependent on number of factors such as network delays, services including SLAs, hardware resources, processing power. The higher the optimization rate, the faster is the throughput and stronger the robustness of the solution. These characteristics are inherent in ACO making it a better choice than the other baselines. It is shown that with the subsequent increase in a number of tasks, net throughput is getting better and overall stable behavior of ACOFTF is depicted over others.

E. AVERAGING OVERHEAD TIME ON VARYING TASKS FROM 1000 TO 14000

Fig. 12 shows average overhead time by baselines over 14000 tasks taken randomly from 100000 datasets. It is observed that ACOFTF has least overhead time over varying tasks. Large overhead cause performance degradation and increase the computational complexity of the system. Therefore, minimizing overhead is the better way to increase the efficiency. The factors contributing to overhead such as VM migration overhead and computing resources overhead have a huge impact on the performance of the system. The ACO because of its high probability and efficiency in finding global optima can significantly reduce the overhead time.

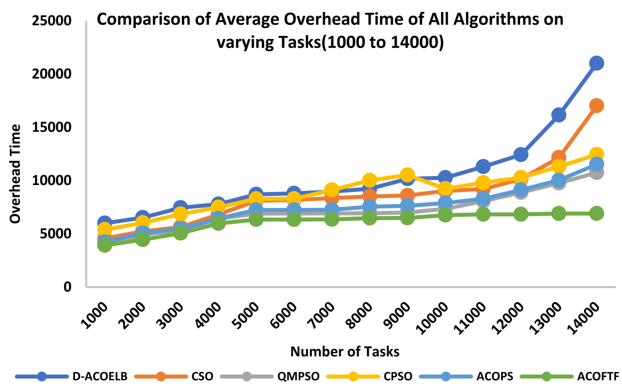


FIGURE 12. Performance comparison of average overhead time of baselines over varying tasks.

Fig. 13 shows overhead time in percentage with respect to ACOFTF which is taken as benchmark. Other algorithms such as QMSO, ACOPS, CPSO, CSO and D-ACOELB have overhead time such as 9%, 11%, 23%, 21% and 36% respectively. Similarly, actual overhead time in percent is also shown by all algorithms. ACOFTF has 12% of total overhead time as compared to QMPSO with 15%, ACOPS 15%, CPSO 18%, CSO 18% and D-ACOELB with 22%. Another important observation is that with the increase in tasks and VMs', overhead time has not frequently increased in the case of ACOFTF as compared to rapid increase in overhead of all other baselines.

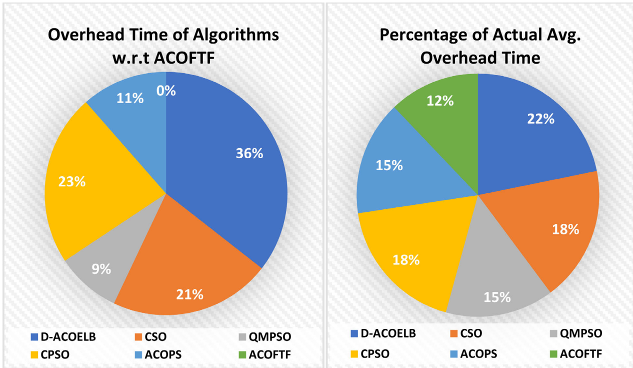


FIGURE 13. Percentage of overhead time w.r.t ACOFTF and actual percentage of ACOFTF with baselines.

F. AVERAGE NUMBER OF SLA VIOLATIONS ON VARYING VMs' FROM 1000 TO 10000

Fig. 14 shows average SLA violations by baselines over 10,000 VMs' and varying tasks from 100000 datasets. It is observed that ACOFTF has done smallest number of SLA violations over varying VMs'. This shows that ACOFTF has a stronger convergence rate and scalability as compared to other baselines. Fig. 15 shows SLA violations in percentage with respect to ACOFTF which is taken as a benchmark. Other algorithms such as QMPSO, ACOPS, CPSO, CSO and D-ACOELB have violated SLAs as 7%, 20%, 22%, 25% and 26% respectively. Similarly, it also shows actual SLAs violation in percent by all algorithms. ACOFTF has a total of 5% SLA violations as compared to QMPSO with 10%, ACOPS 19%, CPSO 21% CSO 22% and D-ACOELB 23%. Another important observation is that with the increase in VMs', SLAs are not frequently increased in the case of ACOFTF as compared to a rapid increase for all other baselines. It means that ACOFTF has quite stable and scalable behavior comparing baselines.

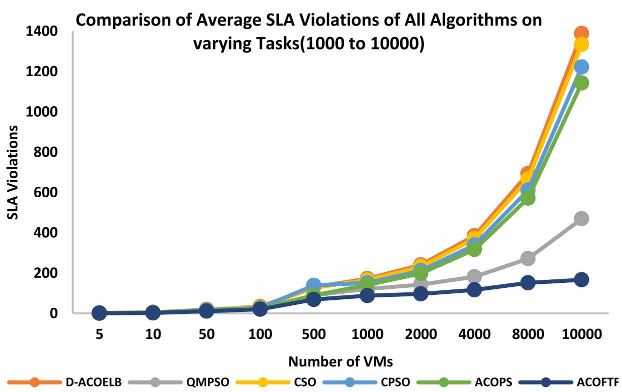


FIGURE 14. Performance comparison of Average SLA violations of baselines over varying VMs' (1000-100000).

G. AVERAGING MIGRATION TIME ON VARYING VMs' FROM 1000 TO 10000

Fig. 16 shows. comparison of Average Migration time of baselines over varying VMs' (1000-100000) It is observed

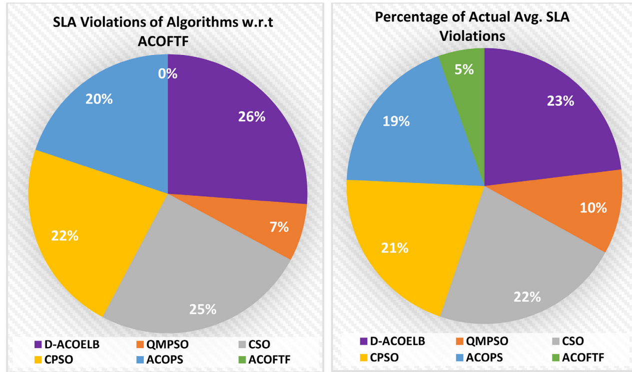


FIGURE 15. Percentage of SLA violations w.r.t ACOFTF and actual percentage of ACOFTF with baselines on VMs' (100000).

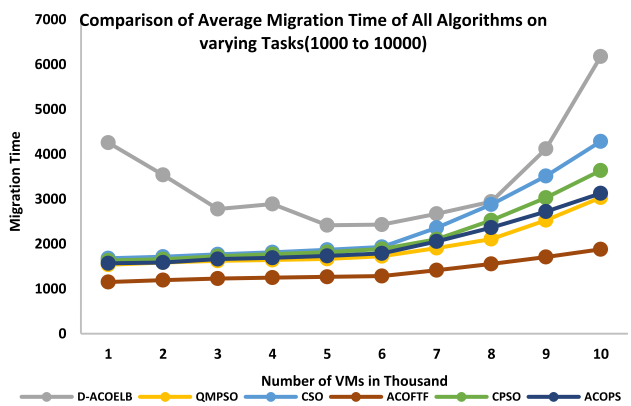


FIGURE 16. Performance comparison of average number of migrations of baselines over varying VMs' (100000).

that ACOFTF has made fewest migrations over varying VMs'. This shows that ACOFTF has a stronger convergence rate and scalability as compared to other baselines. Fig. 17 shows number of migrations in percentage with respect to ACOFTF which is taken as benchmark. Other algorithms such as QMPSO, ACOPS, CPSO, CSO and D-ACOELB have migrated 11%, 13%, 16%, 20% and 40% respectively. Similarly, it also shows an actual number of

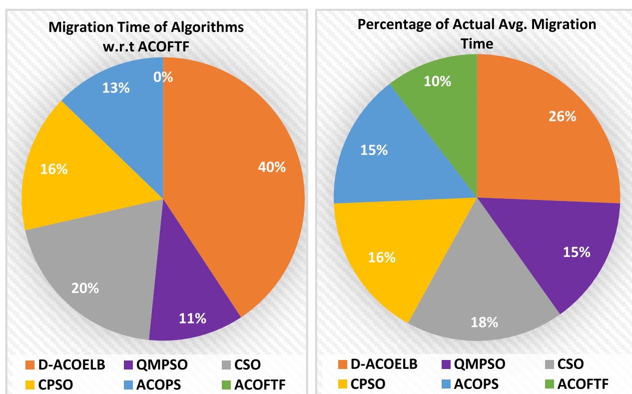


FIGURE 17. Percentage of No. of migrations w.r.t ACOFTF and actual percentage of ACOFTF with baselines on VMs' (100000).

migrations in percent by all algorithms. ACOFTF has a total of 10% migrations as compared to QMPSO with 15%, ACOPS 15%, CPSO 16%, CSO 18% and D-ACOELB 15%. Another important observation is that with the increase in VMs', migrations are not frequently increased in the case of ACOFTF as compared to a rapid increase for all other baselines. It means that ACOFTF has quite stable and scalable behavior comparing baselines.

H. AVERAGING OPTIMIZATION TIME ON VARYING VMs' FROM 1000 TO 10000

Fig. 18 shows average optimization time by baselines over 10000 VMs' gradually increased from 5 VM and varying tasks from 100000 datasets to optimize completely. It is observed that ACOFTF gets optimized itself in the least possible time as compared to baselines over varying VMs'.

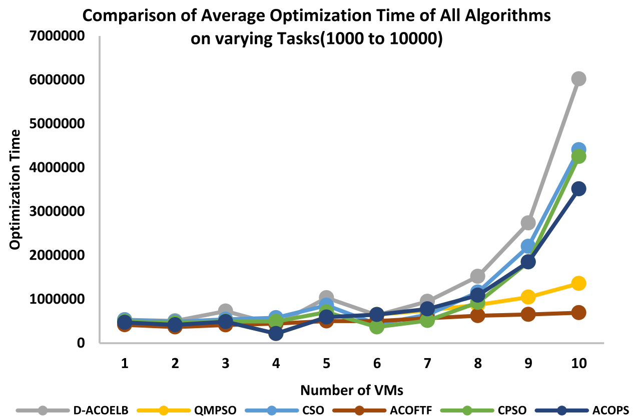


FIGURE 18. Performance comparison of average optimization time of baselines over varying VMs' (1000-100000).

Fig. 19 shows optimization time in percentage with respect to ACOFTF which is taken as a benchmark. Other algorithms such as QMPSO, ACOPS, CPSO, CSO and D-AOCELB have optimized themselves 6%, 17%, 19%, 23% and 35% of total optimization time, respectively. Similarly, actual optimization time in percent is also shown by all baselines.

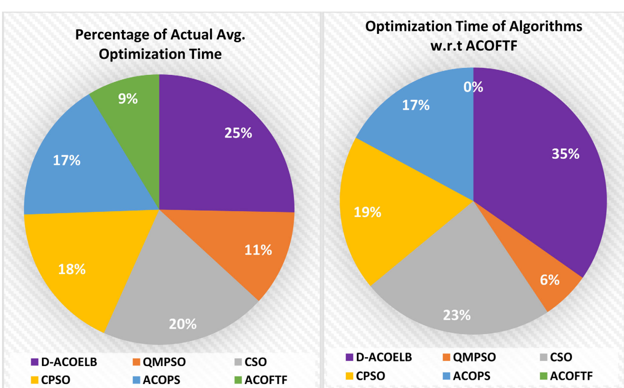


FIGURE 19. Percentage of optimization time w.r.t ACOFTF and actual percentage of ACOFTF with baselines on VMs' (100000).

ACOFTF optimizes in 9% of time as compared to QMPSO with 11%, ACOPS 17%, CPSO 18%, CSO 20% and D-ACOELB 25%. Another important observation is that with the increase in VMs', optimization time not frequently increased in case of ACOFTF where it has jumped rapidly increase for all other baselines. It means that ACOFTF has quite stable and scalable behavior comparing baselines.

I. AVERAGING THROUGHPUT TIME ON VARYING VMs' FROM 1000 TO 10000

Figure 20 shows average throughput time by baselines over up to 10000 VMs' gradually increased from 5 VM and varying tasks. It is observed that ACOFTF has taken least throughput time as compared to baselines over varying VMs'.

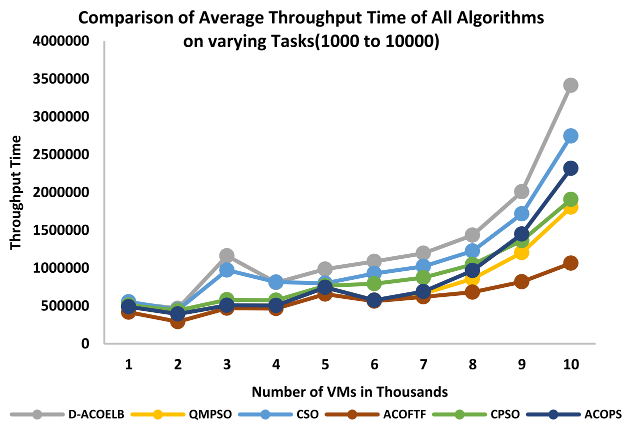


FIGURE 20. Performance comparison of average throughput time of baselines over varying VMs' (1000-100000).

Fig. 21 shows throughput time in percentage with respect to ACOFTF which is taken as a benchmark. Other algorithms such as QMSO, ACOPS, CPSO, CSO, and D-ACOELB have taken throughput time as 9%, 13%, 15%, 27% and 36%, respectively. Similarly, it also shows actual throughput time in percent by all algorithms. ACOFTF has a total of 11%

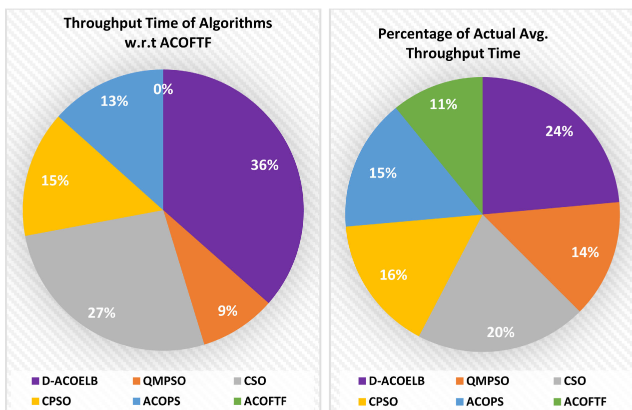


FIGURE 21. Percentage of throughput time w.r.t ACOFTF and actual percentage of ACOFTF with baselines on VMs' (100000).

TABLE 5. Statistical comparison of ACOFTF with baselines.

SLA	CSO	QMPSO	CPSO	ACOPS	D-ACOELB	ACOFTF
SD	11.874539	7.7006051	9.4662022	7.6242634	12.614	3.162963
Mean	63.564102	50.115384	61.461538	59.551282	71	33.96316
p-value	0.0006806	4.5417E-07	1.3279E-05	0.000331	6.701E-07	-
t-value	0.00039	0.00042	0.00034	0.0004	2.287E-09	-
Migration	CSO	QMPSO	CPSO	ACOPS	D-ACOELB	ACOFTF
SD	7646.2561	3452.4034	5394.7909	4968.4642	8985.78	1466.939
Mean	4528.4623	3897.3716	4414.3839	4430.2557	4962	2917.987
p-value	0.0024266	0.0003282	0.0012008	0.0016732	0.04050	-
t-value	0.324266	0.412596	0.333956	0.37975	0.02090	-
Optimize	CSO	QMPSO	CPSO	ACOPS	D-ACOELB	ACOFTF
SD	1071838.02	546027.78	785437.37	566543.82	1183082	259145.3
Mean	779374.46	512912.74	633933.84	525730.17	898411	314728.5
p-value	4.3735E-05	7.675E-7	1.5831E-06	6.001E-08	0.01776	-
t-value	0.08644	0.220345	0.12857	0.18479	2.48E-06	-
Throughput	CSO	QMPSO	CPSO	ACOPS	D-ACOELB	ACOFTF
SD	879132.31	551636.965	759429.38	601151.24	851722.4	170676.4
Mean	706226.59	525778.87	678225.03	571316.59	718490	246494.2
p-value	3.4315E-07	1.358E-07	4.1388E-07	6.794E-08	1.22E-05	-
t-value	0.046171	0.087359	0.055972	0.065556	0.009551	-
Overhead	CSO	QMPSO	CPSO	ACOPS	D-ACOELB	ACOFTF
SD	3005.5647	1745.8645	1938.6172	1852.6520	3866.21	921.141
Mean	8038.6357	6873.6168	8660.8525	7186.3077	9517	6049.5
p-value	0.0001309	0.0003090	0.0001331	7.157E-05	0.018205	-
t-value	0.006037	0.119151	0.000219	0.040104	5.26E-05	-

throughput time as compared to QMPSO with 14%, ACOPS 15%, CPSO 16%, CSO 20% and D-ACOELB with 24%. Another important observation is that with the increase in VMs', throughput time decreased in case of ACOFTF as compared to rapid increase in all other baselines. It means that ACOFTF has quite stable and scalable behavior comparing other baselines.

J. AVERAGING OVERHEAD TIME ON VARYING VMs' FROM 1000 TO 10000

Fig. 22 shows average overhead time by baselines over 10000 VM gradually increased from 5 VM and varying task.

It is observed that ACOFTF has taken least overhead time as compared to baselines over varying VMs'.

Fig. 23 shows overhead time in percentage with respect to ACOFTF which is taken as benchmark. Other algorithms such as QMPSO, ACOPS, CPSO, CSO and D-ACOELB have taken overhead time as 7%, 16%, 17%, 26% and 34% respectively. Similarly, it also shows actual overhead time in percent by all algorithms. ACOFTF has a total of 4% throughput time as compared to QMPSO with 9%, ACOPS 16%, CPSO 17%, CSO 24% and D-ACOELB 30%. Another important observation is that with the increase in VMs', overhead time is not frequently increased in the case of ACOFTF as compared to rapid increase for all other baselines. It means that ACOFTF has quite stable and scalable behavior comparing baselines.

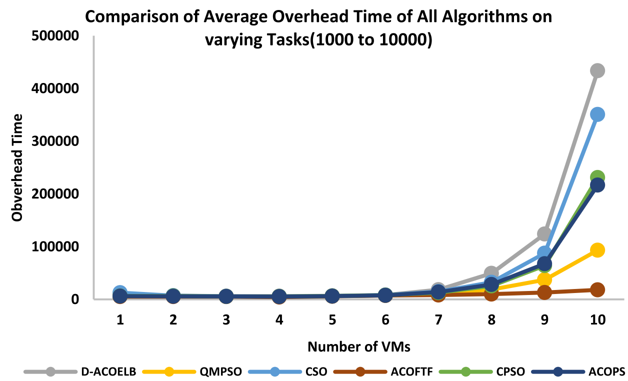


FIGURE 22. Performance comparison of average overhead time of baselines over varying VMs' (1000-100000).

K. SUMMARIZING COMPARISON

In order to check the reliability of the proposed solution and to further make statistical evaluations, there is a need to perform a t-test over the proposed algorithm and other baselines. A T-test is performed by establishing the following hypothesis:

TABLE 6. ANOVA test for QoS metrics of all baselines.

ANOVA Test for Migration Time						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	4.72E+13	15	3.15E+12	19.062	2.92E-15	1.88
Within Groups	7.92E+12	48	1.65E+11	-	-	-
Total	5.51E+13	63	-	-	-	-
ANOVA Test for Optimization Time						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	4.27E+13	15	2.84E+12	27.602	1.6E-18	1.880
Within Groups	4.94E+12	48	1.03E+11	-	-	-
Total	4.76E+13	63	-	-	-	-
ANOVA Test for Throughput time						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	6.14E+13	15	4.09E+12	18.332	6.28E-15	1.880
Within Groups	1.07E+13	48	2.23E+11	-	-	-
Total	7.21E+13	63	-	-	-	-
ANOVA Test for Overhead Time						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	2.27E+08	15	1511619	6.11538	6.9E-07	1.880
Within Groups	1.19E+08	48	2471832	-	-	-
Total	3.45E+08	63	-	-	-	-
ANOVA Test for SLA Violations						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	3663.691	15	244.246	3.21467	0.000294	1.880
Within Groups	9651.818	48	201.0795	-	-	-
Total	13315.51	63	-	-	-	-

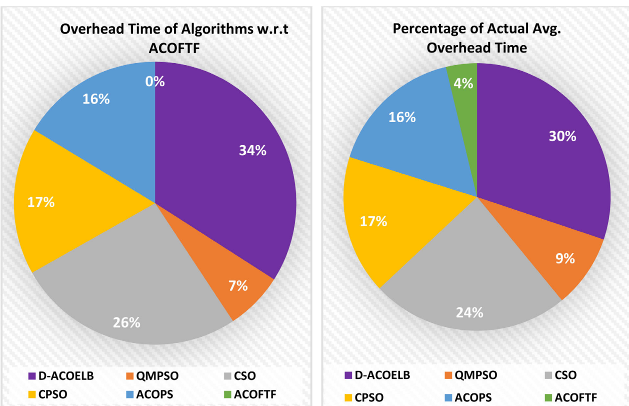


FIGURE 23. Percentage of overhead Time w.r.t ACOFTF and actual percentage of ACOFTF with baselines on VMs' (100000).

Null Hypothesis: There is no difference between ACOFTF and other baselines.

Alternate Hypothesis: There is a significant difference between ACOFTF and other baselines.

The analysis is performed to check that the results obtained are statistically significant not by chance [99]. All five QoS parameters are evaluated against four standard statistical tests including standard deviation (SD), mean, p-test and t-test. The resulting values are given below in Table 5.

The t-test is used to check whether the stated hypothesis is correct or not. Here, we specified the significance level to $p<0.05$. In the table presented above, all p-values are less than 0.05 which means that there is a significant difference between ACOFTF, and other baselines as shown by t-values. Further, all factors such as SLA violation, migration time, optimization time, throughput time and overhead time have shown a significance level of variance less than 0.05 in the t-test which is enough to say that the null hypothesis is rejected and the alternate hypothesis is accepted. In Table 5, a comparison of every algorithm is made separately with ACOFTF in all parameters by averaging their values that result in separate t-values.

We can also use ANOVA (Analysis of Variance) test to make multi comparisons at once in each parameter for all algorithms. ANOVA test is used to compare the mean of two or more groups to see if the difference is significant. So, ANOVA can be applied to verify the statistical significance given in the Table 6:

We have assumed the same hypothesis mentioned above for the ANOVA test. Here p-values for all the F-values such as 19.06 in migration time, 27.60 in optimization time, 18.33 in throughput time, 6.11 in overhead time and 3.21 in SLA violation are less than significance $p<0.05$, so again null hypothesis is rejected and alternate hypothesis is accepted.

VI. CONCLUSIONS

This study has made an innovative contribution to the area of classification by performing file format classifications in cloud computing. To the best of the authors' knowledge, file format classifications have not been performed in literature in cloud computing. The study has made a significant impact on exploring a new area for further research in the cloud. The conducted study has devised a hybrid approach in two phases. In the first phase, SVM is modified for making accurate classifications over several file formats initially 100,000 such as audio, video, image, and text. Datasets/File types are of various sizes and hence take different time in processing. The initially classified data class has shown best classification as compared to known classifiers such as NB, RF, K-NN and CNN over-developed validation metrics comprising of accuracy, sensitivity, specificity, precision, recall, F-Measure, G-Mean, AUC, Kappa value, and Mathews correlation. The values of all these metrics lies between range [0,1] with 0 being no classification and 1 being accurate classification. Anything between [0,1] shows the strength of the classification and correlation. The output of the algorithm FTFSVM is a data class that is fed into ACOFTF for scheduling. ACOFTF is a metaheuristic algorithm that exploits the original ACO objective function in such a way that it fits well for further reducing or improving scheduling time to some significant extent. It is important to mention that number of ACO variants exists in literature in which a number of them adopted hybrid approaches with one objective or number of objectives. This metaheuristic algorithm adopts multi-objective scheduling in which SLA violations, migration time. optimization time, throughput time, and overhead time are taken into consideration. The baselines used for scheduling include QMPSO, ACOPS, ACOFTF, CPSO, CSO, and D-ACOELB.

All these algorithms are metaheuristics naturally inspired behavior algorithms. Except for CSO, all baselines are hybrid multi objectives and have shown good performance with several other algorithms. However, the proposed algorithm ACOFTF has been most successful in all QoS metrics and made huge improvements while outperforming them. Similarly, as far as overall average results are concerned, QMPSO stood second in the list of competitors, ACOPS performed well and placed in third place, CPSO got fourth, CSO falls in fifth place and D-ACOELB the last. VMs' Migration is critical in cloud computing since this whole migration process involves several processes to update such as workloads being migrated, priorities during migration, their sequences, timetabling, the status of the involved processes, performance metrics, and communication. Therefore, it is highly undesired to make many migrations especially VMs'. There are notable observations in this study are:

- The Hybrid approach has a significant impact on improvement on several parameters.
- The Hybrid approach performs extremely well in terms of accuracy and other validation metrics.
- Due to the reduced number of SLA violations, a hybrid approach (if consider energy-aware schemes) has improved energy consumption which is a big issue for data centers.
- Early optimization helps in faster convergence and achieving global optima which further results in reduced iterations and CPU cycles and ultimately reduced overhead time.
- Hybridization provides better optimization which helps in achieving optimum resource utilization. This is possible due to the shared advantages of both approaches.

In future, we will solve a load balancing problem using deep learning approaches and other swarm intelligence techniques and will focus on other performance metrics such as makespan, execution time, and energy consumption.

REFERENCES

[1] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, "Everything as a service (XaaS) on the cloud: Origins, current and future trends," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, New York, NY, USA, Jun. 2015, pp. 621–628.

[2] (Jan. 5, 2017). *Gartner Forecast: Public cloud services, worldwide, 2014-2020, 4Q16 Update*. Accessed: Mar. 12, 2020. [Online]. Available: https://www.gartner.com/doc/3562817/forecast-public-cloudservices-worldwide

[3] *IDC Worldwide Semiannual Public Cloud Services Tracker*. Accessed: Mar. 12, 2020. [Online]. Available: https://www.idc.com/tracker/showproductinfo.jsp?prod_id=881

[4] L. Heilig, E. Lalla-Ruiz, S. Voß, and R. Buyya, "Metaheuristics in cloud computing," *Software: Pract. Exper.*, vol. 48, no. 10, pp. 1729–1733, Oct. 2018.

[5] S. Benabderrahmane, "Combining boosting machine learning and swarm intelligence for real time object detection and tracking: Towards new meta-heuristics boosting classifiers," *Int. J. Intell. Robot. Appl.*, vol. 1, no. 4, pp. 410–428, Dec. 2017.

[6] 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.

[7] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, Mar. 2014.

[8] (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

[9] X. Peng, H. Guimin, L. Zhenhao, and Z. Zhongbao, "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 12, pp. 1–9, 2018.

[10] I. D. I. D. Ariyasingha and T. G. I. Fernando, "Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem," *Swarm Evol. Comput.*, vol. 23, pp. 11–26, Aug. 2015.

[11] C. Mao, L. Xiao, X. Yu, and J. Chen, "Adapting ant colony optimization to generate test data for software structural testing," *Swarm Evol. Comput.*, vol. 20, pp. 23–36, Feb. 2015.

[12] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 651–665, Oct. 2007.

[13] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 321–332, Aug. 2002.

[14] K. M. Salama, A. M. Abdelbar, and A. A. Freitas, "Multiple pheromone types and other extensions to the ant-miner classification rule discovery algorithm," *Swarm Intell.*, vol. 5, nos. 3–4, pp. 149–182, Dec. 2011.

[15] K. M. Salama, A. M. Abdelbar, F. E. B. Otero, and A. A. Freitas, "Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 667–675, Jan. 2013.

[16] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "A new sequential covering strategy for inducing classification rules with ant colony algorithms," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp. 64–76, Feb. 2013.

[17] F. E. B. Otero and A. A. Freitas, "Improving the interpretability of classification rules discovered by an ant colony algorithm: Extended results," in *Proc. Genetic E*, vol. Comput., Conf. (GECCO), Amsterdam, The Netherlands, 2013, pp. 73–80.

[18] N. K. Sreeja and A. Sankar, "A hierarchical heterogeneous ant colony optimization based approach for efficient action rule mining," *Swarm Evol. Comput.*, vol. 29, pp. 1–12, Aug. 2016.

[19] L. Lei and G. Qiao, "Text categorization using SVM with exponent weighted ACO," in *Proc. 31st Chin. Control Conf.*, Hefei, China, 2012, pp. 3763–3768.

[20] H. B. Alwan, "Hybrid ACO and SVM algorithm for pattern classification," Ph.D. dissertation, Dept. Comput., Univ. Utara Malaysia, Changlun, Malaysia, 2013.

[21] T. Liao, K. Socha, M. A. Montes de Oca, T. Stätzle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 503–518, Aug. 2014.

[22] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, Mar. 2008.

[23] T. Thaer, H. El-Bakry, I. Amal, A. Samir, and R. Magdi, "Fast sound verification using support vector machine and particle swarm optimization algorithms," *Int. J. Adv. Res. Comput. Sci. Technol.*, vol. 4, pp. 78–83, Oct. 2016.

[24] E. Houby, E. Yassin, S. N. Omran, "A hybrid approach from ant colony optimization and K-nearest neighbor for classifying datasets using selected features," *Information*, vol. 41, pp. 495–506, Oct. 2017.

[25] 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.

[26] R. Srinivasan, "Multiclass text classification a decision tree based SVM approach," CS294 Practical Mach. Learn. Project, 2006.

[27] G. Karthick and R. Harikumar, "Comparative performance analysis of naive bayes and SVM classifier for oral X-ray images," in *Proc. 4th Int. Conf. Electron. Commun. Syst. (ICECS)*, Coimbatore, India, Feb. 2017, pp. 88–92.

[28] 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.

[29] X. Li, J. Xu, and Y. Yang, "A chaotic parti cle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems," *Comput. Intell. Neurosci.*, vol. 2015, Oct. 2015. Art. no. 718689.

[30] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 1, pp. 514–521, Feb. 2020.

[31] R. Ahmed, R. Tarik, and S. Soran, "Cat swarm optimization algorithm—A survey and performance evaluation," *Comput. Intell. Neurosci.*, vol. 2020, Oct. 2020, Art. no. 4854895.

[32] A. Pradhan, S. K. Bisoy, and P. K. Mallick, "Load balancing in cloud computing: Survey," in *Innovation in Electrical Power Engineering, Communication, and Computing Technology*. Singapore: Springer, 2020, pp. 99–111.

[33] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti, "CSA-WSC: Cuckoo search algorithm for Web service composition in cloud environments," *Soft Comput.*, vol. 22, no. 24, pp. 8353–8378, Dec. 2018.

[34] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment," *Simul. Model. Pract. Theory*, vol. 87, pp. 311–326, Sep. 2018.

[35] A. K. Duggal and M. Dave, "A comparative study of load balancing algorithms in a cloud environment," in *Advances in Computing and Intelligent Systems*. Singapore: Springer, 2020, pp. 115–126.

[36] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, "An autonomous resource provisioning framework for massively multiplayer online games in cloud environment," *J. Netw. Comput. Appl.*, vol. 142, pp. 76–97, Sep. 2019.

[37] M. Ghobaei-Arani, A. Souri, T. Baker, and A. Hussien, "ControCity: An autonomous approach for controlling elasticity using buffer management in cloud computing environment," *IEEE Access*, vol. 7, pp. 106912–106924, 2019.

[38] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth flame optimization algorithm for cyber–physical system applications in fog computing," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, pp. 1–14, Feb. 2020.

[39] 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.

[40] 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, Jun. 2020.

[41] M. Safari and R. Khorsand, "PL-DVFS: Combining power-aware list-based scheduling algorithm with DVFS technique for real-time tasks in cloud computing," *J. Supercomput.*, vol. 74, no. 10, pp. 5578–5600, Oct. 2018.

[42] R. Khorsand, M. Ghobaei-Arani, and M. Ramezanpour, "A self learning fuzzy approach for proactive resource provisioning in cloud environment," *Softw., Pract. Exper.*, vol. 49, no. 11, pp. 1618–1642, Nov. 2019.

[43] E. Strumberger, N. Tuba, M. Bacanin, Beko, and M. Tuba, "Modified and hybridized monarch butterfly algorithms for multi-objective optimization," in *Proc. 18th Int. Conf. Hybrid Intell. Syst. (HIS)*, Porto, Portugal, 2018, pp. 449–458.

[44] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *J. Netw. Comput. Appl.*, vol. 128, pp. 64–77, Feb. 2019.

[45] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid heuristic-Metaheuristic techniques in cloud environment," *J. King Saud Univ. Comput. Inf. Sci.*, pp. 1319–1578, 2019.

[46] 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.

[47] 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.

[48] M. Strumberger, Beko, M. Tuba, M. Minovic, and N. Bacanin, "Elephant herding optimization algorithm for wireless sensor network localization problem," in *Proc. Technol. Innov. Resilient Syst. (DoCEIS)*, Caparica, Portugal, vol. 521, 2018, pp. 175–184.

[49] I. Attiya, M. Abd Elaziz, and S. Xiong, "Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm," *Comput. Intell. Neurosci.*, vol. 2020, Mar. 2020. Art. no. 3504642.

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

[51] 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.

[52] A. Nazia and D. Huifang, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment," *Appl. Sci.*, vol. 8, no. 4, 2018, Art. no. 38.

[53] 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.

[54] 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.

[55] N. Sharma and S. Maurya, "SLA-based agile VM management in cloud & datacenter," in *Proc. COMITCon*, Faridabad, India, 2019, pp. 252–257.

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

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

[58] 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.

[59] 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.

[60] 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 Org.*, vol. 8, no. 1, pp. 29–49, Jan. 2018.

[61] M. B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *ICT Express*, vol. 5, pp. 1–5, Jan. 2018.

[62] M. Sharma and R. Garg, "HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers," *Eng. Sci. Technol., Int. J.*, vol. 23, no. 1, pp. 211–224, Feb. 2020.

[63] 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.

[64] J. K. Naik, "A dynamic ACO-based elastic load balancer for cloud computing (D-ACOELB)," in *Proc. 3rd Int. Conf. Data Eng. Commun. Technol. (ICDECT)*, Hyderabad, India, vol. 1079, Mar. 2019, pp. 11–20.

[65] G. S. Sadasivam and D. Selvaraj, "A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids," in *Proc. 2nd World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2010, pp. 377–382.

[66] E. Tuba, L. Mrkela, and M. Tuba, "Support vector machine parameter tuning using firefly algorithm," in *Proc. 26th Int. Conf. Radioelektronika*, Apr. 2016, pp. 413–418.

[67] S. Shahaboddin, D. Petkovic, T. Nenad, P. S. Ch, A. Torki Altameem, and A. Gani, "Support vector machine firefly algorithm based optimization of lens system," *Appl. Opt.*, vol. 54, no. 1, pp. 37–45, 2015.
[68] G. Anushuya, K. Gopikaa, S. Gokul Prasath, and P. Keerthika, "Resource management in cloud computing using SVM with GA and PSO," *Int. J. Eng. Res. Technol. Etedm*, vol. 6, no. 4, p. 126, 2018.
[69] M. M. Sakr, M. A. Tawfeeq, and A. B. El-Sisi, "Network intrusion detection system based PSO-SVM for cloud computing," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 3, pp. 22–29, Mar. 2019.
[70] E. Tamimi, H. Ebadi, and A. Kiani, "Evaluation of different Metaheuristic optimization algorithms in feature selection and parameter determination in SVM classification," *Arabian J. Geosci.*, vol. 10, no. 22, Nov. 2017, Art. no. 478.
[71] M. Liu, M. Wang, J. Wang, and D. Li, "Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and chinese vinegar," *Sens. Actuators B, Chem.*, vol. 177, pp. 970–980, Feb. 2013.
[72] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit card fraud detection using convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process., (ICONIP)*, Kyota, Japan, Oct. 2016, pp. 483–490.
[73] N. Manohar, Y. H. S. Kumar, and R. G. H. Rani Kumar, "Convolutional neural network with SVM for classification of animal images," *Lect. Notes Electr. Eng.*, vol. 545, pp. 331–341, Oct. 2019.
[74] W. Marco, R. Michiel, M. Embrechts, S. Marijn, A. Meijster, and A. Lambertus, "The Neural Support Vector Machine," in *Proc. 25th Benelux Conf. Artif. Intell. (BNAIC)*, Delft, The Netherlands, Nov. 2013, pp. 247–254.
[75] H. Liu, X. Xiao, Y. Li, Q. Mi, and Z. Yang, "Effective data classification via combining neural networks and SVM," in *Proc. Chin. Control Decision Conf. (CCDC)*, Nanchang, China, Jun. 2019, pp. 4006–4009.
[76] T. He and K. Zhao, "Multispectral remote sensing land use classification based on RBF neural network with parameters optimized by genetic algorithm," in *Proc. Int. Conf. Sensor Netw. Signal Process. (SNSP)*, Xi'an, China, Oct. 2018, pp. 118–123.
[77] S. Shakya and S. Sigdel, "An approach to develop a hybrid algorithm based on support vector machine and naive Bayes for anomaly detection," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Greater Noida, India, May 2017, pp. 323–327.
[78] N. A. S. Selvakumari and V. Radha, "A voice activity detector using SVM and Naïve Bayes classification algorithm," in *Proc. Int. Conf. Signal Process. Commun. (ICSPC)*, Coimbatore, India, Jul. 2017, pp. 1–6.
[79] L. O. Iheme and S. Ozan, "Multiclass digital audio segmentation with MFCC features using naive Bayes and SVM classifiers," in *Proc. Innov. Intell. Syst. Appl. Conf. (ASYU)*, Izmir, Turkey, Oct. 2019, pp. 1–5.
[80] Z. H. Moe, T. San, M. M. Khin, and H. M. Tin, "Comparison of naive Bayes and support vector machine classifiers on document classification," in *Proc. IEEE 7th Global Conf. Consum. Electron. (GCCE)*, Nara, India, Oct. 2018, pp. 466–467.
[81] B. Y. Pratama and R. Sarno, "Personality classification based on Twitter text using naive bayes, KNN and SVM," in *Proc. Int. Conf. Data Softw. Eng. (ICoDSE)*, Yogyakarta, India, Nov. 2015, pp. 170–174.
[82] E. Raczko and B. Zagajewski, "Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images," *Eur. J. Remote Sens.*, vol. 50, no. 1, pp. 144–154, Jan. 2017.
[83] A. Sabat-Tomala, E. Raczko, and B. Zagajewski, "Comparison of support vector machine and random forest algorithms for invasive and expansive species classification using airborne hyperspectral data," *Remote Sens.*, vol. 12, no. 3, 2020, Art. no. 516.
[84] Y. Al Amrani, M. Lazaar, and K. E. El Kadiri, "Random forest and support vector machine based hybrid approach to sentiment analysis," *Procedia Comput. Sci.*, vol. 127, pp. 511–520, May 2018.
[85] H. Mojtaba, K. Kamran, B. Donald, J. Meimandi, and K. B. Laura, "An Improvement of Data Classification using Random Multimodel Deep Learning (RMDL)," *Int. J. Mach. Learn. Cybern.*, vol. 8., pp. 298–310, Oct. 2018.
[86] B. Xu, X. Guo, Y. Ye, and J. Cheng, "An improved random forest classifier for text categorization," *J. Comput.*, vol. 7, no. 12, pp. 2913–2920, Dec. 2012.
[87] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[88] H. H. A. Mukhairez and A. Y. A. Maghari, "Performance comparison of simulated annealing, GA and ACO applied to TSP," *Int. J. Intell. Comput. Res.*, vol. 6, no. 4, pp. 647–654, Dec. 2015.
[89] M. Alhanjouri and A. Belal, "Ant colony versus genetic algorithm based on travelling salesman problem," *Int. J. Comput. Technol. Appl.*, vol. 2, pp. 570–578, Mar. 2011.
[90] M. Michalis and Y. Shengxiang, "Adapting the pheromone evaporation rate in dynamic routing problems," in *Proc. 16th Eur. Conf. Appl. Evol. Comput.*, Vienna, Austria, 2013, pp. 606–615.
[91] M. Mavrovouniotis, I. S. Bonilha, F. M. Muller, G. Ellinas, and M. Polycarpou, "Effective ACO-based memetic algorithms for symmetric and asymmetric dynamic changes," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Wellington, New Zealand, Jun. 2019, pp. 2567–2574.
[92] X. Ren, R. Lin, and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Beijing, China, Sep. 2011, pp. 220–224.
[93] R. Gao and J. Wu, "Dynamic load balancing strategy for cloud computing with ant colony optimization," *Future Internet*, vol. 7, no. 4, pp. 465–483, Nov. 2015.
[94] M. Padmavathi and S. M. Basha, "Dynamic and elasticity ACO load balancing algorithm for cloud computing," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Madurai, India, Jun. 2017, pp. 77–81.
[95] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. Accessed: Mar. 12, 2020. [Online]. Available: http://archive.ics.uci.edu/ml
[96] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia, "A public dataset for YouTube's mobile streaming client," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Vienna, Austria, Jun. 2018, pp. 1–6.
[97] D. Amancio, C. Cesar, C. Dalcimar, T. Gonzalo, B. Odemir, R. Francisco, and C. Luciano, "A systematic comparison of supervised classifiers," *PLoS ONE*, vol. 9, no. 4, 2014, Art. no. e94137.
[98] T. Phan and K. Martin, "Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery," *Sensors*, vol. 18, no. 1, pp. 1–20, 2017.
[99] M. Ghobaei-Arani, A. A. Rahmanian, A. Souri, and A. M. Rahmani, "A moth-flame optimization algorithm for Web service composition in cloud computing: Simulation and verification," in *Proc. Softw., Pract. Exper.*, Jun. 2018, pp. 1865–1892.
[100] B. Chandra Mohan and R. Baskaran, "Survey on recent research and implementation of ant colony optimization in various engineering applications," *Int. J. Comput. Intell. Syst.*, vol. 4, no. 4, pp. 566–582, Jun. 2011.
[101] N. Jiaxu, Z. Changsheng, S. Peng, and F. Yunfei, "Comparative study of ant colony algorithms for multi-objective optimization," *Information*, vol. 10, no. 1, pp. 1–19, 2018.
[102] N. Sakthipriya and T. Kalaipriyan, "Variants of ant colony Optimization—A state of an art," *Indian J. Sci. Technol.*, vol. 8, no. 31, Nov. 2015.
[103] A. Gupta and R. Garg, "Load balancing based task scheduling with ACO in cloud computing," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Doha, Qatar, 2017, pp. 174–179.
[104] X. L. Luan, F. X. Gong, Z. Q. Wei, B. Yin, and Y. T. Sun, "Using ant colony optimization and cuckoo search in AUV 3D path planning," in *Proc. Softw. Eng. Inf. Technol.*, Austin, TX, USA, Feb. 2016, pp. 208–216.

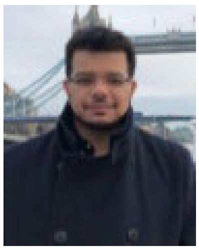
MUHAMMAD JUNAID is currently pursuing the Ph.D. degree with Iqra University, Islamabad, Pakistan. His research interests include cloud computing, blended learning, machine learning, swarm intelligence, and information security.

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

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

ABDULLAH BAZ (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering from UQU, in 2002, the M.Sc. degree in electrical and computer engineering from KAU, in 2007, and the M.Sc. degree in communication and signal processing and the Ph.D. degree in computer system design from Newcastle University, in 2009 and 2014, respectively. He was a Vice-Dean, and then the Dean of the Deanship of Scientific Research with UQU, from 2014 to 2020. He is currently an Assistant Professor with the Computer Engineering Department, a Vice-Dean of DFMEA, the General Director of the Decision Support Center, and the Consultant of the University Vice Chancellor with UQU. His research interests include VLSI design, EDA/CAD tools, coding and modulation schemes, image and vision computing, computer system and architecture, and digital signal processing. Since 2015, he has been served as a Review Committee Member of the IEEE International Symposium on Circuits and Systems (ISCAS) and a member of the Technical Committee of the IEEE VLSI Systems and Applications. He served as a Reviewer in a number of journals, including the IEEE INTERNET OF THINGS, *IET Computer Vision, Artificial Intelligence Review*, and *IET Circuits, Devices and Systems*.

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



HOSAM ALHAKAMI received the B.Sc. degree in computer science from King Abdul-Aziz University, Saudi Arabia, in 2004, the M.Sc. degree in internet software systems from Birmingham University, Birmingham, U.K., in 2009, and the Ph.D. degree in software engineering from De Montfort University, in 2015. From 2004 to 2007, he worked in software development industry, where he implemented several systems and solutions for a national academic institution. His research interests include algorithms, semantic web, and optimization techniques. He focuses on enhancing real-world matching systems using machine learning and data analytics in a context of supporting decision-making.