7-14-2023

DBSCAN inspired task scheduling algorithm for cloud infrastructure

S. M.F.D.Syed Mustapha
Zayed University

Punit Gupta
University College Dublin

Internet of Things and Cyber-Physical Systems 4 (2024) 32–39

Contents lists available at ScienceDirect

Internet of Things and Cyber-Physical Systems

journal homepage: www.keaipublishing.com/en/journals/internet-of-things-and-cyber-physical-systems

DBSCAN inspired task scheduling algorithm for cloud infrastructure

S.M.F D Syed Mustapha [a], Punit Gupta [b,*]

[a] College of Technological Innovation, Zayed University, Dubai, United Arab Emirates
[b] University College Dublin, Dublin, Ireland

ARTICLE INFO

Keywords:
Cloud computing
Density-based spatial clustering of applications with noise (DBSCAN)
PSO (particle swarm optimization)
Ant colony optimization (ACO)
Virtual machine (VM)

ABSTRACT

Cloud computing in today's computing environment plays a vital role, by providing efficient and scalable computation based on pay per use model. To make computing more reliable and efficient, it must be efficient, and high resources utilized. To improve resource utilization and efficiency in cloud, task scheduling and resource allocation plays a critical role. Many researchers have proposed algorithms to maximize the throughput and resource utilization taking into consideration heterogeneous cloud environments. This work proposes an algorithm using DBSCAN (Density-based spatial clustering) for task scheduling to achieve high efficiency. The proposed DBScan-based task scheduling algorithm aims to improve user task quality of service and improve performance in terms of execution time, average start time and finish time. The experiment result shows proposed model outperforms existing ACO and PSO with 13% improvement in execution time, 49% improvement in average start time and average finish time. The experimental results are compared with existing ACO and PSO algorithms for task scheduling.

1. Introduction

Cloud computing provides the convenience and on-demand facility of access to a shared scalable resources pool of virtual machines with high computation power and their services over the network [1]. Cloud computing is a highly trending concept in both public and private domains which is growing simultaneously with time. It provides various features of on-demand scalability, fine accessibility over the network, and various resources on demand. The vendor services provide many other features which are putting their advancement in cloud computing as future technology. The major goal of the existence of cloud computing is to provide computing over the globe based on user requirements and demand. Cloud computing provides the most promising computing paradigms that fulfill the computational requirements in different domains which deal with huge data from various domains like image processing, bioinformatics, earth science, astronomy, big data analysis, data workflow application, etc. [2]. As cloud services are highly in demand, there is an increase in the load over the cloud services and it is also necessary to maintain cloud performance to meet the customer/user requirements and demands. Among the many challenges, task scheduling is key to keeping the throughput to the maximum of cloud services. Task scheduling is the most challenging one because, firstly, it is a non-deterministic polynomial-time (NP) hard problem, secondly, the number of users is increasing daily which adds up to the difficulty for task scheduling [3]. Task scheduling can be categorized as a single-objective or multi-objective optimization problem as it depends on the scheduling policy. Some quality parameters like makespan, cost, energy efficiency, execution time, and many other factors play vital roles in task scheduling and the efficiency influences the task scheduling algorithm [4]. Cloud deals with virtualization, load balancing, fault tolerance, security, and scheduling are the major concerns needed to resolve for better performance and also needed to maximize throughput. Load balancing plays a crucial role in cloud resource utilization. The user submits the request to the data center to process the tasks which the tasks may contain input data and the processing data may be having software requirements, storage values, etc. Each task is executed over the VMs (Virtual Machine) in the cloud. The execution of these tasks is dependent on the task scheduler's scheduling algorithm which maps the arriving tasks accordingly. The mapping of these tasks and VMs should be done in a fine manner to achieve QoS (Quality of Service) [5]. The mapping of the tasks to their most suitable resources is required to be performed successfully. The performance depends on the performance parameters like minimum makespan, least execution time which should be matched with the minimum requirement of resources and many other factors that may be included depending on the task. Researchers are still looking for a solution to achieve optimal QoS and maximize resource utilization with high throughput.

* Corresponding author.
E-mail address: punitg07@gmail.com (P. Gupta).

https://doi.org/10.1016/j.iotcps.2023.07.001
Received 5 April 2023; Received in revised form 6 June 2023; Accepted 8 July 2023
Available online 14 July 2023

In the current scenario, many resource allocation algorithms are proposed using various optimization techniques taking into consideration of the performance parameters such as execution time, start time, waiting time, utilization, cost and power efficiency [6–15]. But these algorithms only consider the performance of the machine in the current situations but not the history of the machine [11–15]. This work considers each resource as an independent component even if it resides in the same data center and has similar behavior. So, in this work, we aim to cluster the machine/resources with similar behavior like computation capacity and cost.

The objective of this work is to overcome the drawback of existing algorithms that consider the cloud as an independent unit in nature. Whereas, in the existing setup, the whole data center is treated as a single unit with multiple resources and with similar performance. This work aims to improve the resource utilization and performance of the system at the same time using DBSCAN algorithm to cluster the resources with similar performance in the past and reduce the problem size by reducing the resource scheduling time with optimal execution time.

In recent years many researchers have proposed many algorithms to improve task scheduling over the cloud such as ACO(Ant Colony Optimization), modified ACO [6], PSO(Particle Swarm Optimization) [10], GA(Genetic Algorithm), WOA(Whale Optimization Algorithm), BAT(Bio-Inspire Bat Algorithm), Firefly, Min-Min, Max-Min and many other algorithms. However, these algorithms lack proper task mapping and scheduling. Whenever the tasks reach the data center, they have to be mapped with suitable efficient VMs over to the cloud. While scheduling, some major points have to satisfy like-resource utilization scalability of VM, throughput, timespan, and many other key points. Scheduling also must be optimized to improve the resource utilization of the data center. The traditional algorithms have some issues, but the DBSCAN algorithm has advantages over ACO, PSO and BAT in the context of mapping. DBSCAN algorithm is a cluster-based algorithm that comes under an unsupervised learning algorithm such that it has the capability to group the data point in similar groups based on specific properties. Here, using the DBSCAN algorithm, the tasks can be grouped based on their required attributes and also using the same DE list of VMs, they can also be grouped together to have a group-specific list of VMs. The proposed scheduling algorithm creates the cluster of VMs with respect to their configuration and schedules the task as per user's tasks. DBSCAN creates the cluster with the DE specification and allocates tasks to the best-fit VM in the cluster as per requirement. It ensures that user tasks are scheduled or mapped to the best-fitted VMs.

Now task scheduling will be more convenient and reliable if the scheduling takes place with these similar groups and also mapping of tasks with VMs will be more efficient which will maximize the throughput and utilization, to increase the throughput over the data centers. The main contributions of this work are as follows:

1. The use of a clustering algorithm to cluster the cloud resources, such that it allows reducing the search space.
2. The proposed approach using DBSCAN algorithm outperforms existing ACO [6] and PSO [10] algorithms using execution time, average start time and average execution time.
3. The proposed approach using DBSCAN algorithm allows clustering and sorting the resources i.e. virtual machine. This allows finding the resources which are underutilized and over-utilized over a period of time.

The work is divided into five sections. In section 2, some of the related works from task scheduling in cloud are discussed. In section 3, the proposed approach utilizing DBSCAN with the complete formulation is discussed. In section 4, experimental results are showcased with some comparative studies. In section 5, the conclusion and the outcomes of the proposed model in its improvement are discussed.

2. Related work

In this section, a set of related work from the domain of task scheduling models is showcased as shown in Table 1. Cloud task scheduling plays an important role in the performance of cloud datacenters and services provided to the client.

In [6], the author has proposed a trust and deadline-aware task scheduling model to improve the reliability of the system using fault and deadline as one of the performance metrics to evaluate the fitness of the data center. The model uses poison distribution to evaluate the occurrence of the fault. The task scheduling is inspired using ant colony optimization algorithm. Whereas the ants are used to find the most suitable food source where fitness is evaluated based on the number of faults that occurred and the execution time required to complete the task. In this work the proposed model is compared with PSO and the existing ACO model.

In [7], Sanaj et al. proposed a multi-objective scheduling model for task scheduling in cloud infrastructure. The model is inspired by the behavior of a squirrel and its mechanism to search for food sources. The proposed model is compared with Bat inspired model, PSO and Hybrid genetic algorithm by taking into consideration energy consumption, cost and utilization as performance parameters. The proposed model improves the performance of cloud in terms of energy and cloud as compared to existing models with increasing task load. In Ref. [8], the author has proposed a Grey wolf nature-inspired algorithm for task scheduling in cloud infrastructure. The model is proposed to improve the makes span on the tasks that are to be scheduled. The work uses execution time as the fitness function in order to find the best solution. The work is compared with existing ACO, PSO and first come first serve algorithms to study the performance of the proposed model. The work shows on increasing the task load proposed model takes the least time to complete the task as compared to the existing models. In Ref. [9], Gupta and Tewari presented a nature-inspired monkey search algorithm for task scheduling in cloud. The scheduling algorithm is inspired by monkey search where the behavior of a monkey climbing the hill is used. The proposed algorithm uses network delay and execution time as the fitness function. The work is compared with ACO, PSO and FCFS algorithms. The work uses the average completion time, execution time and network delay as performance metrics. In Ref. [10], Zeo et al. proposed a

Table 1
Existing models.

Reference	Algorithm Name	Parameters optimized
[6]	Ant Colony Optimization	Fault and Trust
[7]	Squirrel Optimization	Execution time
[8]	Grey Wolf	Execution time, makespan
[9]	Monkey Search	Utilization and Execution Time
[10]	Self Adaptive PSO	Deadline Failure
[11]	HEFT algorithm	Power Efficient
[12]	Ant Colony Algorithm	Time, Cost and Reliability
[13]	Longest Cloudlet Fastest Processing element (LCFP) and Shortest Cloudlet Fastest Processing element (SCFP)	Makespan
[14]	Hybrid model	Turnaround Time and Resource Utilization
[15]	Tournament-Selection Genetic Algorithm	Cost, Resource and Time
[16]	Tournament Selection Genetic Algorithm	Execution time and waiting time
[17]	Genetic algorithm	Memory utilization
[18]	QoS priority	Acceptance Rate and Completion Time
[19]	Oppositional based learning	Execution time and start time
[20]	Ant Colony Optimization	Utilization

deadline-aware self-adaptive PSO model for task scheduling in cloud. The work is a modified PSO for task scheduling which is able to adapt based on situations such as under-loaded, overloaded and average-loaded data center conditions. In Ref. [11], the authors proposed a method using power utilization aware task scheduling that takes care of a power efficient solution to find a task scheduling pattern with the least power consumption. The algorithm is aimed to select a resource with the least utilization in the past based on the past history. The model used the cubic power model to simulate the power consumption of the machine which is considered to be HP workstation.

In [12], the authors proposed an adaptive task scheduling algorithm to perform adaptation on the advantage of other algorithms based on the situations, while considering the distribution and scalability feature of the algorithm. In Ref. [13], the authors proposed two algorithms to minimize the turnaround time and maximize resource utilization in the cloud. To achieve this, the authors consider computation complexity and computing capacity of process elements as matrices to schedule the task. In Ref. [14], the authors did a review on different MapReduce algorithms like FIFO (first in first out), Matchmaking, Delay, and multithreading locality algorithms. In Ref. [15], the authors proposed a genetic base task scheduling algorithm for maximum utilization of resources. The authors proposed Tournament Selection Genetic Algorithm (TS-GA) which is not selected as a good candidate for a solution but having the capability to provide the solution. In Refs. [16,17], the authors proposed and modified GA (Genetic Algorithm) to schedule an independent and divisible task in adaptive manners for separate computation and memory requirements. In Ref. [18], the authors proposed the QoS(Quality of Service) based scheduling algorithm which comprises of QoS matrices of three parameters which are the task arrival time, the cost in use for the communication and the time taken by the task for execution. In Ref. [19], the authors proposed a task scheduling algorithm by merging two algorithms, the first one is cuckoo search algorithm (CSA) and the second one is oppositional-based learning (OBL). These two algorithms build up a hybrid oppositional cuckoo search algorithm (OCSA) algorithm to maximize the throughput with maximum utilization of resources. In Ref. [20], the authors proposed ACO (ant colony optimization) based algorithm to optimize task scheduling with pheromone intensity updating strategies to maximize the utilization. The authors took the intensity in terms of the resources which will reduce with respect to scheduled tasks. In Ref. [21], the author proposed MCC(minimum completion cloud), MEMAX(Median MAX) and Cloud Min-Max normalization algorithms for task scheduling, these scheduling algorithms will work in two phases. In the first phase, MCC single-phase scheduling, while MEMAX and Cloud Min-Max will take place in the second phase. In Ref. [22], the authors investigated the MCC (Mobile cloud computing)-assisted multi-task scheduling problem in the hybrid cloud system and proposed corporative multi-task scheduling based on ACO algorithm. In Ref. [23], the author proposed a pair-based task scheduling algorithm which is based on the famous Hungarian algorithm to minimize layover time where layover time is considered timing gaps between the tasks during the task scheduling. In Ref. [24], the authors proposed a monetary cost optimization algorithm to minimize the execution cost and maximize utilization by following upward and downward approaches together. In Refs. [25–33] various other recent works from the field of optimizations that can be used in the cloud.

In [32] author has proposed an energy-efficient task scheduling algorithm for cloud and fog computing using an AI-based model using HunterPlus. The work proposes a hybrid version of whale algorithm and machine learning model to improve energy efficiency in cloud. The work aims to improve makespan and energy efficiency in the cloud.

In [33] Chandrashekar et al. proposed a hybrid Ant Colony algorithm for task scheduling in cloud. The work is aimed to improve cost and makespan at the same time. The work is compared with basic ACO, min-min, First come first serve and Quantum computing-based task scheduling. The result shows better result but the work does not consider energy.

The work by Saif et al. [34] proposes a new algorithm for task scheduling in cloud-fog computing. The authors highlight the challenges faced in task scheduling in cloud-fog computing environments, such as resource allocation, energy efficiency, and response time. The authors propose a new algorithm called Multi-objective Grey Wolf Optimizer (MOGWO) that aims to optimize multiple objectives simultaneously, including makespan, energy consumption, and response time. The MOGWO algorithm is compared to other state-of-the-art algorithms, including Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), based on various performance metrics. One of the strengths of this article is its focus on a specific problem in cloud computing and propose a novel algorithm to address it. The authors provide a detailed analysis of the MOGWO algorithm and compare its performance with other popular optimization algorithms. The results obtained show that the MOGWO algorithm outperforms other algorithms in terms of makespan, energy consumption, and response time. and load into consideration.

Liang et al. [35] has proposed a new framework for characterizing data-parallel jobs in data centers. The authors highlight the challenges faced in job characterization, such as the complexity of data-parallel jobs and the large amount of job data generated. The authors propose a new framework called DeGTeC that utilizes deep graph and temporal clustering techniques to characterize data-parallel jobs in data centers. The framework consists of three main components: job feature extraction, graph representation learning, and temporal clustering. The results show that the DeGTeC framework outperforms other methods in terms of accuracy and efficiency.

In another work by Saravanan et al. [36] an algorithm for task scheduling in cloud computing using traditional horse optimization algo. The authors highlight the challenges faced in task scheduling, such as minimizing response time and maximizing resource utilization, and propose a new algorithm that combines the Wild Horse Optimization (WHO) algorithm with the Levy Flight algorithm. It proposes a new algorithm that combines the Wild Horse Optimization algorithm with the Levy Flight algorithm and outperforms other popular algorithms. The work tries to improve execution time and waiting time, where the proposed model is compared with the genetic algorithm and PSO(particle swarm optimization).

3. Problem formulation

The literature that is reported shown that various optimization techniques had been used and applied on one or more parameters which are set in the objective functions such as Genetic Algorithm, TS-GA, Squirrel Optimization and Self-adaptive PSO. Other approaches are looking at finding the optimal solution in the search space such as using the Monkey Search, Oppositional-based Learning and ant colony algorithm. These algorithms try to find a solution to schedule the task over the global resources without taking into consideration their current resource utilization, performance and computational capability. This work formulates a new strategy to solving the problem by clustering the resources with similar resources and computational capability and based on the performance. This approach allows the searching algorithm to find a suitable VM for small, medium and large tasks size in an appropriate cluster of VM's. Which reduces the search time and finds the best solution in less time. So that the cluster of the equivalent performing devices can be optimally assigned to the suitable VM. The main objective of this work is to solve task-scheduling problem to improve the user experience and quality of service by improving execution time and task waiting time.

4. Methodology

DBSCAN algorithm is a stochastic algorithm that provides results in the manner of search optimization for real values. The genetic algorithm also comes under the same category, but it works with the evolutionary approach of best fit while DBSCAN provides the solution in multidimensional where the data set comes in the form of real value with

continuity. The proposed approach also has some lakes like, and it does not deal with gradient value in search, but it has higher valued results with nonlinear objects. DBSCAN is an efficient and powerful population-based stochastic optimization algorithm [25] that is applied in many scientific and multiple domains of engineering where it has shown its usability and results make it a favorite algorithm when the solution has to be found on a clustering basis.

DBSCAN algorithm has been used for clustering where clusters as regions of high populated and respond well if all populated areas are dense enough and maintained well distance with the low dense area. Here in cloud computing, data centers are getting requests for task scheduling with their requirements. These requirements have to be fulfilled for better resource utilization and to maximize throughput. Task scheduler comes into action to handle schedules where these tasks are based on the applied scheduling algorithm. Some tasks are static, and some are dynamic, and it needs scaler resources that can expand on demand.

The work is divided into two steps:

1. DBSCAN Clustering
2. Scheduling

4.1. DBSCAN clustering

In this section the clustering phase of the proposed model is explained. DBSCAN stands for Density-based spatial clustering algorithm which clusters the nearest elements depending on whether the points are on boundary point or density point or core point and it discards other points that is considered as noisy points if they are not in the clusters. DBSCAN is a fine clustering algorithm which has the capability to make fine clusters of a given data.

It is executed in three phases:

1. **First phase:** In this phase is selection of epsilon value which is used to mark the distance value from the core point, with this distance, we calculate area of region where data points lie.
2. **Second phase:** In this phase a decision or election of core point or centroid from where we calculate the points whether they lie in a circle or not.
3. **Third Phase:** Third and most important part of the algorithm is border line points or known as boundary points which helps to find out cluster data point, and also to get the data point in cluster. It will find out cluster noisy point which are never included in the cluster, or we can say they always being discarded.

As we can see in the above Fig. 1a and b, *P* is the core point and *e* epsilon value are the two parameters, which will be considered for the distance of data elements. Both a, b are reachable data points in cluster C1, similarly cluster C2 and cluster C3 will be having certain reachable points which are closer to core point or cluster 1. Here *T* and *U* are at the boundary point in cluster C3 and cluster C2 respectively. Data point *S* can also be seen in the figure that shows the noisy value which is excluded from clusters.

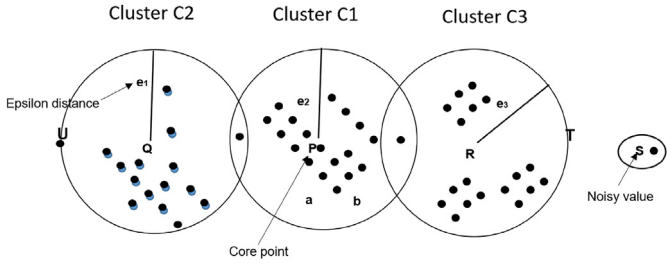

Fig. 1a. Working of DBSCAN

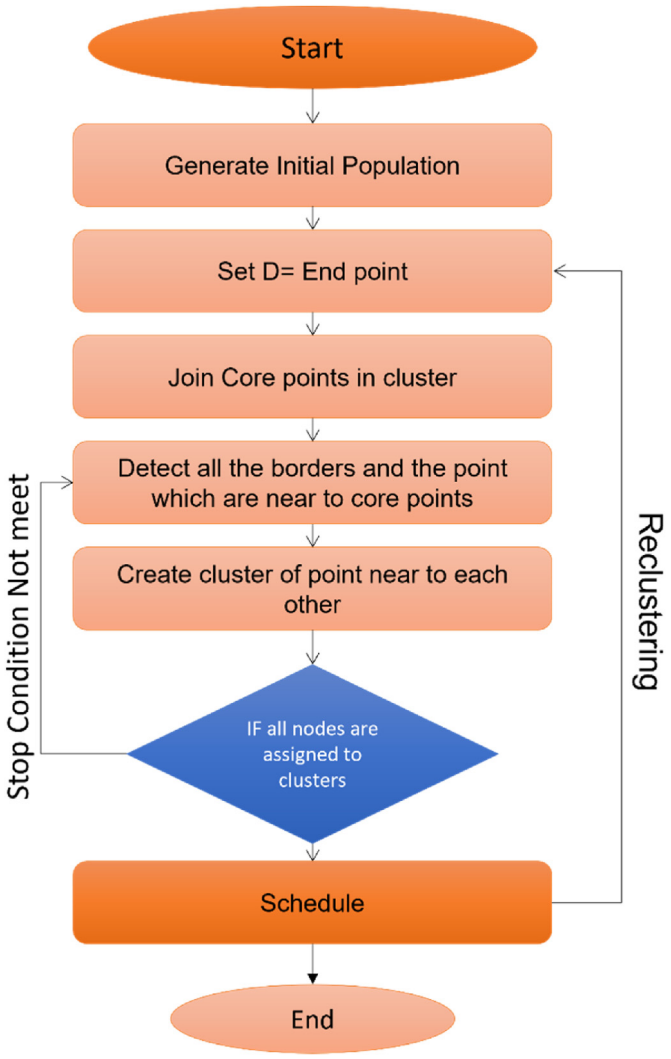

Fig. 1b. Flow diagram of applying DBSCAN algorithm.

As per the DBSCAN properties, the data points which are not considered in the cluster are discarded by the algorithm. As described above, if we summarize the working of DBSCAN in brief, we can say that a similar featured-based data set being clustered in the same region as can be seen in cluster C1,C2,C3 and datasets which have different features has been discarded or considered as Noisy value.

Using the above mechanism, the clustering of tasks is done based on their task size and minimizes the expected total execution time. Where task size is defined by the number of instructions in a task. Based on the DBSCAN clustering and the clustered generated, the virtual machine is assigned which can complete that set of tasks in the cluster using min-min methodology.

Task scheduling mechanism defines the efficiency of user's tasks and also plays a vital role in resource utilization. The proposed approach is based on the DBSCAN algorithm where VMs and tasks are clustered together mapping of tasks and VMs will be done by the proposed task scheduling algorithm. All VMs and tasks are clustered together based on their computing size and load. When the request is received by the data center, the DBSCAN algorithm takes place. The fitness function that is applied to DBSCAN algorithm is the computing power of the VM which is defined by equation (1). The fitness function depends on the number of processor and number of MIPS (million of instruction per second) available for execution. This allows to evaluate both the computational capacity and current load on the virtual machine.

$$Execution\ Time_i = \frac{Task_{Length_i}}{No.\ of\ processor * VM\_MIPS_j} + Network\ Delay_i \quad (1)$$

To have a better understanding of the proposed algorithm let us take a scenario wherein a pool of VMs. The VMs are available in numbers such as i, j, p, m and have to map with tasks (in number from i, j, p, m) in such a manner that each task must fulfill its requirement of resources. The selection of VMs for the task is mapped on the basis of the DBSCAN algorithm in that clusters are made and the algorithm has desired advancement over traditional scheduling algorithms (such as ACO, min-min, min-max, etc). Results show that the application of DBSCAN algorithm has better scheduling output.

4.2. Scheduling

Scheduling is based on selecting a cluster with a VM with the least execution time and network delay. For this process the algorithm used Min-Min algorithm to find the best cluster. Next step after scheduling, a few tasks in the clustering are repeated based on the execution power available and network delay. This allows the algorithm to just check for few cluster not all large amount of virtual machine which reduces the search space and the time taken to find the under loaded VM. Where time complexity of the proposed algorithm is O(nlog n) [37] on the other hand the complexity of particle swarm optimization in $O(n^3)$.

5. Experiment & result

The simulation is carried out on cloudsim 4.0. The simulation is done using 4 data centers with 2 hosts each, 5 & 10 VM's and the tasks are scaled from 1000 to 10000. The simulation is done with a scaling task load to study the performance of the proposed algorithm in underloaded and overloaded conditions. The work uses SWF (Standard Workload Format) files from the parallel-workload repository. The work uses NASA iPSC workload log file. Simulation results show that utilizing DBSCAN as a solution is having an advantage over PSO (particle swarm optimization) and ACO (Ant colony Optimization) algorithms. To simulate the proposed algorithm cloudsim simulation framework is used and a comparative study is done with PSO and ACO algorithms in different aspects. Table 2 shows the 3 type of tasks and their task size and other input parameters. Table 2 also showcases input parameters taken into consideration of ACO and PSO.

Here in Fig. 2a and b comparative study is showcased using execution time as a performance parameter. Execution is defined as the total time to complete all tasks by virtual machines in milliseconds. Execution times that are taken by ACO and PSO are higher as compared to the proposed algorithm with increasing task load.

Fig. 2a and b show the comparison between the execution time in milliseconds against the number of tasks. It can be seen that, as much as we create the tasks, the execution time is getting less in ACO and PSO algorithms whereas in DBSCAN, it stays at a minimum. The study shows the performance with scale resources from 5 VM to 10 VM's, where the DBSCAN performs better than ACO and PSO. We followed the simulation parameter of Rawat et al. [25] as shown in Tables 2 and 3. The

Table 2
Configuration Parameters of user tasks.

Parameters	Values
Task length	300 (small) 2000/3000(medium) 4000(high)
Input File Size	200 Byte
Output File Size	400 Byte
PE	1–2
Population(PSO/ACO)	100
Number of swarm/Ant	10
Fitness Function	Execution time

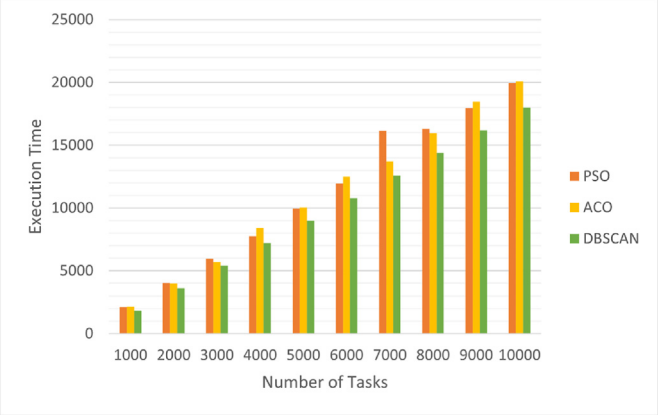

Fig. 2a. Execution time comparisons between PSO, ACO and DBSCAN with 5 VM's.

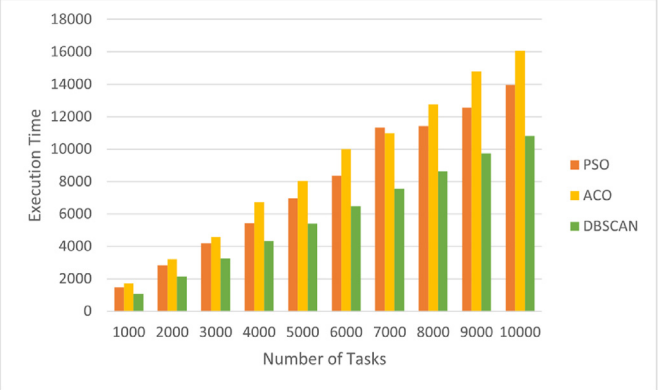

Fig. 2b. Execution time comparisons between PSO, ACO and DBSCAN with 10 VM's.

Table 3
Parameter for existing algorithm.

PSO	Number of Swarms: 100
	Initial inertia weight: 0.9
	Variable inertia weight: 0.2
ACO	**Number of Ants:**100 **Evaporation rate ρ:** 0.1 **Pheromone factor α:**1 **Heuristic factor β:** 1

performance parameters taken into consideration to study the improvement are: Execution Time, Average Start time and Average Finish time. In this experiment workload file in SWF format from parallel workloads are used to generate tasks. The types of VM is listed in Table 4.

Tables 5 and 6 shows the simulation results for execution time as performance metrics with scaling VM's from 5 to 10. The average start time with respect to ACO and PSO, DBSCAN performance is much better as compared to both algorithms as shown in Fig. 3a and b which shows

Table 4
Types of VM's.

	MIPS/Ram (mb)/PE
VM1	1000/512/1 Small machine
VM2	2000/1024/2 Medium machine
VM3	4000/2156/4 Large machine

Table 5
Simulation results of execution time with 5 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	2112.1	2132.1	1800.28
2000	4044.1	3992.1	3600.46
3000	5968.1	5708.1	5400.63
4000	7760.1	8416.1	7200.87
5000	9944.1	10048.1	9001.04
6000	11944.1	12488.1	10801.2
7000	16172.1	13728.1	12601.5
8000	16312.1	15968.1	14401.6
9000	17944.1	18480.1	16201.8
10000	19940.1	20096.1	18002

Table 6
Simulation results of execution time with 10 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	1478.47	1705.68	1080.168
2000	2830.87	3193.68	2160.276
3000	4177.67	4566.48	3240.378
4000	5432.07	6732.88	4320.522
5000	6960.87	8038.48	5400.624
6000	8360.87	9990.48	6480.732
7000	11320.47	10982.48	7560.87
8000	11418.47	12774.48	8640.978
9000	12560.87	14784.08	9721.086
10000	13958.07	16076.88	10801.22

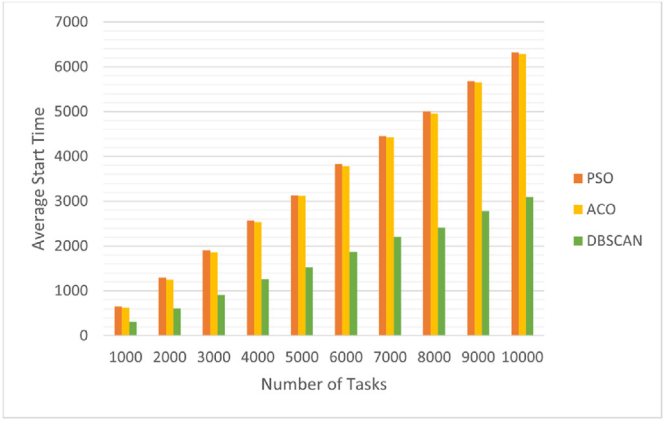

Fig. 3a. Average start time of tasks for 5 VM's.

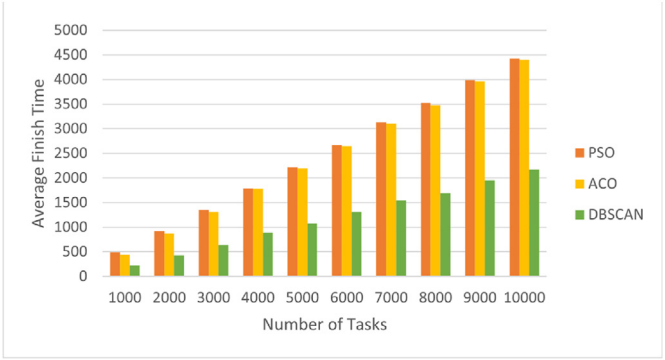

Fig. 3b. Average start time of tasks for 10 VM's.

that the average start time of task execution is less taking into consideration of 5 VM's and 10 VM's.

Tables 7 and 8 shows the simulation result and the study of result with

Table 7
Simulation results of average start time with 5 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	654.3151	625.3151	308.1501
2000	1291.485	1245.485	608.1507
3000	1906.083	1862.083	906.5555
4000	2572.358	2534.358	1260.109
5000	3135.073	3124.073	1531.313
6000	3831.687	3781.687	1870.464
7000	4450.215	4430.215	2200.98
8000	5002.013	4956.013	2408.498
9000	5681.914	5649.914	2783.913
10000	6320.839	6282.839	3098.454

Table 8
Simulation results of average start time with 10 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	458.0206	437.7206	215.7051
2000	904.0395	871.8395	425.7055
3000	1334.258	1303.458	634.5889
4000	1800.651	1774.051	882.0763
5000	2194.551	2186.851	1071.919
6000	2682.181	2647.181	1309.324
7000	3115.15	3101.15	1540.686
8000	3501.409	3469.209	1685.949
9000	3977.34	3954.94	1948.739
10000	4424.587	4397.987	2168.918

increasing task load from 1000 to 10000 tasks and resources with 5 and 10 VM's. Fig. 4a and b shows that the average finishing time when DBSCAN is used to complete the task is lower than compared to ACO and PSO algorithms. It shows resource utilization and performance of the data center in terms of task scheduling is higher and has high throughput. Tables 9 and 10 shows the simulation result for average finish time and the study of result with increasing task load from 1000 to 10000 tasks and resources with 5 and 10 VM's.The above results show that DBSCAN algorithm has high throughput and high resource utilization. DBSCAN algorithm outperforms PSO and ACO in the experimental setup. The DBSCAN algorithm performs better because the other optimization algorithms are iterative and do not consider the resources dependent rather consider all resources as independent which loses the behavior of the clustered resources.

Fig. 5 shows a comparative study of various configurations of ACO and PSO to study the performance that ACO and PSO can achieve as compared to the proposed algorithm. Figure shows the change in existing algorithms can not outperform the proposed algorithm. The proposed algorithm proves to provide better execution time as compared to existing algorithms. Table 11 shows the configurations of ACO and PSO studied for comparison. The proposed algorithm shows a better result because the proposed model can create a cluster of resources based on the computational capacity and performance which allows the algorithm to find a suitable virtual machine for the task based on real-time performance of the resources. Where DBSCAN algorithm allows the best clustering based on multi-parameters.

6. Conclusion

As shown in the result section, the proposed task scheduling for cloud infrastructure surpasses the performance of the existing ACO and PSO algorithm with the scaling load with increasing task load with 5 & 10 virtual machines. The performance is studied using execution time, average start time, average finish time and total execution time as performance parameters. The result shows that the proposed algorithm performs better over the traditional approaches (PSO, ACO) algorithms in the context of average start time and average finish time. The evaluation is done by scaling the number of tasks from 100 to 10000 to study

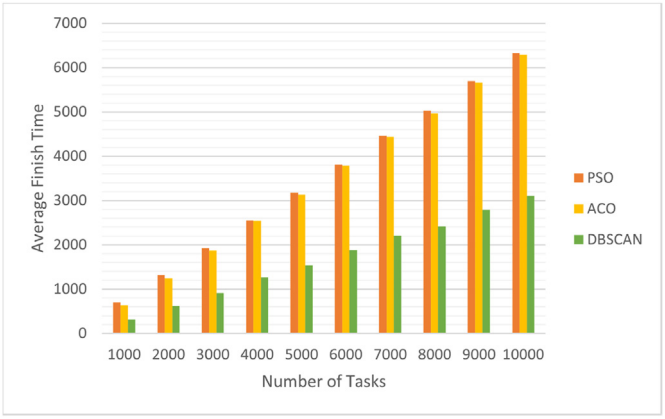

Fig. 4a. Average finish time of tasks for 5 VM's.

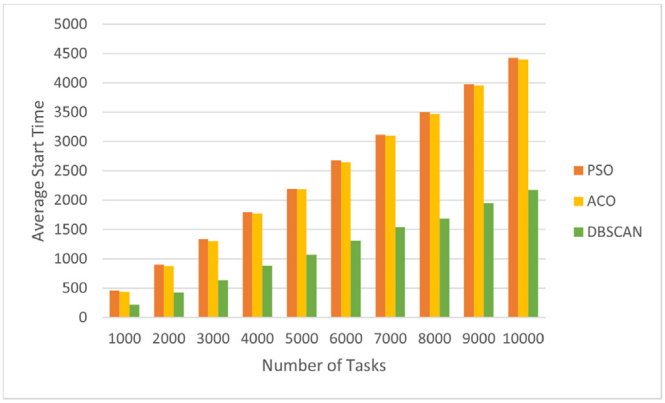

Fig. 4b. Average finish time of tasks for 10 VM's.

Table 9
Simulation results of average finish time with 5 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	699.304	630.304	313.4951
2000	1319.464	1250.464	613.4957
3000	1926.053	1867.053	911.9005
4000	2556.374	2539.374	1265.454
5000	3173.058	3129.058	1536.658
6000	3809.689	3786.689	1875.809
7000	4463.685	4435.685	2206.325
8000	5029.683	4961.683	2413.843
9000	5695.614	5655.614	2789.258
10000	6321.599	6288.599	3103.799

Table 10
Simulation results of average finish time with 10 VM's in milliseconds.

No. of Tasks	PSO	ACO	DBSCAN
1000	489.5128	441.2128	219.4466
2000	923.6246	875.3246	429.447
3000	1348.237	1306.937	638.3304
4000	1789.462	1777.562	885.8178
5000	2221.14	2190.34	1075.66
6000	2666.782	2650.682	1313.066
7000	3124.579	3104.979	1544.427
8000	3520.778	3473.178	1689.69
9000	3986.93	3958.93	1952.481
10000	4425.119	4402.019	2172.659

the performance in underloaded and overloaded conditions and the second evaluation with increasing the number of virtual machines i.e.

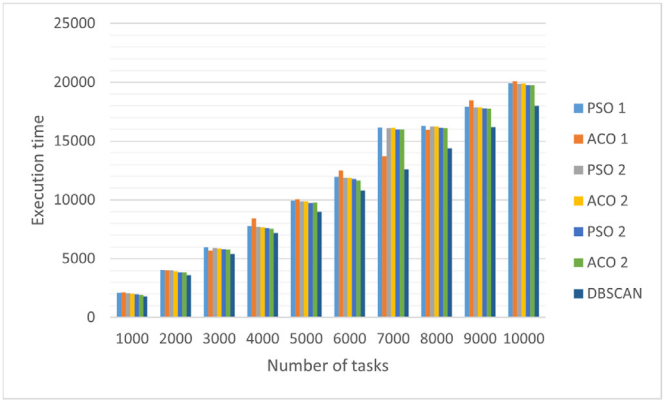

Fig. 5. Execution time comparison for various variation in ACO and PSO.

Table 11
Configuration of PSO and ACO.

PSO 1	Population:100, iteration:100,Initial inertia weight: 0.9, Variable inertia weight: 0.2
ACO 1	**Population:100, iteration:100, Evaporation rate ρ: 0.1 Pheromone factor α:1 Heuristic factor β: 1**
PSO 2	**Population:100, iteration:150,Initial inertia weight: 0.8, Variable inertia weight: 0.3**
ACO 2	**Population:100, iteration:100, Evaporation rate ρ: 0.1 Pheromone factor α:0.8 Heuristic factor β: 0.8**
PSO 2	**Population:100, iteration:200,Initial inertia weight: 0.6, Variable inertia weight: 0.1**
ACO 2	**Population:100, iteration:100, Evaporation rate ρ: 0.2 Pheromone factor α:0.7 Heuristic factor β: 0.7**

scaling resources. Under both the conditions of underloaded, average and overloaded condition the proposed algorithm performs better than the existing model with a 13% improvement in execution time. The work can further be extended with more objective functions like cost and power-efficient algorithms. In the future, the algorithm can be merged with a machine learning algorithm to further improve the performance of the algorithm.

Conflicts of interest

The authors have no conflicts of interest to declare, and there is no financial interest to report.

Data availability

The dataset for simulation is supported by parallel http://workload.com for real-time analysis.

Funding

The research project is funded by Startup Research Grant R21043 from Zayed University, Dubai, United Arab Emirates.

Conflict of interest

The authors whose names are listed immediately below certify that they have NO affi liations with or involvement in any organization or entity with any fi nancial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-fi nancial interest (such as

personal or professional relationships, affi liations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Acknowledgement

The author would like to express gratitude to the assistance provided by Dr Punit Gupta on the aspect of the cloud infrastructure setting and performing the experiment.

References

[1] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Serv. Appl. 1 (2010) 7–18.
[2] R. Nazir, Z. Ahmed, Z. Ahmad, N.N. Shaikh, A.A. Laghari, K. Kumar, Cloud computing applications: a review, EAI Endorsed Transactions on Cloud Systems 6 (17) (2020 May 22).
[3] A.R. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in cloud computing: a literature survey, Future Generat. Comput. Syst. 91 (2019 Feb 1) 407–415.
[4] R. Ghafari, F.H. Kabutarkhani, N. Mansouri, Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review, Cluster Comput. 25 (2) (2022 Apr) 1035–1093.
[5] A. Abdelmaboud, D.N. Jawawi, I. Ghani, A. Elsafi, B. Kitchenham, Quality of service approaches in cloud computing: a systematic mapping study, J. Syst. Software (101) (2015 Mar 1) 159–179.
[6] P. Gupta, S.P. Ghrera, Trust and deadline aware scheduling algorithm for cloud infrastructure using ant colony optimization, February, in: 2016 International Conference on Innovation and Challenges in Cyber Security, ICICCS-INBUSH), 2016, pp. 187–191.
[7] M.S. Sanaj, P.J. Prathap, Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere, Eng. Sci. Tech., Inte. J. 23 (4) (2020) 891–902.
[8] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, M. Zivkovic, November). Task scheduling in cloud computing environment by grey wolf optimizer, in: 2019 27th Telecommunications Forum (TELFOR), 2019, pp. 1–4.
[9] P. Gupta, P. Tewari, Monkey search algorithm for task scheduling in cloud IaaS, December, in: 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017, pp. 1–6.
[10] X. Zuo, G. Zhang, W. Tan, Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud, IEEE Trans. Autom. Sci. Eng. 11 (2) (2013) 564–573.
[11] K. Dubey, M. Kumar, S.C. Sharma, Modified HEFT algorithm for task scheduling in cloud environment, Procedia Comput. Sci. 125 (2018) 725–732.
[12] H. Liu, Research on cloud computing adaptive task scheduling based on ant colony algorithm, Optik (2022 May 1) 1–10.
[13] S. Sindhu, Saswati Mukherjee, Efficient task scheduling algorithms for cloud computing environment, in: International Conference on High Performance Architecture and Grid Computing, Springer, Berlin, Heidelberg, 2011.
[14] Qutaibah Althebyan, et al., Evaluating map reduce tasks scheduling algorithms over cloud computing infrastructure, Concurrency Comput. Pract. Ex. 18 (27) (2015) 5686–5699.
[15] Safwat A. Hamad, Fatma A. Omara, Genetic-based task scheduling algorithm in cloud computing environment, Int. J. Adv. Comput. Sci. Appl. 4 (7) (2016) 550–556.
[16] A. Mahmood, S.A. Khan, R.A. Bahlool, Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm, Computers 6 (2) (2017 Apr 5) 15.
[17] T.T. Tejaswi, M. Azharuddin, P.K. Jana, A GA Based Approach for Task Scheduling in Multi-Cloud Environment, 2015 arXiv preprint arXiv:1511.08707.
[18] Sirisha Potluri, Katta Subba Rao, Quality of service based task scheduling algorithms in cloud computing, Int. J. Electr. Comput. Eng. 2 (7) (2017) 1088.
[19] Pradeep Krishnadoss, Prem Jacob, OCSA: task scheduling algorithm in cloud computing environment, International Journal of Intelligent Engineering and Systems 3 (11) (2018) 271–279.
[20] Weifeng Sun, et al., PACO: a period ACO based scheduling algorithm in cloud computing, in: 2013 International Conference on Cloud Computing and Big Data, IEEE, 2013.
[21] Sanjaya K. Panda, K. Jana Prasanta, Efficient task scheduling algorithms for heterogeneous multi-cloud environment, J. Supercomput. 4 (71) (2015) 1505–1533.
[22] Tongxiang Wang, et al., Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints, Peer-to-Peer Networking and Applications 4 (11) (2018) 793–807.
[23] Sanjaya Kumar Panda, Shradha Surachita Nanda, Sourav Kumar Bhoi, A pair-based task scheduling algorithm for cloud computing environment, J. King Saud Univ. Comput. Inf. Technol. 34 (1) (2022) 1434–1445.
[24] P. Gupta, R.R. Kaikini, D.K. Saini, S. Rahman, Cost-aware resource optimization for efficient cloud application in smart cities, J. Sens.2022 (2022) 1–12.
[25] P.S. Rawat, P. Dimri, P. Gupta, Learning-based task scheduling using big bang big crunch for cloud computing environment, Recent Adv. Comput. Commun.(Formerly: Recent Pat. Comput. Sci. 13 (2) (2020) 137–146.
[26] Pradeep Singh Rawat, Priti Dimri, Punit Gupta, Gyanendra Pal Saroha, Resource provisioning in scalable cloud using bio-inspired artificial neural network model, Appl. Soft Comput. 99 (2021) 106876.
[27] D. Ergu, G. Kou, Y. Peng, Y. Shi, Y. Shi, The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment, J. Supercomput. 64 (3) (2013) 835–848.
[28] P.S. Pillai, S. Rao, Resource allocation in cloud computing using the uncertainty principle of game theory, IEEE Syst. J. 10 (2) (2014) 637–648.
[29] N. Akhter, M. Othman, Energy aware resource allocation of cloud data center: review and open issues, Cluster Comput. 19 (3) (2016) 1163–1182.
[30] X. Wang, X. Wang, H. Che, K. Li, M. Huang, C. Gao, An intelligent economic approach for dynamic resource allocation in cloud services, IEEE Trans. Cloud Comput. 3 (3) (2015) 275–289.
[31] F. Cauteruccio, G. Terracina, D. Ursino, Generalizing identity-based string comparison metrics: framework and techniques, Knowl. Base Syst. 187 (2020) 104820.
[32] S. Iftikhar, M.M.M. Ahmad, S. Tuli, D. Chowdhury, M. Xu, S.S. Gill, S. Uhlig, HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments, Internet of Things vol. 21 (2023) 100667.
[33] C. Chandrashekar, P. Krishnadoss, V. Kedalu Poornachary, B. Ananthakrishnan, K. Rangasamy, HWACOA scheduler: hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing, Appl. Sci. 13 (6) (2023) 3433.
[34] F.A. Saif, R. Latip, Z.M. Hanapi, K. Shafinah, Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing, IEEE Access 11 (2023) 20635–20646.
[35] Y. Liang, K. Chen, L. Yi, X. Su, X. Jin, DeGTeC: a deep graph-temporal clustering framework for data-parallel job characterization in data centers, Future Generat. Comput. Syst. 141 (2023) 81–95.
[36] G. Saravanan, S. Neelakandan, P. Ezhumalai, S. Maurya, Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing, J. Cloud Comput. 12 (1) (2023) 24.
[37] N. Gholizadeh, H. Saadatfar, An improved DBSCAN algorithm for big data, J. Super Comp. (77) (2021) 6214–6235.

Abbreviation Definition

ACO: Ant colony Optimization
PSO: Particle Swarm Optimization
VM: Virtual Machine
RAM: Random access memory
BAT: Bat inspired algorithm
GA: Genetic algorithm
MCC: Mobile cloud computing
DE: Datacenter
WOA: Whale Optimization Algorithm