2528-9705

Örgütsel Davranış Araştırmaları Dergisi
Journal Of Organizational Behavior Research
Cilt / Vol.: 7, Sayı / Is.: S, Yıl/Year: 2022, Kod/ID: 22S0-896

Presenting a Way to Improve Task Scheduling in Cloud Computing Using Ant Colony Algorithm

Tayyebeh.Oladazizmi Ghadikolaei
M.Sc, Department of Computer Engineering Software, North Tehran Branch,Islamic Azad university,Tehran, Iran
E-mail: Tayyebeholadazimi@gmail.com

ABSTRACT

Cloud computing has emerged as a high-performance distributed computing architecture that allows access to a pool of shared resources depending on demand through the Internet in recent years. Cloud computing is still in its infancy, and considerable study on a wide range of areas is required to reap its full benefits. Task scheduling is one of the essential aspects that should be researched in order to obtain optimal cloud performance. Due to the huge solution space and, as a result, the long time it takes to discover an optimal solution, scheduling in cloud computing is classified as an NP-hard issue. The scheduler algorithm's goal in this study is to process users' tasks in the shortest amount of time and for the least amount of money. In order to minimize overall completion time and maximize resource efficiency, all tasks should be evenly allocated across available resources. To achieve the desired goals and address the task scheduling problem, the Ant Clone Algorithm (ACO) was applied. Simulations and comparisons of the suggested method's results with those of the genetic algorithm (GA) and particle swarm optimization (PSO) reveal that the proposed methodology has been able to satisfy consumers while also maximizing resource use.

Keywords: *Cloud Computing, Virtual Machine, Task Scheduling, Ant Colony Algorithm*

INTRODUCTION

Cloud computing is a type of parallel computing, distributed computing, and grid computing that allows users to safely store and process data over the Internet. In the cloud computing environment, services are delivered at three levels: infrastructure as a service (IaaS2), platform as a service (PaaS3) and software as a service (SaaS4). This segmentation enables distinct management support and virtualization technologies for each level (Dillon et al., 2010).
At the IaaS level, task scheduling is a critical procedure that tries to efficiently execute system demands on resources so that other aspects of the cloud environment may be taken into account. Virtual machines are used as scheduling units to assign diverse physical resources to complete tasks in task scheduling. Each virtual computer in the cloud is an abstract unit of computation and storage capacity (Ghorbannia and Arian, 2011).
The issue of scheduling tasks according to the various requirements of service quality index parameters such as bandwidth, cost, time, and availability, as well as resource mapping to tasks with the goal of optimizing bandwidth usage, reducing completion time, lowering costs, increasing interest in resource transfer, and achieving a goal function with greater efficiency and less complexity in the cloud computing environment, is critical. As a result, choosing the right scheduling mechanism can have a big influence on system performance. An effective scheduling algorithm may suit the user's demands while also improving resource productivity, boosting the cloud computing environment's overall performance (Ghaffari, 2010).

For an automated scheduling policy, Hai Zhong et al. (2010) applied an upgraded genetic algorithm. Their method employs the shortest genes and introduces the concept of dividend policy in economics to determine if a demand allocation is optimum or undesirable. In their fitness function, they used a basic sum of CPU, RAM, and hard drive. Sellami et al. (2013) suggested a five-objective evolutionary algorithm for taskflow scheduling to handle the problem of satisfaction limitations in the context of task planning restrictions. For the Cloud Computing Federation, Zhang Z et al. (2010) suggested a load balancing technique based on ant colony optimization. Lee et al. (2011) employed the ant colony load balancing (LBACO) method to schedule independent tasks in all virtual machines with the goal of reducing maketime and taskload.

A hybrid ant algorithm and particle swarm method were used to plan resources in cloud computing in research given by Xiaotang Wen et al. (2012). The particle swarm optimization method picks the next repetition site to lead the particles utilizing information, local extreme information, and global extreme information, whereas the ant algorithm employs a pheromone to transfer information. At the virtual machine level, Medhat Tawfeek et al. (2013) employed ant colony optimization to assign batch input taskloads to virtual machines (VM scheduling).

Medhat Tawfeek et al. (2013) employed ant colony optimization to assign batch input taskloads to virtual machines (VM scheduling) at the virtual machine level. By submitting their computer tasks to the cloud system, millions of users desire to access pooled cloud resources. The cloud computing environment has a hurdle in scheduling these millions of tasks. The optimal resource allocation or task scheduling in the cloud should be determined by the desired number of cloud systems in order to reduce total costs. The user and system levels of cloud service scheduling are separated. There are issues with transactions in user-level planning, such as how to deliver services between suppliers and clients. Resource management in data centers is linked to system-level planning (Fangzhe C et al., 2010). According to what mentioned above, this study aims is to use the ant colony method to optimize task scheduling in cloud computing.

Proposed algorithm

A novel strategy based on the ant colony algorithm was utilized to optimize task scheduling in cloud computing in this study. The goal of this and the suggested method is to arrive at the best answer in the shortest amount of time possible using various operators. For the goal function, many factors have been evaluated in order to decrease machine idle time and hence boost machine efficiency. Furthermore, in order to improve customer satisfaction, their expectations are taken into account, such as reducing the time it takes to complete all operations and lowering the cost of processing and executing tasks.

The suggested algorithm's general phases are depicted as a flowchart in Figure 1. The suggested technique begins by numbering the system's input tasks and creating a pheromone table. The colony population is then established by forming the required number of ants. The target function is used to determine each ant's fitness value. The method then does a global pheromonerization, and if the termination condition is fulfilled, the algorithm finishes, and the best feasible response based on the objective function is shown. Each of the steps will be introduced and discussed in the following chapters.

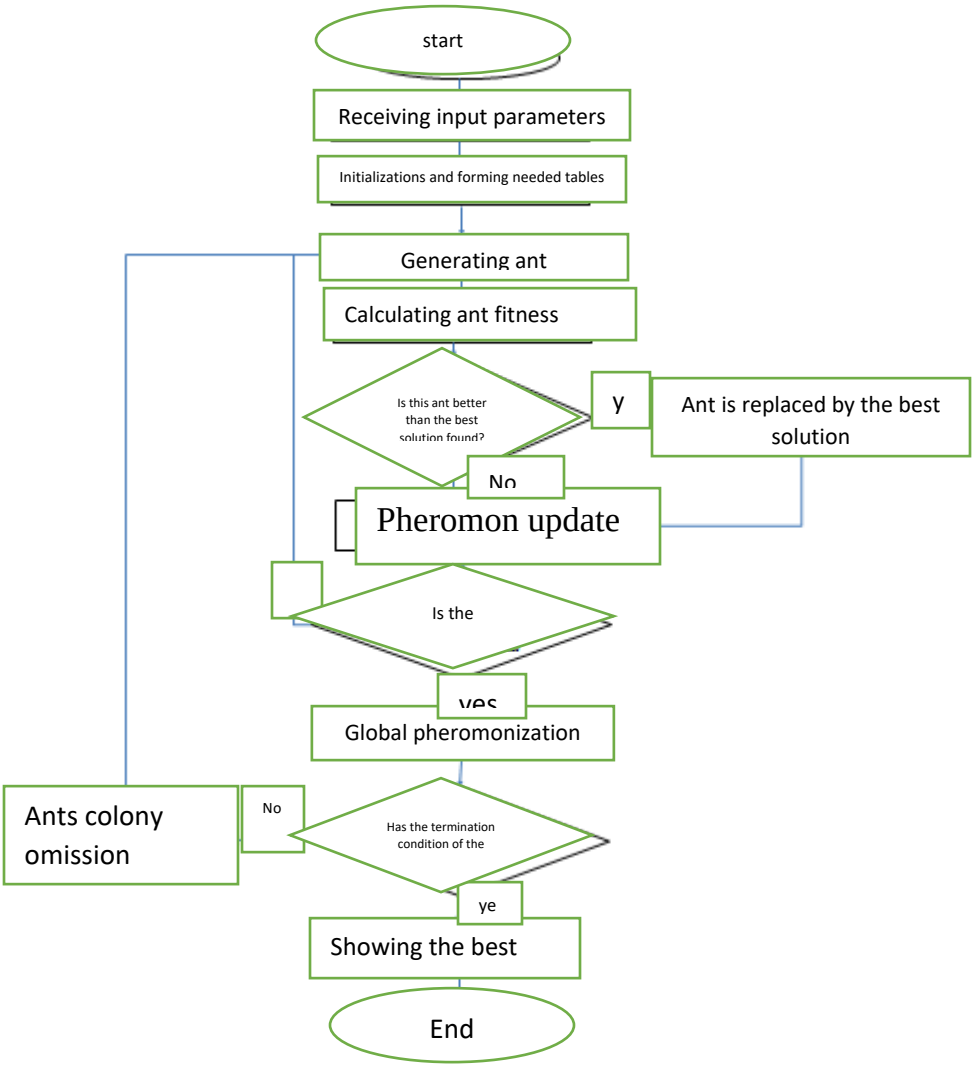

Figure 1. Flowchart of the proposed algorithm steps

The magnitude of the problem parameters is first given to the suggested method. The system then begins coding each task from a single request because one user request may include several tasks. Consider Table 1 to help you comprehend this section (1). There are four columns in a table. The task number, which is unique, appears in the first column. The username appears in the second column, the request number appears in the third column, and the processing status of each task appears in the fourth column. The processing status for all tasks is zero when the table is first created, and after processing is complete, it is set to one.

Table 1. Coding tasks for processing

Processing status	Application Number	User's name	Task (execution) number
0	1	U1	0
0	2	U1	1
0	3	U1	2
0	1	U2	3

0	2	U2	4
0	3	U2	5
0	1	U3	6
0	2	U3	7
0	3	U3	8
0	4	U3	9

The pheromone table is then constructed based on the table produced for numbering the tasks. The ant colony algorithm's pheromone table is a n×n matrix, where n is the number of rows in the coding table. Each cell i × j in the pheromone table depicts the ant's movement from the task I to source j. The initial pheromone value for each component of the pheromone table is defined in this phase. The method is provided the starting pheromone value through the input parameter, which is a modest positive constant value.

Findings

The suggested technique is implemented using the CloudSim simulator. On a machine with an Intel (R) Pentium (R) 4 3.00GHz CPU and 4.00GB of main memory, the method executes. The suggested approach will be put to the test using 11 test datasets. In order to be suitable for the real world, this dataset has been examined in terms of the size of small, medium, and big habitats. It should also be done in a fashion that pleases all users, with tasks distributed among appropriate processors to handle more tasks in less time and decrease downtime in the cloud.

Designing environment simulation parameters

The Cloudsim software is utilized to simulate the suggested approach in this study. Cloudsim, in fact, is a frametask developed by the GRIDS lab at the University of Melbourne that allows for modeling, simulation, and implementation of cloud computing infrastructure architecture. At the vm level, the ant colony method will be utilized to assign input to virtual machines (VM scheduling).

Table 2. Cloudsim simulator parameters

Type of entity	Parameters	Values
Datacenter	Number of data centers	10
Virtual machine	Total number of virtual machines	50
	Number of instructions per second	500-2000
	Bandwidth	500-1000
	RAM Virtual machine memory	256-2048
	Operating system type	Windows, Linux, Unix
	Number of required processors	1-4
	Cost of CPU usage	100-300
Task	Task duration	1000-20000
	Total number of tasks	100-1000

Ant cloning algorithm simulation parameters

The parameter values should be evaluated in Table after developing the ant colony algorithm and evaluating the parameters that impact the algorithm computations (3). The parameters were set to default values of $\alpha = 1$ ، $\beta = 1$ ، $\rho = 0.5$ ، $Q = 100$, and the maximum number of iterations was 1000. To acquire the right values, just one of the parameters was altered in each experiment.

Table 3. Parameters of the proposed algorithm

PARAMETER	VALUE
α	0.3
β	1
ρ	0.4
Q	100

Designing test dataset

Ten test datasets covering small, medium, and large cloud environment systems are created to assess the outcomes of the proposed technique. Table 1 shows the test dataset (4).

Table 4. Designed test dataset

Total number of tasks	Total number of users	Test set name
100	30	Test01
290	48	Test02
320	60	Test03
498	110	Test04
400	100	Test05
350	90	Test06
235	80	Test07
756	130	Test08
600	75	Test09
960	200	Test10

Each user can offer values for processing their tasks based on the values in Tables (3) and (4). Cloud computing, as previously said, is based on a contract between the user and the service provider. For instance, according to the pricing range specified in Table (4) for CPU consumption, the user provides a payment in the range of [1000 and 10000]. The major cost is then computed. In addition, the user includes in his request the time he wants his delivery to be processed and completed in a certain time period, which is a number in a specific period.

Simulation results of the proposed algorithm

The results of the suggested algorithm's implementation for the test dataset in Table (5) are shown in Table (5). (5). The suggested technique was ran 20 times on each example of the test dataset to determine the average fitness value.

Table 5. The results obtained from the implementation of the proposed algorithm

Mean 20 fitness	**Best fitness**	**Best implementation time**	The number of the best iteration	**Test set name**
105.6	102.5	14	493	Test01
299.3	286.3	33	854	Test02
401.9	100.1	49	902	Test03
1264.8	1078.3	85	995	Test04
500.1	481.3	70	803	Test05
423.8	419.6	75	881	Test06
317.4	313.8	78	863	Test07
782.5	756	128	971	Test08
306.9	305.9	83	743	Test09
946.2	923.5	193	950	Test10

One of the objectives of this study was to lower the time it took to complete all tasks. As a result, in Figure (2), you can see the Makespan diagram, which illustrates how long it took each of the test datasets to complete the whole task processing using the suggested genetic algorithm and the particle swarm optimization method. The suggested method was able to process the desired tasks faster than the other two algorithms, as seen in this diagram. This minimizes the time it takes for tasks to be processed, which lowers the system's maintenance costs. Reducing the time it takes to accomplish all requested activities allows limited resources to be allocated to other tasks sooner. This will also help service providers since an increase in the number of tasks handled will boost their profits.

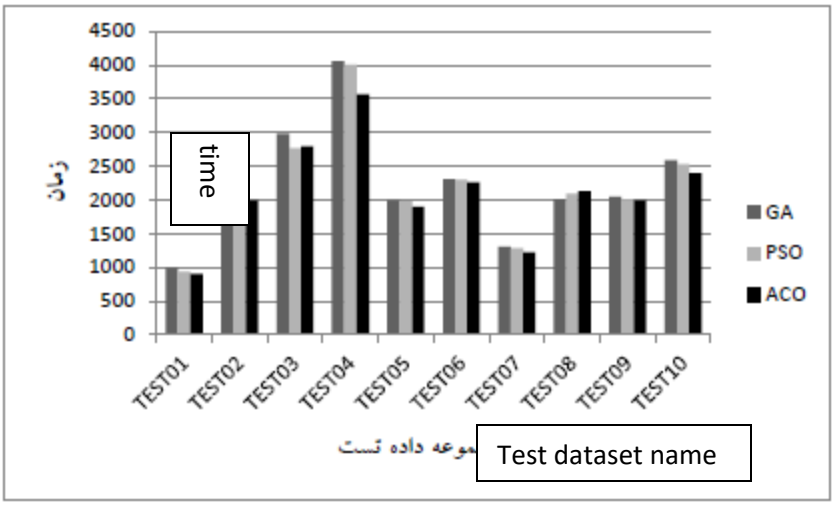

Figure 2: The completion time of total processing of test dataset tasks listed in Table 5

It is preferable to utilize the value obtained from the fitness in different iterations of the test data set to compare the performance of the proposed method with the genetic algorithm and the particle swarm optimization algorithm. The suggested method for - Test05, as shown in Figure (3), was able to lower the Fitness value very rapidly in the same starting repeats, and this reduction process continues until the middle repetitions, yielding superior results to the other

two algorithms. This indicates that the proposed strategy is effective for small-scale scheduling issues. He sought to choose a task for each machine to complete that minimized the processing time of all tasks at the start of the journey.

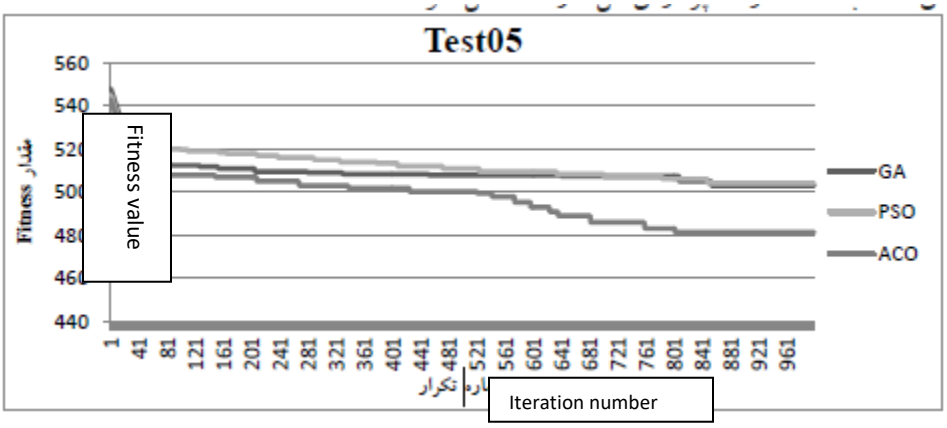

Figure 3. Comparing the fitness chart for Test05 test data

For the Test07 test data, as shown in Figure (4), the proposed method outperformed the other two algorithms and was able to improve the best solution in the same number of iterations.

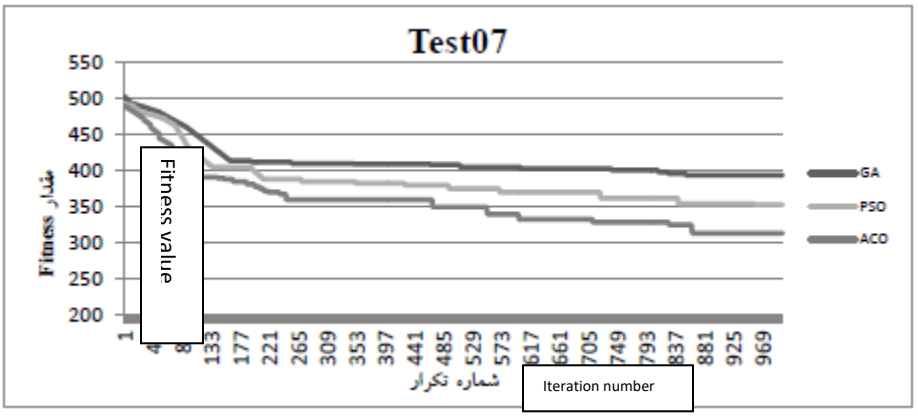

Figure 4: Comparing the fitness chart for Test07 test data

Figure (5) shows how the suggested method improves the best response in each iteration for the Test09 test data. The other two algorithms, like the suggested approach, improved the best answer achieved in the final iterations, and the algorithms are in tight competition. The suggested method, on the other hand, was able to win the competition and obtain a superior result. The fact that the response improved in the last rounds suggests that the suggested algorithm did not reach a point of early convergence.

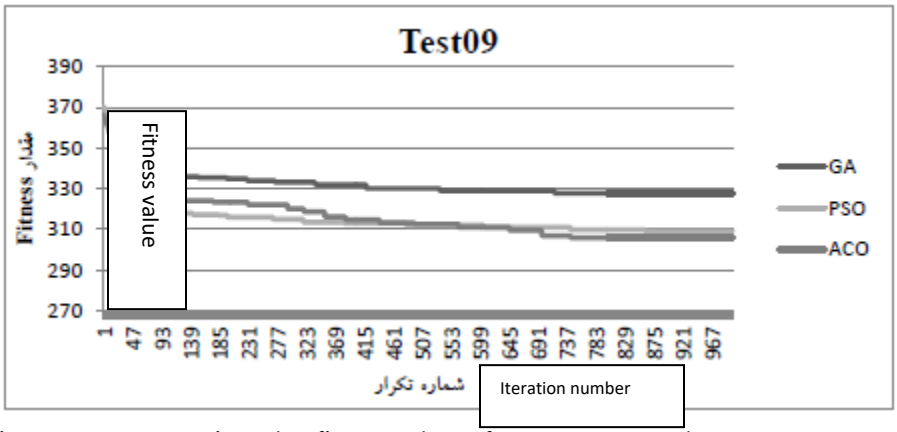

Figure 5: Comparing the fitness chart for Test09 test data

After establishing that the proposed technique has accomplished our aims, the algorithm's stability should be examined (reducing execution time and reducing costs). Stability is defined as a minor variation in the values of the objective function achieved by the algorithm after multiple runs with the same settings on test data. In other words, the algorithm's resistance to the results it has acquired. The suggested approach outperformed the competing algorithms in each iteration, as shown by the Test03 stability diagram in Figure (6). In the last iterations, the best response is acquired. The results acquired in each iteration, however, were less complicated than those obtained in the other two methods.

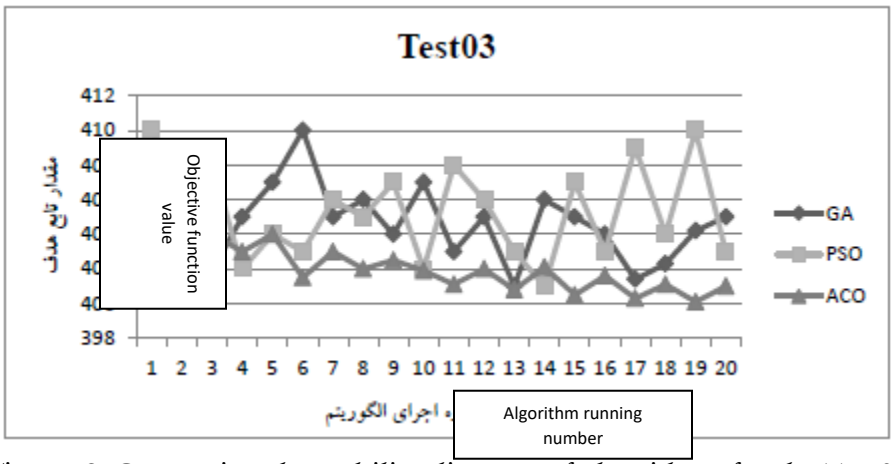

Figure 6: Comparing the stability diagram of algorithms for the Test03 test dataset

The stability comparison chart for - Test06 is shown in Figure (7). The suggested algorithm yielded somewhat varied results, indicating that it has a high level of reliability. The solutions acquired by the particle swarm optimization method, on the other hand, are dispersed, indicating that it is not particularly stable, and its responses are substantially different from the suggested algorithm's replies. The genetic algorithm outperformed the particle swarm optimization technique in this study.

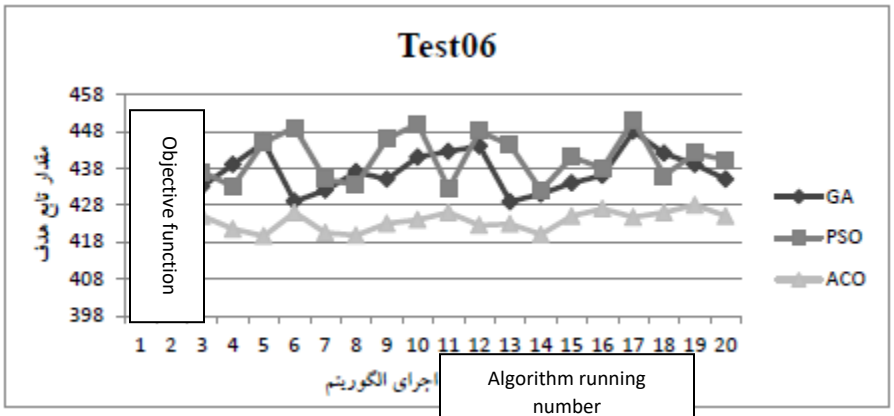

Figure 7. Comparing the stability diagram of algorithms for the Test06 test dataset

The suggested approach has higher stability than the other two algorithms, as shown in Figure (8), which is connected to the Test09 test data set, and the difference between the responses is less than the best answer of the other two algorithms. For this test dataset, the genetic algorithm and the particle swarm optimization technique are both unreliable, with considerable differences in the solutions obtained in each iteration.

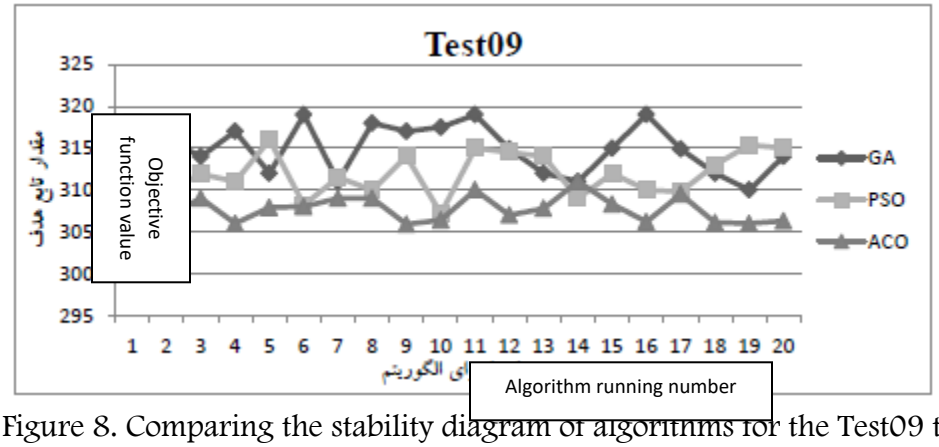

Figure 8. Comparing the stability diagram of algorithms for the Test09 test dataset

According to the diagrams above, the suggested algorithm performs significantly better, allowing user activities to be handled in less time. Processors' profitability and efficiency will improve in the cloud if task time is reduced.

Conclusion

Following an examination of the problem of task scheduling in the cloud environment, a model was suggested in which users' requests in terms of cost and time were taken into account. Many studies have found that getting the optimal solution takes a long time and that all aspects of the cloud environment in the real world, such as bandwidth, cost of utilizing each processor, processing capability of each processor, and so on, have not been taken into account. In this paper, an ant colony algorithm-based technique for job scheduling is given. The suggested approach aims to take into account the majority of the factors in order to produce an effective job scheduling algorithm. The results of the suggested approach were compared to those of the genetic algorithm and the particle swarm optimization algorithm in the CloudSim simulation environment. The findings indicated that the suggested method is very efficient and can shorten

the time it takes to complete a task. Also, to please consumers by striking a balance between time and cost, which are two of the most important factors for service quality.

The utilization of the ant colony optimization method, one of the newest task scheduling techniques in cloud computing, is a novel component of this study. It was also attempted to be examined in this study, unlike many other studies that did not take into account the critical criteria of bandwidth. All efforts in this study are focused on making the most of cloud computing resources, which will boost resource efficiency and lower consumption costs. Furthermore, the suggested cloud environment model was studied in such a manner that it can represent the real environment while taking into consideration all of the criteria required by a user as well as the features of cloud resources.

The suggested method's shortcoming is the ant colony algorithm's entrapment in local optimization and its early convergence. However, with more research, this flaw can be avoided. It is preferable to employ a mix of the two methods to overcome the meta-heuristic algorithm's shortcoming of early convergence and avoid falling into the optimum local trap. When the first algorithm encounters an issue, it employs a mix of ant colony and other local search algorithms to conduct local searches in the problem space, assisting the algorithm in finding a better solution. However, this technique has problems, one of which is that it increases the algorithm's execution time, which is also offered as a remedy to the problem.

Acknowledgment: None
Conflict of Interest: None
Funding: None
Ethical statements: None

References

- Ghaffari, Amir Reza "*Cloud Computing Systems: Examples; Applications; Challenges.*" Shahid Beheshti University, September 2010
- Ghorbannia, Arash and Arian, Yalda. *Presenting a hybrid scheduling method based on genetic algorithm in cloud computing environment*". The first specialized conference on intelligent computer systems and their applications, 2011
- Fangzhe C., Ren J., and Viswanathan R., "Optimal Resource Allocation in Clouds," in Proceedings of the 3rd International Conference on Cloud Computing, Florida, USA, pp. 418- 425, 2010.
- Hai Zhong, Kun Tao, Xuejie Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems," The fifth annual ChinaGrid conference, 2010, pp. 124-128.
- Medhat A.Tawfeek, Ashraf EL-Sisi, Arabi E.Keshk, Fawzy A.Torkey, "Cloud task scheduling based on Ant colony optimization," 8th International Conference on Computer Engineering & Systems (ICCES), pp.64-69, 2013.
- Sellami K, Ahmed-Nacer M, Tiako PF, Chelouah R. Immune genetic algorithm for scheduling service workflows with QoS constraints in cloud computing. South African J Ind Eng 2013;24:68–82.

- Xiaotang Wen, Minghe Huang, Jianhua Shi, “Study on resource Scheduling based on ACO algorithm and PSO algorithm in cloud computing,” 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2012.
- Zhang Z, Zhang X. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. Int Conf Ind Mechatronics Autom 2010;2:240–3.

.