An efficient task scheduling in cloud computing based on ACO algorithm

Zahra Shafahi
Master of Computer architecture
Science and Research Branch,
Islamic Azad University
Tehran, Iran
shafahi.z90@gmail.com

Alireza Yari
Faculty member
ICT Research Institute (Iran Telecom Research Center)
Tehran, Iran
A_yari@itrc.ac.ir

2021 12th International Conference on Information and Knowledge Technology (IKT) | 978-1-6654-0849-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IKT54664.2021.9685674

Abstract— **Resource allocation as a NP-hard problem is a very important part of cloud computing and is examined in the form of scheduling algorithms. An Ant Colony Optimization (ACO) algorithm was proposed in this study to improve the load balancing performance and makespan time parameters. Most of the tasks scheduling algorithms have been proposed to improve one of the service quality parameters for service providers or users and do not address the needs of both at the same time. Since an appropriate scheduling algorithm should be able to consider the quality requirements of users and service providers simultaneously, for this purpose in this paper we have proposed a new algorithm for scheduling tasks in cloud environment. The proposed algorithm is based on the ACO algorithm and studied in comparison to a Particle Swarm Optimization (PSO) algorithm, a Genetic Algorithm (GA) and also another research based on ACO. The proposed algorithm has showed the significant improvements concerning the makespan time, load balancing, execution time and resource utilization against the compared algorithms.**

Keywords— Cloud computing, Task scheduling, Ant Colony algorithm (ACO), Makespan, load balancing

I. Introduction

Cloud computing is the latest computing model for a variety of web applications, data and IT services. In the cloud computing implementation process, task scheduling or resource scheduling is an unavoidable task, which directly determines the entire system's performance. Cloud computing system has a wide range of resources, including heterogeneous resources, extensive user base, and different types of application tasks. Therefore, QoS(Quality of Service) target constraints are different, and cloud computing systems must be integrated with a large number of user tasks and data [1]. Thus, task scheduling strategy in cloud computing, mostly an NP-hard problem, is a challenging issue that is difficult to study. We need an efficient algorithm for scheduling work in the cloud environment [2]. Developing these services and using them is another challenge [3]. priority, deadline, etc., which has a direct impact on user consumption of resources [4].

The cloud service providers offer different types of services, which is why the number of applications and users in the cloud is increasing. It is difficult to handle all the requests from the users in the shortest response time and ensure satisfaction based on service quality parameters. To this end, computer scientists are finding solutions to the optimal use of cloud resources and are proposing scheduling algorithms. The main purpose of these task scheduling algorithms is to guarantee time compatibility. For this purpose, methods are proposed to help service providers perform tasks complying with the logical constraints of transactions. Some researchers try to find high-capacity systems to process and delegate tasks to these systems. Another group of researchers believes in using all systems and consider optimization in load balancing.

But for now, the results show that the two approaches must be considered together. Scheduling tasks without considering the load balancing in the cloud not only does not solve the problem but also creates more problems. This is due to the different resources in the cloud. Because this difference causes all tasks to be directed to high-performance resources, and the problem of creating overhead in those resources is happened, and on the other hand, lower-performance resources remain unused [5].

On the other hand, if the tasks are assigned to machines without considering the required resources, this will increase the execution time and response time as well as the system overhead. By assigning tasks to virtual machines, the cloud creates problems that can be solved with scheduling algorithms. Runtime and cost are issues that drive scheduling algorithms. On the other hand, the resources in the cloud are changing at any time, so the task scheduling is an important issue that has a great impact on the performance of the cloud computing environment.

Due to the NP-hard nature of the task scheduling problem in the cloud environment, various methods have been proposed to provide an efficient solution for it. Each of these methods seeks to improve one of the goals of service quality, some of the most important of which include providing the quality of service parameters required by each program, such as response time; cost reduction for service provider and user; ability to increase capacity and scalability according to user needs; maximizing the use of resources, and creating reliability.

So far, many researchers have used the ACO algorithm for scheduling. Some of them have done the scheduling only using this algorithm and have not paid attention to the load balance. The results of these methods showed that all tasks were assigned to one or two machines and the rest of the machines were left unused. In this research, an attempt has been made to pay attention to both load balance and makespan in scheduling tasks. In the proposed method in this research, using the ACO, we have presented a method for distributing requests among virtual machines. In addition to increasing the load balance and reducing the execution time of the longest task, this method also increases the utilization of resources and reduces the execution time of tasks. The algorithm proposed in the above method has an advantage over the ant colony basis method; So that in the proposed method, through the similarity function and its use in the ant personal opinion section in the probability function of the ant colony algorithm, the most suitable virtual machine can be selected to host each request. In such a way that the resources available in the selected machines are most similar to the request requirements assigned to them. This leads to a fair distribution of requests between different virtual machines and thus increases the load balance due to the proper selection of virtual machines. In addition, in the proposed method, we have changed the probability function of the ant colony algorithm, in which we have also considered the effect of the execution time of the requests. Compared to previous methods, this makes it possible to use the proposed method to select the most suitable machines with the least execution time and the most similarity in terms of resource requirements to host requests. Reducing execution time increases customer satisfaction and load balance, and the utilization of resources increases the satisfaction of service providers.

The difference between this algorithm and the previous research [3] is that in this method, we did not deal directly with high-capacity machines, and on the other hand, we did not focus on machines with the lowest capacity. The goal was to find a machine with the most appropriate amount of resources selected for each task, which could minimize the execution time of the tasks. This balances the load and increases the utilization of resources, reducing the waiting time to perform tasks that require a lot of resources.

In the remaining parts of this paper, we review research background in the next section. Then we introduce the proposed algorithm in the third section and the results of its simulation as well as research results are provided in the section 4. Finally, the conclusion comes in the section 5.

II. RESEARCH BACKGROUND

Cloud-based task scheduling algorithms in distributed systems are mainly classified into three categories: heuristic, meta-heuristic, and hybrid. heuristic algorithms are problem-dependent and perform well for one range of problems but perform poorly for others. The problem-dependent heuristic algorithms are known to perform well for one range of problems, but poorly for others. Heuristic scheduling provides an optimal solution that leverages knowledge bases to make decisions about scheduling [6].

On the other hand, meta-heuristic techniques tackle these problems by providing approximate optimal solutions in a reasonable amount of time [7]. The traditional methods used in optimization are definitive and fast, and usually give accurate answers, albeit working locally. Scheduling tasks is more difficult in a large search space, and the number of solutions available is large, which implies that the tasks must take longer to find the optimal solution. In such situations, there is no well-defined method to solve this problem. However, in the cloud, finding a near-optimal solution is also known to be effective. In the field of IT tasks, methods focus more on meta-heuristic methods [7]. The different types of metaheuristic scheduling algorithms include ACO, GA(Genetic Algorithm), PSO(Particle Swarm Ooptimization), LCA(Life Cycle Assessment) and ABC(Artificial Bee Colony).

Ant Colony Algorithm (ACO) is classified as one of the metaheuristic algorithms. ACO methods are useful for solving discrete optimization problems that involve different ways of achieving the goals. It has been successfully used to solve the Traveling Salesman Problem (TSP), the Knapsack Problem, the second-degree scheduling problem, as well as the scheduling of network and cloud tasks, among many others. The first step toward achieving any solution using ACO is to map (model) the ant system for the problem.

Reference [8] proposed a workflow scheduling algorithm based on the ACO by adding several new features to improve it. Their endeavor was to minimize costs by setting a deadline. For this purpose, two types of pheromones were defined: one is the tendency to minimize cost whereas the other is to minimize the makespan time. They defined three types of heuristic information to help ants search.

Reference [9] in another study, present a method that modifies the cloud task scheduling policy based on the ant colony optimization (MACO) algorithm. The main contribution of the recommended method is to minimize the makespan time and perform the multi-purpose planning process by assigning the amount of pheromone to the performance of the relevant virtual machine.

Reference [10] focuses on the load imbalance problem in System Wide Information Management (SWIM) task scheduling. At the aim of meeting the quality requirements of users for task completion, they studied large-scale network information system task scheduling methods. This research Combined with the traditional ACO, used the hardware performance quality index and load standard deviation function of SWIM resource nodes to update the pheromone, a SWIM ant colony task scheduling algorithm based on load balancing (ACTS-LB). This ACTS-LB algorithm can reduce task execution time and improve utilization of system resources, thus ensuring that the utilization of system resources is met as much as possible.

In their study where they compared the findings of our work, [11] proposed a cloud task scheduling policy based on the ACO algorithm. They compared their results with different FCFS and Round-Robin scheduling algorithms

The main purpose of this algorithm is to minimize the amount of creation of a specific work setting. The algorithms are simulated using the Cloudsim toolkit. Experimental results showed that cloud task programming based on ACO performs better than FCFS and Round-Robin algorithms.

Some researchers have focused on resource load balancing to improve task scheduling performance in the cloud. Reference [12] proposed the LBACO algorithm for scheduling independent tasks with the aim of minimizing time intervals and even loading on all virtual machine [12]. Reference [13], Presented a new ACO algorithm called SACO with slave ants for scheduling tasks in cloud environments. This algorithm schedules the tasks of cloud users for virtual machines in the cloud computing environment efficiently by mapping the optimal parameters.

Reference [14] presented an adaptive load-balanced task scheduling algorithm for cloud computing. The proposed approach has been evaluated and compared with ACO, GA works and the results confirms the improvement concerning the resource utilization, makespan, and SLA violation.

III. PROPOSED ALGORITHMS

In the proposed method, using the ant colony algorithm [15], we present a method for distributing requests among virtual machines to optimize both the makespan time and the resource efficiency. The advantage of our algorithm compared to the basic ACO algorithm is that we use the similarity function in the probability function of the ant colony algorithm. Thus, the proposed method can choose the most suitable virtual machine to host each request by measuring the similarity of the resources available in the selected machines and the request requirements assigned to them. This leads to a fair distribution of requests between different virtual machines and also increases the load balancing, due to the appropriate selection of virtual machines.

In addition, in the proposed method, we have changed the probability function of the ant colony algorithm and also affected the execution time of the requests. This has made it possible to use the proposed method in comparison with the previous methods, to select the most suitable machines with the minimum execution time and the most similarity in terms of resource requirements for hosting requests. Reducing execution time increases customer satisfaction and increases the load balancing and resource efficiency, thus increasing the satisfaction of service providers.

In the first step, we create the structure of each ant as an array so that the index of each member represents the task number and its value is equal to the number of the machine executing that request. The length of each array is the number of tasks.

In the next step, the initial population must be randomly assigned, i.e., we must select a random virtual machine for each request.

In the third step, the termination condition of the algorithm is checked. If the termination condition is not met; in the fourth step to the eighth step, for each ant (step four), each of its tasks (step five), and each virtual machine (step six), we calculate the probability coefficient (step seven). Then in the ninth step, we assign the task to the machine that is most likely. In the tenth step, the available resources of the selected machine are updated. In the twelfth step the amount of makespan of each solution is calculated and the best makespan is updated (steps fourteen and fifteen). In the seventeenth step, the local pheromone value is updated based on the makespan of the solution, and then the global pheromone value is updated using the optimal solution selected in the fourteenth step. These steps are repeated and return to the third step and the termination condition is checked. If the termination condition is met, go to step 21 and the best solution selected in step fourteen is returned as the final output of the algorithm. In the following example, we show the task scheduling map using the ant colony algorithm.

In metaheuristic algorithms, each solution is a permutation of requests on virtual machines. The solution in the ant colony algorithm is ant. Here we consider two ants and one iteration. For example, if we have 5 requests and 3 virtual machines, the first ant permutation can be as follows:

K=1

TABLE I. FIRST ANT PERMUTATION

T1	T2	T3	T4	T5
1	1	2	3	2

For this permutation, the pheromone level is updated as follows. The amount of pheromone is 1/MS; For example, if the MS(Makespan) of this solution is 8, the pheromone level will be 1/8. and this level is only added to the paths on which the requests are located. The amount of pheromone of the first ant is updated according to the table below. Note that in this example the initial pheromone content is assumed to be 0.2.

TABLE II. UPDATED LOCAL PHEROMONE FOR THE FIRST ANT

T1	T2	T3	T4	T5
τ11= 0.2+1/8	T21= 0.2+1/8	τ 31= 0.2+0	τ 41= 0.2+0	τ 51= 0.2+0
τ 12= 0.2+0	τ 22= 0.2+0	τ32= 0.2+1/8	τ 42= 0.2+0	T52= 0.2+1/8
τ 13= 0.2+0	τ 23= 0.2+0	τ 33= 0.2+0	τ 43= 0.2+1/8	τ 53= 0.2+0

Now the second ant is assumed to have chosen the following path.

K=2

TABLE III. SECOND ANT PERMUTATION

T1	T2	T3	T4	T5
2	2	3	1	2

After this selection, the amount of pheromone is updated based on MS of this solution and the amount of available pheromone. For example, if the amount of MS in this solution is 6, the amount of pheromone after selecting the second ant will be updated as shown in the table below.

TABLE IV. UPDATED LOCAL PHEROMONE FOR THE SECOND ANT

T1	T2	T3	T4	T5
τ11= 0.2+1/8 +0	τ21= 0.2+1/8+1/6	τ 31= 0.2+0+0	τ 41= 0.2+1/6	τ 51= 0.2+0+0
τ 12= 0.2+0+1/6	τ 22= 0.2+0+0	τ32= 0.2+1/8+0	τ 42= 0.2+0+0	T52= 0.2+1/8+1/6
τ 13= 0.2+0+0	τ 23= 0.2+0+0	τ 33= 0.2+0+1/6	τ 43= 0.2+1/8	τ 53= 0.2+0+0

After the local update, the pheromone is updated globally. To do this, the pheromone of the best ant (the ant with the lowest amount of MS) is added to the existing pheromones. For example, if in the above example the second ant has the best MS among the solutions, the amount of pheromone at the end of this iteration will be updated as shown in the table below.

TABLE V. GLOBAL PHEROMONE UPDATE AFTER ITRATION USING THE TEMPLATE

T1	T2	T3	T4	T5
τ11= 0.2+1/8 0+0+0	τ21= 0.2+1/8+1/6 +1/6	τ31= 0.2+0+0+0	τ41= 0.2+1/6+ 1/6	τ51= 0.2+0+0+0
τ12= 0.2+0+1/6+ 1/6	τ 22 = 0.2+0+0+0	τ32= 0.2+1/8+0+ 0	τ42= 0.2+0+0+ 0	τ52= 0.2+1/8+1/6 +1/6
τ13= 0.2+0+0+0	τ 23= 0.2+0+0+0	T33= 0.2+0+1/6+ 1/6	τ43= 0.2+1/8+ 0+0	τ53= 0.2+0+0+0

The flowchart of the proposed method is shown in Figure 1.

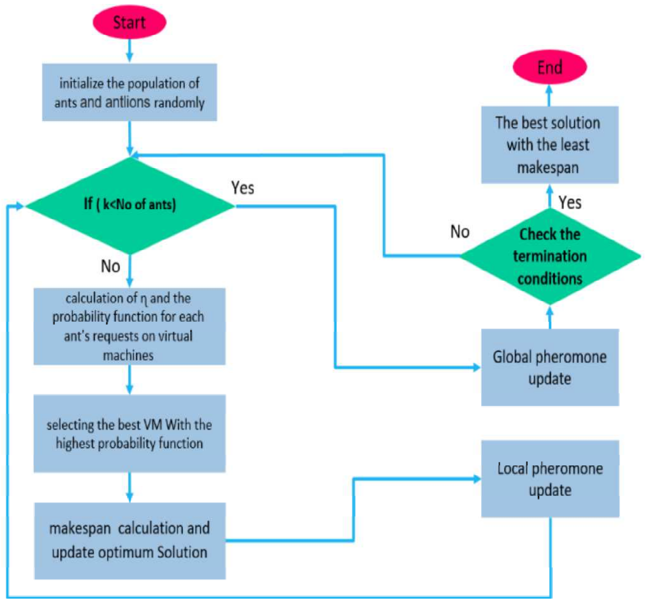

Start
initialize the population of ants and antlions randomly
If (k<No of ants)
Yes
No
calculation of η and the probability function for each ant's requests on virtual machines
selecting the best VM With the highest probability function
makespan calculation and update optimum Solution
Local pheromone update
Global pheromone update
Check the termination conditions
No
Yes
The best solution with the least makespan
End

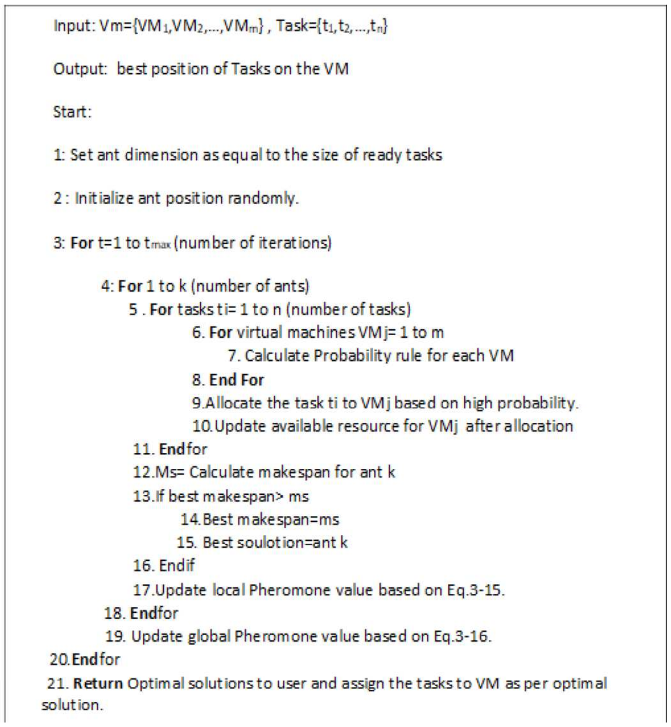
Fig. 1. The flowchart of the proposed method

According to the number of steps of the algorithm, the position of each ant is updated based on the probability function, and also the values of the local pheromone and the global pheromone are calculated based on the position of each ant and group of ants. Finally, the ant with the least of makespan is considered as the answer.

Figure 2 shows the pseudo-code of the above algorithm.

IV. SIMULATION RESULTS

To compare the proposed method with other methods, we have implemented them in the CloudSim simulator. To compare the results, we have considered the parameters of makespan time, degree of load balancing, execution time, resource efficiency.



```
Input: Vm={VM_1,VM_2,...,VM_m} , Task={t_1,t_2,...,t_n}

Output: best position of Tasks on the VM

Start:

1: Set ant dimension as equal to the size of ready tasks

2 : Initialize ant position randomly.

3: For t=1 to t_max (number of iterations)

    4: For 1 to k (number of ants)
        5 . For tasks ti= 1 to n (number of tasks)
            6. For virtual machines VMj= 1 to m
                7. Calculate Probability rule for each VM
            8. End For
            9.Allocate the task ti to VMj based on high probability.
            10.Update available resource for VMj after allocation
        11. Endfor
        12.Ms= Calculate makespan for ant k
        13.If best makespan> ms
            14.Best makespan=ms
            15. Best soulotion=ant k
        16. Endif
        17.Update local Pheromone value based on Eq.3-15.
    18. Endfor
    19. Update global Pheromone value based on Eq.3-16.
20.Endfor
21. Return Optimal solutions to user and assign the tasks to VM as per optimal solution.
```

Fig. 2. Pseudo-code algorithm

Thus, each of the paper swarm optimization algorithms, genetics and the base ACO algorithm [11] run ten times in order to simulate and the values of the objective function are calculated. The average of ten executions is used for each algorithm.

First, we analyze the results of the makespan amount between different methods. The proposed method reduces the makespan in comparison to other methods due to the productivity of this algorithm in searching optimally and without falling into the local optimization and also improving the personal opinion of each ant by selecting a virtual machine tailored to the requirements of requests. In addition, the proposed method reduces the amount of makespan compared to other methods, because the amount of makespan on the pheromones of each ant is reduced.

The results show that the proposed method can select the solutions that have less makespan, by giving more points to the routes with less makespan and more pheromones. Figure 3 shows the makespan time between different methods.

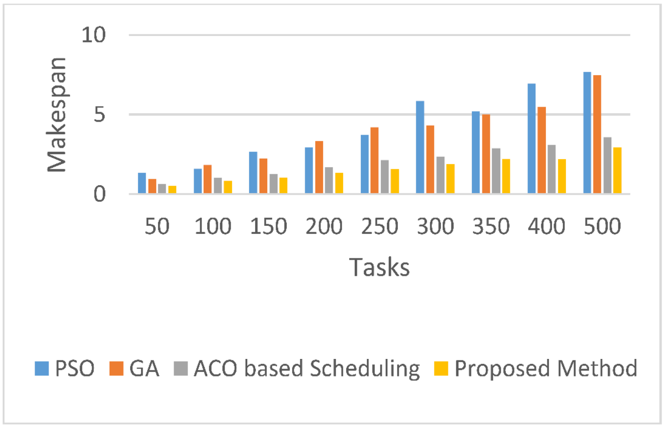


Fig. 3. Comparison of makespan between the proposed method and other methods

The results show that the amount of execution time in the proposed method has been significantly reduced compared to other methods. This is due to the decrease in makespan. Makespan reduction reduces the execution time of tasks in the proposed method compared to other methods. Figure 4 shows the average execution time between different methods.

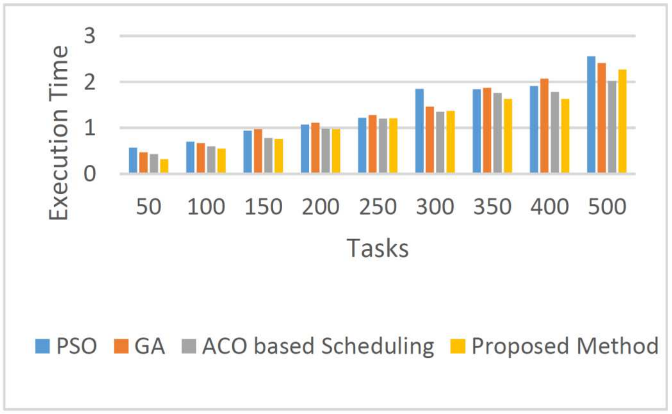


Fig. 4. Comparison of execution time between the proposed method and other methods

The simulations show that the proposed method has improved the load balancing in comparison to other methods. In the proposed method the most suitable virtual machine will select according to the similarity of the required resources, and the existing virtual machine. Figure 5 shows the load balancing for different methods.

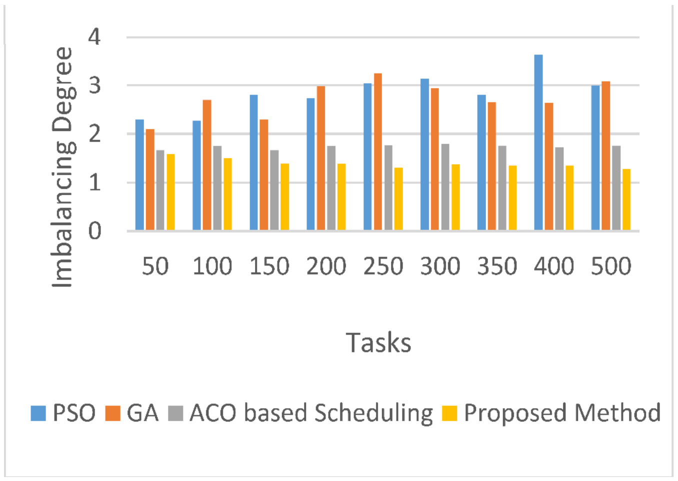


Fig. 5. Comparison the degree of load imbalance between the proposed method and other methods

The results also show that the amount of resource utilization in the proposed method is higher than other methods. Since in the proposed method the amount of makespan is less compared to other methods, the resource utilization increases. Thus, in the proposed method the amount of resource utilization is higher than other methods. Figure 6 shows the resource utilization between different methods.

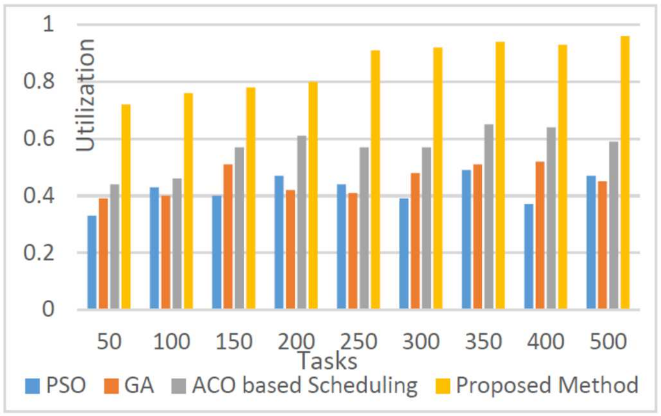


Fig. 6. Comparison of resource utilization between the proposed method and other methods

V. CONCLUSION

In this research, using the ant colony algorithm, we have presented a method for distributing requests among virtual machines. In addition to reducing execution time and makespan time, this method improves the utilization of resources. The advantages of the proposed algorithm in comparison to the basic ant colony method are the use of similarity function and the ant's personal opinion section in the ant colony probability function. In such a way that the resources available in the selected machines are most similar to the request requirements assigned to them.

This leads to a fair distribution of requests between different virtual machines and improvement of the load balancing, due to the appropriate selection of virtual machines. In the proposed method, we have changed the probability function of the ant colony algorithm and also affected the execution time of the requests. This allows to use the proposed method to select the most suitable machines with the least execution time and the most similarity in terms of resource requirements to host requests. Reducing execution time increases customer satisfaction and increases the load balancing and resource efficiency, improves the satisfaction of service providers.

The difference between this algorithm and the previous ones is that in this method, we have not gone directly to high-capacity machines, and on the other hand, we have not focused on machines with the lowest capacity.

The goal was to have a machine with the least amount of resources selected, which could minimize the time required to perform tasks. This reduces the waiting time to perform tasks that require a lot of resources.

Due to the strengths of the proposed algorithm, this model has also faced some limitations. Although the ACO has many advantages, it has a slower convergence rate than other algorithms. This is one of the limitations of ACO. Another limitation that should be considered in using the scheduling algorithm proposed in this research is the great variety of factors and parameters that could be considered in the algorithm. The extent of these criteria, factors or parameters is such that the selection of a limited number of them is inevitable. Although in selecting the factors considered in the proposed algorithm, an attempt was made to consider the most important and relevant factors, but other issues such as security issues, energy consumption, cost consideration and throughput can also be included in future research.

In addition, in this study, we have not examined overhead (time complexity) and solutions to respond to it, which can be considered as a basis for future research.

REFERENCES

[1] Delimitrou, C., & Kozyrakis, C. Quality-of-service-aware scheduling in heterogeneous data centers with paragon. *IEEE Micro*, vol. 34, no.3, pp. 17-30, 2014.

[2] Guo, Q. "Task scheduling based on ant colony optimization in cloud environment," in *AIP Conference Proceedings*, AIP Publishing LLC. vol. 1834, no. 1, p. 040039, April 2017.

[3] Ebadifard, F., & Babamir, S. M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurrency and Computation: Practice and Experience*, vol 30, no. 12, 2018.

[4] Elzeki, O.M., Improved max-min algorithm in cloud computing. *Int. J. Comput. Appl*., vol 50, no. 12, 2012.

[5] Kumar, M., Sharma, S. C., Goel, A., & Singh, S. P. A comprehensive survey for scheduling techniques in cloud computing. *Journal of Network and Computer Applications*, pp.143, 1-33, 2019.

[6] Mathew, T., Sekaran, K. C., & Jose, J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In *2014 International conference on advances in computing, communications and informatics*, pp. 658-664. IEEE. September 2014.

[7] Kalra, M., & Singh, S. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian informatics journal*, vol 16, no. 3, pp. 275-295, 2015.

[8] Chen, W. N., Zhang, J., & Yu, Y. Workflow scheduling in grids: an ant colony optimization approach. In *2007 IEEE Congress on Evolutionary Computation* , pp. 3308-3315, IEEE, september, 2007.

[9] Reddy, G. N., & Kumar, S. P. MACO-MOTS: modified ant colony optimization for multi objective task scheduling in Cloud environment. *International Journal of Intelligent Systems and Applications*, vol.11, no.1,pp 73, 2019.

[10] Li, G., & Wu, Z. Ant colony optimization task scheduling algorithm for SWIM based on load balancing. *Future Internet*, vol.11. no.4, pp.90, 2019.

[11] Tawfeek M. A., El-Sisi, A., Keshk, A. E and Torkey, F. A."Cloud task scheduling based on ant colony optimization," in *2013 8th International Conference on Computer Engineering & Systems* (ICCES), pp. 64-69, 2013.

[12] Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. Cloud task scheduling based on load balancing ant colony optimization. *In 2011 sixth annual ChinaGrid conference*,pp. 3-9. IEEE, August 2011.

[13] Moon, Y., Yu, H., Gil, J. M., & Lim, J. A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. *Human-centric Computing and Information Sciences*, vol 7, no. 1, 28, 2017.

[14] Mubeen A, Ibrahim M, Bibi N, Baz M, Hamam H, Cheikhrouhou O. Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing. *Processes*, vol.9, no.9, pp.1514, september, 2021.

[15] Dorigo, M., Birattari, M., & Stutzle, T. Ant colony optimization. *IEEE computational intelligence magazine*,vol. 1, no.4, pp 28-39, 2006.