JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2798 - 2809
https://publishoa.com
ISSN: 1309-3452

Multi Objective Resource Scheduling for Cloud Environment using Ant Colony Optimization Algorithm

Mufeed Ahmed Naji Saif [1], Abhinava Karantha K[1], S K Niranjan[1], Belal Abdullah Hezam Murshed[2,3]

[1]Department of Computer Applications, Sri Jayachamarajendra College of Engineering, VTU, Mysore, India.
[3]Department of Studies in Computer Science, University of Mysore, Mysore, India,
[4]Department of Computer science, College of Eng. & IT, University of Amran, Amran, Yemen.
[1]mufeed.a.nsaif@gmail.com, [1]abhinavkarantha98@gmail.com, [1] sriniranjan@yahoo.com,
[2]belal.a.hezam@gmail.com
[1]https://orcid.org/0000-0002-0399-6339, [2]https://orcid.org/0000-0003-2187-5044

Abstract: Cloud computing enables efficient resource sharing among several cloud users to run their applications and fulfill the business requirements. Despite the numerous advantages of cloud computing, the heterogeneity, uncertainty, and dynamic nature of user workload complicate the allocation and scheduling of cloud resources. However, improper resource allocation and scheduling result in resource waste and delays the execution of user tasks, which may volatile the Service Level Agreement (SLA). Hence, efficient resource scheduling techniques are highly desirable to maximize resource utilization and ensure efficient execution of user tasks with maintained SLA. In this Article, a multi-objective resource scheduling method is implemented to address the resource scheduling problem in a container-based cloud environment using Ant Colony Optimization algorithm (ACO). This method aims at maintaining a balance between resource utilization and efficient execution of user tasks with minimal time and cost. To evaluate the implemented method, three scenarios were conducted on ContaienrCouldSim toolkit. The experimental analysis reported that the implemented method archives high resource utilization while minimizing the makespan and execution cost in all the scenarios.

Keywords: Cloud Computing, Resource Allocation, Resource Scheduling, Container Scheduling, Ant Colony Optimization, metaheuristic.

1. Introduction

Cloud computing is an emerging computing technology that allows instant access to a pool of virtualized computing resources through the internet on-demand basis. These resources are allocated and de-allocated dynamically based on cloud users' demands. Virtualization has powered the cloud environment by creating virtual instances of cloud physical resources to utilize the resources and minimize the deployment cost [17]. This has enticed cloud users to deploy their applications, such as (big data and Internet-of-Things, etc.) by offering virtualized computational resources such as containers or virtual machines (VMs). Most existing resource scheduling techniques deploy VM instances for every task, requiring maximum start-up time, resulting in high execution times and costs. However, containers are an alternative lightweight virtualized component that has low start-up time to address the problems of high start-up time in VMs based models [1].

Cloud computing has offered the pay-per-use option to the users for utilizing its resources and services, cloud providers provision various services (infrastructure, software, and platform) to the cloud users with instant access and high scalability. Therefore, it is essential to meet the Quality-of-Service (QoS) constraints in order to maintain Service-Level Agreements (SLAs) and satisfy the users' requirements [5]. However, providing high-quality dedicated cloud services and maintaining the SLA constraints is a challenging task in cloud computing. It is the primary concern in existing cloud service models, managing the cloud resources and satisfying the user requirements [19]. Cloud providers need to dynamically allocate large-scale cloud virtual resources to millions of cloud users, without violating SLA and high profit. whereas cloud users need to get their required cloud services at a low cost [6]. Reaching the cloud user satisfaction while maintaining the cloud provider profit, and tackling both objectives is a challenging task. To address this problem, there is a need for multi-objective resource allocation and scheduling techniques to achieve both the cloud providers' and cloud users' objectives

Resource allocation and scheduling in cloud computing are the primary tasks of cloud resource managers. Efficient resource scheduling leads to high resource utilization and less cost. Resource scheduling is an NP problem. Meta-heuristics optimization algorithms are a good tool to deal with such problems. Currently, many researchers have adopted metaheuristics optimization algorithms to address resource allocation and scheduling in cloud computing such as [2, 11, 12, 20, 22, 23, 24]. These multi-objective scheduling techniques are limited to VM based deployment model. To this extent, still few research have considered container-based deployment model, therefore, this research work mainly focusses on addressing the multi-objective resource scheduling in container-based cloud environment using ACO algorithm.

In this article, we implement ACO for addressing the resource scheduling problem in a container-based cloud environment. It is a probabilistic and uncertain global optimization algorithm, which can easily get the global optimal solution. The primary objective of this method is to maintain a balance between resource utilization and the efficient execution of user tasks with minimal time and cost. The main contribution of this article is implementing ACO for multi-objective resource scheduling in a container-based cloud environment and evaluating this algorithm on ConatinerCloudsim to analyze its performance in terms of makespan, resource utilization, and execution cost.

The rest of this article is organized as the following: Sect. II reviews the related work. Sect. III describes the implemented algorithm. In Sect. IV the experimental results and analysis are presented. Finally, Sect. V. concludes the study.

2. Related work

This section briefly describes some of the existing research works related to multi-objective ACO algorithms in cloud computing. To this extent, Malekloo et al. [11] adapted ACO to address the multi-objective optimization problem for VM placement in cloud datacenter with the objective to minimize the energy consumption, resource wastage, and the energy communication cost. Zhu et al. [23] adapted ACO to address the problem of multi-objective optimization based on load balancing and VM placement with an objective to reduce the total resource wastage and power consumption. Ashraf et al. [2] adapted multi-objective ACO for VM consolidation in cloud data centers with the objective to minimize over-provisioning of PMs by consolidating VMs on under-utilized PMs and minimize the number of VM migrations. Pham et al. [14] adapted ACO to address the problem of multi-objective resource allocation in cloud computing with the objective to minimize the energy consumption and balance the load of physical machines.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2798 - 2809
https://publishoa.com
ISSN: 1309-3452

Reddy et al. [16] adapted a modified ACO for task scheduling to address a multi-objective problem improving the performance of task scheduling by reducing makespan. Ming et al. [13] utilized multi-targeted ACO for multi-tenant SaaS service dynamic selection in a cloud environment with an objective to save energy and deployment cost and archive load balancing target. Jia et al. [8] proposed an adaptive workflow scheduling approach based on ACO considering the deadline of the task with an objective to reduce cost and execution time. Xu et al. [22] utilized ACO algorithm for efficient VM allocation based on an improved PM selection strategy to the basic ACO to avoid premature convergence or falling into the local optima, achieving efficient load balancing and maximizing the resource utilization. Malekloo et al. [12] suggested an energy and QoS aware multi-objective ACO technique for VM placement and consolidation to balance the trade-off between system performance, energy efficiency, and SLA compliance.

Reddy et al. [15] suggested a task scheduling method using a modified ACO algorithm for minimizing the makespan. Lin et al. [10] adapted a multi-objective ACO for scheduling microservice on cloud containers considering the utilization, number of requests, storage resources, and failure rate. Wei et al. [20] proposed an improved ACO with an adaptive parameter setting for balancing its fast convergence and robust search capability, ensuring efficient VM placement with an objective to minimize the communication cost and energy consumption over traffic-aware Data Center Network (DCN).

Bindu et al. [3] developed a scheduling approach using ACO for reducing energy, cost, and time. Wei et al. [21] suggested a task scheduling method using improved ACO for a cloud environment. The proposed method aims at searching for the optimal solution for task scheduling to avoid falling into local optimum and considering three objectives: minimizing the waiting time, the degree of load balancing, and the cost of task completion. Huang et al. [7] presented an approach based on ACO for service replicas placement considering scheduling multi-objective: deployment cost and latency.

From the aforesaid detailed literature review related to ACO based resource allocation and scheduling, it can be observed that most of these techniques are intended for VM-based could model, and less attention has been given to resource scheduling in a container based cloud environment. In this research, we aim at analyzing the efficiency of ACO for secluding the resources in a container-based cloud environment.

3. Methodology

This section describes the implemented algorithm for resource scheduling in container-based cloud computing environment.

3.1 Optimal Resource Allocation

The container is the main component in container-based cloud environments, which has the responsibility of executing user tasks. User task has to be effectively scheduled on appropriate container to ensure an efficient execution of user tasks. The required containers for executing the user tasks can be represented by $C = \{c_1, c_2, \qquad c_m\}$ whereas the users' tasks can be represented by $T = \{t_1, t, \qquad t_n\}$. The primary goal of scheduling strategy is enabling an efficient execution of user tasks while meeting both cloud users and providers objectives. it is critical to simultaneously reduce the make-span and execution cost while increasing the resource utilization rate to satisfy both cloud users and providers.

Resource Utilization: it identifies the number of resources utilized executing a user task. This algorithm focusses on maximizing the resource utilization rate. The mathematical formulation for resource utilization can be given as follows:

$$\text{Max } res\_util = \frac{\sum_{i=1}^{n} finish_t(w_i)}{max\ finish_t(w_{final})} \qquad (1)$$

Make-Span: it refers to the completion time of the final task submitted by the users. It is the overall taken time between the submission and execution of the task. Make-span can be obtained using the following formula:

$$\text{Min } M_{span} = max\, f\, inish_t(w_{final}) \qquad (2)$$

where, $finish_t(w_{final})$ represents the finish time or the completion time of final task.

Execution Cost: It is computed by multiplying the price of the instance of container with the computed response time.

$$\text{Min } exe\_\,cos\,t = \sum_{i=1}^{n} (Ft(t_i) - Pt(t_i) - St(t_i)) \times P_{C_i} \qquad (3)$$

where, P_{C_i} represents the price of the containers' instance subjected to run the i^{th}task.

3.2 Ant Colony Optimization Algorithm (ACO)

ACO is a metaheuristic algorithm that simulates ant behavior when foraging for food, it can be used to address complex optimization problems. It mimics the behavior of real ant colonies, which communicate via pheromone trails. They leave pheromones on the path while moving to find a food source. Perceiving the pheromone helps other ants in following the trails to the food source. Most ants choose the shortest path because it has a higher concentration of pheromones. The ACO is adaptable and robust [4]. It can be used for efficient resource scheduling. Here, we briefly describe the basic concepts of ACO applied to converge the decision of container maximining the resource utilization, minimizing the makespan and execution cost. The steps of ACO are described below:

Parameters Initialization: Initially, the required parameters are set; and pheromone trails are initialized. Then, on path segments, the virtual trail is accumulated. Then, ACO constructs ant solutions by moving the ants with probability of the pheromone's concentration. Then the pheromone is updated

Initialize Pheromone: the ants are distributed randomly, then, the pheromone values are initialized:

$$T_i(0) = pNUM_i \times pMIPS_i + Conb_i \qquad (4)$$

where Con is a container, $pNUM_i$ number of Con_i processor $pMIPS_i$ million instructions per second of every Con_i processor $Conb_i$ Con_i communication bandwidth capability.

Selecting the container for upcoming task: For the upcoming task, Con_i is selected by k-ant with probability, when the containers are overloaded, it becomes bottleneck which directly influences the given task by which the makespan get increased.

Local and global pheromone update: let τi (T_p) at any time T be the Con_i pheromone intensity. The update of pheromone is formulated as in the Eq [9].

$$\tau_i (T+1) = (1-\rho) \times \tau_i(T) + \Delta\tau_i \qquad (5)$$

Where ρϵ½ [0, 1] decay coefficient of pheromone trail. The previous solution impact will be low when value of ρ is high. When the ant finishes its tour, the local pheromone on container visited and $\Delta\tau i$ value can be updated by:

$$\Delta ri = 1/tiK \qquad (6)$$

Where t_{iK} K-ant searched shortest path length at ith iteration

If the ant finds the current optimal solution while completing its tour, a higher intensity pheromone is laid on its path while updating the global pheromone on containers visited, and Δri value can be given as:

$$\Delta ri = d/top \qquad (7)$$

Where to_p is the current optimal solution and d is the encouragement coefficient

Algorithm 1 ACO

```
1   Initialize the pheromone value
2   Optimal =null (initialize the best path to null)
3   Initialize the ants
4   For every ant
5       For all the paths
6           determine the value of pheromones on every path
7       End for
8       Best = Compute the fitness function (ants)
9       If (Best < Optimal)
10          Optimal = best //determine the shortest path
11      End If
12      Update the local pheromone (6)
13      Update the global pheromone (7)
14  Repeat until the condition is satisfied.
15  Return the optimal solution (best path)
16  End for
```

4. Results and discussions

To evaluate the efficiency and performance of the ACO algorithm, simulation analysis is conducted on ContainerCloudsim simulation toolkit. We have conducted three scenarios A, B, and C, in the first scenario A, we have created 10 containers, 10 hosts, and 20 Virtual machines VMs, in the second scenario B we have created 50 containers, 10 hosts, and 20 VMs, and in the third scenario C 50 containers, 10 hosts and VMs have been created. Specifications of the VMs and other setup parameters are shown in table 1.

Table1. Cloudsim setup parameters

Entity Type	Parameters	Scenario A	Scenario B	Scenario C
Container	Number of containers	10	25	50
Cloudlet	The number of cloudlets	25 - 150	25 - 150	200 - 1000
	Length	1000 – 10000		
Host	Number of hosts	10		
	RAM	512MB		
	Storage	10000		
	Bandwidth	1000		
Virtual Machine	Number of VMs	20		
	Policy	ACO		
	RAM	512MB		
	Size	1000		
	VMM	Xen		
	OS	Linux		
	Number of CPUs	2		
Datacenter	Number of datacenters	2		

For the experimental analysis, in the first (A) and second (B) scenarios 25-150 clouldlets with random lengths from 1000 to 10,000. In the third scenario (C) 200-800 clouldlets with random lengths from 1000 to 10,000. Running the three scenarios we have obtained the following results as shown in table 2:

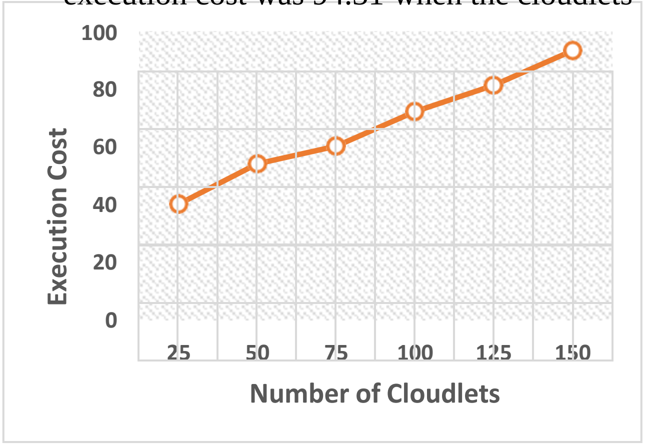
Table 2 Obtained experimental results:

No. of containers	Workload	Execution Cost	Resource Utilization	Makespan	SLA
10	25	40.23	41.21	1749.86	0.002
	50	54.20	57.56	1814.31	0.005
	75	60.32	76.17	2069.96	0.006
	100	72.27	89.19	2256.25	0.012
	125	81.33	98.86	2399.46	0.023
	150	93.38	112.93	2579.16	0.035
25	25	42.73	59.76	1154.11	0.006
	50	51.89	67.42	1213.98	0.011
	75	68.32	91.00	2467.64	0.020
	100	73.11	103.16	1622.53	0.033
	125	81.65	109.18	1791.72	0.045
	150	96.38	125.13	1968.34	0.052
50	200	94.31	130.19	830.01	0.063
	400	102.31	156.44	886.55	0.087
	600	110.38	178.54	973.76	0.098
	800	119.32	189.23	1114.51	0.114
	1000	132.62	197.71	1223.21	0.124

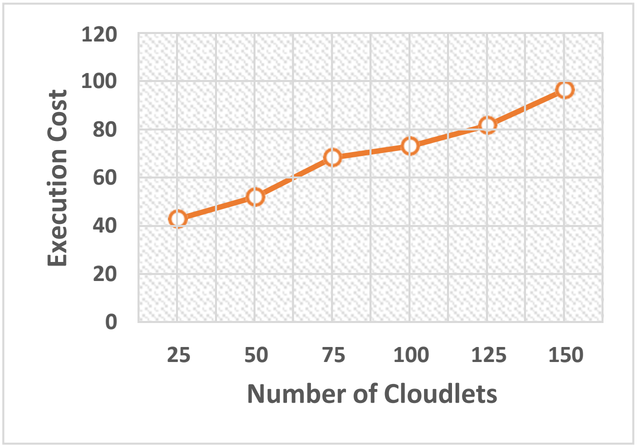
JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2798 - 2809
https://publishoa.com
ISSN: 1309-3452

4.2 Execution Cost

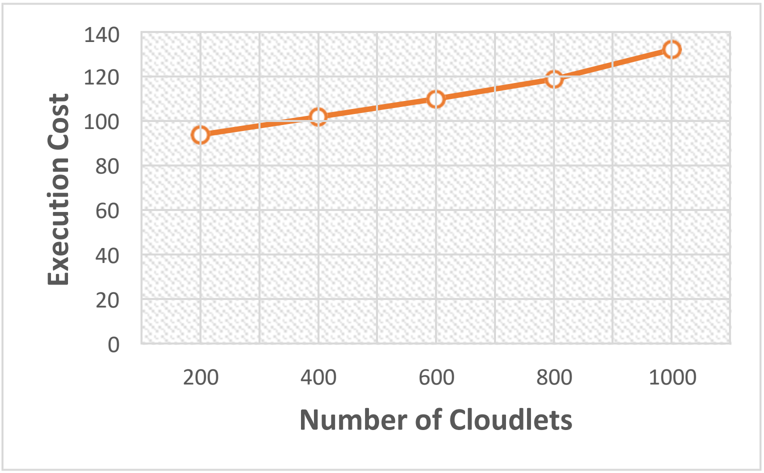
For the evaluation of execution cost, we have executed the three scenarios and obtained the results as shown in table 2. it has been observed that from Fig. 2 ACO algorithm obtained optimal results in terms of execution cost in the first scenario (A) when the number of containers was 10 and the number of cloudlets was 25, the optimal execution cost was 40.23, whereas in the second scenario (B) when the number of containers was 25, the optimal execution cost was 42.73 when the number of cloudlets was 25. But in the third scenario (C) with the number of containers was 50, the optimal execution cost was 94.31 when the cloudlets were 200.



Scenario A (10) Containers



Scenario B (25) Containers

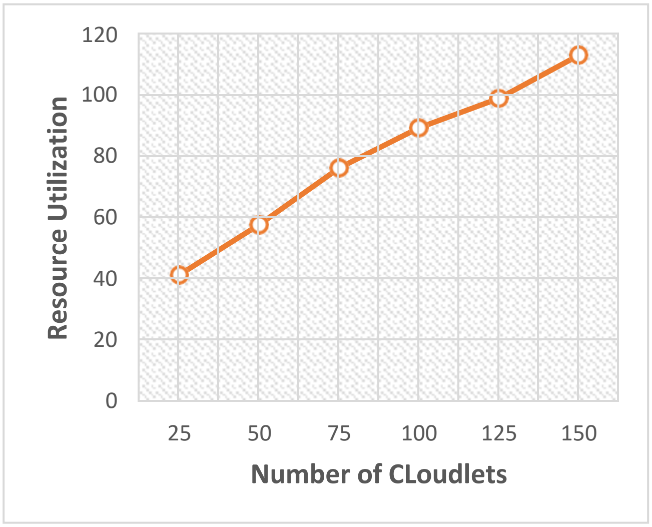


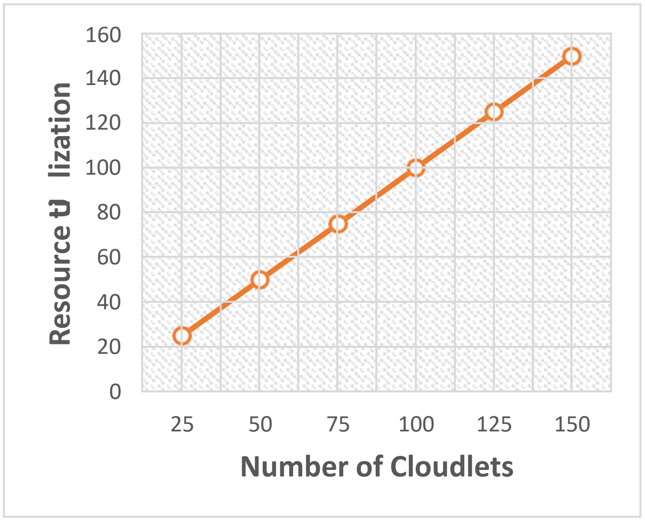
Scenario C (50) Containers

Fig. 2 Execution Cost Vs Number of Cloudlets

4.2 Resource Utilization

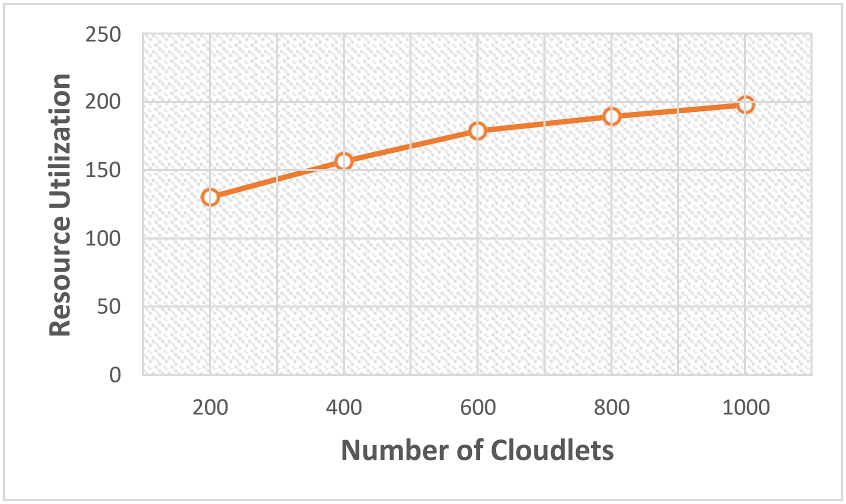
For the evaluation of resource utilization, we have executed the three scenarios and obtained the results as shown in table 2. it has been observed that from Fig. 3 ACO algorithm obtained optimal results in terms of resource utilization in the first scenario (A) when the number of containers was 10 and the number of cloudlets was 25, the optimal utilization was 41.21, whereas in the second scenario (B) when the number of containers was 25, the optimal utilization was 59.76 when the number of cloudlets was 25. But in the third scenario (C) with the number of containers being 50, the optimal utilization was 130.19 when the cloudlets were 200.





Scenario A (10) Containers

Scenario B (25) Containers

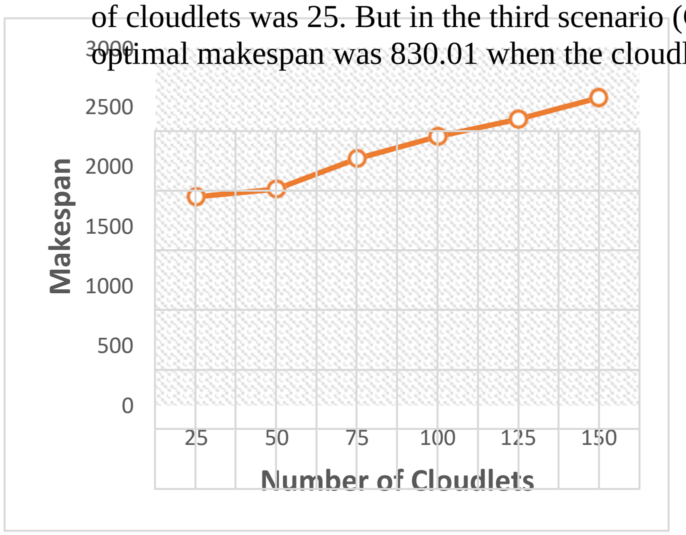


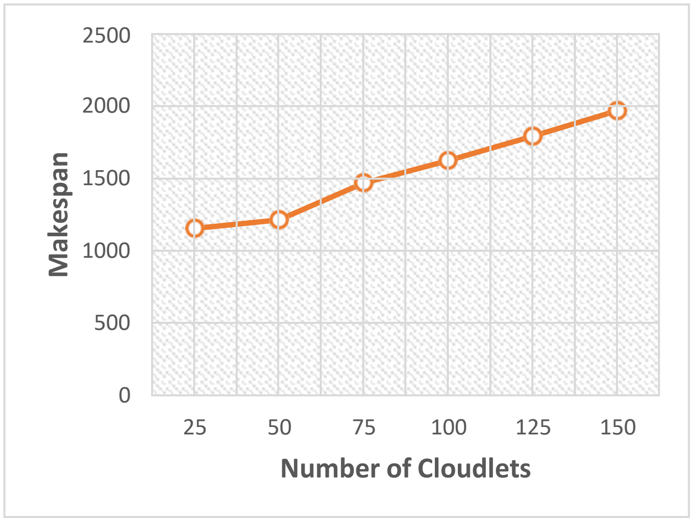
Scenario C (50) Containers

Fig. 3 Resource Utilization Vs Number of Cloudlets

4.3 Makespan

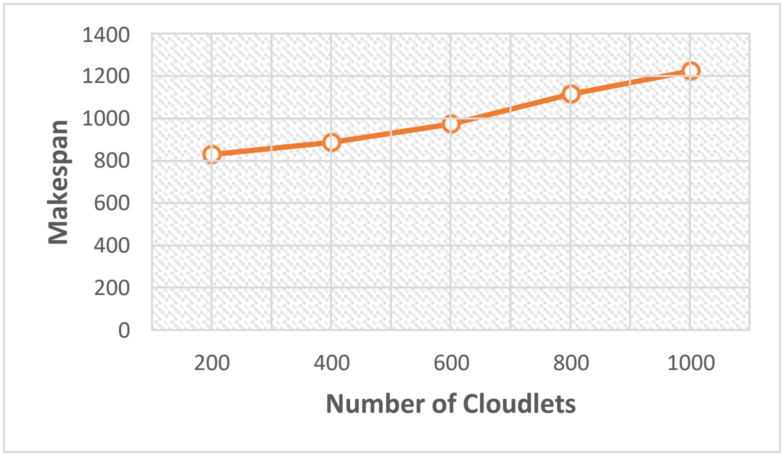
For the evaluation of makespan, we have executed the three scenarios and obtained the results as shown in table 2. it has been observed that from Fig. 4 ACO algorithm obtained optimal results in terms of makespan in the first scenario (A) when the number of containers was 10 and the number of cloudlets was 50, the optimal makespan was 1749.86, whereas in the second scenario (B) when the number of containers was 25, the optimal makespan was 1154.11when the number of cloudlets was 25. But in the third scenario (C) with the number of containers being 50, the optimal makespan was 830.01 when the cloudlets were 200.





Scenario A (10) Containers

Scenario B (25) Containers

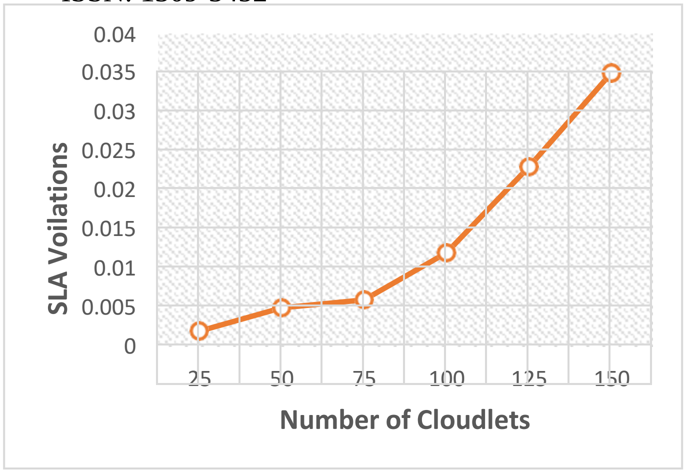


Scenario C (50) Containers

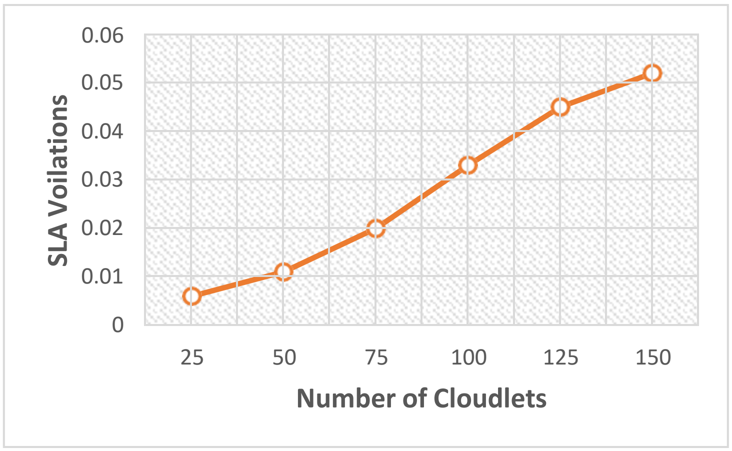
Fig. 4 Makespan Vs Number of Cloudlets

4.2 SLA Violation

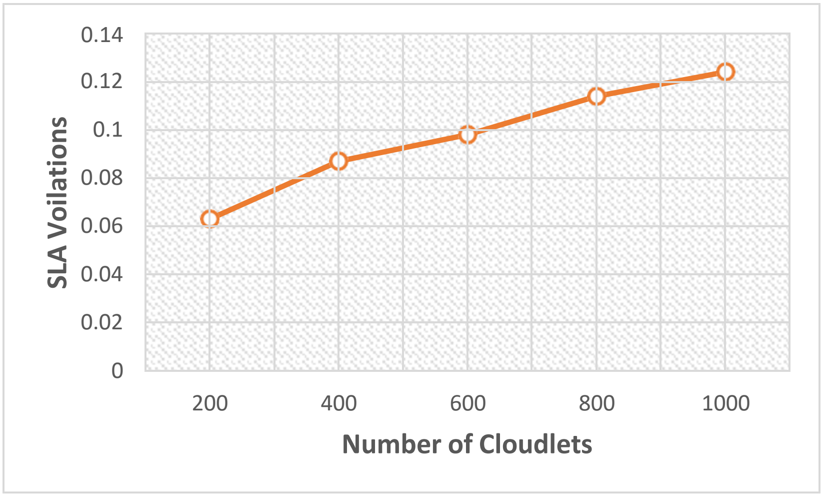
For the evaluation of SLA violation, we have executed the three scenarios and obtained the results as shown in table 2. it has been observed that from Fig. 4 ACO algorithm obtained optimal results in terms of SLA violation in the first scenario (A) when the number of containers was 10 and the number of cloudlets was 25, the optimal SLA violation was 0.002, whereas in the second scenario (B) when the number of containers was 50, the optimal SLA violation was 0.006 when the number of cloudlets was 25. But in the third scenario (C) with the same number of containers, the optimal SLA violation was 0.063 when the cloudlets were 200.



Scenario A (10) Containers



Scenario B (50) Containers



Scenario C (50) Containers

Fig. 5 SLA Violation Vs Number of Cloudlets

The findings significantly showed that employing the ACO allocation algorithm achieves better performance, it optimally obtained 40.23 in terms of execution cost while the 41.21 resource utilization and 1749.86 makespan with a minimum of 0.002 SLA violation. ACO algorithm provides better quality and shows high performance with less execution cost, time, SLA violations, and space complexity. ACO can be suggested as a powerful optimization technique and sufficient for addressing the issues of resource scheduling in a cloud environment.

5. Conclusion

This paper implemented ACO multi-objective container resource scheduling in a cloud environment, the article aimed at analyzing the use of ACO for optimizing multi-objective such as the execution cost, resource utilization, makespan, and SLA violations. The experimental results showed that ACO has performed well. Optimally obtained 830.01 in terms of makespan while it got 75.61 for the resource utilization and 16.42 for the execution cost. In future work, we will compare ACO with other optimization based container resource scheduling and allocation in the cloud from the state-of-art of meta-heuristic algorithms to identify the most efficient and reliable optimization algorithm for optimal resource scheduling in a cloud environment.

References

1. Adhikari, Mainak, and Satish Narayana Srirama. "Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment." *Journal of Network and Computer Applications* 137 (2019): 35-61.
2. Ashraf, Adnan, and Ivan Porres. "Multi-objective dynamic virtual machine consolidation in the cloud using ant colony system." *International Journal of Parallel, Emergent and Distributed Systems* 33, no. 1 (2018): 103-120.
3. Bindu, G. B., K. Ramani, and C. Shoba Bindu. "Optimized resource scheduling using the meta heuristic algorithm in cloud computing." *IAENG International Journal of Computer Science* 47, no. 3 (2020): 360-366.
4. Dorigo, Marco, and Thomas Stützle. "The ant colony optimization metaheuristic: Algorithms, applications, and advances." In *Handbook of metaheuristics*, pp. 250-285. Springer, Boston, MA, 2003.
5. Donyagard Vahed, Nasim, Mostafa Ghobaei-Arani, and Alireza Souri. "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review." *International Journal of Communication Systems* 32, no. 14 (2019): e4068.
6. Guo, Qiang. "Task scheduling based on ant colony optimization in cloud environment." In *AIP conference proceedings*, vol. 1834, no. 1, p. 040039. AIP Publishing LLC, 2017.
7. Huang, Tiansheng, Weiwei Lin, Chennian Xiong, Rui Pan, and Jingxuan Huang. "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing." *IEEE Transactions on Cybernetics* 51, no. 11 (2020): 5595-5608.
8. Jia, Ya-Hui, Wei-Neng Chen, Huaqiang Yuan, Tianlong Gu, Huaxiang Zhang, Ying Gao, and Jun Zhang. "An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, no. 1 (2018): 634-649.
9. Lal, Arvind, and C. Rama Krishna. "Critical path-based ant colony optimization for scientific workflow scheduling in cloud computing under deadline constraint." In *Ambient Communications and Computer Systems*, pp. 447-461. Springer, Singapore, 2018.
10. Lin, Miao, Jianqing Xi, Weihua Bai, and Jiayin Wu. "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud." *IEEE access* 7 (2019): 83088-83100.
11. Malekloo, Mohammadhossein, and Nadjia Kara. "Multi-objective ACO virtual machine placement in cloud computing environments." In *2014 IEEE Globecom Workshops (GC Wkshps)*, pp. 112-116. IEEE, 2014.
12. Malekloo, Mohammad-Hossein, Nadjia Kara, and May El Barachi. "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments." *Sustainable Computing: Informatics and Systems* 17 (2018): 9-24.
13. Ming, Cao, Yu Bingjie, and Liu Xiantong. "Multi-tenant SaaS deployment optimisation algorithm for cloud computing environment." *International Journal of Internet Protocol Technology* 11, no. 3 (2018): 152-158.
14. Pham, Nguyen Minh Nhut, and Van Son Le. "Applying Ant Colony System algorithm in multi-objective resource allocation for virtual services." *Journal of Information and Telecommunication* 1, no. 4 (2017): 319-333.
15. Reddy, G. Narendrababu, and S. Phani Kumar. "MACO-MOTS: modified ant colony optimization for multi objective task scheduling in Cloud environment." *International Journal of Intelligent Systems and Applications* 11, no. 1 (2019): 73.

16. Reddy, G., N. Reddy, and S. Phanikumar. "Multi objective task scheduling using modified ant colony optimization in cloud computing." *Int'l J. Intell. Eng. Sys* 11, no. 3 (2018): 242-250.
17. Saif, Mufeed Ahmed Naji, S. K. Niranjan, and Hasib Daowd Esmail Al-Ariki. "Efficient autonomic and elastic resource management techniques in cloud environment: taxonomy and analysis." *Wireless Networks* 27, no. 4 (2021): 2829-2866.
18. Singh, Harvinder, Anshu Bhasin, and Parag Kaveri. "SECURE: Efficient resource scheduling by swarm in cloud computing." *Journal of Discrete Mathematical Sciences and Cryptography* 22, no. 2 (2019): 127-137.
19. Vinothina, V., and R. Sridaran. "An approach for workflow scheduling in cloud using ACO." In *Big data analytics*, pp. 525-531. Springer, Singapore, 2018.
20. Wei, Wenting, Huaxi Gu, Wanyun Lu, Tong Zhou, and Xuanzhang Liu. "Energy efficient virtual machine placement with an improved ant colony optimization over data center networks." *IEEE Access* 7 (2019): 60617-60625.
21. Wei, Xianyong. "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing." *Journal of Ambient Intelligence and Humanized Computing* (2020): 1-12.
22. Xu, Peng, Guimin He, Zhenhao Li, and Zhongbao Zhang. "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization." *International Journal of Distributed Sensor Networks* 14, no. 12 (2018): 1550147718793799.
23. Zhu, Liwen, Ruichun Tang, Ye Tao, Meiling Ren, and Lulu Xue. "Multi-objective ant colony optimization algorithm based on load balance." In *International Conference on Cloud Computing and Security*, pp. 193-205. Springer, Cham, 2016.
24. Zuo, Liyun, Lei Shu, Shoubin Dong, Chunsheng Zhu, and Takahiro Hara. "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing." *Ieee Access* 3 (2015): 2687-2699.