International Journal of Data Science and Artificial Intelligence (IJDSAI)
Volume 3, Issue 3, May – June (2025)

RESEARCH ARTICLE

COCKROACH SWARM OPTIMIZED SCHEDULE TASKING IN CLOUD ENVIRONMENT

Abdulatif Alabdultif [1,*], Nithya Rekha Sivakumar [2]

[1] Department of Computer Science, College of Computer, Qassim University, Buraydah 52571, Saudi Arabia.
[2] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia.
[*]*Corresponding e-mail: ab.alabdulatif@qu.edu.sa*

Abstract – Cloud computing is one of the major platforms that that maintains high levels of agility, scalability, and resilience while offering users and organizations a wide range of services, including data analytics and computational storage. However, effectively scheduling and managing user-requested workloads across the available cloud resources has become more complex due to the rapid growth in cloud adoption. However, determining the optimal task scheduling solution is thought to be an NP-hard problem, especially when handling large task sizes in a cloud environment. To solve this issue a novel Cockroach swarm Optimization based Task Scheduling (COTS) has been proposed to address this problem and improve the cloud environment's overall responsiveness and efficiency. Finding the best inputs to produce the highest or lowest output at the lowest cost is known as optimization. Cockroach Swarm Optimization can schedule tasks more quickly and find a much better distribution of solutions. The recommended method guarantees that the appropriate task is carried out on the appropriate resource at the appropriate time, improving the cloud environment's overall performance. Measures like CPU time, SLR performance, and waiting time obtained are used to assess the effectiveness of the recommended strategy. The COTS model outperforms the current A2C, EPOSA and GENETIC methods with an accuracy of 97% respectively.

Keywords *– Cloud Computing, Task Scheduling, Cockroach swarm optimization algorithm, virtual machine.*

1. INTRODUCTION

Cloud computing is a pay-per-use model in which users access cloud services without fully understanding the hosting and distribution policies. By providing real time access to resources, such as web facilities, computer servers, storage and web facilities this cuts down on the time required for businesses and reasonable conclusions. [1,2,3]. Customers don't have to contact the facility provider and can access these resources consistently without stress. The goal of cloud infrastructure is to give dynamic applications a user-friendly workspace [4,5].

A cloud can quickly provision a virtual or physical machine; workloads can be easily installed and scaled. Here several virtual machines (VMs) can share on a single host. A data center's bandwidth can also be shared by several virtual machines (VMs) to network virtualization [6,7,8]. Scheduling client demand effectively with a short restructuring period assignments pertaining to client demand is a significant challenge because there are usually a lot of user requests, which ensures optimal system performance [9].

A queuing model is used to formulate the work load problem, which aims to reduce the total waiting time for every task. For efficient handling, task priorities are determined by size, and a waiting time matrix is added to aid in scheduling framework [10]. To improve responsiveness, a parallel method is suggested for scheduling, and the waiting queue is optimized using a Fibonacci heap. However, integrating several different parts, like the Fibonacci heap, parallel scheduling logic, and priority assignment algorithm, adds a lot of computational overhead and makes implementation challenging [11,12]. This can limit the system's scalability or real-time applicability in environments with limited resources or high levels of dynamicity [13]. To overcome this issue a novel based approach has been proposed to effectively schedule user requests and ensure optimal system performance by minimizing turnaround times for tasks associated with user demands. The COT's primary contributions are as follows:

- This study's primary goal is to develop effective approach to enhance efficiency Cloud-based task scheduling and overall system performance environment.
- Initially users submit requests to the data center. which is then examined according to variables like system workload, task complexity, and execution time.
- The right task is carried out on the right resource at the right time with the help of the suggested Cockroach Swarm Optimization Algorithm.

- The effectiveness of the COTS is evaluated using measures such as waiting time, CPU time and SLR performance.

The rest of the paper is organized as Part II goes into great detail about the literature review. Section III provides a description of VANET routing. Section IV displays the findings and observations from the experiment. Section V includes the conclusion and future work.

2. LITERATURE REVIEW

In 2023 Gad et al [14] introduces an OSAPSO (Simulated annealing particle swarm optimizer based on opposition) to resolve early converge problem of PSO. The findings show that OSAPSO outperforms its competitors in cloud system IIOT task scheduling. However, OSAPSO is an algorithmically complex system due to the combination of greedy OES strategies, SA-inspired crossover, and POBL.

In 2024 Ahmed et al [15] suggested datasets for comparing scheduling algorithms. The results of the experiment show that Makespan decreases as the number of virtual machines increases. Even though there are problems with workflow, enhanced HEFT algorithms perform better than the standard HEFT algorithm because they have shorter schedule lengths for running on multiple virtual machines.

In 2024 Devi et al [16] suggested an ideal hybrid metaheuristic algorithm that finds an estimate organizing solution for the BoTS problem by utilizing the advantages of the Artificial Gorilla Troops Optimizer (GTO) and the Honey Badger Algorithm (HBA). According to the results, GTOHBA outperformed the tested metaheuristics by reducing makespan by 8.46–30.97% and energy consumption by 8.51–33.41%.

In 2023 Lipsa et al [17] suggested a parallel task scheduling algorithm where in the heap construction and task priority assignment are carried out concurrently with regard to the non-preemptive and preemptive nature of tasks. The outcomes demonstrate how well our suggested algorithms optimize both the total waiting time and the CPU time used.

In 2023 Hai et al[18] suggested Various HEFT algorithm iterations were changed to produce better results. The outcome demonstrates that the modified HEFT algorithm performs better than the original HEFT algorithm in terms of shorter workflow problem schedule length. The maximum determination of cloud computing scheduling criteria, however, is an optimization problem associated with this.

In 2023 Yadav et al [19] suggested a refined method to meet the minimum makespan goal by narrowing down the vast search area for the optimal scheduling in the shortest amount of time. The optimal schedules produced by this suggested ordinal optimization method and linear regression aid in achieving minimum makespan.

In 2023 Banerjee et al [20] suggested a new approach to cloud computing job scheduling. Although The algorithm's performance in comparison to current scheduling algorithms demonstrated significant improvements which creates reduction and resource utilization, but the comparison was limited to a specific set of algorithms. According to the results, the suggested DynaBit algorithm performs better than the other scheduling algorithms under consideration on a number of evaluation metrics

3. PROPOSED METHODOLOGY

In this section a novel based approach was proposed to arrange tasks according to their preemptive and non-preemptive characteristics. Here, users submit requests to the data center, where they are examined according to variables like system workload, task complexity, and execution time. The best virtual machine (VM) for each task is then found using the Cockroach Swarm Optimization (CSO) algorithm. Lastly, a load balancer ensures optimal performance by allocating the task to the right server based on the CSO scheduling decision. The workflow of proposed methodology was shown in Figure.1

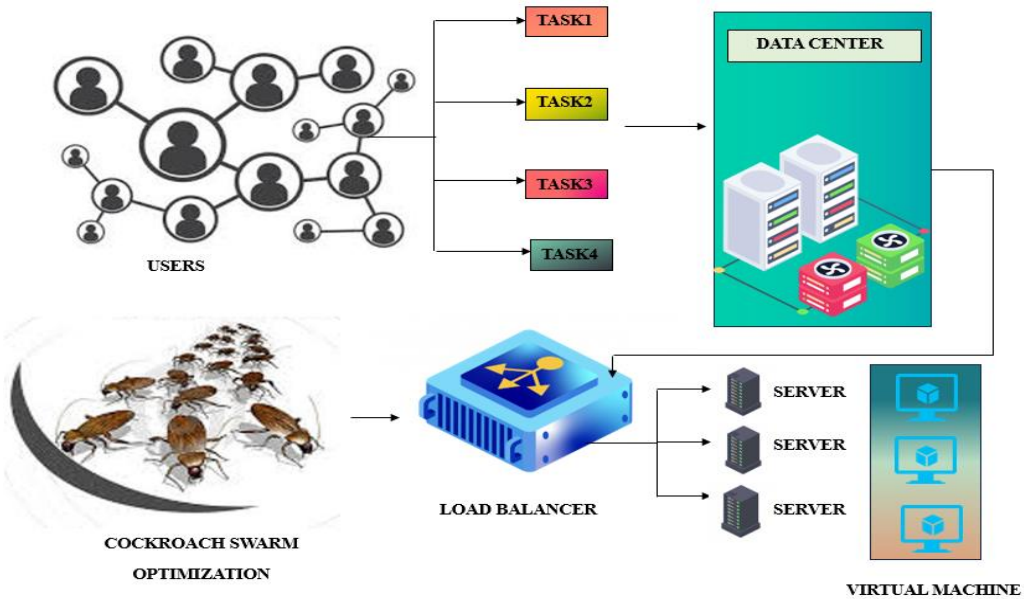

Figure 1. Work flow of COTS methodology

3.1 User Request

The data center receives requests from users. Every request has a specific task that needs to be completed. The system assesses the requirements and nature of the task. This data is utilized for resource allocation and scheduling.

3.2 Data Center

The computing infrastructure required to handle, store, and process enormous volumes of digital data is housed in a centralized location called a data center. Its resources are both physical and virtual. Efficiency and resource consumption can be increased by running several virtual machines (VMs) on a single physical server through the use of virtualization. Scalability is a design feature of data centers.

3.3 Cockroach Optimization Algorithm

The cockroach swarm optimization (CSO) algorithm takes inspiration from how cockroaches follow one another in swarms, scatter in all different directions or flee away in search of food to avoid light. Therefore, the CSO algorithm uses decision rules to model their movement as a group. The decision rules of the CSO algorithm have three generic steps for each cycle or iteration. The first step followed by generating a list of candidate solutions, in most cases, the first solutions that would be created in the search space at random. The CSO algorithm has three procedures as used at each iteration in solving various optimization problems.

The fittest cockroaches possess (Pi) the local best solutions in the new cycle, and they form little swarms and head toward the global optimum (Pg). Each individual (Xi) moves to its local optimum in its visibility range during the chase-swarming process. Because individuals take different pathways than their local optimum, there can be scenarios where a cockroach traveling in a small group becomes the fittest because it finds a better position. A cockroach alone is a local optimum in its own line of sight, and it goes to the world's optimum solution.

The dispersing of individuals is another process. It is occasionally carried out to preserve the variety of cockroaches. This process involves Every cockroach in the search area takes a random step. When the best agent of the moment replaces a random agent then we can also have the situation of ruthless behaviour. The behaviour corresponds to eating weaker cockroaches when food is inadequate.

(1) Chase-Swarming Behaviour

$$X_i = \begin{cases} X_i + step.r\ and.(p_i - x_i) \\ X_i + step.r\ and.(p_l - x_i) \end{cases} \quad (1)$$

where p i is the position for the individual, p l is the best position for the entire world, x_i is the cockroach position, rand is a random number between 0 and 1, and step is a fixed value.

$$s_i = opt_j\{v_j, |x_i - x_j| \leq \text{visual}\} \quad (2)$$

where m = 1,2…, d, l = 1,2…, N, and the perception distance visual is a constant.

$$p_g = opt_i\{v_i\} \quad (3)$$

(2) Dispersion Behaviour

$$w_i = w + rand\,(1, g) \quad l = 1,2,\ldots,d \quad (4)$$

where a D-dimensional random vector that can be set within a specific range is called rand (1, D).

(3) Ruthless Behaviour

$$nm = wz, \quad (5)$$

where w z is the global best position and m is a random integer inside [1, N]. Finally, effective task scheduling is ensured by the Cockroach Swarm Algorithm.

3.4 Load Manager for balancing loads

A load balancer is a network appliance or program that distributes and balances traffic and data flowing into different servers to achieve high performance, high availability, and optimal server utilization. Load balancers disperse workloads to many servers to avoid overburdening any single resource and to make use of resources efficiently. Load balancers will make workloads available and improve availability by reassigning workloads when a server is down. Load balancers improve scalability by easing the addition of new resources when needed, and they optimize performance by the way they distribute traffic.

3.5 Virtual Machine

Virtual machines effectively manage divided workloads. Large tasks are divided into manageable portions by the scheduler. After that, these pieces are split up among several virtual machines (VMs), each of which functions as a separate processing unit. Because virtual machines operate in parallel, execution can be done more quickly and efficiently. Tasks can be moved to another virtual machine in case of a problem, guaranteeing uninterrupted operation. Because of their adaptability, virtual machines are crucial to contemporary, scalable computer systems.

4. RESULTS AND DISCUSSION

This segment centers on the outcomes of the suggested methodology for various performance metrics, compared to other scheduling methods currently being used (A2C, EPSO, and GENETIC). All algorithms were simulated under Python 3.8, while on an HP workstation (Intel Xeon W processor, 3 GHz, 10 cores, and 20MB Intel smart cache, 192 GB DDR4 SDRAM (6×32GB), and Ubuntu operating system). Virtual machines with 1 GB of RAM were run on Oracle VM Virtual Box using Ubuntu as the operating systems.

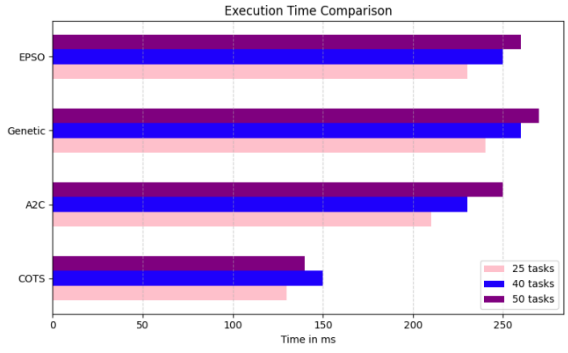

Figure 2. Waiting time comparison

Figure 2 contrasts the results for 25 tasks, 40 tasks, and 50 tasks from A2C, EPO, Genetic, and COTS with the total waiting time calculated using the recommended method.

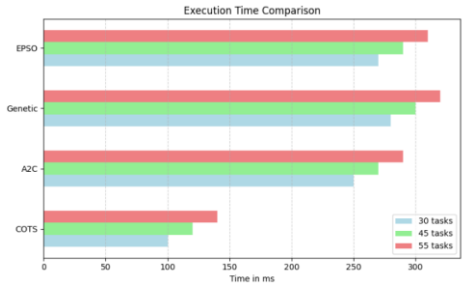

Figure 3. CPU time comparison

s Considering 30 tasks, 45 tasks, and 55 tasks, respectively, Figure 6 shows the total CPU time in milliseconds needed to complete all tasks for A2C, ESO, Genetic, and COTS. The proposed algorithm uses a lot less CPU time than the other methods mentioned because it is parallel**.**

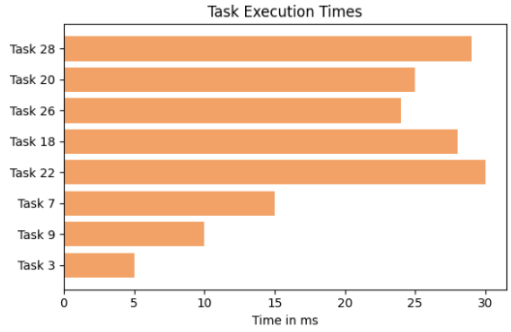

Figure 4. Impact of priority of tasks on waiting time.

Figure 4 illustrates how high-priority tasks affect how long low-priority tasks take to complete. The waiting times for low-priority tasks, like tasks 14, 11, 24, 16, and 5, are at least as long as well as how long high-priority tasks take to process.

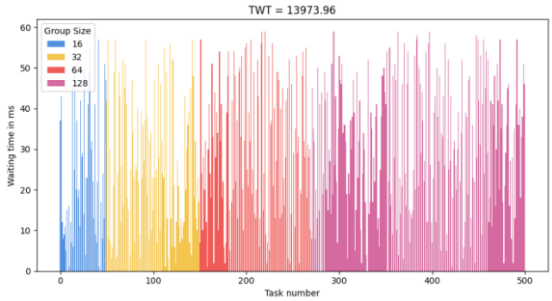

Figure 5. Waiting time of 500 tasks under dynamic cloud computing environment

Figure 5 displays the waiting time for 500 tasks. This figure illustrates that only 16 virtual machines are initially considered for task execution when there are fewer tasks. To reduce the total waiting time, more virtual machines (VMs) are assigned to tasks as the number of functions increases. Furthermore, as shown in Figure 5, the waiting time for the function's ranges from 0 to 60 because of the steady rise in the quantity of assigned virtual machines.

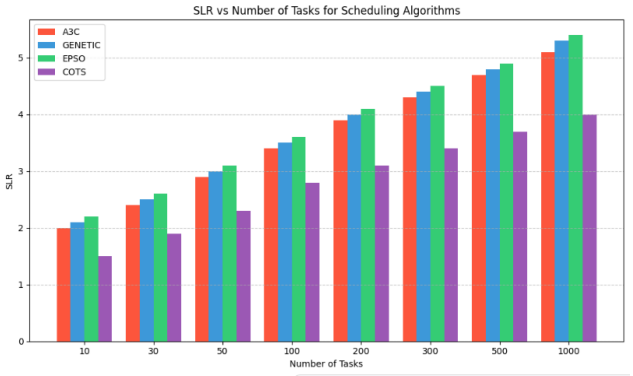

Figure 6. Performance of proposed and existing algorithms

Figure 6 compares the performance of the suggested COTS algorithm with that of the current A2C, EPOSA, and GENETIC algorithms. The findings unequivocally shows the COTS algorithm performs better than the current A2C, EPOSA, and GENETIC approaches on every metric that was assessed. Overall, the COTS approach outperforms earlier methods in terms of accuracy, robustness, and consistency, demonstrating how well it addresses these drawbacks.

5. CONCLUSION

This study proposes a COTS methodology to improve system performance, responsiveness, and efficiency. The suggested approach guarantees that the appropriate task is carried out on the appropriate resource at the appropriate time by utilizing the CSOA technique. Fitness parameters like resource utilization, execution time, system load, and energy consumption are used to determine the best way to allocate tasks. By assigning tasks to appropriate computing units, the suggested scheduling method improves accuracy and dependability. The recommended scheduling method's effectiveness is assessed using metrics like CPU time, overall performance, and waiting time obtained. The suggested algorithms' effectiveness has been evaluated on a range of synthetic data sets and benchmarks. The comparison between the simulated results and the current methods, such as A2C, EPOSA, and Genetic, demonstrates that our suggested algorithms COTS outperform them in terms of maximizing both the total waiting time and the CPU time used. With a 97% accuracy rate, the COTS Scheduling model performs better than the existing A2C, EPOSA, and GENETIC approaches. Future research will employ a hybrid task scheduling approach and compare its results with the most advanced methods in terms of ARUR, Makespan, Throughput, ART, and energy consumption.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest.

FUNDING STATEMENT

Not applicable.

ACKNOWLEDGEMENTS

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

REFERENCES

[1] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions", *Artificial Intelligence Review*, vol. 57, no. 5, pp. 124, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] S.U. Mushtaq, S. Sheikh, and S.M. Idrees, "Next-gen cloud efficiency: fault-tolerant task scheduling with neighboring reservations for improved cloud resource utilization", *IEEE Access*, 2024[CrossRef] [Google Scholar] [Publisher Link]

[3] S.S. Mangalampalli, G.R. Karri, S.N. Mohanty, S. Ali, M.I. Khan, Abdullaev, S. and S.A. AlQahtani, "Multi-objective Prioritized Task Scheduler using improved Asynchronous advantage actor critic (a3c) algorithm in multi cloud environment", *IEEE Access*, vol. 12, pp. 11354-11377, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] S.K. Paul, S.K. Dhal, S.K. Majhi, A. Mahapatra, P.K. Gantayat, and S. Panda, "optimizing task scheduling and resource utilization in cloud environment: A novel approach combining pattern search with artificial rabbit optimization", *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] P. Choppara, and S. Mangalampalli, "Reliability and trust aware task scheduler for cloud-fog computing using advantage actor critic (A2C) algorithm", *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Y. Zhang, and J. Wang. Enhanced Whale Optimization Algorithm for task scheduling in cloud computing environments. Journal of Engineering and Applied Science, vol. 71, no. 1, pp. 121, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] H. Hussain, M. Zakarya, A. Ali, A.A. Khan, M.R.C. Qazani, M. Al-Bahri, and M. Haleem, "Energy Efficient Real-time Tasks Scheduling on High Performance Edge-Computing Systems using Genetic Algorithm", *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[8] F. Yunlong, and L. Jie, "Incentive approaches for cloud computing: challenges and solutions", *Journal of Engineering and Applied Science*, vol. 71, no. 1, pp. 51, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Z. Zhang, C. Xu, S. Xu, L. Huang, and J. Zhang, "Towards optimized scheduling and allocation of heterogeneous resource via graph-enhanced EPSO algorithm", *Journal of Cloud Computing*, vol. 13, no. 1, pp. 108, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Y. Alahmad, and A. Agarwal, "Multiple objectives dynamic VM placement for application service availability in cloud networks", *Journal of Cloud Computing*, vol. 13, no. 1, pp. 46, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[11] M. Aldossary, "Optimizing Task Offloading for Collaborative Unmanned Aerial Vehicles (UAVs) in Fog—Cloud Computing Environments", *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] I.Z. Yakubu, and M. Murali. An efficient IoT-fog-cloud resource allocation framework based on two-stage approach. *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] O.L. Abraham, M.A. Ngadi, J.M. Sharif, and M.K.M. Sidik, "Task Scheduling in Cloud Environment—Techniques, Applications, and Tools: A Systematic Literature Review", *IEEE Access*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] A.G. Gad, E.H. Houssein, M. Zhou, P.N. Suganthan, and Y.M. Wazery, "Damping-assisted evolutionary swarm intelligence for industrial IoT task scheduling in cloud computing", *IEEE Internet of Things Journal*, vol. 11, no. 1, pp. 1698-1710, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] A. Ahmed, M. Adnan, S. Abdullah, I. Ahmad, N. Alturki, and L. Jamel, "An efficient task scheduling for cloud computing platforms using energy management algorithm: A comparative analysis of workflow execution time", *IEEE Access*, vol. 12, pp. 34208-34221, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] A.G. Hussien, A. Chhabra, F.A. Hashim, and A. Pop, "A novel hybrid Artificial Gorilla Troops Optimizer with Honey Badger Algorithm for solving cloud scheduling problem", *Cluster Computing*, vol. 27, no. 9, pp. 13093-13128, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] S. Lipsa, R.K. Dash, N. Ivković, and K. Cengiz, "Task scheduling in cloud computing: A priority-based heuristic approach", *IEEE access*, vol. 11, pp. 27111-27126, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] T. Hai, J. Zhou, D. Jawawi, D. Wang, U. Oduah, C. Biamba, and S.K. Jain, "Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes", *Journal of Cloud Computing*, vol. 12, no. 1, pp. 15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] M. Yadav, and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment", *Journal of Cloud Computing*, vol. 12, no. 1, pp. 8. 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] P. Banerjee, S. Roy, A. Sinha, M.M. Hassan, S. Burje, A. Agrawal, A.K. Bairagi, S. Alshathri, and W. El-Shafai, "MTD-DHJS: makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction", *IEEE Access*, vol.11, pp. 105578-105618, 2023. [CrossRef] [Google Scholar] [Publisher Link]

AUTHORS

Abdulatif Alabdulatif is an associate professor at the School of Computer Science & IT, Qassim University, Saudi Arabia. He completed his Ph.D. degree in Computer Science from RMIT University, Australia in 2018. He received his B.Sc. degree in Computer Science from Qassim University, Saudi Arabia in 2008 and his M.Sc. degree in Computer Science from RMIT University, Australia in 2013. He has published more than 70 academic papers in prominent journals. His research interests include applied cryptography, cloud computing, and E-health

Nithya Rekha Sivakumar is currently an Associate Professor in the Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia from 2017 till date. She received a grant from UGC and completed Ph.D., as a Full-Time Research Scholar in Periyar University awarded with "UGC BSR Research Fellowship in Science for Meritorious Students" by University Grants Commission, New Delhi, Govt. of India from 2010 to 2013. She received a Travel Grant from the Department of Science and Technology (DST), Govt. of India, to go to the United States of America for her Ph.D. research. She received the "BEST DISTINGUISHED RESEARCHER AWARD" for the year 2015-2016 from the College of Computer, Qassim Private Colleges, Buraydah, Kingdom of Saudi Arabia. She has received a Research grant for Research Identity Fast-track Funding Program from the Deputyship for Research & Innovation, Ministry of Education, and the Researchers Supporting Project Award from Princess Nourah bint Abdulrahman University in Saudi Arabia four times.

Arrived: 07.05.2025
Accepted: 11.06.2025