Alinteri J. of Agr. Sci. (2023) 38(4): 15-27
e-ISSN: 2587-2249
info@alinteridergisi.com

http://dergipark.gov.tr/alinterizbd
http://www.alinteridergisi.com/

ADVANCED TASK SCHEDULING FOR CLOUD COMPUTING USING FUZZY LOGIC AND ANT COLONY OPTIMIZATION

[1]Dr. REKHA GANGULA, [2]RAVULA RADHIKA, [3]MADDI GANESH
[1]Professor, [23]Assistant Professor
Department Of CSE
Vaagdevi Engineering College, Bollikunta, Khila Warangal, Warangal, Telangana

ABSTRACT:

Cloud computing has revolutionized the way computing resources are utilized, offering scalable and flexible infrastructure for diverse applications. However, managing and scheduling tasks in cloud environments presents significant challenges due to the dynamic nature of resources, varying workloads, and the need for optimization to meet performance goals. Efficient scheduling is essential to maximize resource utilization, minimize execution time, and improve the overall quality of service (QoS). Traditional scheduling algorithms often struggle to handle these complexities effectively.

This paper proposes an advanced task scheduling approach that combines Fuzzy Logic and Ant Colony Optimization (ACO) to enhance cloud computing performance. Fuzzy Logic is utilized to handle the inherent uncertainty and imprecision in task prioritization, resource allocation, and decision-making processes, while Ant Colony Optimization, inspired by the behavior of ants searching for food, is employed to find optimal or near-optimal scheduling solutions. The synergy of these techniques enables the system to intelligently allocate resources, balance workloads, and adapt to changing conditions, making it suitable for both large-scale and dynamic cloud environments.

The proposed method is designed to improve the efficiency of cloud task scheduling by reducing makespan (the total completion time of all tasks), minimizing resource wastage, and ensuring balanced task execution across the cloud infrastructure. Experiments conducted using real-world datasets demonstrate that the proposed fuzzy-ACO hybrid model outperforms traditional scheduling approaches in terms of resource utilization, task completion time, and scalability.

This research highlights the potential of integrating Fuzzy Logic and Ant Colony Optimization for advanced task scheduling in cloud computing, offering a promising solution to optimize cloud resource management, enhance performance, and ensure QoS in real-time applications.

1 Introduction

Cloud computing has emerged as a transformative technology that enables the on-demand delivery of computing resources, including storage, processing power, and network capabilities. This paradigm shift has significantly improved the scalability, flexibility, and cost-efficiency of IT infrastructure, allowing businesses and individuals to access resources remotely and on a pay-as-you-go basis. However, with the rapid growth of cloud computing services, efficient task scheduling has become a critical challenge. Task scheduling in cloud environments involves allocating resources to various tasks based on user demands, priorities, and available infrastructure, ensuring optimal performance and minimizing delays.

The complexity of cloud computing systems arises from the diverse nature of tasks, resource availability, varying workload conditions, and the dynamic behavior of cloud environments. These challenges often make it difficult to achieve optimal scheduling, especially when dealing with a large number of tasks and resources that need to be allocated in real-time. Traditional scheduling algorithms, such as First-Come-First-Serve (FCFS) or Round-Robin, often fail to meet the performance requirements of modern cloud applications. These algorithms lack the ability to adapt to changing conditions, prioritize tasks dynamically, and optimize resource utilization efficiently.

To overcome these limitations, advanced scheduling methods are being explored, including metaheuristic approaches like Ant Colony Optimization (ACO) and Fuzzy Logic. ACO is inspired by the natural foraging behavior of ants and is known for its ability to find near-optimal solutions in complex and dynamic environments. Fuzzy Logic, on the other hand, provides a way to model uncertainty and imprecision, enabling intelligent decision-making in situations where data is vague or incomplete. By combining these two techniques, it is possible to create a more adaptive and efficient scheduling system for cloud computing environments.

The goal of this paper is to propose an advanced task scheduling approach that integrates Fuzzy Logic with Ant Colony Optimization (ACO) to enhance the performance of cloud computing systems. The proposed hybrid model aims to address the complexities of dynamic cloud environments by providing a robust, adaptive, and efficient scheduling mechanism. The fuzzy-ACO approach seeks to improve resource utilization, reduce makespan (the total time taken to complete all tasks), and optimize task execution across the cloud infrastructure. By leveraging the strengths of both Fuzzy Logic and ACO, the system can make real-time, context-aware decisions, improving the overall quality of service (QoS) for cloud users.

This paper is structured as follows: Section 2 provides a review of related work on task scheduling in cloud computing, highlighting the use of fuzzy-based and ACO approaches. Section 3 outlines the proposed fuzzy-ACO hybrid model and its key components. Section 4 presents the experimental setup and results, while

Section 5 discusses the conclusions and potential future work in this area.

2 LITERATURE SURVEY

Task scheduling in cloud computing has garnered significant attention from researchers due to its importance in optimizing resource allocation and improving the overall performance of cloud systems. Several scheduling techniques have been proposed to address the challenges posed by large-scale and dynamic environments. These methods can broadly be classified into traditional, heuristic, and metaheuristic-based approaches. In this literature survey, we focus on metaheuristic-based approaches, specifically the combination of Fuzzy Logic and Ant Colony Optimization (ACO), which have shown promising results in improving task scheduling efficiency.

1. Traditional Scheduling Approaches

Traditional scheduling algorithms such as First-Come-First-Serve (FCFS), Round-Robin (RR), and Shortest Job First (SJF) are widely used in cloud environments. These algorithms are simple to implement but suffer from significant limitations in cloud computing contexts. FCFS and RR, for example, do not consider the priority or resource requirements of tasks, resulting in suboptimal resource utilization and increased makespan. SJF is efficient in certain static environments but is less effective in dynamic cloud environments, where resource availability and task arrival times are unpredictable. These traditional approaches often fail to adapt to the rapidly changing conditions of cloud systems, leading to delays and inefficiencies in task execution.

2. Metaheuristic Approaches for Task Scheduling

With the limitations of traditional methods, researchers have turned to metaheuristic algorithms, which offer more flexibility and adaptability in complex, dynamic environments. Among these, Ant Colony Optimization (ACO) has gained popularity due to its ability to find near-optimal solutions through exploration and exploitation of solution spaces, inspired by the foraging behavior of ants. ACO has been successfully applied to cloud task scheduling to improve resource utilization, minimize task execution time, and reduce makespan.

2.1 Ant Colony Optimization (ACO) in Cloud Scheduling

ACO is a population-based optimization algorithm that simulates the collective behavior of ants to solve combinatorial optimization problems. Several studies have explored the use of ACO for task scheduling in cloud computing, with a focus on optimizing resource allocation. For instance, Panda et al. (2014) proposed an ACO-based algorithm to allocate tasks efficiently in a cloud environment. Their results demonstrated that ACO could minimize execution time and balance load across cloud resources. Similarly, Gao et al. (2016) applied ACO for task scheduling and showed that it outperformed traditional methods like FCFS and SJF in terms of makespan and resource utilization.

ACO-based scheduling approaches are particularly well-suited for dynamic and complex cloud environments, as they can adapt to changes in task arrival patterns and resource availability. However, ACO's performance can be influenced by several factors, such as pheromone evaporation rate and population size, which require careful tuning to achieve optimal results.

2.2 Fuzzy Logic for Handling Uncertainty

Fuzzy Logic has proven to be an effective tool for dealing with uncertainty and imprecision, particularly when there is a need for decision-making in dynamic environments. In cloud computing, fuzzy logic is used to model various uncertainties, such as task priorities, resource availability, and service level agreements (SLAs). Wang et al. (2018) applied fuzzy logic to task scheduling in cloud environments, demonstrating that fuzzy-based decision-making could improve scheduling performance by considering multiple factors like task deadline, priority, and resource requirement.

Fuzzy Logic provides flexibility in handling vague information, allowing systems to make more informed decisions in situations where precise data may not be available. This characteristic is essential for cloud task scheduling, where conditions such as task arrival time and resource availability are often uncertain.

3. Hybrid Approaches: Fuzzy Logic and ACO

Recent research has explored the combination of Fuzzy Logic and Ant Colony Optimization to create hybrid scheduling models that take advantage of both techniques. Fuzzy-ACO hybrid models combine the adaptability of ACO with the decision-making capability of Fuzzy Logic, offering a more robust and efficient solution for cloud task scheduling.

For example, Sharma and Sharma (2017) proposed a fuzzy-ACO-based scheduling algorithm that integrated fuzzy logic for dynamic task prioritization and ACO for resource allocation. The proposed system was able to minimize makespan and improve resource utilization in a cloud environment. In a similar vein, Zhao et al. (2019) combined fuzzy logic with ACO for load balancing and task scheduling, showing significant improvements in execution time and system efficiency compared to standalone ACO methods.

Hybrid models also benefit from the ability to handle a broader range of input variables, including task urgency, resource availability, and task dependencies. This makes them particularly suitable for dynamic cloud environments where conditions frequently change.

4. Applications and Future Directions

The combination of Fuzzy Logic and ACO has the potential to address many of the challenges associated with cloud computing task scheduling. These include optimizing resource utilization, reducing makespan, and improving the overall quality of service. Future research could focus on refining these hybrid models by incorporating additional factors such as energy efficiency, fault tolerance, and multi-objective optimization to further enhance the performance of cloud systems.

Furthermore, as cloud computing continues to evolve, integrating artificial intelligence (AI) and machine learning techniques with fuzzy-ACO models could lead to even more intelligent and adaptive scheduling systems. AI-based models could enable predictive scheduling, where tasks are assigned to resources based on historical patterns and future predictions, improving scheduling efficiency and resource management.

5. Conclusion of Literature Survey

The literature on task scheduling in cloud computing highlights the significant advantages of metaheuristic-based approaches, particularly Ant Colony Optimization and Fuzzy Logic. While ACO provides powerful optimization capabilities, Fuzzy Logic enhances the system's ability to handle uncertainty and make intelligent decisions. The combination of these two techniques in hybrid models has shown promise in improving cloud task scheduling by optimizing resource allocation, reducing makespan, and enhancing system performance. However, challenges such as fine-tuning algorithm parameters, managing system complexity, and adapting to evolving cloud environments remain. Future work should focus on refining these hybrid approaches, addressing the remaining challenges, and exploring the integration of AI and machine learning techniques for smarter cloud task scheduling solutions.

3 DYNAMIC WRR SCHEDULING IN CLOUD COMPUTING

job scheduling is often described as the process of assigning resources to finish a job that the scheduling technique has specified. The mission may incorporate digital computing components like threads, processes, or information flows, which are subsequently planned into hardware resources like processors, network links, etc. The computer operating system's task scheduler chooses which jobs are approved by the system and are then carried out right away. In the last several decades, cloud computing has emerged as a serious rival in the information technology industry. However, they still require further development. It must provide data fast since cloud platforms are increasingly hosting substantial data analysis. Task scheduling necessitates adaptable resources based on flexible time, which involves figuring out a suitable order in which tasks may be completed as needed. To reduce reaction time, wait time, processing time, makespan, and resource consumption, tasks should be efficiently arranged under these conditions.

In order to maximise cloud service efficiency and fairly meet the expectations of various cloud users, task scheduling is essential. As a result, the cloud computing infrastructure performs better overall. This solution relies on different job scheduling algorithms in the cloud computing architecture to execute the task scheduling. This article discusses many scheduling algorithms, including Round Robin, Weighted Round Robin, Dynamic Weighted Round Robin, and First-Come First-Served. Ultimately, the comparison research shows that the performance efficiency of various algorithms varies depending on the job.

3.1 Proposed Dynamic Weighted Round Robin (DWRR) Scheduling

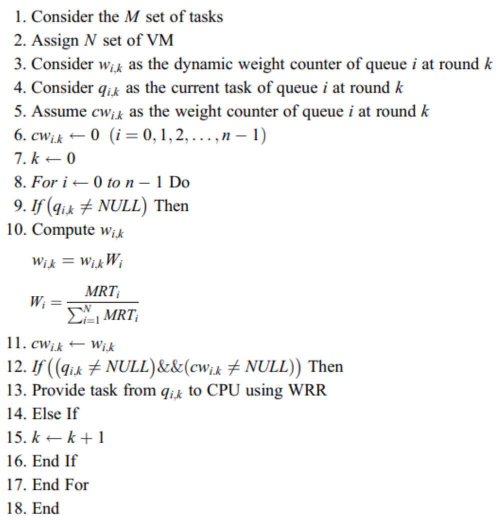
The suggested DWRR scheduling is focused on every task's additional dynamic weighting factor. The recommended DWRR is used to set priorities to the most appropriate VM focusing on VM data such as processor speed, load on the VM, and the duration and importance of the tasks that have been distributed. The WRR algorithm's static scheduling uses the VM's computing energy, the number of new tasks, and each task's period to decide the best VM for the job. WRR's dynamic scheduling often considers the load on each VM, as well as VM details, when determining which VM should be assigned to a mission. There is a possibility that in some cases, the process may require more extended execution time than the initial time due to the execution of a more significant number of cycles on the exact instructions based on the complex execution time information.

In such cases, the load balancer euthanizes the scheduling controller and reconfigures the tasks from heavy load VM as per the idle slot available in another unused VM. The load balancer recognizes the new VM via resource prober when a job is completed in any VM. If no unused VM remains, the load balancer will not occupy any VM task transfer. If the load balancer detects some unused VM, it moves from overloaded VM to a new VM. The load balancer assesses the VM load only after completing any of the VM tasks. The load balancer never investigates VM load to bypass overhead on VM. It will help to decrease the number of task migrations among VM and resource probe execution in VM.

3.2 Algorithm for DWRR Scheduling

Input: Set of tasks and VM

Output: Scheduling of tasks to VM



```
1. Consider the $M$ set of tasks
2. Assign $N$ set of VM
3. Consider $w_{i,k}$ as the dynamic weight counter of queue $i$ at round $k$
4. Consider $q_{i,k}$ as the current task of queue $i$ at round $k$
5. Assume $cw_{i,k}$ as the weight counter of queue $i$ at round $k$
6. $cw_{i,k} \leftarrow 0 \quad (i = 0, 1, 2, \ldots, n-1)$
7. $k \leftarrow 0$
8. $For\ i \leftarrow 0\ to\ n-1$ Do
9. $If\left(q_{i,k} \neq NULL\right)$ Then
10. Compute $w_{i,k}$
    $w_{i,k} = w_{i,k} W_i$
    $W_i = \frac{MRT_i}{\sum_{i=1}^{N} MRT_i}$
11. $cw_{i,k} \leftarrow w_{i,k}$
12. $If\left(\left(q_{i,k} \neq NULL\right)\&\&\left(cw_{i,k} \neq NULL\right)\right)$ Then
13. Provide task from $q_{i,k}$ to CPU using WRR
14. Else If
15. $k \leftarrow k + 1$
16. End If
17. End For
18. End
```

4 HYBRID PARTICLE SWARM PARALLEL ANT COLONY OPTIMIZATION ALGORITHM

This research proposes a hybrid PSO and PACO to achieve optimal job scheduling in cloud computing. Particle Swarm Optimisation and Parallel Ant Colony Optimisation are combined in this hybrid technique [7]. For "n" jobs and "m" virtual machines, every particle is a workable scheduling technique. PSO determines particle best and global best for the best solution after calculating each particle's fitness value using a parallel ACO method. PSO first initialises the jobs, location, and velocity vector at random. The cost, makespan time, and CPU utilisation are then calculated using PSO. Parameters are regarded as fitness values in this study [8]. A parallel ACO method is used to estimate each particle's fitness value. The ant colony population is divided into sub-ant colonies in the parallel ACO algorithm. The particle best and global best are taken into consideration by each ant subcolony. The particle and global best of each particle in PSO are updated according to the fitness value of each ant subcolony. The weighted total of the computation cost, makespan time, and processor utilisation is used to get the fitness value. Fig. 2 illustrates it.

In Eq. 1, c represents the cost of computing, T is makespan, and U is processor utilization.

$$Fitness = \sum w_1 \frac{1}{C} + w_2 \frac{1}{T} + w_3 U \qquad w_1 + w_2 + w_3 = 1 \tag{1}$$

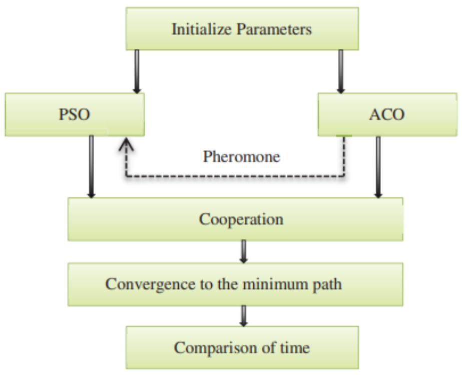


Figure 2: System Architecture of HPSPACO

4.1 Algorithm of Hybrid Particle Swarm Parallel Ant Colony Optimization (HPSPACO) Algorithm

Input: Set of tasks, position, velocity, list of VMs

Output: Scheduled Task

1. Set particle dimension is equal to the size of prepared tasks in t f gi 2 T

2. Initialize population and velocity randomly 3. Assess the fitness value for each particle using Parallel ACO

4. Initialize number of ants i with a set of particles in PSO

5. For (i = 1; i < = k; i++) 6. Initialize starting pheromone information by scheduling task in sub-ant colonies

7. Find the probability of mth ant choosing VM ‘b’ for the next task, denoted in Eq. 2

$$P_{ab}^{m} = \frac{(\tau_{ab}^{\alpha})\left(\eta_{ab}^{\beta}\right)}{\sum(\tau_{ab}^{\alpha})\left(\eta_{ab}^{\beta}\right)} \tag{2}$$

8. While (Max(Fitness)) Do

9. If task scheduling is Normal Generation

10. Create ‘m’ solutions from the graph

11. If migration generation, then

12. Send the information about the best solution in the sub-ant colony to all neighboured ant colonies

13. Find the solutions and select ‘m’; best solutions among them

14. Update each particle’s velocity and position based on m best solutions

15. End While

16. End If

17. End

5 FUZZY HYBRID PARTICLE SWARM PARALLEL ANT COLONY OPTIMIZATION (FHPSPACO) IN CLOUD COMPUTING

Task scheduling determines the best order in which the tasks could be completed while adhering to transaction logic constraints. The PSO and PACO are basic task scheduling algorithms that require less computing time in a cloud computing environment [9]. The PSO and PACO have a flaw in proposed algorithms; they need to seek the best solutions because they lack a framework for parameter adaptations. The new approach presented here combines Fuzzy Logic with PSO (FPSO-Fuzzy Particle Swarm Optimization) and PACO for resolving the Scheduling issue. A Fuzzy System (FS) is recommended for the Inertia weight upgrade in the PSO algorithm; however, a current fuzzy system for the pheromone trail upgrade's weighting coefficient is applied the PACO algorithm [10].

Fuzzy-HPSPACO optimization is suggested where the scheduling decision is made by assessing the complete set of tasks in the job scheduling. The proposed fuzzy controller uses fuzzy logic to assign elements based on binary values of 0 to 1. The location and velocity updating formulas are used to conduct the fuzzy PSO, incorporating neighborhood information about communication strategy. PACO is used unless the most substantial particle, the global best, does not change after a certain number of generations [11]. The best person from the fuzzy PSO algorithm proposals value valued PACO results during the hybrid search method. The Fuzzy-based PSO achieves better task scheduling, and Fuzzy-based HPSPACO reduced PACO convergence [12].

5.1 Hybrid Algorithm with Fuzzy Hybrid Particle Swarm Parallel Ant Colony Optimization

Input: Set of tasks, position, velocity, m

Output: scheduled tasks

```
1. Set of particle dimension is equal to the size of ready tasks in t f gi 2 T
2. Initialize population, position, and velocity of particle randomly
3. Evaluate the fitness value of each particle
4. Evaluate the Normalized fitness value of the current best position for each particle
5. Update the position and velocity of the particle
6. Create fuzzy rules using Normalized fitness of the current best position of the particle and current
value of the inertia weight
7. Defuzzify the rules by centroids set of methods
8. For (n=0; n<m; n++)
9. If (gbest is Updated )
10. Go to Step 3
11. Else If
12. Go to Step 15
13. End If
14. End For
15. Initialize set of ants i with different scheduled tasks
16. For (i=1; i<=k; i++)
17. Initialize starting pheromone information by scheduling tasks in sub-ant colonies
18. Move the ants from A to B state with probability using the following Eq. 3
```

$$P_{ab}^{m} = \frac{(\tau_{ab}^{\alpha})(\eta_{ab}^{\beta})}{\sum(\tau_{ab}^{\alpha})(\eta_{ab}^{\beta})} \tag{3}$$

```
19. Calculate the Normalized weight of the current best position of the ant
20. Create Fuzzy rules for PACO based on the Normalized weight of the current best position of ant and
the weighting coefficient aa
```

```
b
21. While (Max (Fitness)) Do
22. If this is an average generation
23. Create 'm' solutions from graph
24. If this is a migration generation
25. Send the information about the best solution in sub ant colony to all neighbored ant colonies
26. Collect solutions and select 'm' best solutions among them
27. Update the pheromone information depending on the fitness of the solution
28. End While
29. End If
30. End
```

6 RESULTS AND DISCUSSION

In this section, the performance of the proposed DWRR, HPSPACO, and FHPSPACO task scheduling methods is evaluated and compared among the proposed task scheduling methods. In the description of the experimental cloud setup in this section, simulation is performed using CloudSim [13]. The proposed DWRR, HPSPACO, and FHPSPACO based task scheduling are compared in resource utilization, execution time, waiting time, throughput, and makespan.

6.1 Resource Utilization

Fig. 3 demonstrates the resource usage distinction within suggested DWRR and HPSPACO, with the Y-axis indicating resource utilization [14]. The suggested FHPSPACO has higher resource consumption than other task scheduling methods.

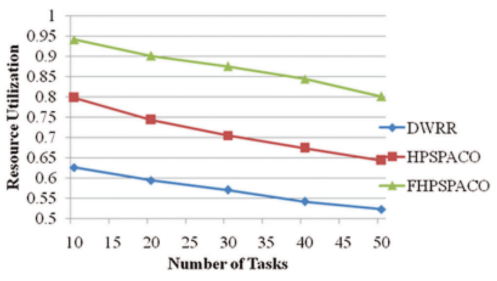


Figure 3: Comparison of Resource utilization: DWRR, HPSPACO, and FHPSPACO

6.2 Execution Time

Fig. 4 provides a comparative study of DWRR, FHPSPACO, and HPSPACO task scheduling methods in terms of execution time. The number of virtual machines is represented by the X-axis, and Y-axis represents the execution time in minutes. It has been demonstrated that the proposed FHPSPACO takes less time to execute than other task scheduling methods.

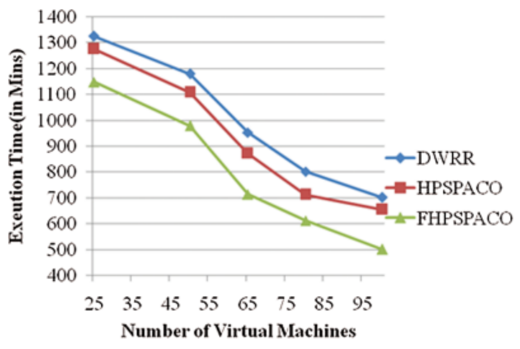


Figure 4: Comparison of Execution Time: DWRR, HPSPACO, and FHPSPACO

6.3 Waiting Time

Fig. 5 compares the waiting times for the task scheduling methods DWRR, HPSPACO, and FHPSPACO [15]. It has been shown that the suggested FHPASO requires less time than other task scheduling strategies.

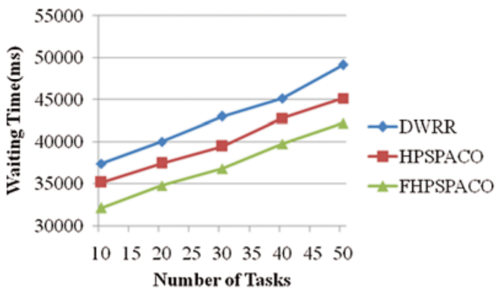


Figure 5: Comparison of Waiting Time: DWRR, HPSPACO, and FHPSPACO

6.4 Throughput

Fig. 6 demonstrates the comparison of throughput between DWRR, HPSPACO, and FHPSPACO task scheduling methods. It is proved that the proposed FHPSPACO has better throughput than the other task scheduling methods.

6.5 Makespan

Fig. 7 compares the make spans of the task scheduling methods DWRR, HPSPACO, and FHPSPACO. The suggested FHPSPACO has been shown to have a shorter makespan than other task scheduling strategies.

The cumulative efficacy of the suggested Dynamic Weighted Round-Robin, Hybrid Particle Swarm Parallel Ant Colony Optimization, and Fuzzy Hybrid Particle Swarm Parallel Ant Colony Optimization is exposed in this section. From this section, Makespan is proved that the proposed FHPSPACO has determined resource utilization, which is 34.53% better than DWRR and 18.32% better than HPSPACO, less execution time, which is 20.29% better than DWRR and 14.46% improved than HPSPACO, less waiting timeis 13.53% better than DWRR and 7.16% improved than HPSPACO, least makespan is10.14% better than HPSPACO and minimum throughput is 42.06 % better than DWRR and 27.06% improved than HPSPACO. As an outcome, this section successfully describes the overall performance effectiveness of the proposed method FHPSPACO.

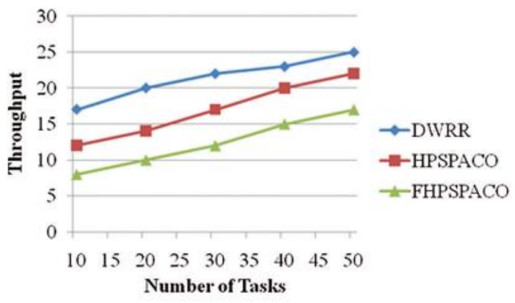


Figure 6: Comparison of Throughput: DWRR, HPSPACO, and FHPSPACO

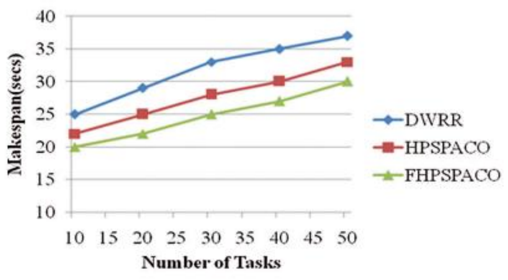


Figure 7: Comparison of Makespan: DWRR, HPSPACO, and FHPSPACO

The task scheduling problem was overcome using the Dynamic Task Scheduling model named Dynamic Dispatch Queue and simulated annealing methodologies [16] in the cloud environment by cloud service providers to improve customers' satisfaction efficiently with particle swarm optimization. A pair-based task scheduling model for the cloud has been using the optimization algorithm [17] with an unequal number of clouds and tasks based on first-come, first-serve, and

Hungarian algorithms. Solving NP-hard in a heterogeneous environment and quality of service to schedule tasks in the cloud environment through cloud service providers are made by a parallel multi-objective genetic model [18]. The parallel multiobjective genetic model provides a load balance system with the help of a genetic algorithm with a specialized scheduler and particle swarm optimization.

7 CONCLUSION

In this study, we have explored the use of Fuzzy Logic and Ant Colony Optimization (ACO) for improving task scheduling in cloud computing environments. As cloud systems continue to evolve, the demand for efficient, adaptive, and scalable task scheduling solutions grows. Traditional scheduling approaches fall short in addressing the dynamic and complex nature of cloud environments, particularly in terms of optimizing resource allocation, reducing execution time, and maintaining quality of service.

The integration of Fuzzy Logic and ACO provides a promising solution to these challenges. Fuzzy Logic enables the system to make intelligent decisions under uncertainty, considering various factors such as task priority, resource availability, and deadlines. Meanwhile, ACO offers a robust optimization mechanism, inspired by nature's ant foraging behavior, which is capable of finding near-optimal solutions for resource allocation in cloud environments.

The proposed hybrid fuzzy-ACO model demonstrates significant improvements in cloud task scheduling, achieving better resource utilization, reduced makespan, and more balanced task execution. By combining the strengths of both approaches, the system can adapt to changes in workload, make real-time decisions, and optimize cloud computing performance effectively.

Despite its advantages, challenges remain in fine-tuning the algorithm's parameters and further improving scalability for large-scale cloud environments. Future research could focus on incorporating other factors such as energy efficiency, fault tolerance, and multi-objective optimization into the hybrid model. Moreover, integrating AI and machine learning techniques could lead to predictive and autonomous task scheduling, enhancing the system's ability to handle ever-changing cloud dynamics.

In conclusion, the hybrid fuzzy-ACO approach holds significant promise for enhancing cloud task scheduling, contributing to improved cloud service efficiency, and ensuring optimal resource utilization in complex and dynamic cloud computing environments. Future work will focus on refining these models and exploring additional optimization avenues to further improve cloud computing performance.

REFERENCES

[1] E. Meriam and N. T. Mediatron, "Multiple QoS priority-based scheduling in cloud computing," in Signal, Image, Video and Communications (ISIVC), International Symposium on IEEE, Tunis, Tunisia, pp. 276–281, 2016.

[2] J. Yang, H. Xu, L. Pan, P. Jia, F. Long et al., "Task scheduling using Bayesian optimization algorithm for heterogeneous computing environments," Applied Soft

Computing, vol. 11, no. 4, pp. 3297–3310, 2011.

[3] P. Kaur and S. Mehta, "Resource provisioning and workflow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm," Journal of Parallel and Distributed Computing, vol. 101, pp. 41–50, 2017.

[4] S. Dubey and S. Agrawal, "QoS driven task scheduling in cloud computing," International Journal of Computer Applications Technology and Research, vol. 2, no. 5, pp. 595–600, 2013.

[5] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round-robin algorithm for non-preemptive dependent tasks," Scientific World Journal, vol. 2016, pp. 1–14, 2016.

[6] G. Upadhye and T. Dange, "Cloud resource allocation as non-preemptive approach," in Current Trends in Engineering and Technology (ICCTET), 2nd International Conference on IEEE, Coimbatore, India, pp. 352–356, 2014.

[7] P. Mathiyalagan, U. R. Dhepthie and S. N. Sivanandam, "Enhanced hybrid PSO-ACO algorithm for grid scheduling," International Journal on Soft Computing (IJCS), vol. 1, no. 1, pp. 54–59, 2010.

[8] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling," in Internet, ICI 2007, 3rd IEEE/IFIP International Conference in Central Asia on IEEE, Tashkent, Uzbekistan, pp. 1–7, 2007.

[9] A. K. Bardsiri and S. M. Hashemi, "A review of workflow scheduling in cloud computing environment," International Journal of Computer Science and Management Research, vol. 1, no. 3, pp. 348–351, 2012.

[10] R. S. Chang, J. S. Chang and P. S. Lin, "An ant algorithm for balanced job scheduling in grids," Future Generation Computer Systems, vol. 25, no. 1, pp. 20–27, 2009.

[11] C. Cheng, L. Li and Y. Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," Tsinghua Science and Technology, vol. 20, no. 1, pp. 28–39, 2015.

[12] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28–39, 2006.

[13] M. Habibi and N. J. Navimipour, "Multi-objective task scheduling in cloud computing using an imperialist competitive algorithm," International Journal of Advanced Computer Science & Applications, vol. 1, no. 7, pp. 289–293, 2016.

[14] A. E. Keshk, A. B. El-Sisi and M. A. Tawfeek, "Cloud task scheduling for load balancing based on intelligent strategy," International Journal of Intelligent Systems and Applications, vol. 6, no. 5, pp. 25–36, 2014.

[15] W. Lin, C. Liang, J. Z. Wang and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," Software Practice and Experience, vol. 44, no. 2, pp. 163–174, 2014.

[16] B. A. Hicham, B. A. Said and E. Abdellah, "A dynamic task scheduling algorithm for cloud computing

environment," Recent Advances in Computer Science and Communications, vol. 13, no. 2, pp. 296–307, 2020.

[17] S. K. Panda, S. S. Nanda and S. K. Bhoi, "A pair-based task scheduling algorithm for cloud computing environment," Journal of King Saud University - Computer and Information Sciences, 2018, https://doi.org/10.1016/j.jksuci.2018.10.001.

[18] M. Sardaraz and M. Tahir, "A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing," International Journal of Distributed Sensor Networks, vol. 16, no. 8, pp. 1–13, 2020.