Journal of Theoretical and Applied Information Technology
15st September 2026. Vol.104. No.17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

QUANTUM PSO BASED TASK SCHEDULING ALGORITHM FOR MEDICAL APPLICATION IN CLOUD COMPUTING ENVIRONMENT

SUSHREE BHARATI[1], PRASANT KUMAR PATTNAIK[2], DIPTI DASH[3]

[1]School of Computer Engg., KIIT, Bhubaneswar, India
E-mail: [1]sushree.bjun94@gmail.com, [2]patnaikprasantfcs@kiit.ac.in, [3]dipti.dashfcs@kiit.ac.in

ABSTRACT

Cloud computing has become a transformative technology in modern computing, allowing individuals and organizations to access software, hardware, and computing platforms remotely through the Internet. In the healthcare sector, it plays a crucial role in managing and processing large volumes of medical data and supporting complex healthcare applications. To ensure efficient operation in cloud environments, effective task scheduling algorithms are essential. These algorithms help reduce execution time and operational costs, improve resource utilization, maintain data privacy, and deliver a high Quality of Service (QoS). The findings of this study indicate that the Quantum Particle Swarm Optimization (QPSO)-based scheduling technique outperforms traditional scheduling methods. By combining intelligent and hybrid optimization strategies, QPSO achieves more efficient resource allocation and task execution. This approach is particularly beneficial in dynamic healthcare environments, where timely processing of medical data is critical. The results show that QPSO significantly improves response time, reduces data center processing time, and lowers overall operational costs, making it a promising solution for cloud-based healthcare systems.
Keywords: *Cloud computing, Comparative analysis, CloudSim, Load balancing, Makespan, Task Scheduling*

1. INTRODUCTION

Cloud computing has transformed the way computing resources are accessed and utilized in modern distributed and grid computing environments. Instead of requiring organizations to invest heavily in physical infrastructure, cloud computing provides on-demand access to resources such as hardware, software, and platforms through the Internet, following a flexible pay-as-you-go model. At the core of cloud computing is virtualization technology, which allows resources to be shared efficiently among multiple users while ensuring scalability and flexibility. This approach enables organizations to access powerful computing capabilities without the complexity of managing extensive infrastructure. As a result, cloud computing offers numerous advantages, including improved performance, reduced operational costs, and greater accessibility. By dynamically distributing workloads across virtualized resources, it helps reduce waiting and execution times, enhances system throughput, and ensures optimal utilization of available resources.

While cloud computing offers many benefits, its implementation in healthcare environments introduces several important challenges. Healthcare systems handle a diverse range of applications, each with dif-ferent processing and storage requirements. Some tasks, such as continuous ECG monitoring, require quick response times and relatively low computa-tional resources. In contrast, advanced applications like medical image processing, patient data analysis, and genomic research demand significant computing power, memory, and storage capacity. In addition, healthcare data exists in many forms and sizes, rang-ing from simple electronic health records to large medical imaging files and continuous streams of data generated by wearable devices. Managing these var-ied workloads efficiently is essential for maintaining system performance and ensuring timely healthcare services.

When tasks are not assigned to appropriate computing resources—for example, when highly demanding applications are allocated to low-capacity vir-tual machines (VMs)—the system may experience slower processing, increased delays, inefficient re-

source utilization, and reduced overall performance.

scheduling are critical for achieving reliable and efficient cloud-based healthcare services. Poor resource allocation in cloud environments can lead to several performance issues. When computationally intensive healthcare applications are assigned to virtual machines (VMs) with insufficient processing capabilities, execution delays may occur, affecting service quality and, in extreme cases, causing system instability. On the other hand, allocating simple tasks to high-performance resources results in inefficient utilization of available infrastructure and unnecessary operational expenses. For this reason, effective task scheduling plays a critical role in cloud-based healthcare systems.

Scheduling mechanisms must ensure that resources are allocated efficiently while meeting important Quality of Service (QoS) requirements such as low response time, high reliability, and cost efficiency. However, conventional scheduling approaches, including First-Come First-Served (FCFS), Round Robin (RR), and other basic heuristic methods, often struggle to manage the dynamic, heterogeneous, and time-sensitive workloads commonly found in healthcare applications. To address these challenges, researchers have explored metaheuristic optimization techniques that can intelligently adapt to changing system conditions. Among these methods, Quantum Particle Swarm Optimization (QPSO) has emerged as a promising solution due to its strong global search capability, rapid convergence, and flexibil-ity in solving complex optimization problems. De-rived from the traditional Particle Swarm Optimiza-tion (PSO) algorithm and inspired by principles of quantum mechanics, QPSO enhances the search process by exploring a wider solution space. This makes it particularly suitable for healthcare cloud environments, where workloads and resource demands can change continuously.

In this study, a QPSO-based task scheduling framework is proposed for healthcare applications operating in cloud environments. The framework analyzes task characteristics, including computational requirements and data size, and dynamically assigns tasks to suitable virtual machines according to current resource availability. The main goals of the proposed approach are to reduce overall completion time, lower execution costs, improve resource utilization, and ensure that critical healthcare services are processed without delay. To evaluate the effectiveness of the proposed framework, eight widely Therefore, intelligent resource allocation and task recognized task scheduling algorithms were implemented and tested using the CloudSim simulation platform. Although some of these algorithms are not the latest developments in the field, they continue to serve as important benchmark methods for performance evaluation. Experiments were conducted under different task loads and virtual machine configurations, and key performance metrics such as makespan, throughput, and load balancing were analyzed. The results are presented through graphical comparisons and detailed discussions, highlighting the strengths and limitations of each scheduling technique. These findings provide useful guidance for researchers and practitioners seeking efficient scheduling solutions for cloud-based healthcare systems.

The primary contribution of the proposed work are summarize as follows:

- Design of a customised QPSO algo sched-ules heterogeneous medical tasks by consid-ering key parameters such as overall response time, overall cost, data centre processing time.
- Comprehensive evaluation of the proposed algorithm using a cloud simulation environment (e.g., CloudSim), benchmarked against tradi-tional scheduling algorithms such as Round Robin, Max Min, Min-Min, classical PSO, Honey bee, Ant colony, Earth worm, Bat colony.

The remainder of this paper is organized as follows: Section II presents a review of related literature on task scheduling in cloud computing. Section III details the implementation of the selected scheduling algorithms. Section IV provides a comprehensive analysis of simulation results and a comparative performance evaluation. Section V concludes the study with a summary of findings and potential directions for future research.

2. RELATED WORKS

Most existing studies in the literature primarily concentrate on developing new scheduling algorithms, with only a limited number dedicated to analyzing and comparing various task-scheduling algorithms. In [1] The authors proposed a Patient Health Monitoring System (PHMS) to remotely track vital signs such as blood pressure, body temperature, and heart rate (ECG). The system includes two modules: a real-time IoT-based task scheduling algorithm and

an optimization module using an objective function to reduce task idle time, thereby enhancing PSO scheduling performance. Using the Libelium e-Health toolkit, the system showed improved effectiveness by lowering task starvation and failure rates.

However, the study did not address key metrics like execution cost, energy consumption, execution time, or makespan.

In [2], a healthcare load balancing model using the Sparrow Search Algorithm (SSA) was proposed to assign high-computation tasks to the most suitable VMs based on fitness values. Simulated with CloudSim, the model showed good results in processing time, makespan, and load balancing. However, it did not consider task criticality or latency, and the repeated fitness evaluations led to high computation time for complex tasks.

In [3], a Hybrid Moth Flame Optimization (HMFO) algorithm integrated with a deep neural network was introduced for task scheduling in cloud-based Internet of Health Things (IoHT) systems. The model was trained using the Google cluster dataset to en-able real-time scheduling decisions. Performance evaluation using CloudSim showed improvements in cost and average run time. However, the method resulted in higher response time and did not consider task priority, delay, or throughput in its analysis.

In [4], the authors introduced a Fog-Cloud IoHT architecture where healthcare data is routed to either the fog or cloud layer based on traffic type. The fog layer handles initial processing before offload-ing data to the cloud. To manage load balancing and data scheduling across devices, a Particle Swarm Optimization (PSO) algorithm was employed. The approach outperformed Random and FCFS meth-ods in terms of latency, throughput, and delivery ra-tio. However, it did not evaluate key factors such as makespan and execution cost.

In [5], a static task scheduling algorithm was introduced for fog computing in healthcare, prioritizing tasks based on their importance rather than task length. The method, called Tasks Classification and Virtual Machines Categorization (TCVC), grouped tasks into high, medium, and low importance lev-els according to the patient's health condition. It was evaluated using the Max–Min algorithm, mea-suring average execution time, waiting time, and finish time. Simulated in CloudSim, the approach showed better results than Max–Min, FCFS, and SJF. However, it did not address makespan, latency, or processing cost, and lacked support for dynamic scheduling, which is essential for real-time applications.

In [6], the authors introduced a model for optimal VM selection in cloud-based healthcare using Parallel Particle Swarm Optimization (PPSO). The approach incorporated neural networks and linear regression to predict chronic kidney disease. The model was evaluated based on total execution time, accuracy, and system efficiency. However, it did not assess key metrics such as makespan, delay, and throughput.

Malik et.al compares various job scheduling techniques such as round robin, PSO, MIN MIN etc based on several parameters. The goal is to identify the most competent task-scheduling algorithm [1][2], The authors conducted a comparative analysis of the Min-Min and Max-Min algorithms, focusing on the makespan. Both algorithms employ a static approach along with task-autonomous and batch-processing scheduling techniques. As Accentuated by Sanjay et al. in [3], these algorithms have been executed Job Orchestration in Cloud Computing, utilizing both temporal and spatial sharing models for virtual machines. The study, conducted using CloudSim, compares four approaches to scheduling in cloud computing: FCFS, STF, LTF, and RR, under Time-multiplexed and space-multiplexed allocation policies. Results show that the Max-Min algorithm excels in load balancing in space-division mode, while Min-Min minimizes task processing delays. Among the algorithms, STF achieved the shortest total completion time with the space-shared policy.

[5] Multiple heuristic algorithms were examined in both homogeneous and heterogeneous environments, taking into account situations with and without workload traces. This thorough evaluation facilitated a detailed analysis of their performance across different conditions. [6]. Authors have conducted a comparative examination of static and dynamic task scheduling algorithms utilized in cloud computing environments. This study offers an in-depth assessment of the effectiveness of these algorithms in optimizing task scheduling within cloud infrastructures.

[7] Thaman et al. present a taxonomy of meta-heuristic and heuristic algorithms for task and job scheduling including ACO, HONEY BEE, and

BCO. Their categorization is based on the objectives and constraints associated with task scheduling algorithms, providing a structured framework for understanding the various approaches within this domain [8]. An algorithmic enhancement has been proposed by Tabak, et al. They have proposed two hybrid algorithms integrated with Max-Min [9] and Sufferage algorithms without compromising the QoS and also reduce the execution time asymptotically.

With the expanding user base of cloud computing systems, the volume of tasks requiring scheduling has surged accordingly. Consequently, there's a pressing demand for enhanced algorithms tailored to task scheduling within these systems. In this study, the focus is placed on a range of task scheduling algorithms, which will be elaborated upon in this section.

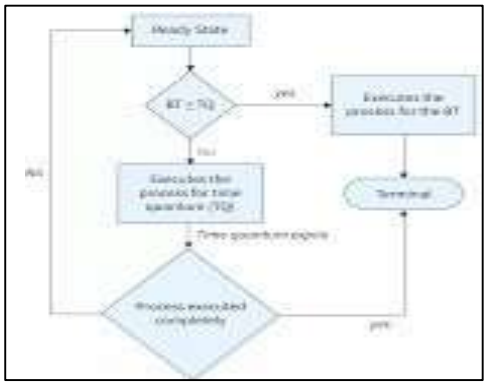


Figure 1- Round robin Flowchart

3. TASK SCHEDULING ALGORITHMS

3.1 Round Robin Scheduling Algorithm (RR)

[13] This approach is limited by the unpredictable nature of cloud workloads and the abstraction of re-sources through virtualization. These uncertainties affect Quality of Service (QoS) due to fluctuating re-source availability and bandwidth. Researchers are addressing these challenges by focusing on predict-ing task completion and queue times to enhance re-source utilization. This work aims to reduce uncer-tainties and optimize cloud services by improving re-source allocation and efficiency.

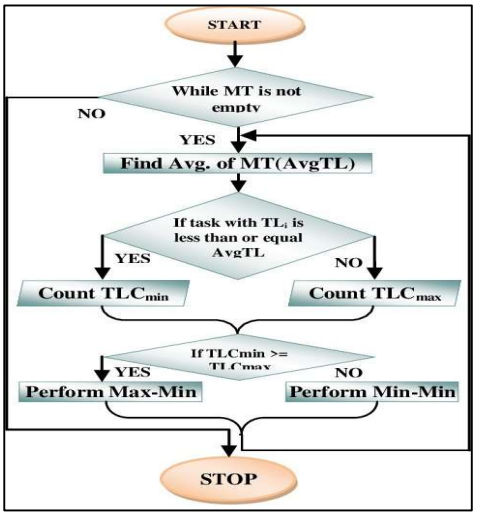


Figure 2- MAX-MIN Flowchart

3.2 MAX-MIN Scheduling Algorithm

[14] Auto-scaling in cloud computing dynamically allocates resources in response to both planned and unexpected demand. Its goal is to promptly assign resources to handle incoming workloads, ensuring efficient utilization by distributing tasks to available virtual machines. Achieving this requires the use of optimal scheduling techniques to effectively allocate tasks across the cloud provider's resources.

3.3 MIN-MIN Scheduling Algorithm

[15] The Min-Min Scheduling Algorithm is a heuris-tic used in distributed computing and parallel pro-cessing environments to allocate resources (like pro-cessors) to tasks in a way that minimizes the overall completion time. It is a straightforward, yet effec-tive, approach to load balancing.

Journal of Theoretical and Applied Information Technology
15st September 2026. Vol.104. No.17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

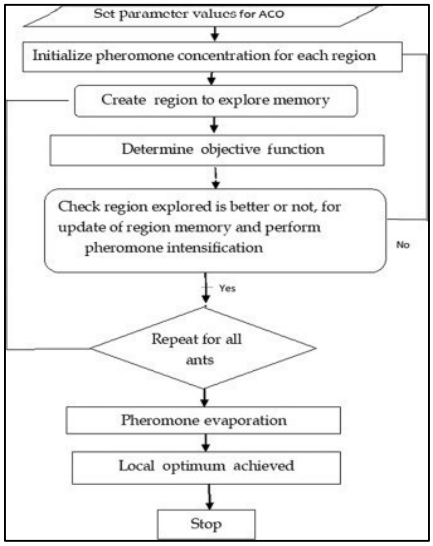


Figure 3- MIN-MIN Flowchart

3.4 Particle Swarm Optimization Algorithm (PSO)

[16] Cloud computing functions as a distributed sys-tem where tasks are allocated across multiple nodes located in different regions. each with varying com-putational power, memory, architecture and network performance. The performance of tasks can vary sig-nificantly depending on the specific nodes on which they are executed.

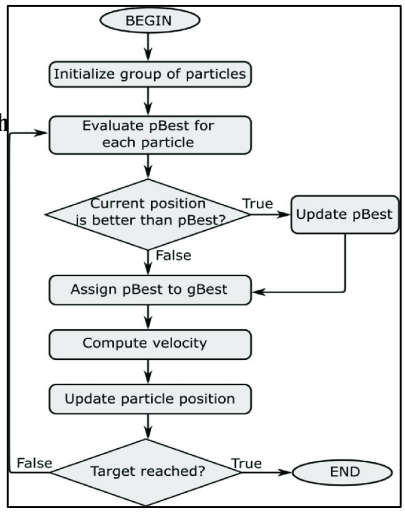


Figure 4- PSO Flowchart

3.7 Earth

3.5 Honey Bee Scheduling Algorithm

[17] The Honey Bee Scheduling Algorithm (HBSA) draws inspiration from the foraging behavior of honey bees, employing their social organization and communication strategies to enhance task schedul-ing, especially in cloud computing settings. This al-gorithm belongs to a wider class of swarm intelli-gence methods that replicate the collective behaviors observed in social organisms.

3.6 Ant Colony Scheduling Algorithm

[18] To efficiently utilize cloud resources, it is es-sential to distribute workloads evenly across vir-tual machines, preventing both under-utilization and over-utilization, and ensuring optimal performance throughout the system.

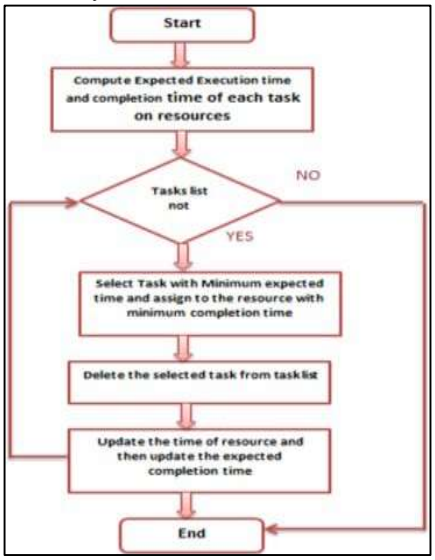


Figure 5- Ant Colony Flowchart

[19]

Earthworm Optimization Algorithm (EWA) is a relatively new bio-inspired algorithm that mimics the movement behavior of earthworms in soil. It has been used to solve various optimization prob-lems. Task scheduling is one of the most important functions in a cloud computing environment, as it determines how tasks are assigned to available vir-tual machines (VMs) and computing resources. The primary objective is to ensure that resources are uti-lized efficiently while achieving optimal system per-formance. Effective scheduling helps reduce over-

all task completion time, minimize energy usage and operational costs, and maintain a balanced workload across available resources. By intelligently matching tasks with suitable computing resources, cloud systems can deliver better performance, improved reliability, and enhanced resource utilization.

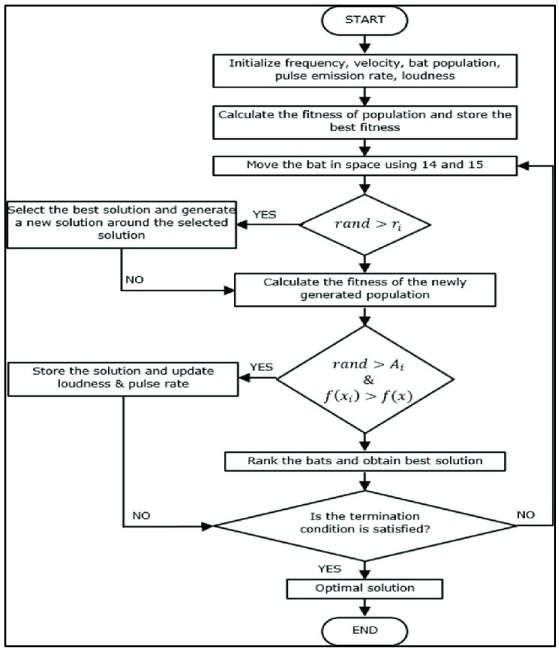


Figure 6-Earth Worm Flowchart

3.8 BAT Colony Optimization

[20] Bat Colony Optimization (BCO) is a bio-inspired algorithm that imitates the echolocation habits of bats. It has shown effectiveness in addressing complex optimization challenges, The ob-jective of task scheduling is to efficiently allocate tasks to virtual machines (VMs), optimizing perfor-mance metrics such as execution time, resource uti-lization, cost, and energy efficiency.

3.9 Quantum PSO Based Algorithm

[21] Quantum Particle Swarm Optimization (QPSO) is an advanced variant of the traditional Particle Swarm Optimization (PSO) algorithm. Using con-cepts from

The third step involves calculating the mean best position, which is the average of all personal best positions in the swarm. This value acts as a key reference point that guides the movement of particles throughout the search process. Based on this reference, each particle adjusts its position using a quantum-inspired update mechanism that considers the average best position of the swarm, the gap between its current position and its personal best position, and a randomly generated factor. Unlike traditional deterministic approaches, this probabilistic movement strategy is inspired by concepts from quantum mechanics and enables particles to explore the search space more effectively. As a result, the algorithm can in-vestigate a wider range of potential solutions and im-prove its ability to identify optimal or near-optimal outcomes. space more broadly compared to classical PSO, helping avoid getting trapped in local optima.

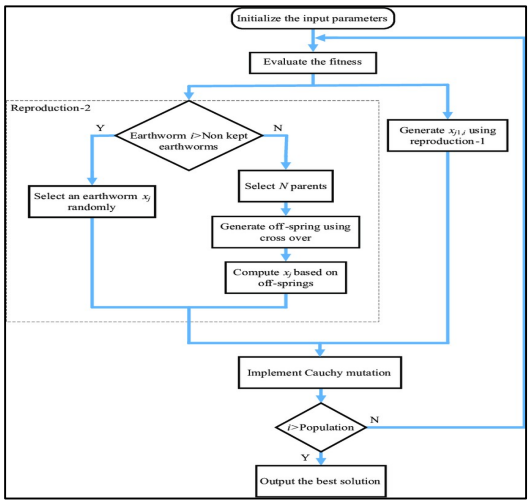


Figure 7-Bat Colony Flowchart

quantum mechanics to improve the search capabilities of the algorithm, leading to more efficient optimization. QPSO has been widely applied to various fields, including engineering, machine learning, and cloud computing task scheduling, due to its ability to perform better global searches and avoid local minima. In Algorithm 1, we have described the quantum PSO(QPSO) pseudocode.

[21] In the first step of the Quantum Particle Swarm Optimization (QPSO) algorithm, a population of particles is initialized, where each particle represents a possible solution to the task scheduling problem in a cloud computing environment. Specifically, each particle corresponds to a mapping of user tasks to available virtual machines (VMs). Along with this, each particle retains a record of its personal best position — the best solution it has found so far — while

the algorithm also keeps track of the global best position, which is the best solution discovered among all particles. The optimization process is then run for a fixed number of iterations or until an acceptable solution is found.

In the second step, during each iteration, calculate the fitness of each particle using a predefined fitness function. This function measures the quality of a solution, which in the context of cloud computing may involve minimizing metrics such as total execution time (makespan), operational cost, or energy consumption. A particle's personal best is updated if its present position gives an increased fitness value than the one before it. Likewise, if this new personal best is better than the current global best, the global best is also updated.

Following this update, the fitness of the new positions is re-evaluated, and the cycle continues. Gradually, the particles converge toward optimal or near-optimal solutions. Ultimately, the best global position represents the most efficient assignment of tasks from machine to machine discovered during the optimization process. Due to its quantum-based movement and enhanced global search capability, QPSO is particularly effective in solving complex dynamic optimization problems in cloud computing environments.

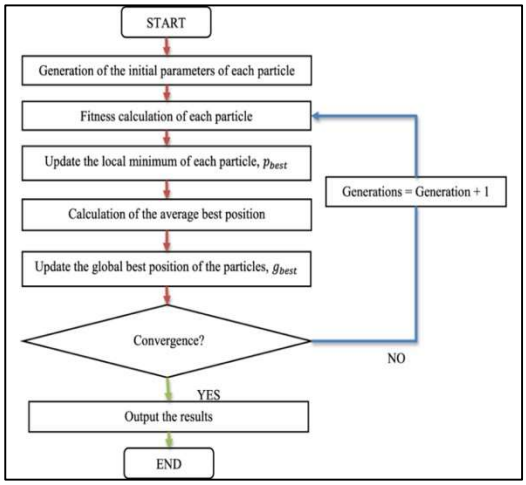


Figure 8- QPSO Flowchart

Why QPSO provides better performance: Quantum Particle Swarm Optimization (QPSO) demonstrates superior performance compared to traditional task scheduling algorithms such as First-Come-First-Serve (FCFS), Round Robin, Genetic Algorithms (GA), and classical PSO etc. This advantage is primarily due to its enhanced global search capabil-ity, faster convergence, and improved resource optimization. Unlike classical PSO, which often suf-fers from premature convergence and local optima entrapment, QPSO leverages quantum mechanics principles—particularly probabilistic position mod-eling—to enable a more comprehensive exploration of the solution space. By eliminating the reliance on velocity vectors, QPSO simplifies the computation process and enhances convergence stability.

It is also well-suited to dynamic and heterogeneous cloud environments where resource availability, task heterogeneity, and user demands are highly variable. Numerous simulation-based studies (commonly implemented in platforms like CloudSim) have consistently shown QPSO to outperform conventional scheduling algorithms across critical performance metrics, including makespan (total completion time), execution cost, resource utilization, load balancing, and energy efficiency. In fields such as healthcare and multimedia cloud computing, where accurate results and timely task execution are essential, QPSO offers significant advantages due to its flexibility and ability to adapt to changing system conditions. Its intelligent optimization process helps maintain high levels of Quality of Service (QoS) by ensuring efficient resource allocation and reliable task scheduling. By integrating the collective search behavior of swarm-based optimization with advanced quantum-inspired techniques, QPSO is able to explore solution spaces more effectively and identify high-quality scheduling decisions in complex cloud environments. theoretical robustness of quantum computation, QPSO provides a reliable, efficient, and scalable solution for complex task scheduling in cloud computing environments.

4. VERIFICATION OF QPSO

Quantum-based Particle Swarm Optimization (QPSO) is a powerful metaheuristic algorithm designed for efficient task scheduling in healthcare cloud environments. It enhances resource allocation by reducing makespan, delay, and cost, while optimizing resource utilization. To ensure QPSO operates reliably in critical healthcare scenarios, verification is essential. This involves formal verification techniques to assess key properties such

as:

- Deadline satisfaction – Ensuring all patient-related tasks are completed within specified time limits.
- Deadlock-freedom – Verifying the algorithm functions consistently without system halts or conflicts.
- Fair resource allocation – Confirming the system is balanced, fault-tolerant, and compliant with healthcare data standards. Such verification provides both mathematical and as-surance and practical validation that QPSO can be trusted in real time high stakes health care system.

5. VALIDATION OF QPSO

Validation of Quantum-based PSO (QPSO) involves assessing its effectiveness through simulations and PSO, Round Robin, and Genetic Algorithms to evaluate gains in scheduling accuracy. Tools such as CloudSim, commonly used to model and simulate QPSO for this purpose. Its performance is typically benchmarked against algorithms like traditional experimental testing using real or synthetic workloads. Key performance indicators include:

- Makespan – The total time required to complete all scheduled tasks
- Throughput – The number of tasks completed within a specific time frame
- Deadline hit rate – The proportion of tasks fin-ished within their assigned deadlines
- Resource utilization – The efficiency with which system resources are used
- Energy and cost efficiency – Optimization of power consumption and operational cost

Validation of Quantum-based PSO (QPSO) involves assessing its effectiveness through simulations and PSO, Round Robin, and Genetic Algorithms to evaluate gains in scheduling accuracy. Tools such as CloudSim, commonly used to model and simulate QPSO for this purpose. Its performance is typically benchmarked against algorithms like traditional experimental testing using real or synthetic workloads. Key performance indicators include:

- Makespan – The total time required to complete all scheduled tasks
- Throughput – The number of tasks completed within a specific time frame
- Deadline hit rate – The proportion of tasks fin-ished within their assigned deadlines
- Resource utilization – The efficiency with which system resources are used
- Energy and cost efficiency – Optimization of power consumption and operational cost

6. TESTING OF QBPSO

Algorithm 1 Quantum Particle Swarm Optimization (QPSO) for Cloud Task Scheduling

Initialize the number of particles N, maximum iterations $t_{\max}$, and QPSO parameters
Randomly initialize X_i for each particle i (task-to-VM mapping)
Set $P_i \leftarrow X_i$ and $G \leftarrow \arg\min_{P_i} F(P_i)$
for $t = 1$ to $t_{\max}$ **do**
  for each particle i **do**
    Calculate fitness $F(X_i)$
    if $F(X_i) < F(P_i)$ **then**
      $P_i \leftarrow X_i$ ▷ Update personal best
    end if
    if $F(P_i) < F(G)$ **then**
      $G \leftarrow P_i$ ▷ Update global best
    end if
  end for
  Compute $M_b \leftarrow \frac{1}{N} \sum_{j=1}^{N} P_j$
  for each particle i **do**
    for each dimension d of X_i **do**
      Generate $u \sim U(0, 1)$
      $X_i[d] \leftarrow M_b[d] \pm \beta |P_i[d] - X_i[d]| \ln(1/u)$
    end for
    Evaluate the new fitness $F(X_i)$
  end for
end for
return global best solution G

Testing of Quantum-based Particle Swarm Optimization (QPSO) involves executing the algorithm on predefined healthcare task scheduling scenarios to evaluate its correctness, stability, and performance. Unlike formal verification or comprehensive validation, testing aims to identify practical issues such as bugs, inefficiencies, or inconsistent behavior during execution. Key objectives of testing include:

- Verifying accurate mapping of tasks to available resources
- Monitoring system response under high work-load

conditions

- Identifying any failures or irregularities in task execution
- Ensuring consistent performance across multi-ple runs

Testing is typically conducted using simulation tools like CloudSim, which provide controlled environments to observe algorithm behavior. Through systematic testing, developers can confirm that QPSO meets the functional requirements of healthcare cloud systems before proceeding to full-scale validation or deployment.

To ensure the reliability of Quantum-based Parti-cle Swarm Optimization (QPSO) in critical health-care applications, it is essential to test the algorithm under challenging conditions beyond normal work-loads. This includes:

- Unfairness Testing Assesses whether QPSO unfairly prioritizes certain task. It involves scheduling a high and low priority task to check for resource imbalance, task starvation or fair-ness violation.
- Critical Input Testing Evaluates QPSO's re-sponse to extreme conditions such as high task volumes, limited resources, or real-time emergency requests. The goal is to detect failures like crashes, missed deadlines, or degraded performance.
- Risk-Based (Worst-Case) Testing Focuses on high-risk scenarios including uniform task deadlines, sudden VM failures, or data issues. Observations include system failure handling, recovery, and performance impact key metrics.

These tests are crucial to confirm that QPSO performs reliably, efficiently, and fairly under stress, making it suitable for deployment in real-time healthcare cloud systems.

7. APPLICATION SCENARIO : MEDICAL RESOURCES

In a modern healthcare cloud system, managing medical resources such as doctors, diagnostic equipment, ICU beds, and virtual machines (VMs) is critical—especially during high-demand periods like pandemics or emergencies. Scenario: A hospital network deploys a cloud-based scheduling system powered by Quantum-based Particle Swarm Optimization (QPSO). The system receives continuous task requests such as patient record analysis, diagnostic imaging processing, real-time monitoring data, and appointment scheduling.

QPSO's Role:

- Assigns tasks (e.g., processing MRI scans) to the most suitable available computing resource or expert system.
-
- Ensures critical patients are prioritized without starving routine cases.
- Optimizes resource usage by reducing waiting times, energy consumption, and task deadlines.
- Handles sudden surges in demand (e.g., during a disease outbreak) while maintaining system performance and fairness.

This scenario demonstrates QPSO's practical value in efficient, fair, and adaptive resource allocation in life-critical healthcare environments.

8. MEMORY MANAGEMENT IN QPOS

In cloud-based healthcare environments, memory management is vital to ensure smooth task scheduling and data handling. When using QPSO, memory is used to store particle states, fitness values, task queues, and resource mappings. Key strategies include:

- Efficient storage of particle data and fitness re-sults
- Dynamic memory updates for tasks and virtual resources
- Isolation of memory for real-time or emergency healthcare tasks
- Quick cleanup of unused memory after task completion

Optimizing memory ensures that QPSO remains scalable, fast, and reliable even under high workloads in critical medical systems.

9. POWER MANAGEMENT

In healthcare cloud environments, power efficiency is essential due to continuous data processing and resource use. When implementing QPSO for task scheduling, power management ensures reduced energy consumption without compromising performance. Key aspects include:

- Optimized task-resource mapping to avoid overloading and idle resource usage
- Dynamic VM scaling to power down unused servers
- Deadline-aware scheduling to reduce process-ing time and energy waste
- Load balancing to distribute work evenly, minimizing power peaks

QPSO improves power management by selecting energy-efficient scheduling paths, making it suitable for green, sustainable healthcare systems.

10. GREEN MANAGEMENT

Green management focuses on reducing the environmental impact of cloud-based healthcare systems. When using QPSO for task scheduling, it contributes to eco-friendly operations by:

- Minimizing energy usage through efficient task allocation
- Reducing carbon footprint by avoiding overuse of computing resources
- Optimizing VM utilization to limit unnecessary power consumption
- Encouraging sustainable practices like dynamic scaling and idle VM shutdown

By promoting energy-aware and resource-efficient scheduling, QPSO supports green cloud computing in healthcare, ensuring performance without harming the environment. While this is comparable to classical PSO, QPSO may introduce slightly greater computational cost due to its advanced update mechanism. Nonetheless, the added complexity is often justified by improved convergence and global optimization performance mechanism. Nonetheless, the added complexity is often justified by improved convergence and global optimization performance.

Key insights: QPSO outperforms others in critical, high-stakes environments (like healthcare) due to its global search ability, high fairness, and better convergence. Round Robin, though simple, is not resource-aware, making it less suitable for dynamic cloud systems. Min min and max min works for small system but fail under high complexity or urgency. PSO, QPSO, Earth worm, Bat colony shows strong adaptability and efficiency. Suitable for real time health care need.

11. EXPERIMENTAL RESULTS AND COMPARATIVE PERFORMANCE EVALUATION

We use the CloudSim simulation toolkit to comparative performance comparison analysis. Our cloudlets (tasks) are configured with a single CPU, a file size of 300, an output size of 300, and a randomly generated cloudlet length within CloudSim. The virtual machines (VMs) have the following specifications: 512 MB of RAM, 250 MIPS (Million Instructions

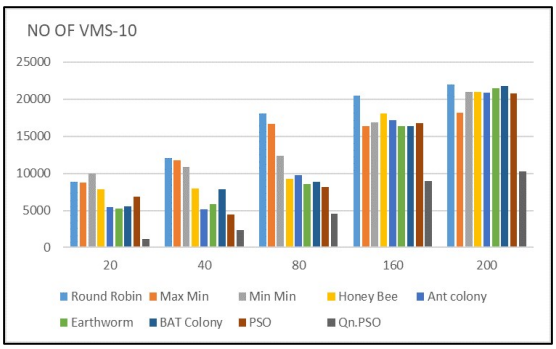


Figure 9- Makespan when VM =10

Journal of Theoretical and Applied Information Technology
15st September 2026. Vol.104. No.17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

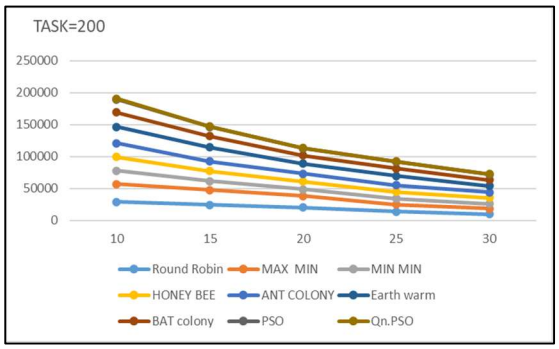


Figure 10 – Makespan When Task=200

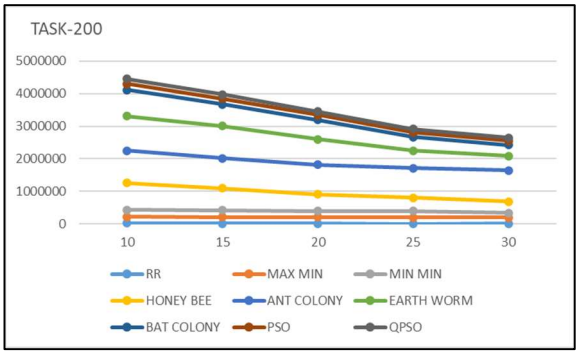


Figure 13- Makespan When Task=200

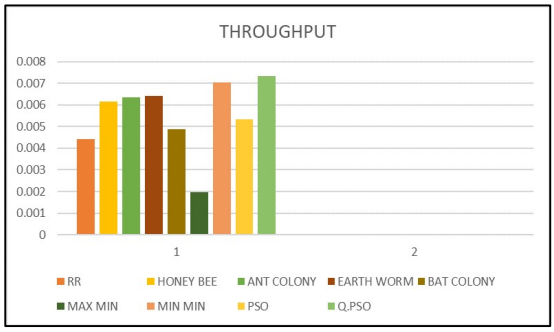


Figure 11-THROUGHPUT 1

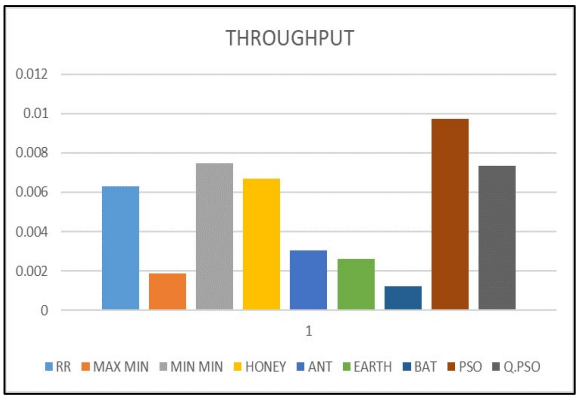


Figure 14-THROUGHPUT 2

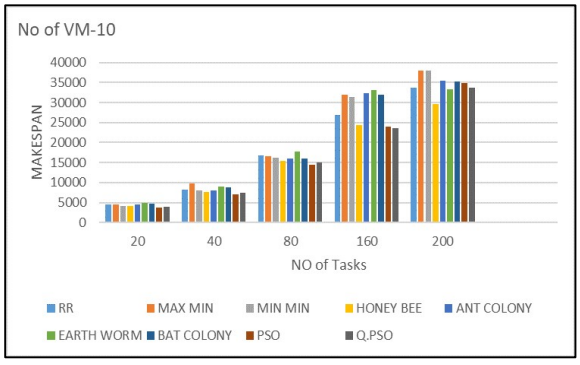


Figure 12- Makespan When Vm =10

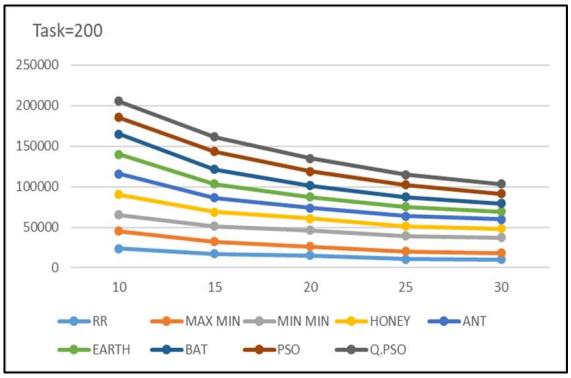


Figure 15-Makespan When Task=200

-

Table 1: Qpso Vs. Other Scheduling Algorithm

Algorithm	***Type***	***Nature***	***Complexity***	***Resource Util***	***Adaptability***	***Load Bal***	***Schd Eff.***	***Remarks***
Round Robin	*Deterministic*	*Static*	*O(n)*	*Low*	*Low*	*Poor*	*Low*	*Fair but ignores task size and resource power*
MIN MIN	*Heuristic*	*Static*	$O(n^2)$	*Moderate*	*Low*	*Poor*	*Moderate*	*Favors short tasks; long tasks suffer*
MAX MIN	*Heuristic*	*Static*	$O(n^2)$	*Moderate*	*Low*	*Moderate*	*Moderate*	*Prioritizes large tasks*
PSO	*Metaheuristic*	*Dynamic*	*O(N×D×I)*	*High*	*High*	*Good*	*High*	*May get trapped in local optima*
HONEY BEE	*Bio-inspired*	*Dynamic*	*O(N×I)*	*Moderate*	*High*	*Good*	*High*	*Inspired by foraging behavior*
ANT COLON	*Bio-inspired*	*Dynamic*	$O(n^2 \times t)$	*Moderate*	*Moderate*	*Good*	*Moderate*	*Pathfinding-based; slow convergence*
EARTH WORM	*Bio-inspired*	*Dynamic*	*O(N×I)*	*High*	*High*	*Good*	*High*	*Mimics soil optimization; fast convergence*

BAT COLON	*Metaheuristic*	*Dynamic*	*O(N×D×I)*	*High*	*High*	*Good*	*High*	*Good convergence, but sensi-tive to parameters*
QPSO	*Metaheuristic*	*Dynamic (Quantum)*	*O(N×D×I)*	*Very High*	*Very High*	*Excellent*	*Very High*	*Stronger global search; better for critical systems*

Per Second), a bandwidth of 1000, one CPU, and the Virtual Machine Monitor (VMM) named ”Xen”. These configurations are consistently applied across all algorithms during the simulation in CloudSim.

We evaluate throughput, load balancing, and makespan by varying the virtual machines and the number of tasks. we get the result of all the 9 algorithms using three categories of place-ment policy such as optimize response time(ORT), closest data center(CDC), and reconfigure dynam-ically(RD). The ”Closest Data Center” policy in CloudAnalyst is a straightforward and efficient ap-proach for routing user requests to the most appropri-ate data center within a cloud environment. This pol-icy focuses on minimizing data transaction latency by directing requests to the geographically nearest data center. Reducing the distance between users and data centers enhances response times and sig-nificantly improves the overall user experience.

Makespan is the time difference between starting and completing a series of tasks. Figures 10 and 11 present the average makespan for all the algo-rithms compared. Figure 10 illustrates the average makespan as the number of tasks varies. In this figure, QPSO demonstrates superior performance in terms of makespan, as it effectively finds near-optimal solutions. PSO also reduces the makespan by executing smaller tasks concurrently while pro-cessing longer tasks. On the other hand, algorithms like RR, MAX-MIN, and other meta-heuristic ap-proaches experience higher makespan due to in-creased context switching when handling a large number of tasks, resulting in poorer performance compared to the other methods.

Figure 11 depicts the average makespan as the num-ber of virtual machines (VMs) varies.

Expanding the quantity of VMs clearly results in a significant re-duction in makespan. Similarly, QPSO consistently yields the lowest makespan in this scenario andthen, the RR AND meta-heuristics approach gives a better makespan figure 12 shows the throughput of all the algorithms.

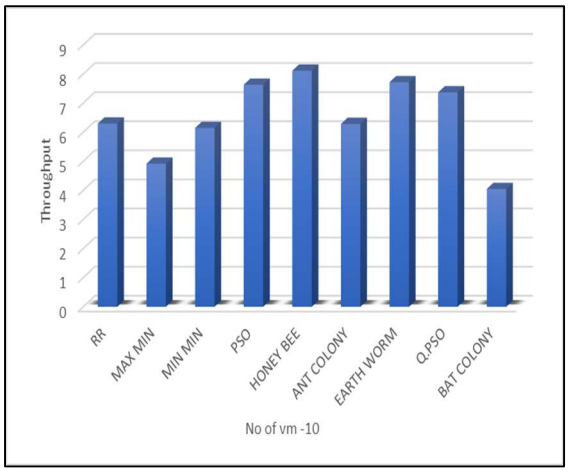


Figure 16-Makespan when VM=10

Figure 13 illustrates the average makespan as the number of tasks changes. In this figure, QPSO demonstrates superiority among the algorithms with respect to makespan, as it effectively finds near-optimal solutions. PSO also contributes to a re-duced makespan by executing smaller jobs concur-rently while handling longer tasks. In contrast, al-gorithms such as RR, MAX-MIN, and other meta-heuristic methods experience the highest makespan due to increased context switching when managing a large number of tasks.

Figure 14 presents the average makespan as the num-ber of virtual machines (VMs) varies. raising the quantity of VMs clearly contribute to a significant reduction in makespan. Similarly, QPSO continues to yield the lowest makespan in this scenario. and then, the RR AND meta

then, the RR AND meta heuristics approaches gives better makespan figure 18 shows the throughput of all the algorithms.

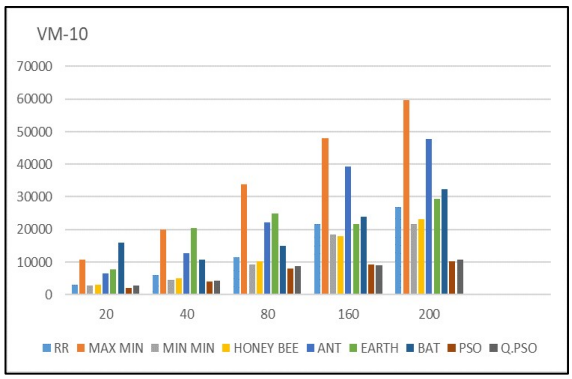


Figure 17-THROUGHPUT 3

heuristics approaches gives better makespan figure 15 shows the throughput of all the algorithms.

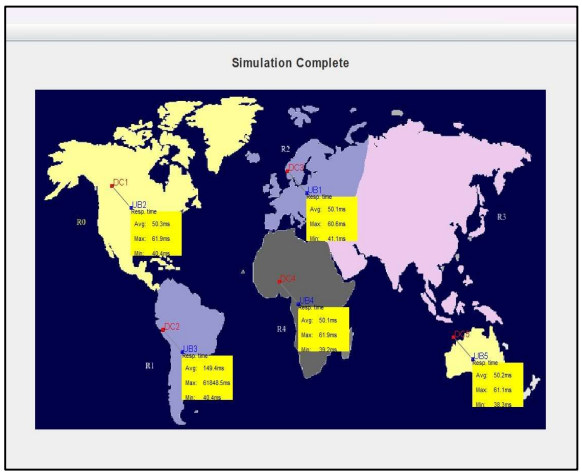


Figure 18- Simulation Environment In Cloud Analyst With DATACENTER Configuration

Figure 16 illustrates the average makespan as the number of tasks varies. In this figure, QPSO demonstrates superior performance among the algorithms in terms of makespan, as it is capable of finding near-optimal solutions. PSO also contributes to a re-duced makespan by allowing smaller jobs to be exe-cuted concurrently while longer jobs are processed. Conversely, algorithms like RR, MAX-MIN, and other meta-heuristic approaches exhibit the highest makespan due to increased context switching when dealing with a large number of tasks.

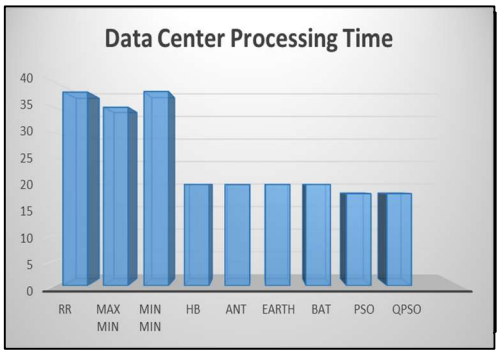
Figure 17 presents the average makespan as the number of virtual machines (VMs) changes. It is evident that increasing the number of VMs leads to a noticeable decrease in makespan. Similarly, QPSO consistently yields the lowest makespan in this context and

Figure 19 depicts the Cloud Analyst simulation environment, where nine different task scheduling policies are simulated. This simulation calculates Cost, processing time of requests, and response time, presenting the results in a tabular format as table I,II,III. In figure 19, five data centers (DC) and one user base (UB) are located in different regions, resulting in a total of six regions, each identified by a unique region ID or cloud ID: North America (ID 0), South America (ID 1), Europe (ID 2), Asia (ID 3), Africa (ID 4), and Oceania (ID 5). Each region has distinct users, peak simulation times, durations, bandwidths, and latencies. The figure illustrates the placement of the UB and DC in Regions 0, 1, 2, 3, and 4, showcasing the minimum, maximum, and average processing times for each respective region.

Table 2: OVERALL PROCESSING COST

Policy	AVG	MIN	MAX
RR WITH CDC	0.02	0.02	2.83
RR WITH ORT	0.01	0.01	2.83
RR WITH RD	16.01	0.02	16.03
MinMax WITH CDC	2.51	0.02	2.83
MinMax WITH ORT	2.51	0.01	2.83
Min Max WITH RD	16.19	0.02	16.51
MAX MIN WITH CDC	2.51	0.02	2.83
MAX MIN WITH ORT	2.51	0.01	2.83
MaxMin WITH RD	16.19	0.02	16.51
HB WITH CDC	2.51	0.02	2.83
HB WITH ORT	2.51	0.01	2.83
HB WITH RD	15.30	0.02	15.62
ANTC WITH CDC	2.51	0.02	2.83
ANTC WITH ORT	2.51	0.01	2.83
ANTC WITH RD	16.19	0.02	16.51
EW WITH CDC	2.51	0.02	2.83
EW WITH ORT	2.51	0.01	2.83
EW WITH RD	16.19	0.02	16.51
BATC WITH CDC	2.51	0.02	2.83
BATC WITH ORT	2.51	0.01	2.83
BATC WITH RD	16.19	0.02	16.51
PSO WITH CDC	2.51	0.02	2.83
PSO WITH ORT	2.51	0.01	2.83
PSO WITH RD	16.19	0.02	16.51
QPSO WITH CDC	2.51	0.02	2.83
QPSO WITH ORT	2.51	0.01	2.83

Figures 20 and 21 display the overall response time and the total processing cost of the system. show-ing that the processing cost in the QPSO algorithm is



significantly lower than that of the other scheduling algorithms data center (DC) processing time of the system, respectively. The QPSO algorithm in both figures demonstrates the shortest completion times

Table 3: DATA CENTER PROCESSING TIME

Policy	AVG	MIN	MAX
RR WITH CDC	0.02	0.02	2.83
RR WITH ORT	0.01	0.01	2.83
RR WITH RD	16.01	0.02	16.03
MinMax WITH CDC	2.51	0.02	2.83
MinMax WITH ORT	2.51	0.01	2.83
Min Max WITH RD	16.19	0.02	16.51
MAX MIN WITH CDC	2.51	0.02	2.83
MAX MIN WITH ORT	2.51	0.01	2.83
MaxMin WITH RD	16.19	0.02	16.51
HB WITH CDC	2.51	0.02	2.83
HB WITH ORT	2.51	0.01	2.83
HB WITH RD	15.30	0.02	15.62
ANTC WITH CDC	2.51	0.02	2.83
ANTC WITH ORT	2.51	0.01	2.83
ANTC WITH RD	16.19	0.02	16.51
EW WITH CDC	2.51	0.02	2.83
EW WITH ORT	2.51	0.01	2.83
EW WITH RD	16.19	0.02	16.51
BATC WITH CDC	2.51	0.02	2.83
BATC WITH ORT	2.51	0.01	2.83
BATC WITH RD	16.19	0.02	16.51
PSO WITH CDC	2.51	0.02	2.83
PSO WITH ORT	2.51	0.01	2.83
PSO WITH RD	16.19	0.02	16.51
QPSO WITH CDC	2.51	0.02	2.83
QPSO WITH ORT	2.51	0.01	2.83
QPSO WITH RD	16.19	0.02	16.51

Figure 19- Overall Response Time

Figure 20-Data Center Processing Time for tasks, indicating that the waiting time is min-imized with the QPSO scheduling algorithm com-pared to other eight algorithms. Figure 22 presents data center processing time. similarly tables 1 , Ta-ble 2 & Table 3 shows the data of overall response time, Data center processing time and overall cost.

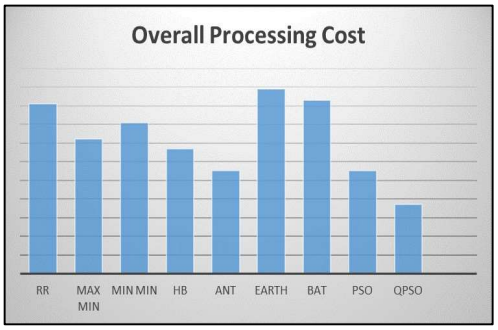


Figure 21- Overall Processing Cost

Table 4: Overall Response Time

Policy	AVG	MIN	MAX
RR WITH CDC	1.57	0.32	2.83
RR WITH ORT	1.57	0.32	2.83
RR WITH RD	8.17	0.32	16.03
MinMax WITH CDC	1.57	0.32	2.83
MinMax WITH ORT	1.57	0.32	2.83
Min Max WITH RD	8.41	0.32	16.51
MAX MIN WITH CDC	1.57	0.32	2.83
MAX MIN WITH ORT	1.57	0.32	2.83
MaxMin WITH RD	8.41	0.32	16.51
HB WITH CDC	1.57	0.32	2.83
HB WITH ORT	1.57	0.32	2.83
HB WITH RD	7.97	0.32	15.62
ANTC WITH CDC	1.57	0.32	2.83
ANTC WITH ORT	1.57	0.32	2.83
ANTC WITH RD	8.41	0.32	16.51
EW WITH CDC	1.57	0.32	2.83
EW WITH ORT	1.57	0.32	2.83
EW WITH RD	8.41	0.32	16.51
BATC WITH CDC	1.57	0.32	2.83
BATC WITH ORT	1.57	0.32	2.83
BATC WITH RD	8.41	0.32	16.51
PSO WITH CDC	1.57	0.32	2.83
PSO WITH ORT	1.57	0.32	2.83
PSO WITH RD	8.41	0.32	16.51
QPSO WITH CDC	1.57	0.32	2.83
QPSO WITH ORT	2.51	0.32	2.83
QPSO WITH RD	14.42	0.32	14.42

12. CONCLUSION

Cloud computing plays a vital role in research and healthcare by enabling efficient task scheduling through virtual resource management. In this study, nine popular scheduling algorithms were evaluated, and Quantum-based Particle Swarm Optimization

QPSO) outperformed others in metrics like average waiting time, turnaround time, makespan, throughput and load balancing. While algorithms like Ant Colony, PSO, Bat Colony, and Honey Bee showed strengths in specific areas, QPSO offered a balanced performance. In healthcare, where real-time task execution, deadline adherence, and resource efficiency are critical, QPSO proves highly effective. Its strengths include global optimization, flexibility, and fairness; however, challenges remain such as param eter sensitivity, higher computational overhead, and non-determinism. Future work will explore adaptive and hybrid scheduling models tailored for healthcare cloud systems.

REFERENCES

[1] Malik, Babur Hayat, Mehwashma Amir, Bilal Mazhar, S.H. Ali, Rabiya Jalil and Javaria Khalid. ”Comparison of Task Scheduling Algorithms in Cloud Environment.” International Journal of Advanced Computer Science and Applications 9 (2018): n. pag.

[2] Neha Sharma and D. Sanjay Tyagi, ”A comparative analysis of min-min and max-min algorithms based on the makespan parameter”, International Journal of Advanced Research in Computer Science, vol. 8, no.03, April 2017.

[3] Sanjay Kumar and Atul Mishra, ”Application of min-min and maxmin algorithm for task scheduling in cloud environment under time shared and space shared vm models”, International Journal of Computing Academic Research (IJCAR), vol. 04, no. 06, pp. 182–190, December 2015.

[4] Elzeki O. M.; Reshad M. Z.; Elsoud M. A.,” Improved Max-Min Algorithm in Cloud Comput-ing”, International Journal of Computer Applica-tions, Vol.50 (12), pp.22-27, July 2012.

[5] F. Alhaidari, T. Balharith and E. AL-Yahyan, ”Comparative Analysis for Task Scheduling Algo-rithms on Cloud Computing,” 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-6.

[6] Madni, Syed Hamid Hussain, Muhammad Shafie Abd Latiff, Mohammed Abdullahi, Shafi’I. Muham-mad Abdulhamid, and Mohammed Joda Usman. ”Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment.” PloS one 12, no. 5 (2017): e0176321.

[7] Akilandeswar P, Srimathi H. Survey and analysis on TaskschedulingnCloud environment t. Indian Journal of Science and Technology. 2016;:9(37):1-6.

[8] Thaman j,singh M current prospective in task scheduling techniques in cloud computing:A review
.international journal in foundation of computer science and technology 2016;6:65-85.

[9] Tabak EK,cambazoglu bb, aykanat c, improving the performance of independent task assignment heuristic min min ,max-min and suffrage. IEEE taransaction on parallel and distributed systems.2014;25(5):1244-56.

[10] Siddique, Md & Sharmin, Selina & Ahammad, Tanvir. (2020). Performance Analysis and Comparison Among Different Task Schedul-ing Algorithms in Cloud Computing. 1-6. 10.1109/STI50764.2020.9350466.

[11] Mukherjee P, Pattnaik P, Swain T, Datta A. Task scheduling algorithm based on multi criteria deci-sion making method for cloud computing environ-ment: TSABMCDMCCE. Open Computer Science . 2019;9(1): 279-291.

[12] Syed Hamid Hussain Madni, Muhammad Shafie Abd Latiff, Mohammed Abdullahi, Shafi’i Muham-mad, ”Performance comparison of heuristic algo-rithms for task scheduling in IaaS clood nvironment” PLOS ONE, May 2017.

[13] T. Balharith and F. Alhaidari, ”Round Robin scheduling algorithin CPU and Cloud computing:A review: 2019 2nd International conference on computer applications and information secu-rity(ICCAIS),Ryadh Saudi arabis.2019,pp 1-7.

[14] Mao, Y., Chen, X., Li, X. (2014). Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing. In: Patnaik, S., Li, X. (eds) Proceedings of International Conference on Com-puter Science and Information Technology. Ad-vances in Intelligent Systems and Computing, vol 255. Springer, New Delhi.

[15] Huankai Chen, F. Wang, N. Helian and G. Akanmu, ”User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing,” 013 National Conference on

Parallel Computing Technologies (PARCOMPTECH), Bangalore, India, 2013, pp. 1-8.
[16] Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. Arch Computat Methods Eng 29, 2531–2561 (2022).
[17] Dhinesh Babu L.D., P. Venkata Krishna,Honey bee behavior inspired load balancing of tasks in cloud computing environments,Applied Soft Com-puting,Volume 13, Issue 5,2013.
[18] C .-Y. Liu, C. -M. Zou and P. Wu, "A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing," 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineer-ing and Science, Xi'an, China, 2014,pp. 68-72.
[19] Kumar, M. Santhosh, and Ganesh Reddy Karri. 2023. "EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework" Sensors 23, no. 5: 2445.
[20] BAT algorithm used for load balancing pur-pose in cloud computing: an overview Arif Ullah, Nazri Mohd Nawi, and Mubashir Hayat Khan In-ternational Journal of High-Performance Computing and Networking 2020 16:1, 43-54 Bo Wang, Zhifeng Zhang, Ying Song, Ming Chen, Yangyang Chu, Ap-plication of Quantum Particle Swarm Optimization for Task Scheduling in Device-Edge-Cloud Cooper-ative Computing, Engineering Applications of Artificial Intelligence, Volume 126, Part C,2023,107020.
[21] I qbal N, Imran S, Ahmad RA, Kim DH (2021) A scheduling mechanism based on optimization us-ing IOT-tasks orchestration for efcient patient health monitoring. Sensors 21(16):1–29.
[22] Abdulhammed OY (2022) Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm. J Supercomput 78(3):3266–3287.
[23] Arivazhagan N et al (2022) Cloud-internet of health things (IOHT) task scheduling using hybrid moth fame optimization with deep neural network algorithm for e healthcare systems. Sci Pro-gram.
[24] Chudhary R, Sharma S (2021) Fog-cloud assisted framework for heterogeneous internet of health-care things. Procedia Comput Sci 184(2019):194–201.
[25] Aladwani T (2019) Scheduling IoT Healthcare Tasks in Fog Computing Based on their Importance. Procedia Comput Sci 163:560–569.
[26] Abdelaziz A, Elhoseny M, Salama AS, Riad AM (2018) A machine learning model for improving healthcare services on cloud computing environ-ment. Meas J Int Meas Confed 119:117–128.