Enhanced Quantum-Driven Bacterial Colony Optimization for Efficient Load Balancing in Cloud Computing

R M ARAVIND
Ph.D. Research Scholar
Department of Computer Science
Sri Vasavi College, Erode, India
Email id: aravi4909@gmail.com

Dr R. PRAGALADAN
Head & Associate Professor
Department of Computer Science
Sri Vasavi College, Erode, India
Email id: pragaladanr@gmail.com

Abstract : Effective load balancing is essential in the dynamic and resource-intensive world of cloud computing to guarantee optimal resource utilization, decreased latency, and improved service reliability. An enhanced BCO algorithm based on quantum mechanism (QBCO) is presented in this study to solve load balancing issues in cloud environments. In the suggested hybrid strategy, the improved search efficiency of quantum-inspired methods like superposition and quantum bit representation is combined with the exploratory and adaptable capabilities of BCO. The algorithm uses these quantum principles to avoid local optima, achieve faster convergence, and adapt dynamically to changing workloads. In terms of resource usage, response time, and energy economy, experimental results show that QBCO works better than conventional load balancing techniques. This creative method meets the increasing needs of high-performance and sustainable computing by offering a reliable and scalable solution for load distribution optimization in contemporary cloud infrastructures.

Keywords : QuantumComputing,Load Balancing,Bacterial Colony Optimization,Convergence Rate,Swarm Intelligence.

Introduction

The distribution of computer services, including as servers, storage, databases, networking, software, analytics, and intelligence, via the internet is known as "cloud computing" ("the cloud"). It gives customers more flexibility, scalability, and cost-effectiveness by allowing them to access and store data and apps on distant servers as opposed to local devices. Cloud computing has grown to be a crucial component of contemporary IT infrastructure, providing effective and scalable solutions for a range of uses. Cloud systems' dynamic nature, however, frequently results in unequal resource consumption, which can cause performance snags and lower service quality. The equitable distribution of workloads among available resources is ensured by load balancing, a crucial approach to address this difficulty. The bio-inspired optimization technique known as BCO is based on the collective behavior of bacteria, namely their capacity to adapt and maximize survival tactics amid shifting environmental conditions. BCO imitates these tendencies in computers to resolve intricate optimization issues, such as load balancing. Quantum-inspired techniques improve conventional optimization algorithms by utilizing concepts from quantum computing, such as entanglement and superposition[1]. These techniques enable more effective solution space exploration, which could lead to the discovery of superior solutions more quickly than with traditional methods. To enhance BCO's efficiency in load balancing tasks, the proposed QBCO incorporates quantum computing techniques into BCO.

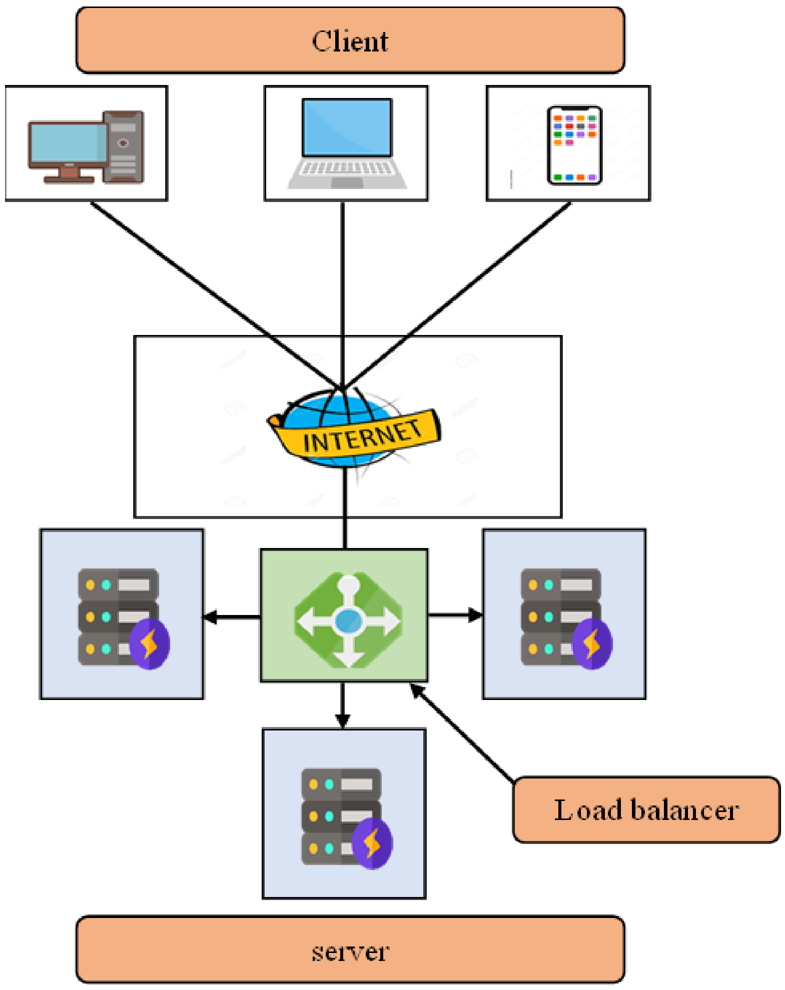

Figure 1 : Framework for Load balancing

The goal of this hybrid strategy is to improve the optimization process's accuracy, speed, and efficiency in cloud environments. A more optimal load distribution is ensured by the algorithm's ability to escape local optima and explore a larger solution space thanks to the quantum technique. QBCO offers real-time load balancing and can adjust to the dynamic workload fluctuations in cloud environments. The BCO algorithm's convergence is accelerated by the quantum-inspired method, which shortens the time needed to obtain an ideal or nearly ideal solution. The approach works effectively in large-scale cloud environments where resource availability and workload fluctuate regularly. The contributions of the research as follows,

- The study presents a new algorithm that uses quantum-inspired techniques to improve on the conventional BCO.
- Specifically, the load balancing issue in dynamic cloud systems is addressed by the suggested QBCO algorithm.
- The QBCO algorithm outperforms alternative metaheuristic algorithms and conventional load-balancing techniques, according to thorough simulation and testing.

1. Related works

Establishing the basis and importance of the suggested research is greatly aided by the related work section. An overview of current load balancing techniques in cloud computing, including both conventional and metaheuristic methods, is given in the related work section. S. Janakiraman et al. (2023) [2] develop a new load balancing method based on a Hybrid Grey Wolf and Improved PSO with Adaptive Intertial Weight-based multi-dimensional Learning Strategy (HGWIPSOA) for enhancing precision and rapidness in task scheduling and assignment of resources to Virtual Machines (VMs) in cloud environments. The Grey Wolf Optimization Algorithm (GWOA) is incorporated into PSO for considering the highest fitness particle as alpha wolf search agent, such that the objective of allocating tasks to VMs is attained effectively and efficiently. It then integrates chaos, Adaptive Inertial Weight (AIW) and Dimensional Learning (DL) into PSO specifically to prevent premature convergence and achieve better convergence speed and global search ability depending on the best experience determined by particles for effective LB. P. Pirozmand et al. (2023) [3] developed a new task scheduling method based PSO in order to shorten the execution time of the original PSO algorithm for task scheduling in the cloud computing environment, a multi-adaptive learning strategy is employed. In its initial population phase, the proposed Multi Adaptive Learning for PSO (MALPSO) defines two sorts of particles: ordinary particles and locally best particles. During this phase, the population′s variety is reduced and the likelihood of reaching the local optimum rises.

M. A. N. Saif (2023) [4] proposed an autonomic CSO-ILB load balancer to ensure the elasticity of the cloud system and balance the user workload among the available containers in a multi-cloud environment. The concept of multi-loop has been utilized in this approach to enabling efficient self-management before load balancing. The tasks are scheduled to the containers using an extended scheduling algorithm called Deadline-Constrained Make-span Minimization for Multi-Task Scheduling (DCMM-MTS). Based on the task scheduling, the load in each container is computed and then balanced using the proposed load balancer algorithm CSO-ILBT. Saba et al. (2023) [5] proposed secured data management with distributed load balancing protocol using PSO, which aims to decrease the response time for cloud users and effectively maintain the integrity of network communication. It combines distributed computing and shift high-cost computations closer to the requesting node to reduce latency and transmission overhead. Moreover, the proposed work also protects the communicating machines from malicious devices by evaluating the trust in a controlled manner. P. Neelakantan et al. (2023) [6] proposed a novel Load balancing methodology between VMs using the Hybrid Krill herd and Whale-based Deep Belief Neural model (HKHW-DBNM). The aims to improve the system's performance by balancing the Load between the VMs, optimizing the makespan, improving resource usage, reducing the degree of imbalance. M. Sumathi et al. (2023) [7] HHO and ACO are hybridized in the proposed technique. The factors that are analysed in the proposed system are the average waiting time (A_{WT}), average execution time (A_{ET}), average response time (A_{RT}), make-span, throughput analysis, turnaround time and LB time. By the allocation of workloads to VMs will shows which one gives the best efficiency in LB among the VM's. K. Ramya et al. (2023) [8] developed a new hybrid method based on dingo and whale optimization algorithm (WHA)-based load balancing mechanism (HDWOA-LBM) for effective load balancing those aids in maximized throughput, reliability, and resource utilization in the clouds. It utilized the merits of whale optimization for improving the exploitation phase of DOA to balance the trade-off between local and global search.

R. Kaviarasan et al. (2023) [9] developed new load balancing method based on improved lion optmizeation method. The developed method approach has better exploration and exploitation rate when compared with other method and it does not get struck into local optima during the search process of identifying the underutilized node. K. Malathi et al. (2023) [10] proposed new method based on lion optimizer to balance the loads by developing the optimal parameter selection for virtual machines. Two selection probabilities like task scheduling probability and virtual machine selection probability are developed for refining the selection procedure. Fitness criteria based on the task and the virtual machine properties are used for the lion optimizer. As the second contribution, a genetic algorithm is developed by modifying the global search criteria with relevance to the lion optimizer. M. A. Selvan (2024) [11] presents a comprehensive approach to load balancing using advanced optimization techniques integrated with multipath routing protocols. The primary focus is on dynamically allocating network resources to manage the massive volume of data generated by modern applications. By leveraging algorithms such as GA and PSO, the proposed method efficiently distributes data across multiple paths, ensuring balanced network utilization. The combination of these algorithms with multipath routing significantly reduces congestion and improves overall network performance. S. Karimunnisa et al .(2023) [12] proposed an improved density based clustering method (IDCM) for task scheduling approaches. The first is the preprocessing the user tasks for improved accuracy, classifying the tasks with respect to resource demand and execution time. The second phase deals with enhanced coot optimization algorithm for task scheduling (ECOA-TS) that proceeds and proves its novelty by adopting Cauchy mutation overcoming the convergence backdrop for generating an optimal mapping between clustered user tasks and VMs. G. Saravanan et al. (2023) [13] developed the improved scheduling efficiency algorithm using Wild Horse Optimization called IWHO for addressing the problems of lengthy scheduling time, high-cost consumption, and high virtual machine load in cloud computing task scheduling. First, a cloud computing task scheduling and distribution model is built, with time, cost, and virtual machines as the primary factors. Second, a feasible plan for each whale individual corresponding to cloud computing task scheduling is to fnd the best whale individual, which is the best feasible plan; to better find the optimal individual, used the inertial weight strategy for the IWHO to improve the local search ability and effectively prevent the algorithm from reaching premature convergence. S. S. Sefati et al. (2023) [14] proposes a new routing scheme with load-balancing capability using the Markov Model (MM) and the Artificial Bee Colony (ABC) algorithm. LEACH algorithm is used to maintain load balancing between Cluster Heads (CHs). Then the MM and the ABC called (MMABC) algorithm were used to find the best candidate nodes of each cluster to be turned into a CH. the proposed method surpasses the compared methods in terms of energy efficiency, number of alive nodes, and the number of delivered packets to BS and CH.

P. Suresh *et al. (2024)* [15] presents an optimized fault tolerant load balancing method using multi-objective cat swarm optimization (CSo) algorithm called MCSOFLB and the results are then compared against other powerful optimization algorithms. The experimental results evidently show that the proposed algorithm ranks first on the whole. The MCSOFLB method produces an average improvement of 31 % makespan, 6 % resource utilization, 12 % cost, 6 % success rate and 32 % average throughput over other benchmark algorithms. M. Menaka et al. (2024) [16] proposed a method to fully utilize virtual machines with a similar weight distribution, a strategy-oriented mixed support and load balancing structure has been developed in this work. To minimize make-span time and accomplish initial load balancing, the SPSO-TCS technique combines Time-Conscious Scheduling with Supportive PSO. Finding the optimal make span time minimization for each virtual environment is the aim of this stage. Its main objective is to discover the sequence of activities with the least computation time and to reduce the time required to finish each operation. Utilizing the hybrid idea leads to a decrease in makespan and the use of the least amount of energy. P. Geetha et al. (2024) [17] propose a novel optimal load balancing (LB) method in the cloud. For this a new Intercrossed Chimp and Bald Eagle Algorithm (IC&BA) algorithm is introduced that combines the algorithms like bald eagle search (BES) optimization and chimp optimization (CA) algorithm. Thereby, the proposed load balancing model works under the consideration of (i) Energy Consumption (ii) execution cost (iii) migration cost (iv) Make span and (v) load balancing parameters like response time, Turnaround time. R. Gowrishankar et al. (2024) [18] proposes a novel approach for load balancing using the Firefly Algorithm (FA). The algorithm is applied to dynamically distribute tasks among nodes in a distributed computing environment. The contribution lies in adapting the FA specifically for load balancing purposes in distributed computing systems. The study explores the effectiveness of this approach in improving system performance and resource utilization. The FA effectively redistributes tasks among nodes, reducing processing delays and improving overall system efficiency. S. Mohapatra et al. (2024) [19] proposed framework integrates Dragonfly (DF) and PSO algorithm. The performance of the proposed method is compared with PSO and Dragonfly algorithm. The performance is evaluated in different measures such as best fitness value, response time by varying the user base and response time. It is observed that the proposed method outperforms the other approached for load balancing. Z. R. Wang et al. (2024) [20] proposed SNSK-IPSO algorithm, which develops as a two-phases algorithm: enumerating all distributed solutions between VMs and tasks, finding the optimal solution through IPSO. It not only minimizes the execution time, but also improves resource utilization and load balance

2. Task scheduling

In cloud computing, mapping all jobs to accessible VMs and determining the best solution is a difficult issue. Thus, to distribute all user tasks to the appropriate resource and maintain VM load balance, a productive task scheduling method is needed. The direct importance of the tasks, resources, and framework structure is illustrated in this study. Each task must be assigned to a single VM (VM) to maximize machine usage and minimize makespan in cloud computing. This is the primary goal of the suggested technique. Suppose a cloud consists of x tasks such as $T = \{T_1, T_2, \ldots, T_X\}$ and y number of heterogeneous VMs such as: $VM = \{VM_1, VM_2, \ldots, VM_X\}$, but the condition for execution of such tasks is $x > y$. Every task has length T_1 and is communicated as MI (million instruction). The execution speed of y^{th} VM is showing as millions of instructions per second (MIPs) to achieve our goal and calculate the service rate of VM as follows,

$$\sum_{j-1}^{y} VM_{sr} = \sum_{j-1}^{y} VM_{mips} \times \sum_{j-1}^{y} VM_{cpu} \tag{1}$$

The expected execution time (EET) of task T_i, where $i = 1,2,..,x$ execute on VM VM_j where $j = \{1,2,..,y\}$ can be represented as follows

$$EET_{i,j} = \sum_{i=1}^{x} T_1 \times \sum_{j=1}^{y} VM_{sr} \tag{2}$$

A single VM (VM) can have one or more jobs in it. When the number of tasks in the VM (VM) increases, the overall file size and task dependency are displayed as follows:

$$T_{totalfilesize} = T_{individualfilesize} \times T_{dependency} \tag{3}$$

There's a chance that transferring a task between VMs will require some transfer time. The transmission of individual file sizes and the VM's bandwidth capacity determine the task transfer speed $TT_s(i,j)$. The formula for task transfer speed is as follows:

$$TT_s = \sum_{i=1}^{x} T_{individualfilesize} / \sum_{j=1}^{y} VM_{bw} \tag{4}$$

The duration required to move the job file size between VMs is known as the task transfer time. As a result, the time that task i transmits to VM j is represented by task transfer time $TT_t(i,j)$, and the task i communication time on machine time j is computed as follows:

$$CT_{i,j} = T_{totalfilesize} / TT_t \tag{5}$$

Lastly, as the following equation illustrates, the total execution time at every VM_j is the product of the expected task execution time and task transfer time.

$$ET_i = EET_i + TT_s + CT_{i,j} + DV_{i,j} \tag{6}$$

Where $DV_{i,j}$ is the binary decision variable which is defined as follows,

$$DV_{x,y} = \begin{cases} 1, & if\ T_i\ is\ allocated\ toVM_j \\ 0 & otherwise \end{cases} \tag{7}$$

When more activities are completed successfully in the data center, makespan time can decrease. This affects system throughput. The total completion time of each work is specified by makespan, which may be calculated using the following equation. The maximum of ET_i is the makespan (MS).

$$MS = \max \{ET_i\} \tag{8}$$

The goal of optimization is to increase resource utilization (R_u) as shown in the following equation. The calculation of resource consumption is derived from dividing the total execution time of all tasks by the highest makespan execution time. This is how the average utilization is displayed:

$$R_u = \frac{ET_i}{MS} \tag{9}$$

$$R_{avgu} = \frac{\sum_{j=1}^{y} R_u}{y} \tag{10}$$

The fitness function of the BCO algorithm is now defined as the following equation, which provides a particle with an optimal solution and a better location.

$$F = WT + MS + R_u \tag{11}$$

WT stands for waiting time. Now is the time to wait to assign the task to a separate VM.

1.1 VM availability

The following equations illustrate how many resources are used when $'x'$ number of tasks are allocated to $'y'$ number of resources.

$$VM_{use} = \sum_{i=1}^{x} T_i \tag{12}$$

1.2 VM selections

A VM's task allocation is contingent upon the task's arrival time. In other words, tasks with the earliest arrival times are allotted resources first, and shorter tasks will be completed sooner. The VM that is selected to complete the task depends on its capacity, current workload, and impending task length—all of which are determined using the following equation. VM capacity is based on available CPU, memory, bandwidth, and processing speed.

$$R_s = \frac{T_{i,l} * VML_j}{VM_{sr}} \tag{13}$$

All-time schedulers aim to assign all tasks to the available VMs, although this relies on the VM's load at that specific moment. Thus, determined the VM's load and capacity using the equation VML_j,

$$VML_j = \frac{T_i \times \sum_{i=1}^{x} T_1}{VM_{sr}} \tag{14}$$

$$VM_c = VM_{sr} + VM_{bw} \tag{15}$$

Algorithm 1: BCO algorithm

```
Step 1: Set up the required parameters
Step 2: For all bacterial colonies
Step 3: Chemotaxis and communication
Step 4: Reproduction and elimination
Step 5: Migration
Step 6: If the ultimate state is not reached, step 2 should be taken; if not, the process should
        be terminated.
Step 7: The final position is considered as the best position
```

1.3 VM state

There are three states for each VM: overload, underload, and balanced. A VM is considered underloaded if it uses less than 25% of its capacity, and overloaded if it uses more than 80% of its capacity. Equations that follow show that everything else is perfectly balanced. To ensure that the whole load is balanced, tasks are transferred from the overburdened VM to the understanding VM. The jobs are moved from a VM that is overloaded to one that is underloaded. Rather than using the VM migration strategy, our suggested algorithm made use of the task migration approach.

$$VM = \begin{cases} R_u < |VM_c \times 25\%|, & underload \\ R_u < |VM_c \times 80\%|, & overload \\ R_u = VM_c, & balanced \end{cases} \tag{16}$$

3. BCO

BCO is a population-based method planned by Niu et al. [21] The BCO is used in numerous real-world applications such as fuzzy clustering [22], data clustering [23-28], feature selection [29], disease detection[30], and scheduling [31]. To tackle the above problem, a novel bacterial algorithm called BCO was developed with SI characteristics to expedite the optimization process. stages of BCO include chemotaxis and communication, elimination and reproduction, migration, and the remaining four stages. The entire BCO process takes advantage of the chemotaxis and communication phase. The bacteria use the population statistics to modify their swimming and tumbling habits. To update the positions of the microorganisms, a unique chemotaxis and communication method is applied. Throughout their lives, bacteria can be divided into two types of chemotaxis: swimming and tumbling. A stochastic direction participates in the actual swimming process when tumbling. The combined effects of the best searching director in Tumbling and the turbulent director have an impact on the updated locations and search direction of each bacterium. These effects are expressed as follows:

$$Position_i(T) = Position_i(T-1) + C(i) * [f_i.(G_{best} - Position_i(T-1)) + (1-f_i) * (P_{best_i} - Position_i(T-1)) + turb_i] \quad (17)$$

However, bacteria lack a turbulence director to steer swimming toward an optimal state, which could be put as follows:

$$Position_i(T) = Position_i(T-1) + C(i) * [f_i.(G_{best} - Position_i(T-1)) + (1-f_i) * (P_{best_i} - Position_i(T-1))] \quad (18)$$

$$C(i) = C_{min} + \left(\frac{Iter_{max} - Iter_j}{Iter_{max}}\right)^n (C_{max} - C_{min}) \quad (19)$$

Where, $turb_i$- turbulent direction variance. $f_i \in \{0,1\}$, $C(i)$ - chemotaxis step size. The personal and global best value is denoted by $P_{best}$and G_{best} respectively. n is the linearly reducing way of the chemotaxis step. $Iter_{max}$, $Iter_j$ - maximum number of iterations and present iteration respectively. In the phase of elimination and reproduction, the sick bacterium will be replaced by the high-energy bacterium, which will replicate them to build the newest people. The high energy shows that the bacterium hunts for resources with remarkable efficiency. The bacterium can migrate within a certain search space range during the migration phase when certain conditions are met. During the migration phase, the bacteria typically go toward the most recent nutrients according to a specific likelihood.

4. Quantum mechanism

Quantum computing is a branch of computing that uses quantum physics to carry out calculations. Quantum computation makes use of quantum bits (qubits), as opposed to classical computation, which represents information using binary digits (bits). Superposition is the possibility that the qubits are in more than one state at once. This feature allows quantum computers to perform some tasks far more quickly than conventional computers [32]. A qubit's state may be represented as a vector in a two-dimensional complex vector space called a Bloch sphere. A qubit system consists of two states, $|0\rangle$ and $|1\rangle$. As mentioned earlier, a qubit may exist in a superposition of both $|0\rangle$ and $|1\rangle$ state, which is denoted by $|\Psi\rangle$ as follows,

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad (20)$$

where the probability amplitudes of states 0 and 1 are represented by the complex numbers α and β, respectively. An observation procedure is used to gauge a qubit's state. After the observation phase is over, a qubit's state can be identified as either 0 or 1. The probabilities that the qubit exists in states 0 and 1, respectively, are indicated by the modulus $|\alpha|^2$ and $|\beta|^2$, and both α and β should meet the normalization requirement.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{21}$$

Consequently, a qubit can be expressed as a pair of complex integers using the formula $q = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. Likewise, a multi-qubit system with a n number of qubits can be shown as

$$Q = [q_1, q_2, \dots . q_n] = \left[\begin{matrix} \alpha_1 \\ \beta_1 \end{matrix}\middle| \begin{matrix} \alpha_2 \\ \beta_2 \end{matrix}\middle| \begin{matrix} \cdots \\ \cdots \end{matrix}\middle| \begin{matrix} \alpha_n \\ \beta_n \end{matrix}\middle|\right] \tag{22}$$

A qubit's state can be updated via quantum gates such the NOT-gate, CNOT-gate, Hadamard-gate, etc. For a qubit q_i, the quantum angle can be expressed as follows:

$$\theta_i = arctan \frac{\beta_i}{\alpha_i} \tag{23}$$

5. Proposed quantum BCO

An improved form of the conventional BCO is the Quantum-Based BCO (QBCO) for load balancing in cloud computing. By using quantum-inspired methodologies to improve the optimization process, QBCO's main goal is to increase work allocation and resource utilization in cloud environments. In order to achieve optimal load balancing, this method seeks to improve local exploitation, global exploration, and convergence. The way bacteria forage serves as the model for the conventional BCO algorithm. It uses a population of bacteria (solutions) that move toward more advantageous areas of the solution space, reproduce, and mutate in order to improve iteratively. To help the bacterium better explore the search space and stay out of local optima, QBCO incorporates quantum-inspired concepts. In order to direct the investigation and exploitation of solutions, the quantum approach presents the idea of quantum behavior. A new and effective method for load balancing in cloud computing systems is the QBCO. BCO's advantages are combined with quantum-inspired methods in QBCO to optimize cloud system performance, balance loads, and enhance resource consumption.

6. Experimental results and analysis

Understanding the performance, scalability, efficiency, and efficacy of various load-balancing algorithms and techniques depends on the experimental results of cloud computing LB. Researchers and professionals can assess how well different load-balancing methods operate in actual or simulated cloud settings thanks to experimental data. They can evaluate the performance of various algorithms under various workload scenarios by contrasting parameters

Algorithm 2: Load balancing using QBCO

```
Step 1: Compute the EET, TTS, and CT
Step 2: Compute the CRU, Load, and capacity of the VM
Step 3: Check the load balancing possibility
Step 4: Check the availability of VM
Step 5: Set up the required parameters.
Step 6: For all bacterial colonies
Step 7: Chemotaxis and communication
Step 8: Reproduction and elimination
Step 9: Migration
Step 10: Updating the position using quantum mechanism
Step 11: Continue doing this until all loads are balanced and all tasks are assigned
appropriately.
```

Table 1 : Parameter settings for BCO

Parameter	Value	
Population	100	
Chemotaxis step	100	
Swimming step size	4	
Reproduction step	4	
Elimination step	2	
Elimination and dispersal probability	0.25	
C_{max} and C_{min}	0.12 and 0.5	

like reaction time, throughput, resource use, and system stability. An evaluation and comparison with other current approaches are conducted for the suggested BCO technique. The performance of QBCO is compared with some well-known methods such as BCO, BFO [33], CPSO[34], PSO [35], ACO [36], and GA [37]. The CloudSim is used to implement the suggested BCO. The method was configured with an Intel (R) Core (TM) i5-1235U running at 1.30 GHz and 8 GB of RAM. Windows 10 was the 64-bit operating system. The simulation's parameters are displayed below. This work is motivated by the desire to reduce makespan and increase resource usage in a dynamic environment. Table 1 lists the BCO algorithm's several features. Table 2 illustrates the non-pre-emptive tasks that are free that we have taken into consideration for the test and tasks are assigned to a variety of heterogeneous VMs within a data center.

4.1 Parameter settings

In BCO, the behavior and performance of the algorithm are largely determined by its parameters. Few parameters decide the performance of BCO and optimal parameter settings can help to achieve better performance in a given solution. The population size, chemotaxis and swimming step size, reproductive and elimination step, and its probability values are parameters that are considered in this work as per Tabe 1. To optimize BCO performance and strike a balance between exploration and exploitation, it is imperative to choose the right parameters.

4.2 Performance metrics

In cloud computing, performance metrics are essential to load balancing. By effectively distributing workloads among several servers, load balancing seeks to maximize resource utilization, reduce response times, and prevent any one server from becoming overloaded. A quantitative foundation for assessing and enhancing load balancing tactics is provided by metrics.

- **Makespan (MT):** Makespan measures the time span from the submission of the first task to the completion of the last task. An optimal load balancing strategy minimizes the makespan, ensuring that tasks are completed as quickly as possible. It provides insights into the overall efficiency and performance of the system, guiding the development and refinement of load balancing strategies to meet performance goals.
- **Execution time (ET):** Execution time is the amount of time needed for a particular job or process to finish running on a computer resource. This statistic is essential for assessing and improving load balancing tactics in cloud computing. In cloud computing, execution time metrics are crucial for efficient load balancing. They direct the allocation of tasks, optimize the use of resources, and boost system performance in general. Cloud systems can provide faster, more dependable, and more economical services by reducing execution times.
- **Resource utilization (RU):** Making sure that the computing resources in a cloud environment are utilized as efficiently as possible to complete activities and workloads is known as resource utilization in load balancing. By dividing up the workloads, efficient load balancing maximizes the utilization of available resources, cutting down on idle time, enhancing performance, and lowering expenses. It is another quantitative statistic that is

Table 2 : Properties Settings

Task	
Range of task	10 - 50
Length	1000 - 6000
Size of the file	500
VM	
VM ranges	3 - 5
Speed	225 - 300
Memory	256 - 512
CPU	1 - 5
Bandwidth	1000
VMM	XEN

dependent on the parameters. The objective is to enhance the effectiveness of employing the cloud environment's resources.

4.3 Results analysis

The results analysis offers empirical support for the claim that improved load balancing in cloud environments is achieved by the QBCO algorithm. Because it confirms the suggested QBCO algorithm, shows that it is better than current approaches, and offers a thorough understanding of its performance, the results analysis is essential to the study. This part lays the groundwork for next developments in cloud computing load balancing in addition to validating the research contributions. Virtual machines (VMs) are crucial parts of load-balancing systems in cloud computing settings. LB distributes incoming network traffic among several servers in order to boost throughput, reduce response times, optimize resource usage, and prevent overloading any one server. Because metaheuristic techniques offer efficient algorithms for optimizing resource allocation, decreasing response times, and improving overall system performance, they are crucial for LB in cloud computing systems. In the metaheuristic study of LB in cloud computing, the performance of the implemented algorithms is evaluated in terms of several parameters, including as resource consumption, reaction time, throughput, scalability, and fault tolerance. Examine how well each method distributes the load and allocates resources in different workload circumstances. Examine how effectively the LB algorithm divides up CPU, memory, and bandwidth among virtual machines in order to meet the demands of incoming requests. Determine how much resource the system is using too little or too much, and assess how well the algorithm performs in terms of resource optimization. The current section examines the capabilities of QBCO approaches, which are one of the novel LB techniques that were the subject of the current paper.

Tables 3 and 4 and figures 2 and 3 are shows the performance of load balancing algorithms based on Makespan value obtained for 3 and 5 VMs, respectively. Tables 5 and 6 and figures 4 and 5 are shows the performance of load balancing algorithms based on Resource utilization obtained for 3 and 5 VMs, respectively. Tables 7 and 8 and figures 6 and 7 are shows the performance of load balancing algorithms based on Execution time obtained for 3 and 5 VMs, respectively. Tables 9 and 10 and figures 8 and 9 are shows the performance of load balancing algorithms based on Memory utilization obtained for 3 and 5 VMs, respectively. The effectiveness and efficiency of the method are determined by a number of crucial indicators when assessing QBCO performance for load balancing in cloud computing. Among these are makespan, execution time, memory usage, and resource usage.

Table 3 : Makespan value obtained for 3 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	292	323	347	368	391	412	428
20	368	397	425	458	487	508	524
30	482	529	558	571	610	641	665
40	697	715	733	759	788	805	824
50	835	869	881	897	910	934	955

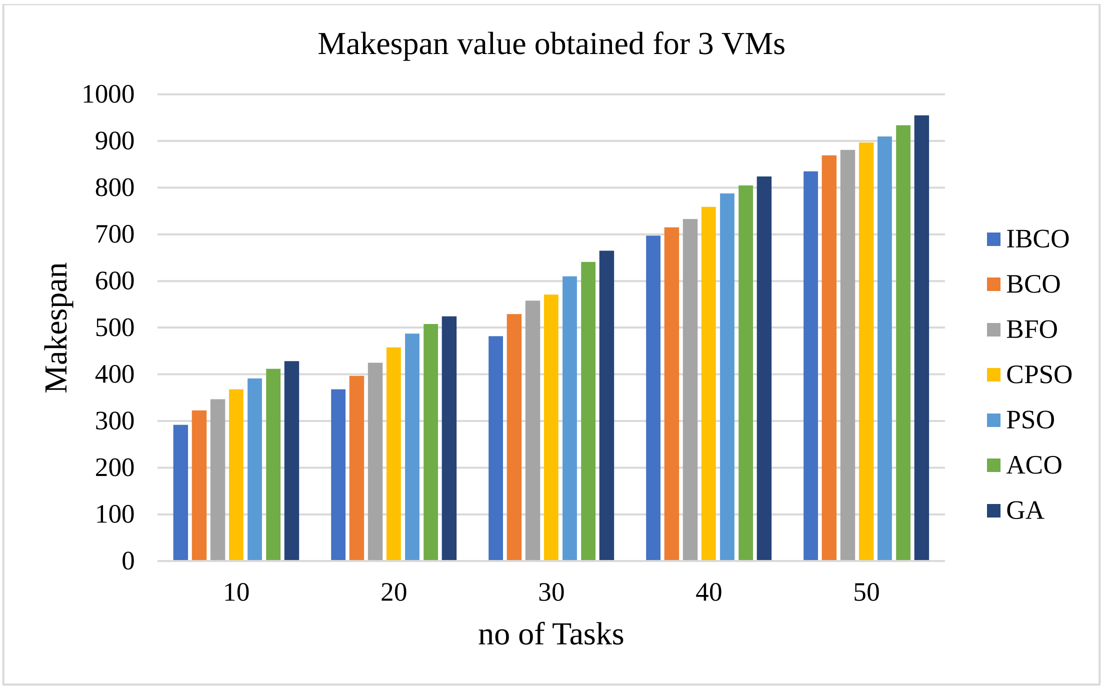

Figure 2: Makespan value obtained for 3 VMs

Table 4 : Makespan value obtained for 5 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	145	174	210	239	248	267	285
20	310	338	364	397	420	443	463
30	487	505	531	558	579	607	620
40	628	657	688	704	725	749	762
50	788	805	824	854	865	887	898

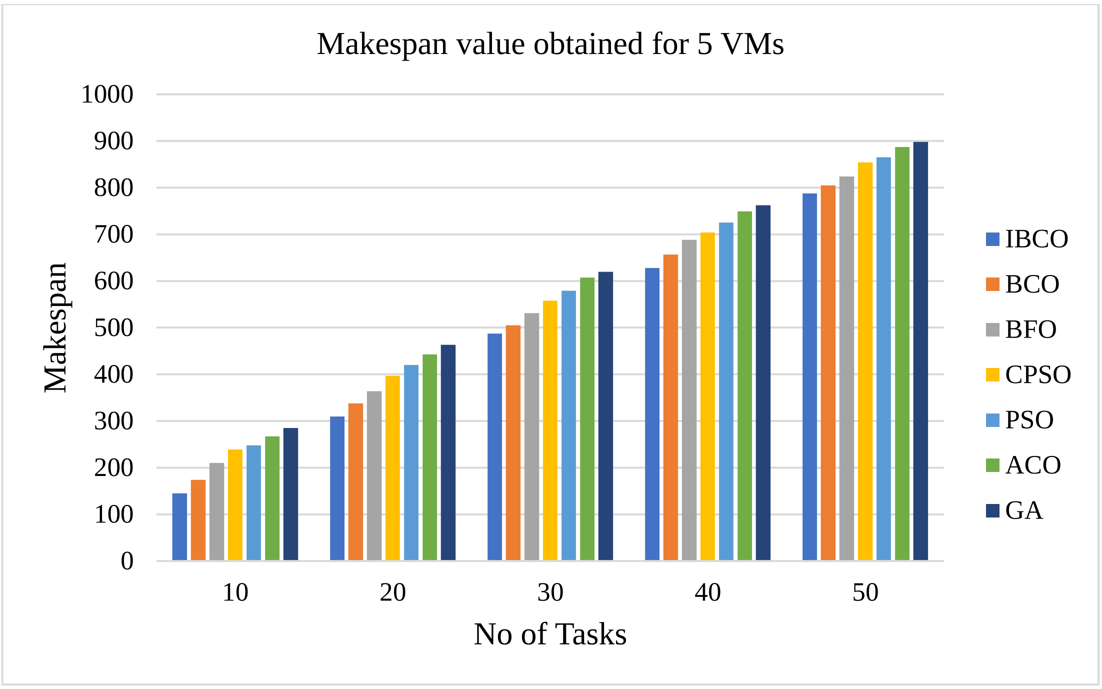

Figure 3 : Makespan value obtained for 3 VMs

Table 5 : Resource utilization obtained for 3 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	0.70	0.62	0.42	0.38	0.32	0.29	0.27
20	0.75	0.64	0.58	0.51	0.45	0.39	0.31
30	0.83	0.73	0.68	0.61	0.55	0.49	0.43
40	0.85	0.83	0.71	0.61	0.53	0.44	0.37
50	0.88	0.86	0.78	0.69	0.58	0.49	0.41

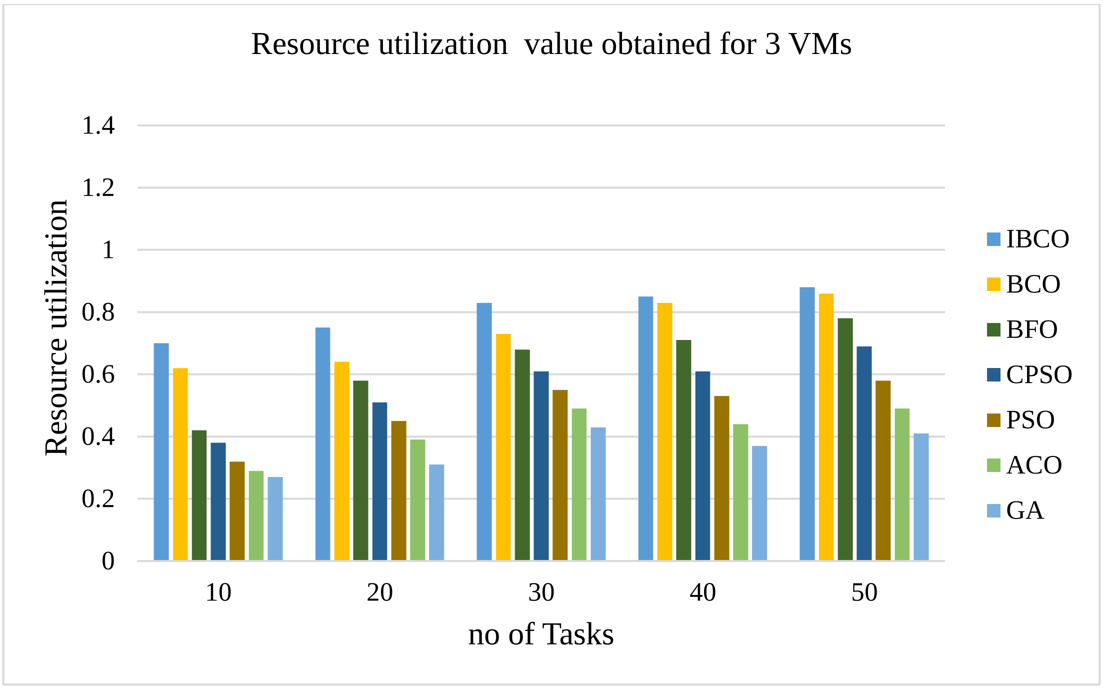

Figure 4 : Resource utilization value obtained for 3 VMs

Table 6 : Resource utilization obtained for 5 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	0.60	0.48	0.43	0.37	0.30	0.25	0.20
20	0.89	0.70	0.63	0.58	0.41	0.35	0.24
30	1.06	0.98	0.86	0.76	0.66	0.52	0.48
40	1.11	1.09	0.95	0.88	0.71	0.64	0.59
50	1.32	1.29	1.11	1.01	0.92	0.85	0.71

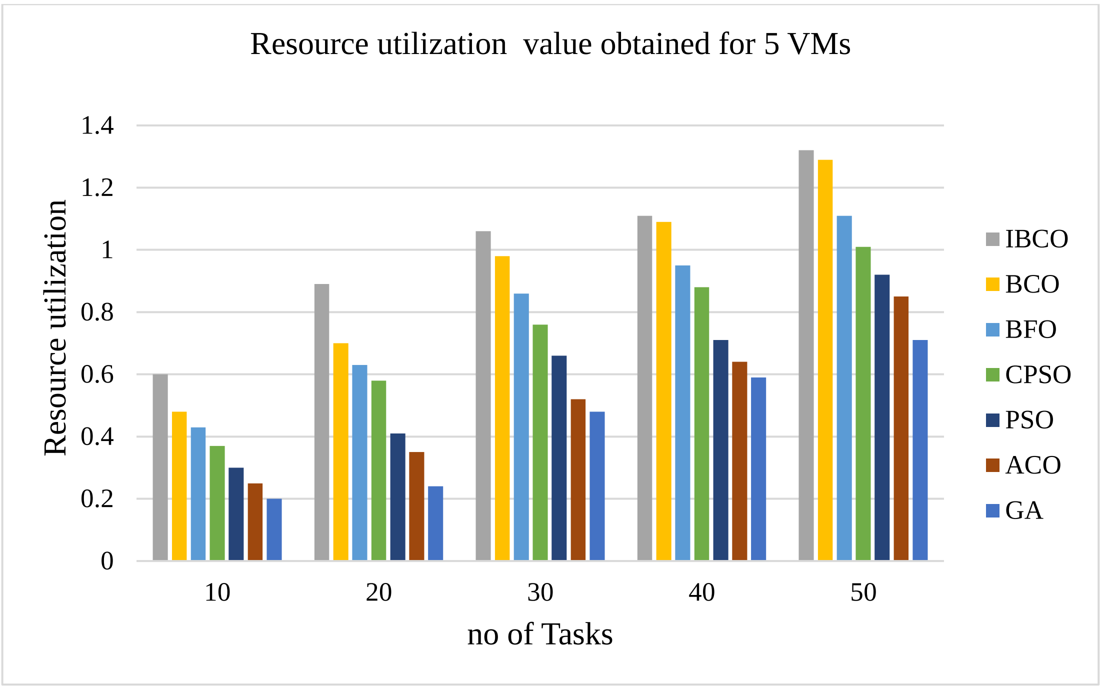

Figure 5 : Resource utilization obtained for 5 VMs

Table 7 : Execution time obtained for 3 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	115.65	131.46	148.16	159.11	164.11	175.22	190.41
20	159.29	168.95	179.33	188.93	197.29	208.22	215.95
30	168.26	179.40	184.67	192.10	214.25	226.63	238.59
40	181.42	197.40	211.33	239.85	258.67	273.42	298.72
50	224.24	237.11	256.71	272.30	288.32	302.30	312.72

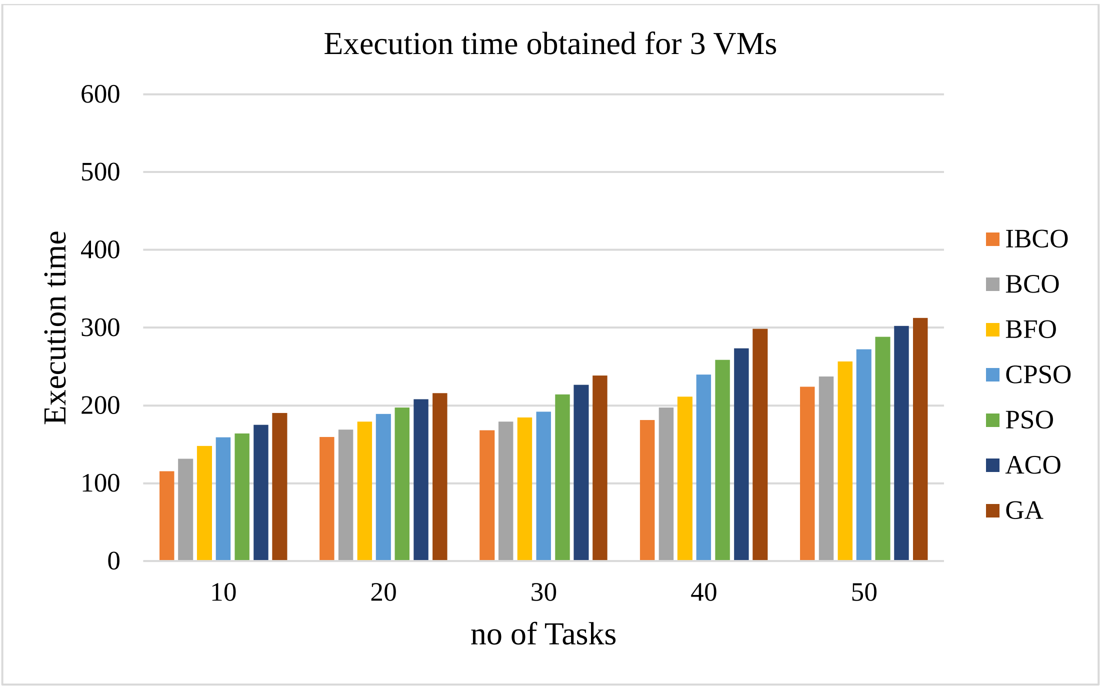

Figure 6 : Execution time obtained for 3 VMs

Table 8 : Execution time obtained for 5 VMs

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	131.46	147.56	159.18	171.54	180.46	191.88	204.54
20	169.54	175.63	209.67	225.20	247.62	269.28	289.42
30	321.72	348.26	372.56	416.34	437.48	463.72	475.26
40	357.12	394.84	420.61	431.86	452.75	478.66	498.20
50	370.52	398.43	425.06	457.34	488.41	514.63	537.14

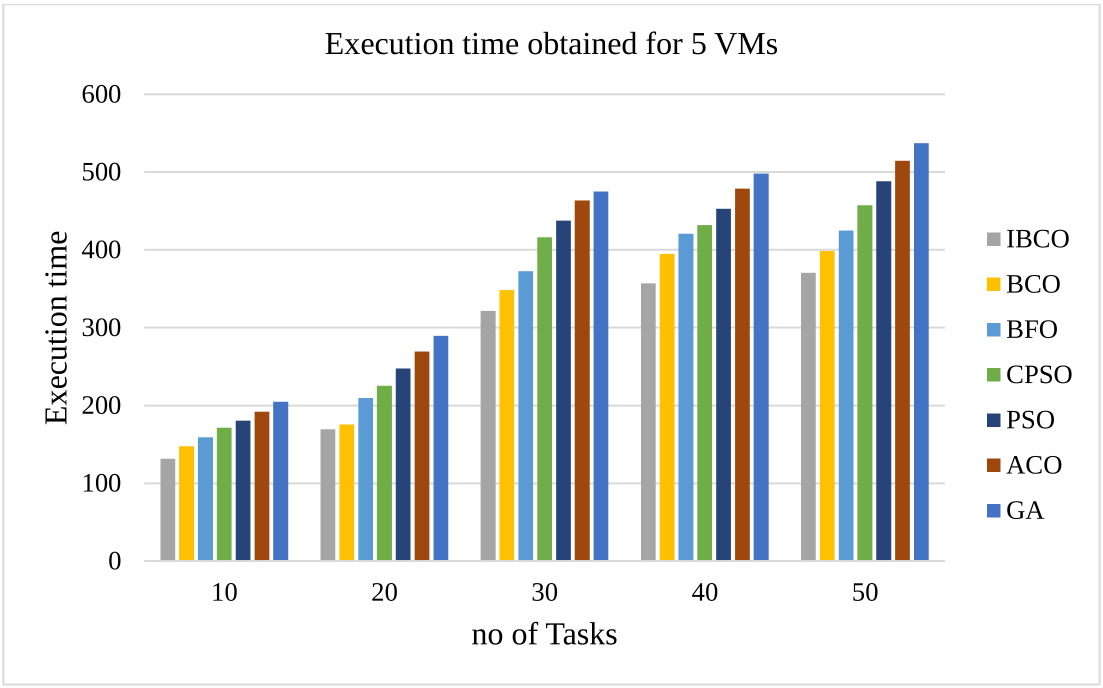

Figure 7 : Execution time obtained for 5 VMs

Table 9 : Memory utilization obtained for 3 VMs in percentage (%)

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	48	53	53	55	57	62	66
20	50	51	55	58	62	65	70
30	54	57	59	63	69	72	76
40	59	61	66	70	76	83	84
50	62	65	68	76	80	85	89

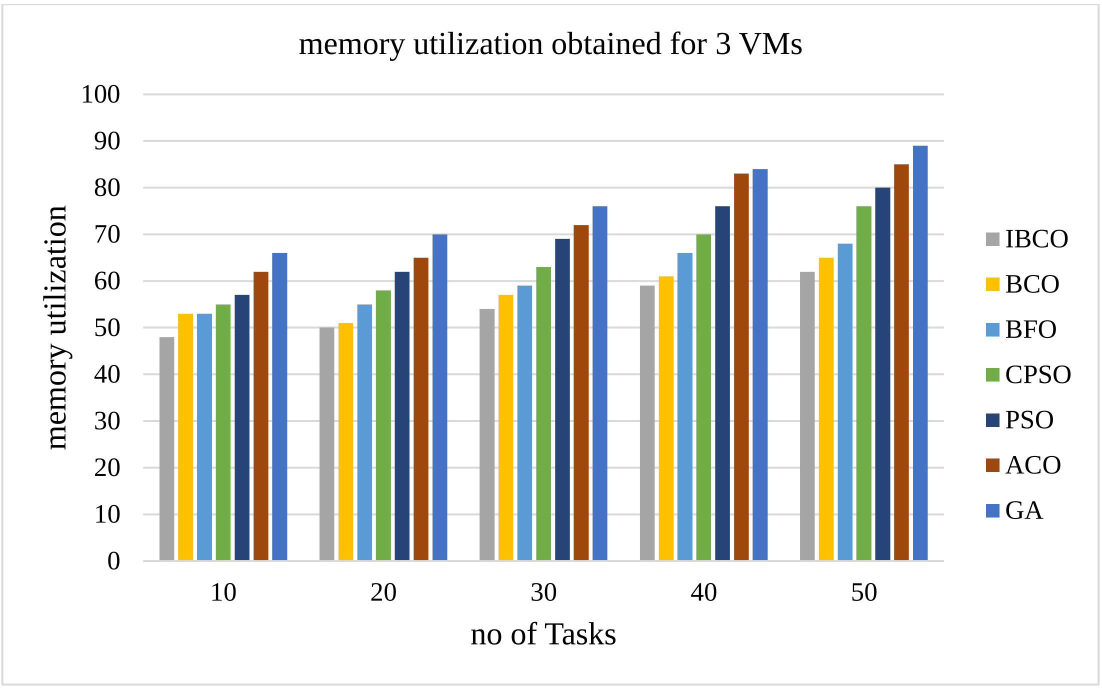

Figure 8 : Memory utilization obtained for 3 VMs in percentage (%)

Table 10 : Memory utilization obtained for 5 VMs in percentage (%)

Tasks	QBCO	BCO	BFO	CPSO	PSO	ACO	GA
10	51	52	55	60	66	71	78
20	55	58	64	71	76	80	83
30	60	64	67	76	81	88	92
40	64	70	78	83	88	92	95
50	71	77	79	82	86	92	97

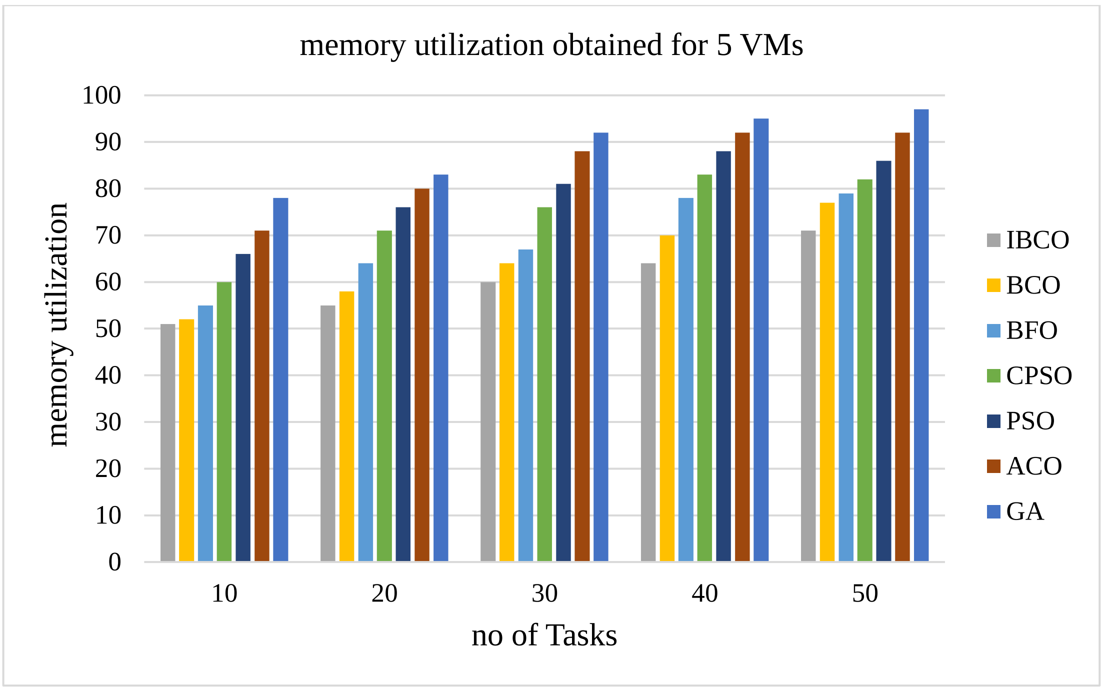

Figure 9 : Memory utilization obtained for 5 VMs in percentage (%)

7. Conclusions

This study addressed the crucial problem of load balancing in cloud computing systems by presenting an QBCO algorithm based on quantum techniques. The suggested approach successfully blends the exploration efficiency of quantum computing principles with the advantages of BCO. Superior load distribution across cloud resources was shown using the QBCO algorithm. As a result, duties were distributed more fairly, avoiding underuse and resource overload. QBCO is appropriate for real-time dynamic applications since it has demonstrated scalability across several cloud environments and adaptability to varying workloads. QBCO proceeded better than other metaheuristics and conventional algorithms, according to a comparative investigation. Better global search capabilities and faster convergence were made possible by its quantum-inspired improvements.

References

[1] J. Balicki, "Many-objective quantum-inspired particle swarm optimization algorithm for placement of virtual machines in smart computing cloud," *Entropy,* vol. 24, no. 1, p. 58, 2021.

[2] S. Janakiraman and M. D. Priya, "Hybrid grey wolf and improved particle swarm optimization with adaptive intertial weight-based multi-dimensional learning strategy for load balancing in cloud environments," *Sustainable Computing: Informatics and Systems,* vol. 38, p. 100875, 2023.

[3] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing,* vol. 14, no. 4, pp. 4313-4327, 2023.

[4] M. A. N. Saif, S. Niranjan, B. A. H. Murshed, F. A. Ghanem, and A. A. Q. Ahmed, "CSO-ILB: chicken swarm optimized inter-cloud load balancer for elastic containerized multi-cloud environment," *The Journal of Supercomputing,* vol. 79, no. 1, pp. 1111-1155, 2023.

[5] T. Saba, A. Rehman, K. Haseeb, T. Alam, and G. Jeon, "Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence," *Cluster Computing,* vol. 26, no. 5, pp. 2921-2931, 2023.

[6] P. Neelakantan and N. S. Yadav, "An optimized load balancing strategy for an enhancement of cloud computing environment," *Wireless Personal Communications,* vol. 131, no. 3, pp. 1745-1765, 2023.

[7] M. Sumathi, N. Vijayaraj, S. P. Raja, and M. Rajkamal, "HHO-ACO hybridized load balancing technique in cloud computing," *International Journal of Information Technology,* vol. 15, no. 3, pp. 1357-1365, 2023.

[8] K. Ramya and S. Ayothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment," *Transactions on Emerging Telecommunications Technologies,* vol. 34, no. 5, p. e4760, 2023.

[9] R. Kaviarasan, G. Balamurugan, and R. Kalaiyarasan, "Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm," *e-Prime-Advances in Electrical Engineering, Electronics and Energy,* vol. 6, p. 100326, 2023.

[10] K. Malathi and K. Priyadarsini, "Hybrid lion–GA optimization algorithm-based task scheduling approach in cloud computing," *Applied Nanoscience,* vol. 13, no. 3, pp. 2601-2610, 2023.

[11] M. A. Selvan, "Multipath Routing Optimization for Enhanced Load Balancing in Data-Heavy Networks," 2024.

[12] S. Karimunnisa and Y. Pachipala, "Task Classification and Scheduling Using Enhanced Coot Optimization in Cloud Computing," *International Journal of Intelligent Engineering & Systems,* vol. 16, no. 5, 2023.

[13] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," *Journal of Cloud Computing,* vol. 12, no. 1, p. 24, 2023.

[14] S. S. Sefati, M. Abdi, and A. Ghaffari, "QoS-based routing protocol and load balancing in wireless sensor networks using the markov model and the artificial bee colony algorithm," *Peer-to-Peer Networking and Applications,* vol. 16, no. 3, pp. 1499-1512, 2023.

[15] P. Suresh *et al.*, "Optimized Task Scheduling Approach with Fault Tolerant Load Balancing using Multi-Objective Cat Swarm Optimization for Multi-Cloud Environment," *Applied Soft Computing,* p. 112129, 2024.

[16] M. Menaka and K. S. Kumar, "Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing," *International Journal of Cognitive Computing in Engineering,* vol. 5, pp. 192-198, 2024.

[17] P. Geetha, S. Vivekanandan, R. Yogitha, and M. Jeyalakshmi, "Optimal load balancing in cloud: Introduction to hybrid optimization algorithm," *Expert Systems with Applications,* vol. 237, p. 121450, 2024.

[18] R. Gowrishankar, B. Senthilkumar, E. Jananandhini, and D. Ramasamy, "SWARM INTELLIGENCE APPROACH FOR LOAD BALANCING IN DISTRIBUTED COMPUTING SYSTEMS USING FIREFLY ALGORITHM," *ICTACT Journal on Soft Computing,* vol. 14, no. 4, 2024.

[19] S. Mohapatra, S. Mohanty, H. K. Nayak, M. K. Mallick, J. V. N. Ramesh, and K. V. Dudekula, "DPSO: A Hybrid Approach for Load Balancing using Dragonfly and PSO Algorithm in Cloud Computing Environment," *EAI Endorsed Transactions on Internet of Things,* vol. 10, 2024.

[20] Z. R. Wang, X. X. Hu, P. Wei, and B. Yuan, "An improved particle swarm optimization algorithm for scheduling tasks in cloud environment," *Expert Systems,* vol. 41, no. 7, p. e13529, 2024.

[21] B. Niu and H. Wang, "Bacterial colony optimization: principles and foundations," in *Emerging Intelligent Computing Technology and Applications: 8th International Conference, ICIC 2012, Huangshan, China, July 25-29, 2012. Proceedings 8*, 2012: Springer, pp. 501-506.

[22] K. Vijayakumari and V. Baby Deepa, "Fuzzy C-means hybrid with fuzzy bacterial colony optimization," in *Advances in electrical and computer technologies: select proceedings of ICAECT 2020*, 2021: Springer, pp. 75-87.

[23] V. Prakash, V. Vinothina, K. Kalaiselvi, and K. Velusamy, "An improved bacterial colony optimization using opposition-based learning for data clustering," *Cluster Computing,* vol. 25, no. 6, pp. 4009-4025, 2022.

[24] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Bacterial colony optimization for data clustering," in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2019: IEEE, pp. 1-4.

[25] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Hybrid data clustering approaches using bacterial colony optimization and k-means," in *IOP Conference Series: Materials Science and Engineering*, 2021, vol. 1070, no. 1: IOP Publishing, p. 012064.

[26] K. Tamilarisi, M. Gogulkumar, and K. Velusamy, "Data clustering using bacterial colony optimization with particle swarm optimization," in *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2021: IEEE, pp. 1-5.

[27] S. S. Babu and K. Jayasudha, "A Simplex Method-Based Bacterial Colony Optimization for Data Clustering," in *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2021*: Springer, 2022, pp. 987-995.

[28] S. S. Babu and K. Jayasudha, "A simplex method-based bacterial colony optimization algorithm for data clustering analysis," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 36, no. 12, p. 2259027, 2022.

[29] H. Wang, L. Tan, and B. Niu, "Feature selection for classification of microarray gene expression cancers using Bacterial Colony Optimization with multi-dimensional population," *Swarm and Evolutionary Computation,* vol. 48, pp. 172-181, 2019.

[30] S. İlkin, T. H. Gençtürk, F. K. Gülağız, H. Özcan, M. A. Altuncu, and S. Şahin, "hybSVM: Bacterial colony optimization algorithm based SVM for malignant melanoma detection," *Engineering Science and Technology, an International Journal,* vol. 24, no. 5, pp. 1059-1071, 2021.

[31] B. Niu, T. Xie, Y. Bi, and J. Liu, "Bacterial colony optimization for integrated yard truck scheduling and storage allocation problem," in *Intelligent Computing in Bioinformatics: 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings 10*, 2014: Springer, pp. 431-437.

[32] M. Bey, P. Kuila, B. B. Naik, and S. Ghosh, "Quantum-inspired particle swarm optimization for efficient IoT service placement in edge computing systems," *Expert Systems with Applications,* vol. 236, p. 121270, 2024.

[33] S. T. Milan, L. Rajabion, A. Darwesh, M. Hosseinzadeh, and N. J. Navimipour, "Priority-based task scheduling method over cloudlet using a swarm intelligence algorithm," *Cluster Computing,* vol. 23, no. 2, pp. 663-671, 2020.

[34] T. Prem Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wireless Personal Communications,* vol. 109, no. 1, pp. 315-331, 2019.

[35] M. Kumar and S. C. Sharma, "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustainable Computing: Informatics and Systems,* vol. 19, pp. 147-164, 2018.

[36] R. Gao and J. Wu, "Dynamic load balancing strategy for cloud computing with ant colony optimization," *Future Internet,* vol. 7, no. 4, pp. 465-483, 2015.

[37] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing," *Procedia Technology,* vol. 10, pp. 340-347, 2013.