International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/49 Volume 11, Issue 6, November – December (2024)

RESEARCH ARTICLE

Energy-Aware Optimization of Cloud Request Placement and Resource Monitoring Using an Evolutionary Algorithm for Cloud-assisted Systems

Santosh Kumar Paul
Faculty of Science (FOS), Sri Sri University, Cuttack, Odisha, India.
santoshkumarpaul22@gmail.com

Sunil Kumar Dhal
Faculty of Science (FOS), Sri Sri University, Cuttack, Odisha, India.
sunildhal@srisriuniversity.edu.in

Rakesh Nayak
Department of Computer Science and Engineering, School of Engineering, OP Jindal University, Raigarh, CG, India.
rakesh.nayak@opju.ac.in

Umashankar Ghugar
Department of Computer Science and Engineering, School of Engineering, OP Jindal University, Raigarh, CG, India.
✉ ughugar@gmail.com

Received: 08 August 2024 / Revised: 19 November 2024 / Accepted: 30 November 2024 / Published: 30 December 2024

Abstract – The exponential growth of data generated by various aspects of life, particularly through internet-enabled devices, has introduced significant challenges in processing such data within strict time constraints. Cloud computing has emerged as a potential solution due to its ability to handle heterogeneous, energy-constraint, and non-cooperative data. However, the task scheduling problem in cloud computing, being NP-hard, demands efficient solutions that balance system performance and energy consumption. Current methods often fail to address the imbalanced system loads and fluctuating cloud requests effectively, leading to increased energy usage and degraded performance. This paper tackles these challenges by proposing an energy-efficient load balancing strategy coupled with an optimized cloud requests placement method. Task scheduling is approached using the binary chaotic Jaya (BCJaya) algorithm, which leverages evolutionary techniques to ensure high performance. The proposed algorithm is evaluated against key metrics, including Makespan, virtual machine (VM) utilization, energy consumption, and load balancing rate. Additionally, the BCJaya algorithm's efficacy is demonstrated using a real-world benchmark dataset and is compared against established baselines. The results show that BCJaya consistently outperforms alternative methods, particularly in scenarios involving increasing tasks and VMs, making it a robust solution for cloud scheduling challenges.

Index Terms – Cloud Requests Placement, Resource Monitoring, Task Scheduling, Resource Monitoring, Chaotic Jaya, Cloud Computing.

1. INTRODUCTION

In today's era of pervasive digitalization, every aspect of human life is increasingly dependent on computational systems and internet-enabled devices. This dependence has led to an exponential increase in data generation, making effective management of computational resources a critical necessity. Cloud computing has emerged as a cornerstone for addressing these challenges, offering scalable, on-demand access to resources for storing, processing, and managing data. However, the rapid growth of cloud computing has also raised significant concerns about energy consumption, resource optimization, and environmental impact. These challenges are magnified by the heterogeneous, latency-sensitive, and dynamic nature of workloads, necessitating innovative solutions for resource management and energy efficiency.

Request placement and resource monitoring are essential components in cloud computing, significantly influencing the performance and efficiency of cloud systems. Poor task placement and ineffective resource monitoring can result in inefficient resource utilization, higher energy consumption, elevated operational costs, and a decline in Quality of Service (QoS). Addressing these issues requires energy-aware optimization techniques that not only improve resource

RESEARCH ARTICLE

utilization but also minimize energy expenditure. Cloud computing has transfigured the business horizon as to how computational resources are delivered and consumed. It enables organizations to access computing power without significant upfront investments, promoting agility, flexibility, and cost savings. However, these benefits come with challenges, especially in terms of energy efficiency. A typical cloud datacenter houses thousands of servers, networking devices, and cooling systems, all of which contribute to substantial energy consumption. Studies suggest that datacenters globally consume approximately 1-2% of the world's total electricity, and this figure is expected to grow as the demand for cloud services increases.

Task consolidation and resource management are now feasible choices as a result of the development of cloud computing which offers end users on-demand services over the Internet based on a pricing structure. Cloud computing, which provides virtualization techniques for the dynamic scheduling of cloud requests, appears to be a popular solution. Cloud requests must be handled promptly to reduce average waiting time and execution time while effectively increasing resource utilization. To make this feasible, heuristic algorithms or standard scheduling cannot yield the best results. The exponential growth of cloud computing can be attributed to its dynamic environment, scalability, customization, and on-demand access to computing resources. The abstraction of virtual resource management, which makes the technical complexities easier to understand, is one of its main benefits.

1.1. Problem Statement

Effective task scheduling or cloud request placement for users requesting services is necessary for efficient resource monitoring in cloud computing [1-4]. When there are few tasks and resources, scheduling is easy; however, it becomes difficult when different user demands call for different levels of service quality. Task consolidation and resource monitoring, which are considered NP-hard problems, in cloud computing is challenging because of the heterogeneous and dynamic nature of datacenters, which makes it difficult to solve NP-hard problems. Metaheuristic-based scheduling techniques have proven to be more effective than traditional and heuristic approaches [5, 6]. These non-deterministic techniques, like meta-heuristic algorithms, have proven to be appropriate and perform satisfactorily when used to solve NP-hard optimization problems [7].

Numerous advantages come with cloud computing, including fewer maintenance requirements, increased flexibility, and consistent quality because of outside management. However, finding the best mapping algorithm to allocate resources among cloud requests is necessary for optimizing the resource utilization (NP-hard [8]) rate. Improved results are obtained by expanding the search space with population-based algorithm-based metaheuristics. Withstanding the advantages of such algorithms, these metaheuristics produce better results. Because scheduling strategies guarantee that the cloud requests are distributed among compatible virtual machines (VMs), they are critical to effective cloud computing solutions. One major problem that impacts QoS, performance, and user experience is the placement of cloud requests on suitable VMs. There is a dearth of research on resolving these issues with the JAYA variant available in the literature [9]. It is very difficult to get the best solution for placement and resource monitoring problems in a given amount of time when competing metrics are involved and tasks are dynamically assigned to different resources. A heterogeneous environment is defined by a fluctuating number of tasks and virtual machines, which makes it difficult to verify the algorithm's effectiveness in both homogeneous and heterogeneous environments. Metaheuristic techniques can yield approximate optimal solutions within predictable timeframes, according to prior research [8].

1.2. Motivation

Several challenges complicate the implementation of energy-aware optimization in cloud computing environments: Task Heterogeneity: Cloud requests vary in terms of computational requirements, priority, and deadlines, making uniform optimization strategies ineffective. Dynamic Resource Availability: Resource availability changes dynamically due to varying workloads, hardware failures, and maintenance activities. VM Consolidation Overheads: While consolidating VMs onto fewer servers can save energy, excessive VM migrations can introduce overheads, negating potential savings. Trade-offs Between QoS and Energy Efficiency: Ensuring high QoS while minimizing energy consumption requires balancing conflicting objectives, particularly for latency-sensitive applications. NP-Hard Nature of Task Scheduling: Task scheduling in cloud environments is inherently NP-hard, meaning that finding an optimal solution requires significant computational effort. To address these challenges, this study introduces an evolutionary approach to optimize cloud request placement and resource monitoring. The proposed method integrates Binary Chaotic Jaya (BCJaya) optimization to address the limitations of existing methods. BCJaya, an advanced metaheuristic algorithm, combines chaotic maps with the Jaya algorithm to enhance exploration and exploitation capabilities in the solution space. The research focuses on addressing critical challenges such as:

- Dynamic Workload Adaptability: Ensuring that the optimization framework can adapt to fluctuating workloads and resource availability.
- Energy-Aware Task Scheduling: Incorporating energy metrics into the scheduling process to minimize power consumption without compromising QoS.

RESEARCH ARTICLE

- Load Balancing: Achieving a balanced workload distribution across active servers to prevent resource bottlenecks and underutilization.
- Reduction of VM Migration Overheads: Minimizing the frequency and cost of VM migrations to achieve energy efficiency without excessive overheads.

Using an evolutionary approach, this research attempts to address cloud request placement on the underlying VMs with effective resource monitoring throughout the system. In particular, the goal of the study is to improve the traditional Jaya algorithm by introducing a variation called Chaotic Jaya (CJaya) to speed up convergence and produce ideal solutions. Additionally, a binary version of the CJaya (BCJaya) algorithm is designed to display the produced solution within a range of 0 and 1. The primary motive for using the Jaya metaheuristic approach for this problem is its simplicity and power to strike a balance between local and global optima. Both the best and worst solutions are estimated in a single equation by considering very few control parameters which makes it more powerful and efficient than others. Particle swarm optimization (PSO) is a commonly used and accepted standard method among various metaheuristic algorithms [10]. Nevertheless, PSO fails to overcome local optima as a global search technique, despite its advantages. As the number of tasks within the problem domain increases, it also faces challenges related to early convergence [11]. In contrast, CJaya demonstrates a high convergence rate and does not get trapped in local optima due to its simultaneous consideration of both the best and worst solutions in a single equation [12]. Another popular and highly advantageous optimization method is genetic algorithms (GA). However, because there are a lot of control parameters in GA, it has a high computational cost. On the other hand, because solutions are updated using a single equation in a single step, BCJaya has low computational complexity and requires less time. Furthermore, BCJaya preserves a trade-off in the problem space between particle exploration and exploitation capacity. BCJaya reduces computational time because it doesn't require additional control parameter tuning, unlike GA and PSO.

1.3. Contributions

To summarize, the following is a list of the major contributions:

- To optimise the scheduling and resource monitoring, and accomplish efficient load balancing, the Binary Chaotic Jaya algorithm is proposed in conjunction with a resource monitoring strategy,
- Taking into account a fitness function to protect the interests of the user and the cloud service provider, making sure the goals of load balancing, resource usage, and energy consumption are maintained,
- Running simulations in environments in a heterogeneous to show how effective the suggested algorithm is,
- To measure the scheduling metrics and validate the algorithm's effectiveness, a benchmark dataset is used.

1.4. Flow of the paper

The remaining sections of the document are arranged as follows: Section 2 reviews the current state of relevant research. The various system models are shown in Section 3, which is followed by the problem formulation. The proposed methodology is explained in Section 4, with special attention to the Binary Chaotic Jaya algorithm and the resource monitoring strategy. In Section 5, test cases and dataset utilization are experimentally evaluated. A comparative analysis with existing algorithms is provided, and a discussion follows. The paper is finally concluded with some future perspectives in Section 6.

2. RELATED WORK

Cloud data centers strive for energy efficiency, often achieved through dynamic virtual machine (VM) consolidation. By dynamically consolidating VMs onto fewer active servers, power consumption is significantly reduced. However, server workloads fluctuate, necessitating frequent consolidation adjustments. The following research papers propose metaheuristic or evolutionary-based approaches to solve task mapping problem considering various constraints. For instance, Mishra and Majhi [13] proposed a hybrid load-balancing approach integrating Genetic Algorithm (GA) with Jaya optimization to efficiently schedule dynamic medical data in cloud systems for biomedical applications. GA generates the initial population of solutions, while Jaya optimization identifies the most suitable virtual machines (VMs) for executing these solutions. This integration leverages the strengths of both algorithms, resulting in improved scheduling and load-balancing efficiency. However, the proposed method does not account for thermal aspects or fluctuating workloads, which limits its applicability to dynamic and real-time scenarios. The research also lacks consideration of energy-aware metrics and dynamic VM consolidation, leaving room for further exploration in these areas. Zahedi et al. [14] introduced a "thermal-aware consolidation" technique to address energy efficiency and workload distribution in cloud datacenters. The model categorizes servers based on their energy efficiency and prioritizes VM placement on highly efficient servers. This approach effectively reduces energy consumption and mitigates thermal challenges by preventing hotspots. Despite these advantages, the method does not adequately address dynamic workloads or task prioritization. The lack of consideration for Quality of Service (QoS) requirements and workload variability in dynamic cloud environments highlights an important research gap that future studies could

RESEARCH ARTICLE

address. Zhang et al. [15] developed a reservation-based VM allocation strategy using an evolutionary algorithm to optimize server energy consumption. The approach strategically allocates VMs based on their instruction-to-energy ratios, achieving significant energy savings. However, the study primarily focuses on energy efficiency and does not sufficiently address task heterogeneity or workload imbalance. Additionally, the method overlooks real-time workload fluctuations and dynamic task scheduling, which are critical for enhancing cloud resource management under varying conditions. Mishra et al. [16] proposed a dynamic load scheduling approach for Infrastructure-as-a-Service (IaaS) cloud ecosystems using the binary Jaya algorithm integrated with a load-balancing technique. This approach minimizes the number of active servers, ensuring efficient task distribution and improved energy efficiency. However, the model is limited by excessive VM migrations and the absence of thermal management considerations. Moreover, the study lacks an energy-aware framework and overlooks thermal-aware consolidation techniques, which could enhance its applicability in dynamic cloud environments. Llager et al. [17] tackled the issue of "blind consolidation" in dynamic VM consolidation, where excessive VM migrations can negate energy savings due to high migration overheads. Their energy and thermal-aware consolidation model ensures balanced workload distribution while mitigating hotspots. This approach effectively addresses energy efficiency and thermal challenges. However, it does not consider task deadlines or QoS constraints, which limits its applicability in scenarios requiring strict service-level agreements (SLAs). The lack of focus on resource heterogeneity further highlights areas for improvement. Azizi et al. [18] proposed a two-phase VM migration algorithm to address resource imbalance in powered-on servers. The algorithm relocates VMs from overloaded servers to efficient, powered-off servers, thereby improving resource utilization and overall system efficiency. While this method enhances resource distribution, it incurs high overhead due to frequent VM migrations. The study also lacks applicability to heterogeneous environments and dynamic task scheduling, which are essential for real-world cloud datacenter operations. Yavari et al. [19] introduced a hybrid approach combining heuristics and metaheuristics to address multi-dimensional constraints such as CPU, memory, and temperature during VM consolidation. This method improves performance by targeting VM migrations and balancing resources effectively. However, its scalability is limited when applied to large cloud systems. Additionally, the study does not fully optimize energy consumption or QoS requirements, leaving a gap in achieving comprehensive efficiency in cloud resource management. Abdessamia et al. [20] explored energy-efficient VM placement using the Binary Gravitational Search Algorithm (BGSA). The algorithm guides VMs toward high-efficiency physical machines, enhancing energy savings. Despite its benefits, the method primarily focuses on energy efficiency without addressing thermal metrics or QoS constraints. Furthermore, the absence of dynamic scheduling and workload heterogeneity limits its practical application in diverse cloud environments. Abualigah et al. [21] proposed a hybrid Differential Evolution-AntLion Optimization (DE-ALO) algorithm for multi-objective task scheduling in cloud environments. The method maximizes resource utilization and minimizes Makespan, demonstrating significant performance improvements. However, the high computational complexity of the hybrid approach makes it unsuitable for real-time or large-scale scenarios. Additionally, the study does not address dynamic workloads or energy efficiency, which are critical for adaptive cloud systems. Mishra and Majhi [22] introduced a load-balancing approach based on Binary Bird Swarm Optimization (BBSO) for cloud computing environments. This method effectively balances system loads and improves resource utilization. However, the approach is limited to homogeneous environments with a small number of tasks, reducing its significance for real-world, large-scale, and heterogeneous cloud systems. The absence of a focus on energy efficiency and dynamic task scheduling further limits its utility in modern cloud datacentre. The summary of these related works is presented in Table 1.

Table 1 Summary of Related Works

Authors (Reference)	Methodology Used	Advantages	Limitations	Research Gaps
Mishra & Majhi [13]	GA + Jaya optimization	Efficient scheduling and load balancing	Ignores thermal aspects	No dynamic VM consolidation or energy metrics
Zahedi et al. [14]	Thermal-aware consolidation	Reduces energy and addresses thermal issues	Limited dynamic workload handling	No QoS or workload variability consideration

RESEARCH ARTICLE

Zhang et al. [15]	Reservation-based VM allocation using evolutionary algorithm	Optimizes energy via instruction-to-energy ratios	Ignores task heterogeneity and workload imbalance	Lacks dynamic task scheduling
Mishra et al. [16]	Binary Jaya algorithm with load balancing	Minimizes active servers	Excessive VM migrations, no thermal management	No energy-aware or thermal-aware consolidation
Llager et al. [17]	Energy and thermal-aware VM consolidation	Prevents hotspots	Lacks task deadline handling	No QoS or resource heterogeneity consideration
Azizi et al. [18]	Two-phase VM migration	Improves resource utilization	High VM migration overhead	Limited to homogeneous environments
Yavari et al. [19]	Hybrid heuristics/metaheuristics	Balances CPU, memory, and temperature	Limited scalability	Does not optimize energy consumption or QoS
Abdessamia et al. [20]	Binary Gravitational Search Algorithm	Enhances energy efficiency	Ignores thermal and QoS metrics	No dynamic scheduling or workload heterogeneity
Abualigah et al. [21]	Hybrid DE-ALO	Maximizes resource utilization, minimizes Makespan	High computational complexity	Lacks dynamic workloads or energy efficiency
Mishra & Majhi [22]	Binary Bird Swarm Optimization	Balances system loads	Tested on homogeneous environments	Limited to small-scale, homogeneous environments

This research focuses on the integration of a resource monitoring strategy and a requests placement technique to achieve both an equitable workload distribution and an optimal task mapping to virtual machines (VMs) for better resource utilization. Workloads are divided among the VMs by the Broker using a particular load balancing technique. It also uses the proposed BCJaya-based scheduling algorithm to figure out how to allocate tasks to virtual machines in the most efficient way. Iteratively distributing the workload and optimizing resource utilization, this process keeps going until a balanced state is achieved through a compatible–based resource monitoring strategy for an increasing number of tasks on heterogeneous VMs.

3. SYSTEM MODELS AND PROBLEM FORMULATION

The Cloud-assisted scheduling and resource monitoring framework is depicted in Figure 1. This figure illustrates a Cloud computing framework designed for efficient task scheduling and resource allocation using the Binary Chaotic JAYA (BCJaya) optimization algorithm. The architecture is centred around a Cloud datacenter, which acts as the computational backbone for executing user tasks. The framework incorporates various components that interact to manage resources effectively, optimize task placement, and meet Quality of Service (QoS) requirements.

The Cloud users represent the end-users who submit tasks to the cloud. These tasks are sent via a request and response channel, which facilitates communication between the users and the cloud system. Once a task is submitted, it is passed through the broker, an intermediary responsible for managing task requests and responses between users and the datacenter.

Within the Cloud datacenter, the heart of the system lies in the computational framework, where the BCJaya algorithm

RESEARCH ARTICLE

operates. The framework begins with resource monitoring and load balancing, which tracks the status of physical resources and ensures that tasks are distributed evenly across the infrastructure. Tasks submitted by users are temporarily stored in a task buffer, awaiting scheduling based on their priority and resource requirements. The BCJaya algorithm is used for both initial scheduling and final placement of tasks, focusing on optimizing factors such as execution time, energy efficiency, and resource utilization. The framework also evaluates QoS metrics, which guide decision-making to meet user expectations for performance and reliability.

The system relies on a pool of heterogeneous resources, which includes physical machines with varying computational capabilities, such as servers, desktops, and mobile devices. These resources execute the tasks assigned by the scheduling framework. A dedicated Cloud Information Service (CIS) provides real-time metadata about resource availability, capacity, and status, enabling informed decisions during the scheduling process.

Finally, the placement and scheduling module determines the most suitable physical machines for executing tasks, ensuring optimal use of the infrastructure while adhering to constraints like workload balancing and energy efficiency. Once tasks are completed, the results are transmitted back to the users through the broker.

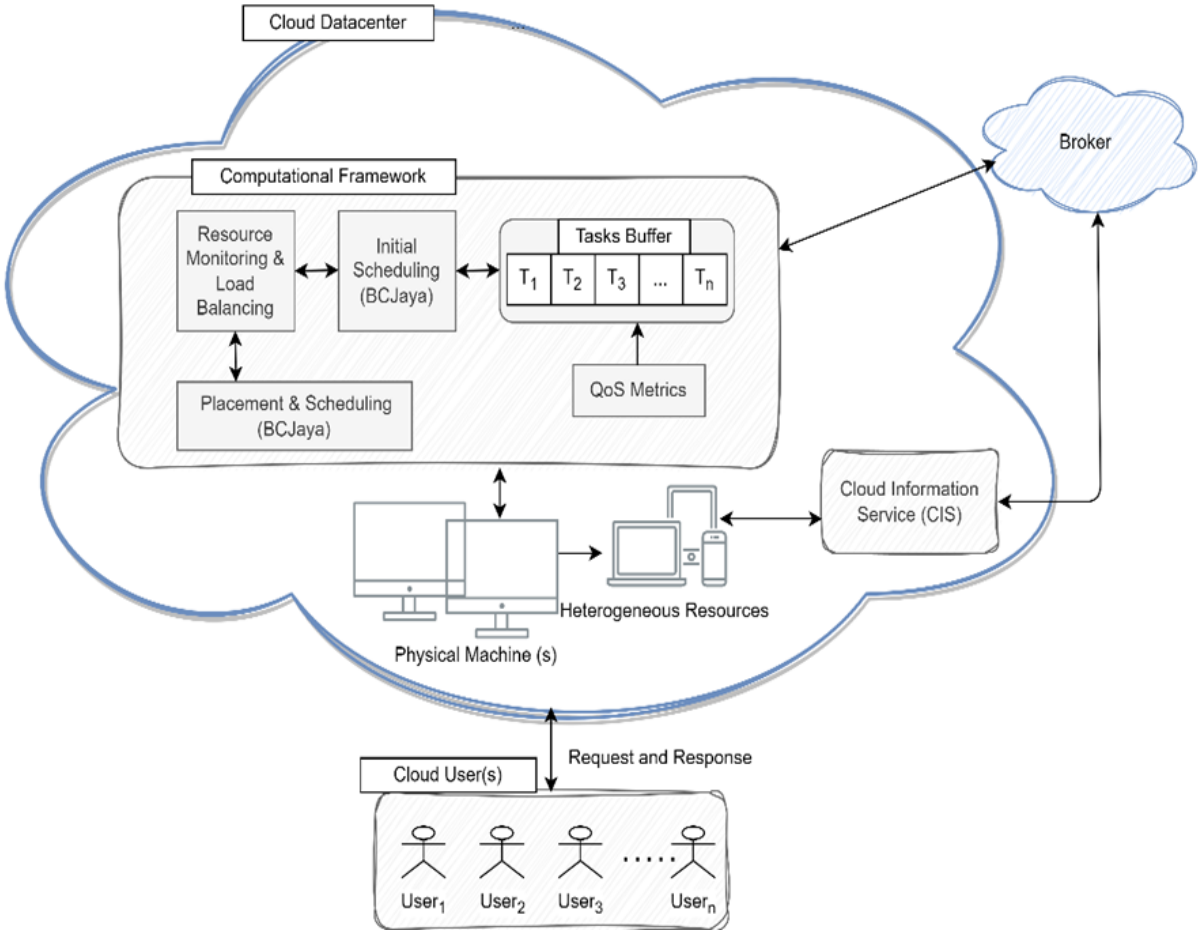

Figure 1 Cloud-Assisted Request Placement and Resource Monitoring Framework

This architecture ensures an efficient, scalable, and QoS-aware cloud computing environment. By leveraging the BCJaya algorithm for scheduling and load balancing, it achieves optimal resource utilization and maintains high performance while meeting the dynamic demands of cloud users.

3.1. Task-VM Model

A tremendous number of tasks is generated from the internet-enabled devices by cloud users on a daily basis which requires to be processed by computationally-rich and resource-rich nodes. The generated tasks are of disparate nature in terms of

RESEARCH ARTICLE

size, specifications, resource requirements, bandwidth requirements, speed, processing cores required to get executed, etc. Therefore, the tasks are expressed in million instructions (MI) and are assumed to be independent in nature. These are denoted as $T = \{T_1, T_2, \dots, T_i, \dots, T_N\}, i \in N$ and are characterised in terms of size (l), processing core (c), memory (m), and storage (s) expressed as $\langle T_i^l, T_i^c, T_i^m, T_i^s \rangle$. Likewise, the computationally-rich nodes are deployed in the Cloud layer for processing these requests. The Cloud layer encompasses a series of hosts deployed in a rack of servers. Furthermore, a number of VMs are created under a series of hosts through virtualization technology. The VMs are further classified as homogeneous and heterogeneous in nature. Homogeneous in nature, we mean, the technical specifications of all the VMs are the same whereas different in a heterogeneous environment. The VMs are expressed as $VM = \{M_1, M_2, \dots, M_J, \dots, M_{VM}\}, j \in VM$ and are characterised in terms of processing core, speed in MIPS, memory in GB, storage in GB and bandwidth denoted as $\langle M_j^c, M_j^m, M_j^s, M_j^{BW} \rangle$. This research considers a dynamic range of tasks and heterogeneous VMs for the simulation and validation of the effectiveness of the proposed strategy.

3.2. Scheduling Parameters

These parameters are also called Quality of Service (QoS) parameters. These parameters hold significance in appraising the performance of the proposed algorithm over others.

MAKESPAN (Makespan): it is the maximum execution time of all the allocated services for each fog node, and is expressed as shown in Equation (1) [23]:

$$\text{Makespan} = \max \sum_{i=1}^{N} \text{Exe}_t(M_j) \quad (1)$$

The execution time of the j^{th} VM $\left(\text{Exe}_t(M_j)\right)$ is the time required to process all the cloud requests on a VM and is computed as in Equation (2):

$$\text{Exe}_t(M_j) = \sum_{T_k^j \epsilon M_j} \left(\text{Proc}_t^j + \text{Prop}_t^j\right) \quad (2)$$

Where

$$\text{Proc}_t^j = \frac{L(T_k^j)}{\rho(M_j) \times \text{CPU}_{\text{rate}}(M_j)} \quad (3)$$

$$\text{Prop}_t^j = \frac{D((x_1,x_2)(y_1,y_2))}{3 \times 10^8} \quad (4)$$

Where Proc_t^j is the processing time of the j^{th} VM, Prop_t^j is the propagation time for sending an cloud request from an internet-enabled device to the j^{th} VM through distance $D(\,)$, $L(T_k^j)$ is the number of instructions in the k^{th} request, $\rho(M_j)$ is the number of core present in the j^{th} VM, and $\text{CPU}_{\text{rate}}(M_j)$ is the processing capability of the j^{th} VM.

VM UTILIZATION (Util_M): It's the level of virtual machine (VM) usage. Minimising the makespan is the aim of load balancing, which aims to optimise resource utilisation. On average Equation (5) is used to determine the average utilisation of all virtual machines (VMs), where M is the total number of VMs [16].

$$\text{Util}_M^{avg} = \frac{\sum_{j=1}^{M} \text{Exe}_t(M_j)}{\text{makespan} \times M} \quad (5)$$

DEGREE OF IMBALANCE (doi): The gauge for determining task imbalances among VMs is the degree of imbalance [22]. Equations (6) and (7) are used to measure it, where Texe_i^{max} and Texe_i^{min} represent the highest and lowest execution times of task T_i across all VMs. Moreover, Texe_i^{avg} represents the average execution time. The task's length is represented by Len, the number of cores in the j^{th} VM is denoted by $M_j^{\#c}$, and a total MIPS assigned to the j^{th} VM is represented by M_j^{MIPS}.

$$\text{doi} = \frac{\text{Texe}_i^{max} - \text{Texe}_i^{min}}{\text{Texe}_i^{avg}} \quad (6)$$

$$T_i = \frac{\text{Len}}{M_j^{\#c} \times M_j^{MIPS}} \quad (7)$$

3.3. Energy Consumption Model

An additional objective of this study is to optimize resource utilization while minimizing energy consumption. Efficient resource utilization can be achieved by ensuring that only the necessary processors are active, while others remain idle or powered down. When services are distributed across multiple processing nodes, it often leads to some nodes being underutilized, consuming energy inefficiently. Idle nodes can still use approximately 30-40% of their peak energy consumption. Therefore, to reduce the overall energy consumption of virtual machines (VMs) in the cloud, it is crucial to schedule services effectively. The energy usage of the j^{th} VM and the total energy consumed by all VMs in the datacenter are mathematically represented by Equations (8-9) [23].

$$E_C^{M_j} = \text{Exe}_t(M_j) \times \text{active}_j + \left(\text{Makespan} - \text{Exe}_t(M_j) \times \text{idle}_j\right) \quad (8)$$

$$E_C^{Total} = \sum_{j=1}^{M} E_C^{M_j} \quad (9)$$

3.4. Problem Formulation Model

Assume a problem matrix with $N \times d$ -dimension $\{X = \{X_{11}, X_{22}, X_{33}, \dots, X_{ij}, \dots, X_{nm}\}, X_{ij} \{T \in (1,2,3, \dots, i, \dots, N); M \in (1,2,3, \dots, j \dots, M)\}\}$ represents each particle in the problem space. The objective of this research is to map the tasks or cloud requests to compatible VMs

RESEARCH ARTICLE

optimally to achieve the following objectives: (1) minimize makespan (Makespan) and maximize average VM utilization ($\text{Util}_{\text{M}}^{\text{avg}}$) shown in Equation (10), (2) minimize the degree of imbalance (doi) shown in Equation (11), and (3) minimize total energy consumption ($\text{E}_{\text{C}}^{\text{Total}}$) shown in Equation (12). Each objective is stated with respect to the constraints in the following equations.

$$\mathbb{O}_1 = \min_{T_{n,i}^{m} M_{n,i}^{m}} \sum \frac{1}{\text{Makespan}} \times \text{Util}_{\text{M}}^{\text{avg}}, \quad m \in M, \quad n \in T; \tag{10}$$

$$\mathbb{O}_2 = \min_{M_{n,i}^{m}} \sum \text{doi}, \quad m \in M, \quad n \in T; \tag{11}$$

$$\mathbb{O}_3 = \min_{M_{n,i}^{m}} \sum \text{E}_{\text{C}}^{\text{Total}}, \quad m \in M; \tag{12}$$

Hence, the objective function considering minimization and maximization of makespan and VM utilization (Equation (10)), minimization of the degree of imbalance (Equation (11)), and minimization of Energy consumption (Equation (12)) has been formulated as shown in Equation (13):

$$\mathbb{O} = \mathbb{O}_1 \times \alpha + \mathbb{O}_2 \times \beta + \mathbb{O}_3 \times \gamma \tag{13}$$

subject to:

$\text{C1}: \alpha + \beta + \gamma = 1;$

$\text{C2}: \sum_{i \in T} \sum_{j \in M} 0 \leq X_{i,j} \leq 1;$

Equation (10) indicates that a datacenter's makespan should be minimised and resource utilization to be maximized, contingent on the required durations of tasks and execution times of individual virtual machines. Though the makespan and resource utilization are associated in a reverse linear relationship, it is considered in one equation. The minimization of the system's degree of imbalance is discussed in Equation (11). In order to lessen the severity of imbalance, this goal makes sure that loads are distributed evenly amongst VMs. Equation (12) shows that resource utilisation should be maximised. The weighted average of each sub-objective is shown in Equation (13).

The primary objective ($\mathbb{O}$) will have a balancing factor of one, according to Constraint (C1). Many weight values are used in the simulation; it was found that α=0.5, β=0.25, and γ=0.25 are the most significant values. It is crucial to remember that it takes a lot of work to figure out the exact weight values. After we conducted tests and tracked our results using several weight values, the precise weight values were taken into account. Constraint C2 states that the optimal mapping of tasks to virtual machines (VMs) should be represented by either 0 or 1. It is a binary variable.

4. PROPOSED STRATEGY

This section explains the proposed methodology implemented to carry out this research. The proposed strategy is a two-fold mechanism. First, a mapping of tasks to VMs would be initiated based on the resource monitoring. If the system is identified as imbalanced, the resource monitoring algorithm would be triggered to redistribute the workloads among VMs based on resource availability and resource adaptability. In addition, the tasks scheduling algorithm called BCJaya (Binary Chaotic Jaya) would be executed for scheduling the tasks among VMs. As a result, the optimal mapping of tasks into VMs would take place resulting in a reduced makespan, energy consumption, degree of imbalance and improved VM utilization. In this regard, sub-section 4.1 presents the cloud request placement strategy (task scheduling) using a binary chaotic Jaya (BCJaya) algorithm. Next, the load balancing and resource monitoring strategy is presented in sub-section 4.2.

4.1. Cloud Requests Placement Strategy Using BCJaya

Jaya is a population-based metaheuristic optimization algorithm used to address constrained and unconstrained optimization problems [16]. This algorithm consists of two entities, such as particles and food sources. Likewise, our cloud requests placement strategy consists of tasks as particles and VMs as food sources. In the Jaya-based algorithm, each particle competes among itself to get into the compatible food source. Analogously, the tasks compete among themselves to get into a compatible VM in our problem domain. The position of each particle is updated in each iteration and so are tasks. The fitness of each particle is estimated according to the fitness value, and the best and worst positions are estimated through a single equation in the Jaya algorithm. The fitness of each task is evaluated through a defined fitness function in Equation (13). The position of each particle is updated through the following Equation (14).

$$X_i^{t+1} = X_i^t + r_1(B_i - |X_i^t|) - r_2(W_i - |X_i^t|) \tag{14}$$

Here, X_i^t and X_i^{t+1} are the current and the updated positions of the particle i at t and $t+1$ iterations, respectively, r_1 and r_2 are arbitrary numbers between 0 and 1, B_i and W_i are the best and worst solutions of the particle i.

However, the standard Jaya suffers from slow convergence and consequently, local entrapment of the particles may take place. Therefore, the chaotic JAYA (CJAYA) theory of chaos serves as the foundation for this work. This technique is used to overcome the drawback of being slower at the JAYA algorithm's standard convergence rate. This is intended to hasten exploration without becoming ensnared in local optima. The standard JAYA and the CJAYA operate on similar principles. Because of CJAYA, an arbitrary, chaotic number generator is used to create the random numbers that make up the initial population. Because of its simplicity, the tent map function is used as a chaotic random number generator in this study rather than other chaotic map functions. Equation (15) is used to express it.

RESEARCH ARTICLE

$$X_i^{t+1} = \begin{cases} \frac{X_i^t}{0.7}, & X_i^t < 0.7 \\ \frac{10}{3}(1 - X_i^t), & X_i^t \geq 0.7 \end{cases} \quad (15)$$

Note. X_i^t is the previous random number and X_i^{t+1} is the newly generated chaotic random number. Since the initial value affects the drifting pattern of some of the chaotic maps, it is set to 0.7.

Initially, the continuous optimization problem is the focus of the standard JAYA and its variations. Additionally, this has been enhanced to address more dynamic optimization issues, such as cloud computing task scheduling, that are limited to 0s and 1s, which requires the solutions to be converted to binary. Therefore, the tangent hyperbolic logistic transfer function is used [16]. It is represented by Equations (16) and (17).

$$\tanh\left(\left|X_i^{k+1}\right|\right) = \frac{e^{\left(\left|2X_i^{k+1}\right|\right)-1}}{e^{\left(\left|2X_i^{k+1}\right|\right)+1}} \quad (16)$$

$$X_i^{k+1} = \begin{cases} 1, & if\ rand() < \tanh\left(\left|X_i^{k+1}\right|\right) \\ 0, & otherwise \end{cases} \quad (17)$$

The pseudo-code for the proposed BCJaya placement algorithm is presented in Algorithm 1.

BEGIN

1. Initialize *T* and *M* set, Particles' position using Chaos map theory Eq. 15;

2. LOOP

for 1: N

Estimate the fitness of each task using Equation (13);

$if(\mathbb{0} < B_i)$

Update the task's new fitness as the B_i;

End;

End;

for 1: n

Initiate a new placement by estimating the position with Equation (14);

Modify the tasks' position through binary solutions with Equation (16) and Equation (17);

end;

Till the maximum iteration is reached;

3. Optimal placement of tasks to VMs is accomplished;

END

Algorithm 1 Cloud Requests Placement Strategy Using BCJaya

4.2. Load Balancing and Resource Monitoring Strategy

This strategy estimates the workloads and capacities of all the underlying VMs. By comparing the capacity with workload, it estimates the state of the VM and categorises them into one of the groups Overutilized VM (OVM), Underutilized VM (UVM) and Normalized VM (NVM). Next, the used resources and available resources of all the UVMs are estimated and then the tasks of OVMs are evaluated against each UVM to check the adaptability for the offloading to bring a balance across the total workloads in all the VMs. These are delineated as follows:

Step 1: estimate the doi and check if the system is balanced or not. If not, then trigger the load balancing operation as follows.

Step 2: estimate the capacity of each VM and the capacity of all the VMs.

The capacity of a VM (Equation (18)) is defined as the maximum workload handled by a VM and is expressed with respect to its attributes. The capacity of all the VMs is denoted in Equation (19).

$$Cap(M_j) = M_j^c \times M_j^m \times M_j^s \times M_j^{BW} \quad (18)$$

$$Cap(M) = \sum_{j=1}^{M} Cap(M_j) \quad (19)$$

Step 3: estimate the workload of each VM, the total workload, and the average workload

The load on a virtual machine (VM) is characterized as the total number of tasks assigned to it at a given time t. This load is quantified using Equation (20), which evaluates the ratio of the number of tasks allocated to a VM to its execution time at the same time t. Similarly, the overall workload across all VMs and the average workload of the datacenter are determined using Equations (21) and (22), respectively. These equations collectively provide an estimation of individual and collective workloads within the cloud environment.

$$L(M_j^t) = \frac{NT(t)}{Exe_t(M_j)} \quad (20)$$

$$L = \sum_{j=1}^{M} L(M_j^t) \quad (21)$$

$$L_{avg} = \frac{1}{M}\sum_{j=1}^{M} L(M_j^t) \quad (22)$$

Step 4: identify the state of the VMs, group them into three classes, and sort them depending on their loads

The VMs' state can be found by contrasting the VM's load with the average load. For instance, if the load of a VM is greater than the average loads $\left(L(M_j^t) > L_{avg}\right)$, it can be treated as an overutilized VM (OVM). Similarly, if the load of a VM is less than the average loads $\left(L(M_j^t) < L_{avg}\right)$, it can be treated as an underutilized VM (UVM), otherwise, it is referred to as normalized VM (NVM). Initially, the UVM

RESEARCH ARTICLE

group's underloaded resources are arranged according to increasing loads. The second step involves rearranging the underloaded resources in descending order of resource utilisation while maintaining the same state of loads. Finally, the OVM group's overloaded resources are arranged according to decreasing load orders.

Step 5: Resource Monitoring for UVM

Resource monitoring is an essential step before migrating tasks from an OVM to a suitable UVM for a trade-off. Hence, the total resources available and total resources used by all the UVMs are to be estimated. It is expressed in Equation (23) and Equation (24). Here, vector $\overrightarrow{T_i}$ is the resource usage pattern (core, memory, storage) of all the tasks, and vector $\overrightarrow{T_R}$ denotes the total resources of a UVM.

$$\overrightarrow{T_{R_{used}}}\left(M_{j(UVM)}\right) = \sum_{i=1}^{N} \overrightarrow{T_i} \quad (23)$$

$$\overrightarrow{T_{R_{avail}}}\left(M_{j(UVM)}\right) = \overrightarrow{T_R} - \overrightarrow{T_{R_{used}}} \quad (24)$$

Step 6: Identifying the adaptability and migration of tasks of OVM to UVM based on adaptability value.

After estimating the available resources of UVM, it is pivotal to identify the compatibility between the tasks of OVM with all the UVMs for better placement of tasks and improved performance. Hence, the adaptability is estimated in Equation (26), where the value 0.5 is initialized for α. If the value of the adaptability is less, it means better compatibility exists and hence, migration or placement of that task to the respective UVM is feasible. For checking the compatibility, the similarity (Equation (25)) is identified between the task of an OVM ($\overrightarrow{T_O}$) with the available resources of the UVM.

$$similarity = \cos^{-1}\left(\frac{\overrightarrow{T_O} \times \overrightarrow{T_{R_{avail}}}}{|\overrightarrow{T_O}|\left|\overrightarrow{T_{R_{avail}}}\right|}\right) \quad (25)$$

$$adaptability = \alpha \times angle + (1-\alpha) \times Util_M \quad (26)$$

5. PERFORMANCE EVALUATION AND ANALYSIS

This section demonstrates the experimental setup and presents the compared baselines and dataset utilization for carrying out the simulations. Lastly, the obtained results are demonstrated through graphs for different considered service parameters.

5.1. Experimental Setup

The current study's simulation tests leverage a cloud environment to boost the capabilities of the CloudSim simulator. The environment consists of both homogeneous and heterogeneous virtual machines (VMs) housed in a cluster with an arbitrary mesh topology. Numerous reliant tasks that are dynamically generated need to be scheduled on the virtual machines. The research uses 46 to 246 virtual machines (VMs) and 500 to 2500 tasks. To create heterogeneity between the VMs, each one's processing power must adhere to a consistent distribution between 2000 and 20000 MIPS. Virtual machines' power consumption ranges arbitrarily from 80 to 200 watts when they are in active mode. It is believed that during idleness, 70% of the power used in the active state is used. A 1000 Mbps bandwidth is allotted to the communication channel, and a propagation delay of 1 to 3 milliseconds is considered to exist between the virtual machines. The tasks are divided into three categories: hard real-time activities, soft real-time tasks, and firm real-time tasks. Hard real-time tasks are arbitrarily generated between 100 and 372 MI, with task sizes that are always between 100 and 500 MI. The second and third task types have sizes of 1028–4280 MI and 2400–6800 MI, respectively, with matching deadlines of 500–2500 MI and 1500–4500 MI. A random selection is made for the input and output file sizes for each task type, ranging from 100 to 10,000 KB, 50 to 1000 KB, and 1 to 500 KB, respectively. The Java programming language simulation experiments are implemented using the CloudSim simulator. The trials are performed on a laptop with Windows 11 that has an Intel® Core i7-6600U CPU, four cores, and a 2.6 GHz clock speed. It also has 16 GB of RAM. To ensure accurate results, each experiment is conducted thirty times, and the average of the results is displayed. For the suggested algorithm, the maximum number of iterations is set to 750 and the population size is set to 30.

In order to evaluate this algorithm's performance, the authors considered actual workloads. The GoCJ: Google Cloud Jobs dataset for distributed and cloud computing infrastructures, which Google published in September 2018, was the dataset that the authors used [24]. This dataset, which is kept in the Mendeley data repository, consists of 19 text files, each with a different number of jobs per million instructions (MI). Each job is treated as a cloudlet based on its length in MI. For the execution of the corresponding VMs, the number of cloudlets (1000–2500) with different instruction sizes, ranging from 1000 to 5000, is taken into consideration.

5.2. Results Analysis

To appraise the effectiveness of the proposed algorithm, the authors compared it with other baselines such as binary bird swarm optimization (BBSO) [22], the standard Jaya, Binary Jaya (BJaya) [16], and a hybrid genetic algorithm and Jaya algorithm (GAYA) [13].

All these algorithms are validated for a set of scheduling parameters like Makespan, VM Utilization, Load balancing (doi), and Energy consumption. A distinct dataset with varying task ranges in terms of lengths, MI, and execution times is used to assess the performance parameters in order to realise the impact of a growing set of tasks and virtual machines on the suggested algorithm's scalability and performance. The obtained results present the mean values of

RESEARCH ARTICLE

these performance parameters for the various task and virtual machine ranges in a heterogeneous environment.

5.2.1. Performance Analysis for Makespan

The Makespan metric is a critical indicator of efficiency in task scheduling algorithms, measuring the total time required to execute all tasks. The provided graph compares five algorithms—BCJaya, Binary Bird Swarm Optimization (BBSO) [22], Standard Jaya, Binary Jaya (BJaya) [16], and GAYA (GA + Jaya) [13]—across increasing workloads, from 500×46 to 2500×246 tasks and VMs. BCJaya consistently outperforms the others, achieving the lowest Makespan across all configurations, reflecting its superior scalability and adaptability in handling dynamic, large-scale cloud environments. By incorporating chaotic mappings into the Jaya optimization framework, BCJaya effectively explores the search space, avoids premature convergence, and ensures efficient task allocation to virtual machines (VMs), minimizing idle times and maximizing resource utilization. Figure 2 shows the impact of makespan on the proposed technique.

In contrast, BBSO performs the worst, showing the highest Makespan values across all workloads. Its limited exploration capabilities and tendency to get stuck in local optima result in inefficient task allocation and scalability issues. Standard Jaya shows moderate performance, achieving better Makespan values than BBSO but lagging behind the hybrid approaches. Its steady increase in Makespan with workload growth indicates its limited adaptability in heterogeneous and dynamic cloud settings.

Binary Jaya (BJaya) improves upon Standard Jaya by introducing binary representation for task scheduling, better aligning with the problem's discrete nature. While BJaya outperforms Standard Jaya, it lacks the advanced adaptability and efficiency provided by BCJaya's chaotic mappings. GAYA, which combines Genetic Algorithm (GA) for generating initial solutions with Jaya for refinement, demonstrates better performance than BJaya and Standard Jaya. Its hybrid approach effectively balances exploration and exploitation, achieving lower Makespan values. However, GAYA's performance slightly declines as workloads grow, indicating its limitations in handling large-scale and dynamic environments compared to BCJaya.

The Makespan trends reveal significant differences in scalability. BCJaya exhibits the slowest growth in Makespan as workloads increase, demonstrating its ability to balance tasks effectively and maintain efficiency under high workloads. In contrast, BBSO and Standard Jaya show steep increases in Makespan, reflecting their inefficiency and limited scalability. BJaya and GAYA offer moderate improvements but fail to match BCJaya's robustness and efficiency.

The superior performance of BCJaya highlights the importance of advanced optimization techniques. By integrating chaotic mappings, BCJaya avoids local optima and ensures faster convergence, resulting in consistently lower Makespan values. This adaptability makes BCJaya highly suitable for modern cloud environments, where workloads are dynamic and heterogeneous. On the other hand, the poor scalability of BBSO and Standard Jaya underscores the limitations of traditional heuristic approaches. While BJaya and GAYA represent steps forward, they fall short of BCJaya's performance, particularly for large workloads.

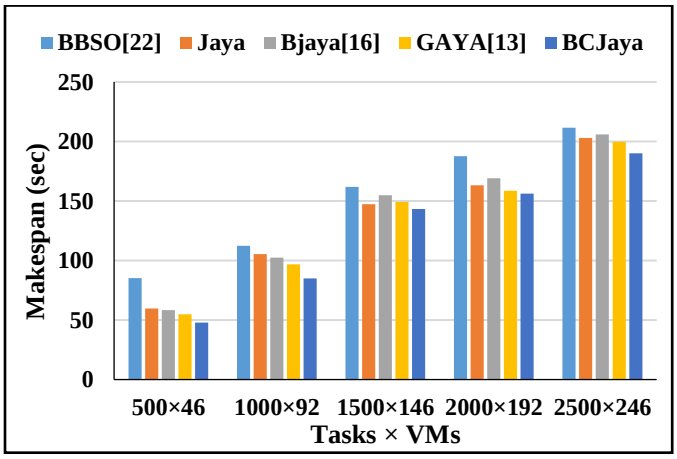

Figure 2 Performance Analysis of the Proposed BCJaya for Makespan

5.2.2. Performance Analysis for VM Utilization

VM Utilization is a critical performance metric in virtualized environments. It measures the extent to which a virtual machine's resources (primarily CPU and memory) are being used. High VM utilization indicates efficient resource allocation, while low utilization suggests underutilization or potential resource overprovisioning.

Figure 3 compares the VM Utilization performance of several techniques: Binary Bird Swarm Optimization (BBSO), Standard Jaya, Binary Jaya, GAYA, and the proposed BCJaya. The x-axis represents different task and VM configurations, while the y-axis indicates VM Utilization percentage.

Across all task and VM configurations, BCJaya consistently demonstrates superior VM Utilization compared to other techniques. This indicates that BCJaya effectively allocates resources and schedules tasks, maximizing the utilization of virtual machines. As the task and VM configurations increase in complexity, the performance gap between BCJaya and other techniques widens. This suggests that BCJaya's advantages are more pronounced in larger and more intricate environments. BBSO, Standard Jaya, and Binary Jaya exhibit some improvement over GAYA, but they still lag behind

RESEARCH ARTICLE

BCJaya. This highlights the impact of hybridization and chaotic map incorporation in BCJaya.

BCJaya's optimization algorithms likely excel at assigning tasks to VMs, ensuring optimal resource allocation and minimizing idle time. The proposed technique may effectively allocate CPU and memory resources to VMs, preventing over-provisioning and underutilization. BCJaya is capable of adapting to changing workloads and resource constraints, ensuring sustained high utilization.

In a nutshell, Figure 3 demonstrates the significant potential of BCJaya in optimizing VM Utilization. Its ability to effectively allocate resources and schedule tasks makes it a promising solution for improving the performance and efficiency of virtualized environments.

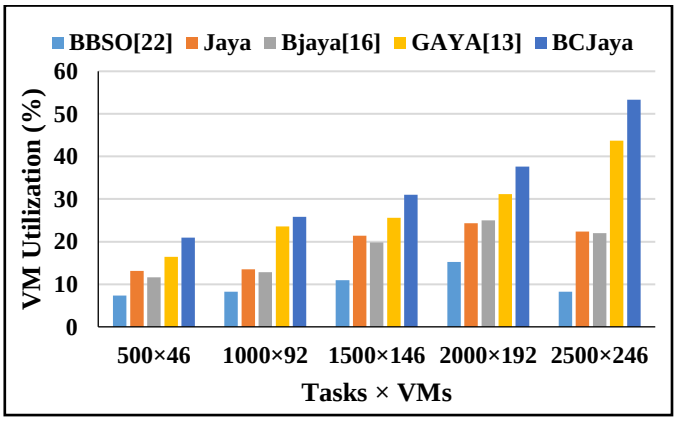

Figure 3 Performance Analysis of the Proposed BCJaya for VM Utilization

5.2.3. Performance Analysis for Energy Consumption

Figure 4 illustrates the performance metric of Energy Consumption for various techniques, including Binary Bird Swarm Optimization (BBSO), Standard Jaya, Binary Jaya, GAYA, and the proposed BCJaya. A key observation is that BCJaya consistently demonstrates the lowest energy consumption across different task and VM configurations. This superior performance highlights BCJaya's ability to efficiently allocate resources and schedule tasks, minimizing idle time and unnecessary energy expenditure.

While BBSO, Standard Jaya, and Binary Jaya exhibit some improvement over GAYA, they still consume significantly more energy than BCJaya, particularly in larger and more complex scenarios. This underscores the impact of BCJaya's hybridization and chaotic map incorporation, which enhance its ability to optimize resource utilization and reduce energy consumption. Reduction in makespan and increased efficient utilization of resources lead to reduce the consumption of energy in the datacentre. The reduction in energy consumption achieved by BCJaya has significant implications for datacenters and cloud computing environments. Lower energy consumption translates to reduced operational costs for datacenter operators and cloud service providers. By minimizing energy usage, BCJaya contributes to a reduced carbon footprint and promotes sustainable computing practices. Efficient resource utilization can lead to improved system reliability and reduced downtime. Because optimal VM utilisation reduces server energy consumption, there is a close, direct linear relationship between energy consumption and VM utilisation.

5.2.4. Performance Analysis for Load Balancing Rate

Figure 5 illustrates the performance metric of Load Balancing Rate for various techniques, including Binary Bird Swarm Optimization (BBSO), Standard Jaya, Binary Jaya, GAYA, and the proposed BCJaya. A key observation is BCJaya's consistent outperformance across different tasks and VM configurations, indicating its superior ability to effectively distribute workload across VMs. This superior performance is particularly evident in larger and more complex scenarios, suggesting the efficacy of BCJaya's hybridization and chaotic map incorporation.

While BBSO, Standard Jaya, and Binary Jaya exhibit some improvement over GAYA, they still lag behind BCJaya, highlighting its potential to optimize workload distribution and improve load balancing. By effectively distributing workload, BCJaya can minimize the overall job completion time, leading to reduced system response times and improved user experience. Optimal workload distribution ensures that VMs are utilized efficiently, preventing idle resources and maximizing system capacity. Balanced workload distribution through the proposed similarity-and-compatibility-based load balancing method helps prevent system overload and improve overall system reliability.

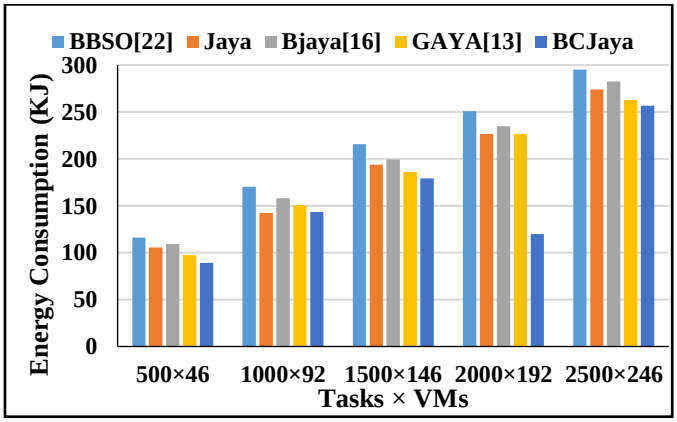

Figure 4 Performance Analysis of the Proposed BCJaya for Energy Consumption

The enhanced load balancing achieved by BCJaya is closely linked to its ability to reduce makespan and increase VM utilization. By efficiently distributing tasks across VMs,

RESEARCH ARTICLE

BCJaya minimizes idle resources and maximizes system throughput. This, in turn, leads to reduced job completion times and improved system performance.

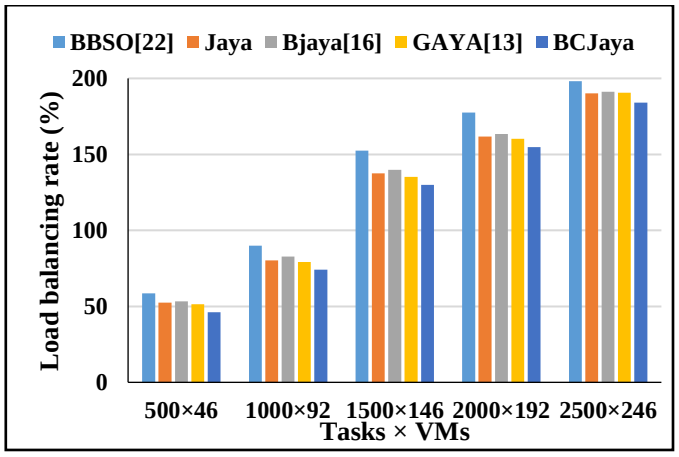

Figure 5 Performance Analysis of the Proposed BCJaya Load Balancing Rate (%)

6. CONCLUSIONS AND FUTURE DIRECTIONS

This study introduces a dynamic scheduling approach combined with an efficient load balancing mechanism. The objective function is designed to optimize multiple parameters simultaneously: reducing the makespan, energy consumption, and degree of imbalance while enhancing VM utilization. The approach adopts a two-step strategy—first, it ensures an even distribution of workloads across VMs to achieve load balancing. This, in turn, significantly improves Quality of Service (QoS) parameters. Following this, the Binary Chaotic JAYA (BCJaya) algorithm is applied for task scheduling. The chaotic principle integrated into Jaya helps address the exploration-exploitation trade-off inherent in the standard Jaya algorithm. Furthermore, the binary adaptation allows for precise task-to-VM mapping in a discrete space. The method is evaluated in a heterogeneous environment where the number of tasks and virtual machines (VMs) dynamically increases, providing a robust test for the algorithm's efficiency. Simulation results highlight substantial improvements in key performance metrics, including makespan, load balancing efficiency, energy consumption, and VM utilization. Compared to other algorithms, the proposed approach consistently delivers superior performance, demonstrating its effectiveness in dynamic cloud environments.

For future work, prioritization of cloud requests could be done to reduce the latency and incurred delay. Different natures of tasks could be explored and simulated to witness the efficacy of the proposed algorithm. A hybrid algorithm could be devised to surmount the inherent limitations of the underlying algorithm.

REFERENCES

[1] Mishra, K., & Majhi, S. (2020). A state-of-art on cloud load balancing algorithms. International Journal of computing and digital systems, 9(2), 201-220.

[2] Zhang, Z., Zhao, M., Wang, H., Cui, Z., & Zhang, W. (2022). An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty. Information Sciences, 583, 56-72.

[3] Ghafari, R., Kabutarkhani, F. H., & Mansouri, N. (2022). Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. Cluster Computing, 25(2), 1035-1093.

[4] Mahapatra, A., Mishra, K., Pradhan, R., & Majhi, S. K. (2023). Next Generation Task Offloading Techniques in Evolving Computing Paradigms: Comparative Analysis, Current Challenges, and Future Research Perspectives. Archives of Computational Methods in Engineering, 1-70. https://doi.org/10.1007/s11831-023-10021-2

[5] Zade, B. M. H., Mansouri, N., & Javidi, M. M. (2022). A two-stage scheduler based on New Caledonian Crow Learning Algorithm and reinforcement learning strategy for cloud environment. Journal of Network and Computer Applications, 202, 103385.

[6] Manikandan, N., Gobalakrishnan, N., & Pradeep, K. (2022). Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. Computer Communications, 187, 35-44.

[7] Pradhan, A., Bisoy, S. K., & Das, A. (2022). A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. Journal of King Saud University-Computer and Information Sciences, 34(8), 4888-4901.

[8] Ullman, J. D. (1975). NP-complete scheduling problems. Journal of Computer and System sciences, 10(3), 384-393.

[9] Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. Egyptian informatics journal, 16(3), 275-295.

[10] Xu, L., Wang, K., Ouyang, Z., & Qi, X. (2014, August). An improved binary PSO-based task scheduling algorithm in green cloud computing. In 9th International Conference on Communications and Networking in China (pp. 126-131). IEEE.

[11] Kaur, G., & Sharma, E. S. (2014). Optimized utilization of resources using improved particle swarm optimization based task scheduling algorithms in cloud computing. International Journal of Emerging Technology and Advanced Engineering, 4(6), 110-115.

[12] Rao, R. V. (2019). Jaya: an advanced optimization algorithm and its engineering applications, 770-780.

[13] Mishra, K., & Majhi, S. K. (2023). A novel improved hybrid optimization algorithm for efficient dynamic medical data scheduling in cloud-based systems for biomedical applications. Multimedia Tools and Applications, 1-35. https://doi.org/10.1007/s11042-023-14448-4

[14] Zahedi Fard, S. Y., Ahmadi, M. R., & Adabi, S. (2017). A dynamic VM consolidation technique for QoS and energy consumption in cloud environment. The Journal of Supercomputing, 73(10), 4347-4368.

[15] Zhang, X., Wu, T., Chen, M., Wei, T., Zhou, J., Hu, S., & Buyya, R. (2019). Energy-aware virtual machine allocation for cloud with resource reservation. Journal of Systems and Software, 147, 147-161.

[16] Mishra, K., Pati, J., & Majhi, S. K. (2022). A dynamic load scheduling in IaaS cloud using binary JAYA algorithm. Journal of King Saud University-Computer and Information Sciences, 34(8), 4914-4930.

[17] Ilager, S., Ramamohanarao, K., & Buyya, R. (2019). ETAS: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. Concurrency and Computation: Practice and Experience, 31(17), e5221.

[18] Azizi, S., Zandsalimi, M. H., & Li, D. (2020). An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Cluster Computing, 23, 3421-3434.

[19] Yavari, M., Ghaffarpour Rahbar, A., & Fathi, M. H. (2019). Temperature and energy-aware consolidation algorithms in cloud computing. Journal of Cloud Computing, 8(1), 1-16.

RESEARCH ARTICLE

[20] Abdessamia, F., Zhang, W. Z., & Tian, Y. C. (2020). Energy-efficiency virtual machine placement based on binary gravitational search algorithm. Cluster Computing, 23, 1577-1588.
[21] Abualigah, L., & Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing, 24, 205-223.
[22] Mishra, K., & Majhi, S. K. (2021). A binary Bird Swarm Optimization based load balancing algorithm for cloud computing environment. Open Computer Science, 11(1), 146-160.
[23] Singh, S., & Vidyarthi, D. P. (2023). An integrated approach of ML-metaheuristics for secure service placement in fog-cloud ecosystem. Internet of Things, 22, 100817. https://doi.org/10.1016/j.iot.2023.100817
[24] Hussain, A., & Aleem, M. (2018). GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures. Data, 3(4), 38.

Authors

Santosh Kumar Paul completed his MCA, M.Tech.(Comp.Sc.). He is currently pursuing Ph.D. (Computer Science) at Sri Sri University Cuttack. He has 15 years of experience in teaching. Currently, he is working on load-balancing optimization algorithms in cloud computing.

Prof (Dr.) Sunil Kumar Dhal working as a Professor at Sri Sri University. He has completed his Master's in mathematics, M.Tech.(Comp.Sc) and Ph.D. in Computer Science. He has 25 years of teaching experience. His current research focuses on Quantitative Methods for Management, Software Project Management, Data Analysis, Management Science, Cloud computing, Wireless Sensor networks, ERP etc. He has many publications and patents and he authored many books.

Dr. Rakesh Nayak is author five books, presently working as Asst. Dean and Head of the Department of Computer Science and Engineering at O P Jindal University, Raigarh, Chhattisgarh. He received his Master degree in Computer Applications from Indira Gandhi National Open University in the year 2007 and M.Tech (CSE) from Acharya Nagarjuna University in 2010 and his Ph.D degree in Computer Science from Behrampur University in 2013. He has more than 22 years of teaching and administrative field in the technical level institutions. He has guided 12 M.Tech Students. He has many publications in international journals/conferences to his credit.

Dr. Umashankar Ghugar earned his full-time doctoral degree from Berhampur University, Odisha, in 2021, and his M. Tech degree in Computer Science from Fakir Mohan University, Balasore, in 2012. and his B.E. degree in IT from Utkal University in 2006. Currently, He is working as an Assistant Professor (Sr. Grade) in the Department of CSE, OP Jindal University, Raigarh, India. He has 15 years of teaching and research experience in different organizations. He has published 40 articles, including reputed journals, book chapters, and conferences in international publishers. His research interests are in Computer Networks, Network Security in WSN. He is a Reviewer of IEEE Access, IEEE Transaction on Education, IEEE Transactions on Neural Networks and Learning Systems, Security and Privacy (Wiley), International Journal of Communication Systems (Wiley), International Journal of Distributed Sensor Networks (Hindawi), International Journal of Knowledge Discovery in Bioinformatics (IGI Global), and International Journal of Information Security and Privacy (IGI Global) and a member of IEEE, IACSIT, CSTA, and IRED.

How to cite this article:

Santosh Kumar Paul, Sunil Kumar Dhal, Rakesh Nayak, Umashankar Ghugar, “Energy-Aware Optimization of Cloud Request Placement and Resource Monitoring Using an Evolutionary Algorithm for Cloud-assisted Systems”, International Journal of Computer Networks and Applications (IJCNA), 11(6), PP: 821-834, 2024, DOI: 10.22247/ijcna/2024/49.