MG-CS: Micro-Genetic and Cuckoo Search Algorithms for Load-Balancing and Power Minimization in Cloud Computing

Jun ZHOU*, Youyou Li
College of Artificial Intelligence, Jiaozuo University, Jiaozuo, Henan, 454000, China

***Abstract*—Cloud computing has emerged as a transformative technology, offering remote access to various computing resources. However, efficiently managing these resources while curbing escalating energy consumption remains a critical challenge. In response, this paper presents the Micro-Genetic Algorithm with Cuckoo Search (MG-CS), a novel approach for enhancing cloud computing efficiency. MG-CS optimizes load balancing and power reduction and significantly contributes to reducing operational costs, ensuring compliance with service level agreements, and enhancing overall service quality. Our experiments showcase MG-CS's versatility in achieving a well-balanced distribution of workloads, resource optimization, and substantial energy savings. This multifaceted approach redefines cloud resource management, offering an environmentally sustainable and cost-effective solution. By introducing MG-CS, this research addresses the pressing challenges in cloud computing, aligning it with environmental responsibility and economic efficiency.**

Keywords*—Resource utilization; cloud computing; energy consumption; optimization

I. Introduction

Cloud computing enables cloud users to access a wide range of configurable computing resources, such as networks, servers, storage, services, and applications, conveniently and on-demand [1]. It has become a transformative technology widely discussed and currently prevalent in numerous commercial sectors. The cloud environment is categorized into private, public, and hybrid/federated clouds [2]. A private cloud represents a dedicated computing environment exclusively utilized by a single organization. It offers benefits like isolation, customization, and heightened security. The hosting can either be on-premises or managed by a third-party provider [3].

On the other hand, a public cloud operates as a shared cloud computing environment accessible to the general public. It provides advantages such as convenience, cost-effectiveness, and scalability, with resources delivered by third-party service providers via the Internet [4]. A hybrid/federated cloud integrates elements of both private and public clouds. This approach enables organizations to distribute workloads across multiple cloud deployment models, offering flexibility, seamless integration, and redundancy. Multi-provider clouds are becoming increasingly popular in cloud infrastructure, where multiple providers are used to distribute workloads across the environment. Organizations can enhance flexibility, redundancy, and resource allocation by leveraging multiple providers. Moreover, there are specialized cloud environments designed to cater to specific services [5]. IoT cloud services are a prime example that caters to IoT devices' data analysis and management. These services are equipped with capabilities to process and derive insights from the massive volumes of IoT-generated data efficiently. Mobile cloud services employ cloud computing to provide applications and services to mobile devices. This approach allows mobile users to access cloud applications and data, providing flexibility, scalability, and improved performance [6].

Cloud computing encompasses three primary cloud service models, each catering to specific needs: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) [7]. Software applications are provided to users over the Internet by SaaS, a cloud computing model that uses a subscription-based approach. SaaS enables users to access and utilize these applications via a web browser, eliminating the need for local installation and maintenance. The responsibility of hosting, maintaining, and updating the software lies with the SaaS provider. PaaS, however, provides developers with a platform and environment to build, deploy, and manage applications without the complexities of managing the underlying Infrastructure. PaaS includes essential tools, runtime environments, databases, and other services required for seamless application development and deployment. Finally, IaaS grants users' access to virtualized computing resources via the Internet. Users can rent virtual machines, storage, and networking components pay-as-you-go. IaaS empowers organizations with the flexibility to create and manage their virtual data centers without the burden of owning physical hardware.

Cloud computing is built upon service-oriented architecture, which enables it to offer various services such as Database-as-a-Service (DbaaS), Identity-as-a-Service (IDaaS), and the broader concept of Anything-as-a-Service (XaaS). This architecture has revolutionized resource management in industry and academia, providing an efficient and dynamic approach [8]. The cloud system's dynamic nature is a crucial characteristic, accommodating numerous users, devices, networks, organizations, and resources that frequently connect and disconnect from the system. This adaptability is essential for meeting the diverse needs of cloud users. Several factors come into play when deciding on the appropriate cloud service model to implement. These factors include flexibility, scalability, interoperability, and service control. Evaluating these aspects is crucial in determining the best-fit cloud service model to address specific requirements and optimize performance.

In the context of cloud computing, users have the flexibility to request resources from both the cloud service provider and the cloud resource broker. When functioning as a cloud service provider, the cloud resource broker is responsible for selecting the most suitable resource, considering the user's stipulated time constraints and budget considerations. This dynamic approach ensures the seamless delivery of on-demand services to users. Nevertheless, the proliferation of users and applications within the cloud ecosystem can lead to an escalation in workload and web application traffic, particularly for those deployed on virtual machines (cloud resources). To manage this expanding landscape effectively, the cloud resource broker necessitates a proficient algorithm capable of distributing tasks equitably among the active virtual machines. Such an algorithm becomes instrumental in minimizing the proportion of tasks that are rejected due to resource constraints. The overarching goal of load distribution within the cloud milieu is to optimize several critical aspects, including scalability, response time, and resource utilization.

Effective load-balancing not only leads to the attainment of minimum makespan times for tasks but also contributes to overall system performance enhancement. Furthermore, load-balancing acts as a preventive measure against system bottlenecks stemming from disparities in load distribution. This realm presents substantial research challenges within the realm of cloud computing, focusing on the equitable distribution of workload among virtual machines. Load-balancing in the cloud encompasses two pivotal stages: task scheduling and virtual machine monitoring. Task scheduling, a well-recognized optimization problem (NP-Complete), becomes intricate due to the heterogeneous resource configuration within the cloud and the swift fluctuations in on-demand requests. The intricate nature of this landscape renders the prediction and computation of all conceivable task-resource mappings within the cloud environment arduous.

Consequently, the development of an efficient task-scheduling algorithm assumes paramount importance. Such an algorithm is instrumental in the judicious distribution of tasks, thereby mitigating scenarios where certain virtual machines endure overload or under-load conditions. These algorithms play a pivotal role in fostering balanced resource utilization and fostering optimal performance within cloud computing systems. As a result, they constitute an indispensable component in the pursuit of achieving equilibrium and excellence within the dynamic cloud computing landscape.

Various techniques, including meta-heuristic algorithms, machine learning, and deep learning, have been integrated into cloud load balancing strategies to address the increasing demand for cloud services and ensure optimal resource utilization. Meta-heuristic algorithms, such as Ant Colony Optimization (ACO) [9], Particle Swarm Optimization (PSO) [10], sine cosine algorithm [11], and imperialist competitive algorithm [12], provide efficient methods for task scheduling and resource allocation, contributing to equitable workload distribution and enhanced system performance. Machine learning techniques enable cloud systems to learn from historical data, adapt to changing workloads, and make real-time load-balancing decisions [13-15]. Deep learning, with its neural networks, enhances predictive accuracy and aids in proactive load management [16, 17]. Cloud load balancing ensures that the various components of these transportation systems, such as ticketing, scheduling, and real-time tracking, operate efficiently and respond to dynamic demands [18, 19]. By utilizing the power of cloud computing and the aforementioned advanced techniques, public transportation services can offer improved reliability, scalability, and cost-effectiveness, ultimately benefiting commuters and the environment.

The demand for cloud services has led to the rapid expansion of extensive data centers, resulting in a significant increase in electricity consumption. This heightened energy consumption has raised concerns about its environmental impact and economic sustainability. Researchers have explored innovative approaches to optimize cloud resource management to address these challenges while simultaneously upholding high-quality service levels. In this context, the integration of metaheuristic algorithms has shown great promise in tackling complex optimization problems frequently encountered in cloud computing. Our paper introduces a novel approach, the Micro-Genetic and Cuckoo Search (MG-CS) algorithm, which is tailored for power reduction and load-balancing in cloud computing. The primary contribution of this research is the development of an efficient and multifaceted approach that concurrently addresses various critical objectives:

- Load balancing excellence: MG-CS aims to achieve a well-balanced distribution of workloads across cloud resources, ensuring optimal resource utilization and averting potential performance bottlenecks.
- Dedicated power minimization: Our approach reduces energy consumption within cloud data centers, promotes environmental sustainability, and optimizes operational costs.
- Strategic cost reduction: We target minimizing resource wastage and optimizing cloud service delivery to make cloud infrastructure more cost-effective.
- Time optimization initiatives: MG-CS endeavors to improve response times and task completion rates, enhancing the overall user experience and operational efficiency of cloud services.
- SLA compliance assurance: Our approach ensures that cloud services meet predefined service level agreements (SLAs), aligning with performance and availability requirements defined by consumers.
- QoS elevation strategies: The research aims to elevate the quality of cloud services, covering aspects of reliability, scalability, and data security, thereby providing an enhanced user experience and meeting customer expectations.

The paper is organized as follows: Section II provides an overview of related work, Section III details our proposed load balancing framework using the MG-CS algorithm, Section IV presents the experimental results, and Section V concludes our research, summarizing the contributions and potential future directions.

II. Related Work

Yakhchi, et al. [20] presented a method rooted in the CS algorithm to identify over-utilized hosts within a cloud environment. Subsequently, they employed the Minimum Migration Time (MMT) policy to systematically transfer VMs from over-utilized hosts to alternative hosts, ensuring that the migration process did not inadvertently lead to new instances of over-utilization. Following this, the researchers categorized all hosts except the over-utilized ones as underutilized, aiming to efficiently relocate VMs from these underutilized hosts to different hosts and transition the former to a sleep mode. This strategic maneuver effectively optimized both resource utilization and energy consumption. The research employed simulation using the CloudSim simulator, yielding compelling results. Specifically, their approach yielded the lowest energy consumption compared to several well-established algorithms, reaffirming the efficacy of their proposed method.

Sharma, et al. [21] have employed the bat algorithm as an approach to cloud load balancing. The bat Algorithm draws inspiration from the echolocation behavior of bats and has been proposed for this purpose. Bats, in their pursuit of prey, exhibit erratic flight patterns by altering various parameters such as velocity, pulse emission rate, position, frequency, and loudness. These alterations are made based on the proximity between the bat and its prey. The adjustment of velocities and positions of bats is incorporated in a manner similar to the PSO algorithmic. The bat algorithm is structured to achieve optimal results by running the algorithm through multiple iterations. In the context of this study, the bat algorithm is utilized to determine the most suitable server from a pool of available servers for the execution of incoming tasks. When a new task is introduced into the task pool, the load balancer initiates the bat algorithm to identify the best-suited server that matches the requirements of the incoming task. The bat algorithm takes into account factors such as task type and required resources when selecting the optimal VM for task execution. Upon selecting the appropriate server, the load balancer allocates the task to that server. If the load on the chosen server surpasses that of all other servers, the task is then distributed across multiple servers.

Devaraj, et al. [22] introduced an innovative load-balancing algorithm named FIMPSO, which represents a hybrid amalgamation of the Firefly (FF) algorithm and the Improved Multi-Objective Particle Swarm Optimization (IMPSO) technique. The FIMPSO algorithm synergizes the strengths of the FF algorithm to effectively narrow down the search space while harnessing the capabilities of the IMPSO technique to attain enhanced responsiveness. The IMPSO algorithm takes a unique approach to select the global best (gbest) particle. It does so by considering the proximity of a point to a line, enabling the identification of candidates for the gbest particle. This method significantly refines the search process, ultimately facilitating the pursuit of an optimal solution. The proposed FIMPSO algorithm is validated through its notable accomplishment in load balancing. This achievement translates to improved resource utilization and diminished task response times. The outcomes of simulations underscore the superiority of the FIMPSO model in comparison to alternative methods. Specifically, the FIMPSO algorithm exhibited exceptional performance metrics such as average response time (13.58ms), CPU utilization (98%), memory utilization (93%), reliability (67%), and throughput (72%). Additionally, the FIMPSO algorithm achieved an impressive makespan of 148, outperforming all other methodologies considered for comparison.

Jena, et al. [23] introduced an inventive approach to dynamically balance the load across VMs utilizing a hybrid strategy named QMPSO, which amalgamates a modified Particle Swarm Optimization (MPSO) technique with an enhanced Q-learning algorithm. Within the QMPSO algorithm, this fusion mechanism fine-tunes the velocity of MPSO by incorporating insights from both the global best (gbest) and personal best (pbest) solutions. These solutions are derived from the optimal actions identified through the improved Q-learning algorithm. The primary objectives driving this hybridization are to elevate the performance of virtual machines through load balancing, amplify the throughput of VMs, and uphold equilibrium between task priorities by optimizing their waiting times. To validate the robustness of the QMPSO algorithm, a comprehensive comparison was conducted. The algorithm's outcomes, gleaned from both simulation-based assessments and actual platform measurements, were juxtaposed with those generated by existing load-balancing and scheduling algorithms. The empirical evidence unequivocally demonstrated the superiority of the proposed QMPSO algorithm, underscoring its prowess in achieving load-balancing and fine-tuning the performance of virtual machines within a cloud environment.

Sefati, et al. [24] harnessed the Grey Wolf Optimization (GWO) algorithm as a means to attain effective load-balancing while considering the resource reliability capacity. In this endeavor, the GWO algorithm was employed to discern nodes that were either idle or occupied within the cloud environment. Once these nodes were identified, the algorithm proceeded to compute the threshold and fitness function for each node. The researchers conducted a simulation using CloudSim, wherein the proposed approach, leveraging the GWO algorithm, was assessed in comparison to other load-balancing methods. The results of this assessment highlighted significant advantages, including reduced costs and response times. Moreover, the solutions obtained were deemed optimal, serving as a testament to the efficacy of the load-balancing methodology founded on the GWO algorithm.

Latchoumi and Parthiban [25] have introduced a groundbreaking approach, termed the Quasi-Oppositional Dragonfly Algorithm for Load-balancing (QODA-LB), with the primary aim of attaining optimal resource scheduling within a cloud computing framework. The QODA-LB algorithm strategically integrates three pivotal variables – execution time, execution cost, and charge – to formulate an objective function. This objective function serves as the foundation for task allocation to Virtual Machines (VMs), predicated on their inherent potential. A noteworthy aspect of the QODA-LB algorithm is the incorporation of the Quasi-

Oppositional Based Learning principle. This principle confers a distinctive edge by elevating the standard convergence rate of the Dragonfly algorithm (DA). The integration of this principle enhances the efficacy of load-balancing and resource scheduling within the cloud environment. A comprehensive series of experiments was meticulously conducted to assess the QODA-LB algorithm's performance. The ensuing results were scrutinized from diverse angles to validate its heightened efficiency. The outcomes of simulations substantiated the algorithm's exceptional load-balancing efficiency, positioning it as a superior alternative to other foundational approaches for load-balancing and resource scheduling in the realm of cloud computing.

Haris and Zubair [26] introduced a dynamic load-balancing algorithm named Mantaray modified multi-objective Harris hawk optimization (MMHHO) that draws inspiration from hybrid optimization algorithms. This innovative approach leverages the strengths of the Harris Hawk Optimization (HHO) algorithm, enhancing its search space through integration with the Manta Ray Foraging Optimization (MRFO) algorithm. The hybridization process strategically melds various factors, including cost, response time, and resource utilization, to streamline the load-balancing process. The MMHHO algorithm sets its sights on optimizing system performance by bolstering VM throughput, achieving equilibrium in load distribution among VMs, and harmonizing task priorities by adjusting their waiting times. The implementation of the MMHHO-based load-balancing algorithm is realized through the utilization of the CloudSim tool. This platform provides the means to assess the algorithm's effectiveness across various parameters and compare its performance against other established load-balancing algorithms. Upon meticulous analysis and simulation, the results unequivocally underscore the supremacy of the proposed MMHHO load-balancing scheme. In terms of system performance and efficiency, the MMHHO algorithm surpasses its counterparts, thereby validating its potential to elevate load-balancing processes and enhance the overall effectiveness of the system.

III. Proposed Load-Balancing Framework

A. Problem Statement

The importance of autonomic load-balancing in the cloud computing domain stems from its capacity to elevate throughput through the optimized utilization of resources. The load-balancing strategies and power management strategies put forth in this proposal are geared towards the automatic and efficient allocation of computational resources within the cloud infrastructure. This is achieved by evaluating the suitability of all tasks concerning resource availability. The effectiveness of the load-balancing approach is determined using intersection formulas, with the most common ones being represented by Eq. (1) and Eq. (2). These formulas play a crucial role in the assessment of task-resource mapping, enabling the system to achieve improved performance and better resource allocation in the cloud environment.

$$K = (G + 2\sqrt{g})/3G \tag{1}$$

$$X = \begin{bmatrix} 0 & gen \\ 1 & gen \end{bmatrix} X \frac{\pi}{2} \tag{2}$$

Fig. 1 depicts the architecture of the load-balancing framework, encompassing three fundamental stages:

- Optimal resource utilization: This phase focuses on achieving efficient resource utilization by effectively managing cloud resources and handling the workload coming from cloud users. Clustering and VM deployment support are employed to ensure optimal resource provisioning.
- Workload submission and demand-based processing: During this phase, cloud users submit their requests and workloads based on their specific demands. The system takes into account energy consumption while processing and managing the workload.
- Minimizing power consumption: The framework emphasizes minimizing power consumption to reduce the environmental impact and operational costs within the cloud.

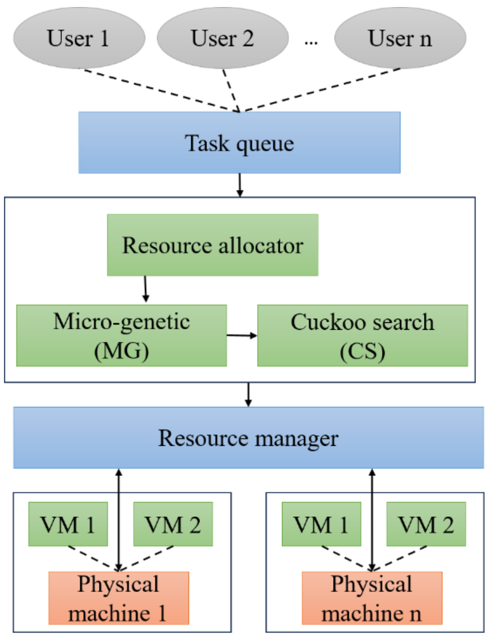

Fig. 1. Proposed load-balancing framework

Key terminologies and components within the load balance framework are as follows:

- Cloud users: Entities, whether individuals or businesses, who make use of cloud storage services to conveniently manage, store, and access their computing resources from any location.
- SLA administration: Service-level agreements (SLAs) provide assurance to customers and enable cloud providers to prioritize the fulfillment of their particular needs and expectations. Active management of SLAs is crucial, as they represent more than just guidelines and function as contracts.

- Workload scheduling and clustering: This method clusters and schedules similar workloads to the corresponding virtual machines based on Quality of Service (QoS) and SLA considerations.
- QoS management: This component is responsible for managing any Quality of Service (QoS) specifications linked to workload access, ensuring that the workload is effectively handled in accordance with its specific QoS requirements.

Algorithm 1 outlines the process of grouping workloads based on their center points by optimizing an objective function to achieve the minimal value. The algorithm aims to create clusters by utilizing the designated center points as group representatives. Each workload in a cluster is highly likely to belong to the cluster represented by its nearest center point. However, workloads can only belong to a single cluster, except for the center point, which may be part of multiple clusters.

Algorithm 1. Workload clustering

```
No of Clusters commit to complete which will be decided by C.
STEP-1: START
STEP-2: The non-empty subclasses of object C will be divided at random.
STEP-3: Cluster centroids are currently the separating seed points of clusters.
STEP-4: An object will be paired with another object whose seed points are closer.
STEP-5: END
```

B. *Micro-genetic Algorithm*

Genetic Algorithms (GAs) belong to the family of Evolutionary Algorithms (EAs) and are widely recognized as one of this family's earliest and most well-known members. In GAs, elitism, which involves preserving the best individuals (solutions) in the population, is promoted through two fundamental mechanisms:

- Environmental selection: Environmental selection aims to remove the worst-fitted individuals from the current population, ensuring they do not contribute to the next generations. This allows the fittest individuals to have a higher chance of survival and progression.
- Parent selection: In the parent selection process, the algorithm promotes generating offspring solutions from the population's best individuals (the elite solutions). Non-elite solutions are excluded from this process, further reinforcing the elitism aspect of the algorithm.

Micro-GAs (MGs) are a specific type of GA with minimal populations. Due to the use of such small populations, MGs exhibit a high level of elitism. In MGs, the environmental selection has lower survivor rates than canonical GAs, as the focus is on preserving only the best solutions. Additionally, the parent selection process only allows elite solutions to generate offspring, further enhancing the elitism effect. The genetic algorithm demonstrates the capability to quickly generate high-quality local optimal solutions while maintaining competitiveness in the long term. This makes it an effective approach for solving problems with computationally intensive fitness functions.

Nevertheless, when dealing with problems that encompass high-dimensional parameter spaces, attaining the convergence of all model parameters within a specified margin of error can present difficulties and consume a substantial amount of time. As the count of model parameters expands, the genetic algorithm necessitates a larger population size, resulting in an increased volume of cost-function analyses. This can be computationally expensive, especially when dealing with high-dimensional problems. In such scenarios, micro-genetic algorithms offer a viable alternative. These algorithms operate with very small populations, which help reduce the computational burden while maintaining a high level of elitism. The smaller population size allows for a more focused search, and the algorithm can swiftly converge to promising solutions without the need for a large number of cost-function evaluations.

C. *Cuckoo Search Algorithm*

The CS algorithm is inspired by the egg-laying strategy of cuckoo birds, where they lay their eggs in the nests of other bird species. This nature-inspired optimization technique simulates this behavior to explore complex search spaces and discover optimal solutions. Cuckoos employ a Levy flight strategy to select nests, frequently opting for nests where the host bird has recently deposited its own eggs. This behavior enhances the likelihood of their eggs successfully hatching. Notably, certain female cuckoos mimic the colors and patterns of host eggs to decrease the chances of their eggs being rejected, thereby amplifying their reproductive success. The foraging behavior of animals, including insects, follows a quasi-random pattern, effectively resembling a random walk. This behavior has been observed in many animals and has been mathematically modeled as Lévy flights. Lévy flights involve making successive movements with step lengths drawn from a Lévy distribution, which allows for long jumps that facilitate efficient exploration of large search spaces. Based on this concept, researchers have applied Lévy flights to optimization and search problems, resulting in the development of the CS algorithm. Preliminary results have shown promising capabilities of this algorithm in finding optimal solutions for a wide range of optimization problems. By imitating the natural behavior of cuckoos and incorporating Lévy flights, the Cuckoo Search Algorithm offers a powerful and efficient approach for tackling complex optimization challenges. The CS algorithm models the natural behavior of cuckoos and can be described using the following idealized rules:

- Each cuckoo lays a single egg at a time, selecting a nest at random for deposit. Nests with superior egg quality (improved solutions) are more likely to persist across subsequent generations.
- The count of available host nests is constant, represented as 'n,' and the host bird has a probability of detecting an alien egg within the range of [0, 1].

- When an alien egg is detected, the host bird has the choice to either discard it or desert the nest to construct a new one at a distinct location.

To simplify, this final assumption can be approximated using a probability of pa for each of the n nests. With these rules in mind, the fundamental steps of the CS algorithm can be succinctly summarized in pseudocode as follows:

```
1) Initialize the population of cuckoos (solution candidates).
2) Evaluate the quality (fitness) of each cuckoo.
3) Identify the best cuckoos and their nests for further reproduction
4) Repeat until stopping criteria are met:
5) Generate new cuckoo solutions by performing Levy flights
6) Evaluate the fitness of newly generated cuckoos
7) Replace the old cuckoos with the new cuckoos in the nests based on their fitness
8) Abandon and rebuild nests (cuckoos) with a probability of $p_a$
9) If a host bird discovers an alien egg with probability $p_a$:
10) Throw away the alien egg or abandon the nest and build a new one
11) Identify the best solution found and return it as the final result
```

In the CS algorithm, the movement of each cuckoo from generation *t* to *t+1* is represented by a vector *x* with entries $X_i(t+1)$ and is calculated by Eq. (3).

$$X_i(t+1) = X_i(t) + \alpha \oplus \text{Lévy}(u) \tag{3}$$

Where $X_i(t)$ is the current position of the i^{th} cuckoo at generation t, $\alpha > 0$ is the step size, which depends on the scale of the given problem, $\oplus$ represents entry-wise multiplication, and *Lévy(u)* is determined using the Lévy flight, a random step-length process. The expression for *Lévy(u)* is given by:

$$\text{Lévy}(u) = t^{(-\lambda)} \tag{4}$$

Where λ is a parameter, typically within the range $1 < \lambda \leq 3$, t is the current generation. The Lévy flight results in a power-law step-length distribution with a heavy tail, making cuckoos more exploratory. In the real world, if a cuckoo's egg closely resembles the host's eggs, it is less likely to be discovered by the host bird. To mimic this behavior, the CS algorithm performs a random walk in a biased way with some random step sizes. This biased random walk, guided by the Lévy flight, allows the algorithm to explore the search space more effectively, discovering better solutions in complex optimization problems.

D. MG-CS Algorithm

Consider a cloud environment with multiple VMs where diverse workloads are dynamically generated based on user demands. The goal is to efficiently distribute these workloads across the available VMs to ensure optimal resource utilization, prevent overloads or under-utilization, and ultimately enhance the overall performance of the cloud system. The algorithm follows the steps described below.

- Initialization: The algorithm begins by randomly entering tasks into a task memory divided into replaceable and non-replaceable tasks. Various parameters such as states, positions, steps, and visual parameters are set up during this phase.
- Task selection: The algorithm selects tasks from the task memory for further processing.
- Crossover and mutation: The selected tasks undergo crossover and mutation operations to generate new potential solutions.
- Patronize: The algorithm evaluates the fitness level of each potential solution.
- New tasks and convergence: The algorithm tracks the best MG values and investigates their behaviors. If the fitness of MG exceeds the predefined threshold (bulletin value), the MG's fitness is updated in the bulletin.
- Filter and external memory: The algorithm uses a filter to refine the solutions and stores valuable information in the external memory, facilitating feedback with both sides of the task memory.
- Final solution: The CS step performs the optimal solution chosen from the population and decodes it to determine the most appropriate resource assignment to tasks based on their availability and throughput.

IV. Experimental Results

The experiment was conducted in CloudSim, a simulation tool devised by cloud laboratories situated in Melbourne. Within this experiment, a total of 50 tasks were examined within a simulation framework encompassing 25 VMs. Each VM was equipped with 2048 MB of RAM. Fig. 2 to 7 present various performance metrics and comparisons of the proposed MG-CS method with existing approaches in a cloud simulation environment. The experiments were conducted with different numbers of workloads and servers. Fig. 2 depicts the availability rate in relation to various workloads. MG-CS exhibited a diverse spectrum of results, with an availability rate of up to 55% with 500 workloads. As the workload increased, the availability rate decreased, reaching 90% with 3000 workloads. Fig. 3 provides an illustration of the reliability rate as it correlates with different workloads. The MG-CS again demonstrated a diverse range of results. It achieved a reliability rate of up to 48% with 1000 workloads, which decreased as the workload increased. Ultimately, it achieved a reliability rate of 72% with 3000 workloads, outperforming the existing system's performance.

In Fig. 4, the resource utilization pattern is displayed alongside varying workloads. The MG-CS approach demonstrated notable efficacy, achieving a peak resource utilization of 80% when subjected to 2500 workloads. Fig. 5 provides a visual representation of the SLA violation rates in relation to varying workloads. Notably, the MG-CS system presented a remarkably low violation rate compared to its counterparts. Specifically, it exhibited a mere 4% violation rate when confronted with 500 workloads and a slightly higher

10.5% violation rate when handling 3000 workloads. Fig. 6 presents the energy consumption during workload processing. The introduced MG-CS approach effectively minimized energy consumption in comparison to comparative ones. To illustrate, when subjected to 2000 workloads, the energy consumption was notably reduced to 400 kW. Fig. 7 provides a visual contrast of execution times across diverse methodologies and workloads. Impressively, the MG-CS system consistently accomplished the processing of 500 to 3,000 workloads within a time span of 5000 to 6,000 seconds. This remarkable efficiency in execution time sets the MG-CS approach apart from existing techniques, highlighting its superior performance.

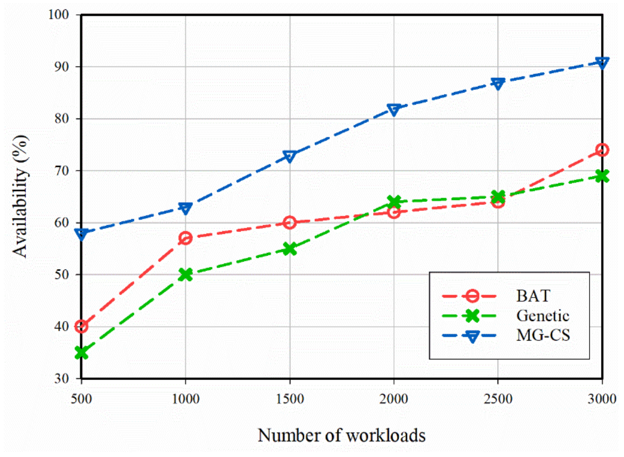

Fig. 2. Availability comparison

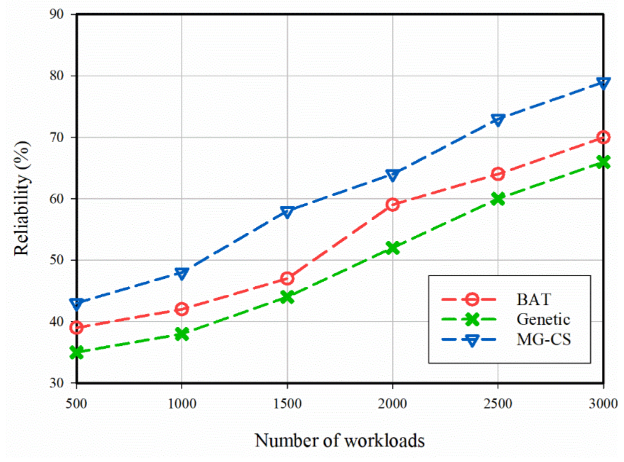

Fig. 3. Reliability comparison

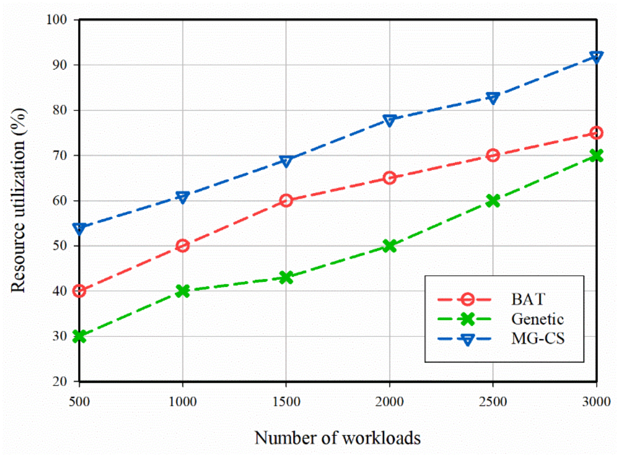

Fig. 4. Resource utilization comparison

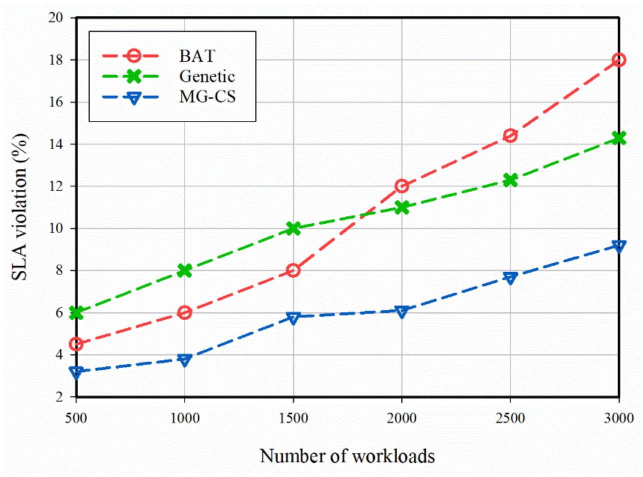

Fig. 5. SLA violation comparison

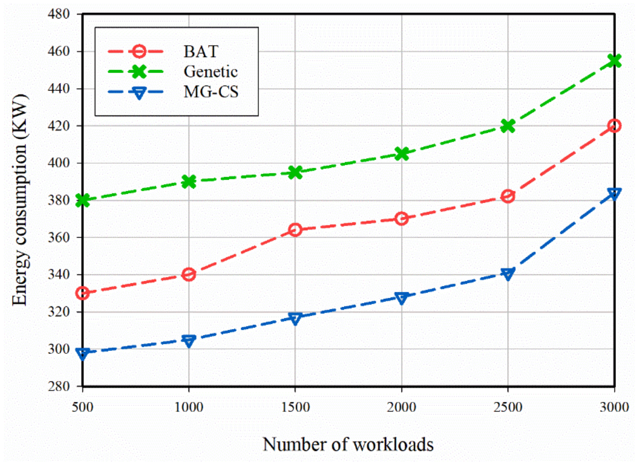

Fig. 6. Energy consumption comparison

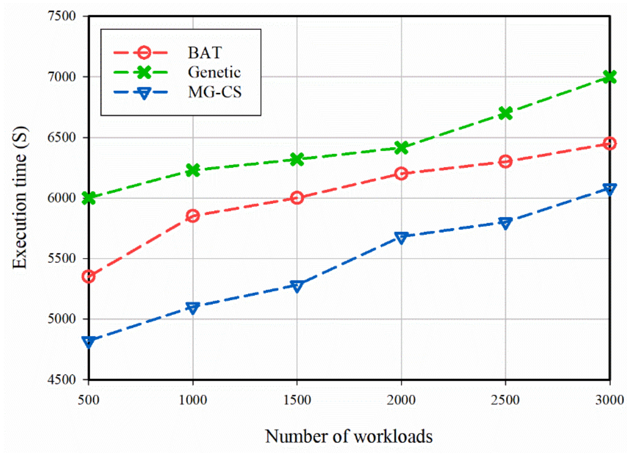

Fig. 7. Execution time comparison

The experimental results, as presented in Fig. 2 to 7, reveal the significant impact of the proposed MG-CS approach on various performance metrics in a cloud simulation environment. Notably, the results illustrate the adaptability and efficacy of MG-CS across a range of workloads and server configurations. In terms of availability, Fig. 2 demonstrates that MG-CS exhibited a diverse spectrum of results, achieving an availability rate of up to 55% with 500 workloads. As the workload increased, availability decreased but remained robust, reaching 90% with 3000 workloads. Similarly, in Fig. 3, the reliability rate showcased a wide range of outcomes. MG-CS achieved a reliability rate of up to 48% with 1000 workloads, surpassing existing systems. Fig. 4 showcases the resource utilization pattern, with MG-CS achieving a remarkable peak utilization of 80% when subjected to 2500 workloads, signifying its efficiency. In terms of SLA violation rates (Fig. 5), MG-CS demonstrated an impressively low violation rate, with just 4% for 500 workloads and a slightly higher 10.5% for 3000 workloads, highlighting its ability to meet service level agreements. Furthermore, Fig. 6 illustrates the energy consumption during workload processing, with MG-CS effectively minimizing energy consumption, reducing it to 400 kW when subjected to 2000 workloads. Lastly, Fig. 7 provides a visual contrast of execution times, underscoring MG-CS's remarkable efficiency in processing 500 to 3,000 workloads within a period of 5,000 to 6,000 seconds. These findings emphasize the significance of the MG-CS approach in enhancing cloud resource management, achieving load balance, and optimizing operational efficiency while meeting service level agreements and reducing energy consumption.

Table I presents the dimensions of diverse synthetic datasets along with the associated task quantities. The "extra-large" dataset encompasses 800-1000 tasks, and each task's magnitude falls within the range of 100,000-200,000MI. Similarly, the "large" dataset comprises 600-700 tasks, with task sizes ranging from 70,000-100,000MI. Correspondingly, the "medium-sized" dataset entails 400-500 tasks, and the tasks vary in size between 50,000-70,000MI. Likewise, the "small-sized" dataset encompasses 100-200 tasks, with task sizes spanning from 30,000-50,000MI. It is noteworthy to mention that task sizes were generated randomly during runtime, and their size is denoted in Millions of Instructions (MI). Moreover, the research utilized a total of 80 servers, each characterized by distinct resource capacities and loads. Each server hosted different types of VM instances, featuring varying CPU and memory capacities, as outlined in Table II. Fig. 8 illustrates the outcomes obtained through the proposed method concerning CPU utilization. The graph clearly demonstrates that, in comparison to FIMPSO, MG-CS consistently achieved the highest CPU utilization across all task categories.

TABLE I. DATASETS DESCRIPTION

Type of tasks	Size of tasks (MI)	Number of tasks
Small	100-200	30000-50000
Medium	400-500	50000-70000
Large	600-700	70000-100000
Extra-large	800-1000	100000-200000

TABLE II. TYPES OF VM INSTANCES

Type of tasks	Memory capacity (GB)	CPU capacity (MIPS)
Small	5	10000
Medium	10	20000
Large	15	25000
Extra-large	20	35000

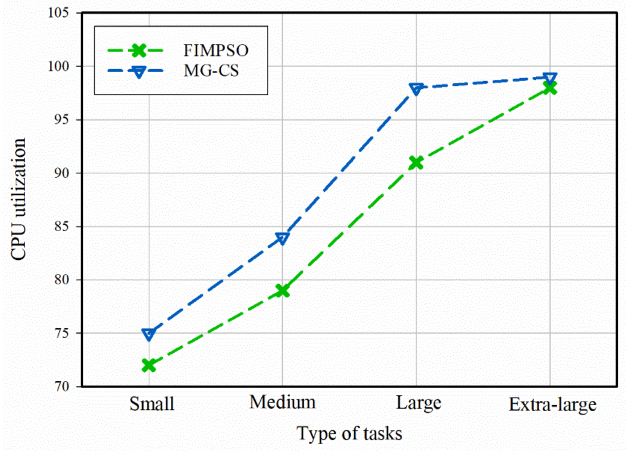

Fig. 8. CPU utilization comparison.

V. CONCLUSION

This paper presented MG-CS, a load-balancing resource allocation approach aimed at improving the utilization of cloud resources. The proposed method's experimental results demonstrate its effectiveness in addressing various QoS factors, including availability, reliability, resource utilization, SLA violation, energy Consumption, and execution time. The proposed method is compared with existing algorithms for QoS parameter efficiency, and the experimental results show that the MG-CS technique outperforms existing GA and BAT algorithms in terms of cost, timing, and energy. The experimental findings prove that MG-CS is the effectiveness of MG-CS in accomplishing both optimal scheduling and load-balancing objectives. Furthermore, the approach ensures the preservation of superior QoS standards while steadfastly adhering to SLA requirements during cloud services. In the future, the researchers plan to incorporate artificial intelligence self-learning methods to facilitate large-scale data sources. The method can enhance its performance and adaptability by integrating AI capabilities in handling complex and dynamic cloud environments. These advancements are expected to contribute to more efficient and robust cloud resource allocation, making cloud services more reliable and cost-effective for users.

MG-CS offers several notable benefits in the context of cloud load balancing and power minimization. It excels in achieving superior load distribution across cloud resources, ensuring optimal resource utilization, and averting performance bottlenecks. Moreover, the MG-CS algorithm effectively reduces energy consumption within cloud data centers, promoting environmental sustainability and cost efficiency. The method optimizes cloud service delivery, enhancing resource utilization and minimizing operational expenses. Furthermore, the approach enhances response times, task completion rates, and overall QoS, improving the user experience. However, like any approach, there are limitations to consider. The computational complexity of MG-CS may pose challenges in large-scale cloud environments. Additionally, the algorithm's performance could be influenced by the specific workload characteristics, and it may require fine-tuning for optimal results. Moreover, while MG-CS demonstrates robust performance in our experiments, its generalizability to diverse cloud infrastructures and real-world scenarios may need further investigation. These limitations underscore the need for ongoing research to fine-tune and adapt MG-CS for various cloud computing contexts and scenarios.

REFERENCES

[1] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[2] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[3] V. Hayyolalam, B. Pourghebleh, A. A. P. Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[4] K. N. Qureshi, G. Jeon, and F. Piccialli, "Anomaly detection and trust authority in artificial intelligence and cloud computing," Computer Networks, vol. 184, p. 107647, 2021.

[5] Q. Yu, L. Chen, and B. Li, "Ant colony optimization applied to web service compositions in cloud computing," Computers & Electrical Engineering, vol. 41, pp. 18-27, 2015.

[6] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," Journal of Network and Computer Applications, vol. 97, pp. 23-34, 2017.

[7] F. Ebadifard and S. M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment," Cluster Computing, vol. 24, no. 2, pp. 1075-1101, 2021.

[8] I. Z. Yakubu and M. Murali, "An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 3, pp. 2981-2992, 2023.

[9] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," IEEE computational intelligence magazine, vol. 1, no. 4, pp. 28-39, 2006.

[10] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," IEEE Access, 2022.

[11] L. Abualigah and A. Diabat, "Advances in sine cosine algorithm: a comprehensive survey," Artificial Intelligence Review, vol. 54, no. 4, pp. 2567-2608, 2021.

[12] S. Mahmoudinazlou, A. Alizadeh, J. Noble, and S. Eslamdoust, "An improved hybrid ICA-SA metaheuristic for order acceptance and scheduling with time windows and sequence-dependent setup times," Neural Computing and Applications, pp. 1-19, 2023.

[13] B. M. Jafari, M. Zhao, and A. Jafari, "Rumi: An Intelligent Agent Enhancing Learning Management Systems Using Machine Learning Techniques," Journal of Software Engineering and Applications, vol. 15, no. 9, pp. 325-343, 2022.

[14] S. R. Abdul Samad et al., "Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection," Electronics, vol. 12, no. 7, p. 1642, 2023.

[15] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Journal of Water Reuse and Desalination, 2022.

[16] S. Aghakhani, A. Larijani, F. Sadeghi, D. Martín, and A. A. Shahrakht, "A Novel Hybrid Artificial Bee Colony-Based Deep Convolutional Neural Network to Improve the Detection Performance of Backscatter Communication Systems," Electronics, vol. 12, no. 10, p. 2263, 2023.

[17] W. Anupong et al., "Deep learning algorithms were used to generate photovoltaic renewable energy in saline water analysis via an oxidation process," Water Reuse, vol. 13, no. 1, pp. 68-81, 2023.

[18] S. Vairachilai, A. Bostani, A. Mehbodniya, J. L. Webber, O. Hemakesavulu, and P. Vijayakumar, "Body Sensor 5 G Networks Utilising Deep Learning Architectures for Emotion Detection Based On EEG Signal Processing," Optik, p. 170469, 2022.

[19] M. Khodayari, J. Razmi, and R. Babazadeh, "An integrated fuzzy analytical network process for prioritisation of new technology-based firms in Iran," International Journal of Industrial and Systems Engineering, vol. 32, no. 4, pp. 424-442, 2019.

[20] M. Yakhchi, S. M. Ghafari, S. Yakhchi, M. Fazeli, and A. Patooghi, "Proposing a load balancing method based on Cuckoo Optimization Algorithm for energy management in cloud computing infrastructures," in 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), 2015: IEEE, pp. 1-5.

[21] S. Sharma, A. K. Luhach, and S. Sinha, "An optimal load balancing technique for cloud computing environment using bat algorithm," Indian J Sci Technol, vol. 9, no. 28, pp. 1-4, 2016.

[22] A. F. S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. L. Lydia, and K. Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments," Journal of Parallel and Distributed Computing, vol. 142, pp. 36-45, 2020.

[23] U. Jena, P. Das, and M. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 6, pp. 2332-2342, 2022.

[24] S. Sefati, M. Mousavinasab, and R. Zareh Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," The Journal of Supercomputing, vol. 78, no. 1, pp. 18-42, 2022.

[25] T. P. Latchoumi and L. Parthiban, "Quasi oppositional dragonfly algorithm for load balancing in cloud computing environment," Wireless Personal Communications, vol. 122, no. 3, pp. 2639-2656, 2022.

[26] M. Haris and S. Zubair, "Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 10, pp. 9696-9709, 2022.