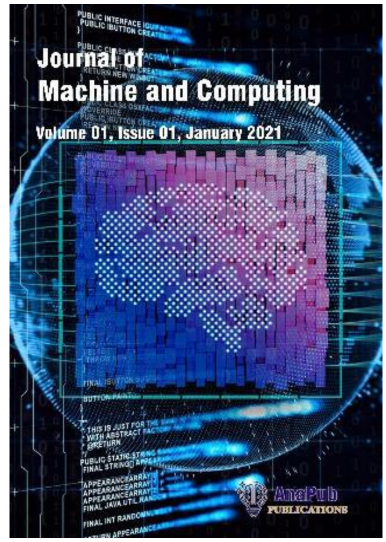
Journal Pre-proof



Efficient Resource Allocation in Cloud Environment: A Hybrid Circle Chaotic Genetic Osprey Solution

Rajgopal K. T, Manoj T. Gadiyar H, Nagesh Shenoy H and Goudar R H

DOI: 10.53759/7669/jmc202505021

Reference: JMC202505021

Journal: Journal of Machine and Computing.

Received 10 May 2024

Revised form 02 October 2024

Accepted 15 November 2024

Efficient Resource Allocation in Cloud Environment: A Hybrid Circle Chaotic Genetic Osprey Solution

[1a,b] Rajgopal K. T., [*2a,b] H. Manoj T. Gadiyar, [3] Nagesh Shenoy H, [4] R H Goudar
[1a]Department of Computer Science and Engineering,
Canara Engineering College, Mangalore, Karnataka, India– 574219.
[1b]Research Scholar,
Visvesvaraya Technological University, Belagavi, Karnataka, India.
[*2a]Department of Information Science and Engineering,
Canara Engineering College, Mangalore, Karnataka, India– 574219.
[*2b]Research Supervisor, Visvesvaraya Technological University, Belagavi, Karnataka, India.
[3]Department of Computer Science and Engineering,
Canara Engineering College, Mangalore, Karnataka, India– 574219.
[4]Department of Computer Science and Engineering,
Visvesvaraya Technological university, Belagavi, 590018.
[1]rajgopal.kt@canaraengineering.in [2]hmanojtgadiyar@gmail.com [3]h.nagesh.shenoy@gmail.com [4]rhgoudar@vtu.ac.in

---

Abstract: Organizations and individuals now access and use computing resources in a completely new way due to cloud computing. However, efficient resource allocation remains a significant challenge in cloud environments. Existing techniques, such as static, dynamic, heuristic, and meta-heuristic, often lead to locally optimal solutions, suffering from slow convergence rates that hinder the achievement of global optimality. To address this challenge, this paper presents a novel Hybrid Circle Chaotic Genetic Osprey Optimization Algorithm (HC^2GOO). This innovative approach synergizes the strengths of the Osprey Optimization Algorithm (O^2A) and Genetic Algorithm (GA) to significantly enhance resource allocation efficiency in cloud environments. The HC^2GOO incorporates a circle chaotic map to replace the random initialization values in the Osprey population update phase. Furthermore, the integration of the GA effectively balances the exploration and exploitation processes of the osprey optimization, facilitating the discovery of optimal solutions. The effectiveness of the HC^2GOO algorithm is assessed using the GWA-T-12 Bitbrains dataset and is benchmarked against established algorithms. The results indicate that HC^2GOO outperforms existing methods, achieving significant improvements in key performance indicators: energy consumption (36 kWh), host utilization ([illegible]3,800), SLA violations (7.2), average execution time (16.2 ms), service cost ($12.5), number of migrations (3,050), and throughput (28.6%) based on 100VMs. Overall, the HC^2GOO algorithm represents a substantial advancement in the field of cloud resource allocation, offering more effective solutions for optimizing computing resource management.

Keywords: Circle chaotic, Cloud computing, Genetic algorithm, Internet, Optimization, Osprey optimization, Resource allocation, Service level agreement (SLA).

I. INTRODUCTION

Cloud computing has fundamentally transformed the landscape of distributed computing, concealing traditional paradigms such as mainframe and client-server architectures. This revolutionary approach provides a comprehensive suite of features and services that organizations and individuals increasingly adopt as they embrace cloud-centric operations [1]. Functionality across cloud services spans critical areas, including communication, integration, management, platform delivery, and networking, illustrating the versatility and depth of cloud solutions personalized to meet specific operational needs [2]. Consequently, cloud computing has become integral across diverse sectors, encompassing education, geospatial sciences, technology, manufacturing, engineering, healthcare, data-intensive applications, and numerous scientific and business fields [3].

The advantages of cloud computing are substantial, offering organizations significant cost savings, enhanced data security, scalability, increased mobility, robust disaster recovery options, comprehensive control over resources, and a competitive edge in the marketplace. These benefits have solidified cloud computing's position as a reliable and indispensable technology within the contemporary business environment [4]. Three main service models, Infrastructure as a Service (IaaS), Platform as a Service

(PaaS), and Software as a Service (SaaS), deliver virtualized resources, which form the foundation of cloud computing architecture [5]. IaaS provides essential hardware resources such as memory, CPU, servers, and storage, with notable examples including Microsoft Azure, Apple iCloud, Google Drive, and Amazon Web Services (AWS) [6] [7]. One example of a platform as a service (PaaS) is Google App Engine, which provides developers with an OS and framework to build, test, run, and manage apps [8] [9][10]. SaaS offers applications as services that users can access through an internet interface, eliminating the need for local installation examples include Google Apps, Cisco WebEx, and Salesforce [11][12].

Despite these capable advantages, cloud computing faces significant challenges shaped by user demands and provider constraints. A critical issue is resource scheduling, an NP-hard problem that profoundly influences cloud system performance [13]. As cloud computing endeavours to provide shared resources as on-demand services, efficient job scheduling is paramount to optimize resource utilization, especially with the numerous resources offered by cloud service providers, including virtual machines (VMs) [14]. Effective VM allocation is not only essential for accommodating diverse user needs but also for maximizing resource efficiency.

The operational efficacy of cloud systems hinges on the optimal performance of all applications. Thus, efficient resource management and job scheduling are foundational requirements for sustaining high operational efficiency in cloud environments [15]. This allocation process involves assigning available resources to incoming applications within designated timeframes, subsequently enhancing the Quality of Service (QoS) for each application [16]. Constraints specified by both cloud service providers and clients are used to strategically divide various projects over different sorts of resources [17].

Due to factors such as rising need for digital transformation, rising costs, and more and more people using cloud-based services, the cloud computing market is expected to experience substantial growth in the near future [18]. From 2024–2029, the market is projected to expand from an initial 2023 valuation of about $587.78B to a final 2029 valuation of between $947.3B and $1.806B, representing a CAGR of 13.3% to 18.49%. However, the market also faces challenges, including inefficient resource allocation, which can lead to underutilization of cloud resources, with approximately 35% of cloud resources remaining underutilized. Optimized use of cloud services can lead to significant cost savings, with AWS reporting that customers may achieve up to 70% savings.

The implementation of effective resource allocation techniques necessitates advanced real-time decision-making capabilities to mitigate instances of underutilization and overutilization, thereby ensuring compliance with Service Level Agreements (SLAs) [19]. Non-compliance can lead to detrimental effects for both customers and service providers, creating financial challenges and reducing profitability [20]. Consequently, cloud providers strive to accommodate a maximized number of incoming requests, focusing on profitability while adhering to the QoS standards delineated within SLAs [21]. To accomplish this, the cloud must have efficient mechanisms for allocating resources in response to user demands; these mechanisms must minimize response times and costs while taking availability, dependability, and response time restrictions service level agreements (SLAs) into account [22].

On-demand resource allocation embodies inherent complexities, recognized as an NP-complete challenge in cloud environments [23]. Algorithms created to handle these problems become more complicated as the amount of resources allocated increases [24]. Although extensive research has been aimed at cloud resource allocation, the domain is influenced by a variety of factors, including substantial request volumes, heterogeneous workloads, dynamic network circumstances, flexible resource provisioning and de-provisioning, fluctuating request, and intricate pricing models [25]. Therefore, it is essential to create a plan for allocating resources that satisfies the needs of service providers as well as those of the end customers.

While several heuristic algorithms have been proposed to approach cloud resource allocation, such as particle swarm optimization (PSO) [26], harmony search (HS) [27], Hill climbing algorithm (HCA) [28], and Nearest Neighbor heuristic (NHH) [29], have not provided satisfactory solutions within practical timeframes. So many researchers nowadays use nature-inspired algorithms for cloud resource allocation, such as genetic algorithm (GA) [30], simulated annealing (SA) [31], and ant colony optimization [32], which are inspired by natural phenomena and are used to elucidate complex optimization difficulties. However, these possess numerous constraints, including raised energy consumption, excessive host utilization, diminished network stability, significant computational complexity, and high-cost utilization. Motivated by these challenges, this paper presents a novel HC^{2}GOO, which is specifically designed to enhance resource allocation in cloud environments while effectively addressing user demand. The key contributions of this research are outlined as follows:

- A hybrid circle chaotic genetic osprey optimization (HC^{2}GOO) algorithm is proposed to identify optimal solutions for scientific applications while meeting end-user demands.
- A model for optimizing power consumption and costs associated with computational resources is developed, focused on significantly reducing energy usage and overall deployment costs.
- The performance and effectiveness of the developed framework are validated across various workloads, with comparisons made against existing algorithms.

Research Questions:

- ☞ How does HC^{2}GOO minimize energy consumption in cloud environments?
- ☞ How does HC^{2}GOO allocate resources in cloud environments, and what are the key performance indicators (KPIs) to measure its effectiveness?

- Can HC^2GOO reduce costs associated with resource allocation, energy consumption, and host utilization in cloud environments?
- How does HC^2GOO compare to existing nature-inspired and meta-heuristic algorithms in terms of optimization performance, computational complexity, and scalability?

The rest of the paper is organized as follows: A thorough analysis of relevant literature about state-of-the-art methods for allocating resources in cloud systems is given in Section 2. The proposed HC2GOO-based virtual machine allocation mechanism is detailed in Section 3. In Section 5, the study is concluded and future directions for this field of study are outlined. In Section 4, the results and discussions surrounding the proposed model are presented.

II. RELATED WORKS

An analysis and description of a survey of different methods currently in use for allocating resources in a cloud environment are provided below.

The efficient resource scheduling algorithm can dynamically schedule tasks on cloud infrastructure, reducing the entire cost of rental virtual machines while ensuring efficient resource utilization. Devi et al. [33] developed a genetic algorithm known as the Genetic Encoded Chromosome for Dynamic Resource Scheduling Policy (GEC-DRP). This approach was tested on both the Google and NASA datasets, achieving a throughput of 95% when scheduling 100 tasks. However, as the amount of tasks augmented to 1000, the throughput decreased to 46%, highlighting the challenges posed by the high computational complexity associated with the GEC-DRP method.

In order to schedule work on already-existing virtual machines (VMs), Shooli et al. [34] devised an efficient resource allocation technique that coupled fuzzy logic with the Gravitational Search Algorithm (GSA). They employed an approach that involved mass creation through the combination of job sequences allocated to numerous machines, GSA for identifying the best assignments, and fuzzy logic for evaluating the interactions between these masses. The performance of the algorithm was evaluated using three metrics: Make-span, Mean Flow Time, and Load Imbalance, demonstrating improved results compared to traditional genetic algorithms and GSA without fuzzy logic. However, the algorithm's utility was constrained in very large-scale cloud environments due to its significant computational resource requirements.

To enhance task scheduling efficiency and promote fairness while minimizing idle time, Manavi et al. [35] developed a hybrid algorithm that integrated genetic algorithms with neural networks. This approach aimed to achieve performance improvements in execution time, cost, and response time. It outperformed cutting-edge techniques, showing improvements of 3.2% in execution time, 13.3% in cost, and 12.1% in reaction time. Nonetheless, the model faced scalability issues when applied to larger datasets or complex task dependencies.

For dynamic resource allocation, Abedi et al. [36] introduced an Improved Firefly Algorithm based on load balancing optimization, termed IFA-DSA. This method sought to efficiently utilize resources and maximize productivity by balancing workloads across existing virtual machines, thereby reducing completion time. Experimental results indicated that the proposed method outpaced the ICFA method in the makespan criterion by an average of 3%. However, IFA-DSA relied on heuristic methods for initial population creation, which may not consistently yield optimal solutions.

In order to optimize resource allocation time and meet task deadlines, Selvapandian et al. [37] created a hybrid optimized allocation model that integrated the PSO algorithm and the Bat Optimization Algorithm (BOA) for resource allocation in multi-cloud environments. This model minimized energy usage. The evaluation of the BOA-PSO model utilized a dataset of 500 tasks with varying requirements and resource availability. The results indicated an allocation time of 47 seconds while achieving a minimum energy consumption of 200 kWh. However, the BOA-PSO model encountered scalability issues when dealing with larger datasets.

Moazeni et al. [38] developed a dynamic resource allocation strategy utilizing a multi-objective teaching-learning-based optimization (AMO-TLBO) algorithm for dynamic effective resource allocation in cloud data centers. This algorithm aimed to efficiently allocate resources for fine-grained computational tasks using datasets generated through simulation tools. The evaluation yielded an impressive resource utilization rate of 80% across 100 tasks. Still, the AMO-TLBO method was limited by its high computational complexity.

In order to minimize execution times, task failure rates, and power consumption, Gupta et al. [39] used a hybrid technique that integrated artificial neural networks (ANN) with the Harmony Search Algorithm (HAS) to optimize resource allocation in cloud computing. The performance of the HAS-ANN model was evaluated using real-world cloud data, yielding an execution time efficiency of 78%. However, this model faced challenges related to high host utilization.

Du et al. [40] developed a cloud computing distribution algorithm based on an enhanced ant colony approach. The goal of this technique was to find the nodes with the fastest response times among all of the available resources and then pick the best ones to meet quality standards. The model was verified through MATLAB simulation experiments, achieving an execution time of 679 seconds; however, it struggled with low throughput performance.

Abouelyazid et al. [41] introduced the Deep-Hill algorithm, which combined a 5-layer Deep Neural Network (DNN) with a Hill-Climbing algorithm to enhance cloud resource allocation by accurately predicting SaaS instance configurations. The performance of the Deep-Hill algorithm was assessed using historical data on SaaS configurations, user demand, and resource allocation, achieving an accuracy of 96.33%. Nevertheless, the Deep-Hill algorithm faced challenges associated with high-cost consumption.

Vhatkar et al. [42] developed a hybrid model known as the Whale Random Update Assisted Lion Algorithm (WR-LA) to improve container resource allocation in cloud-based microservices. This model utilized container resource allocation data derived from cloud computing environments, yielding a performance throughput of 67%. However, it was constrained by longer execution times. The survey of existing techniques with their performance and limitations is explained in Table 1.

Table 1. Survey of existing techniques

Author name and reference	**Technique used**	**Aim**	**Performance**	**Limitation**
Devi et al. [33]	GEC-DRP	Minimize total cost of rental virtual machines while ensuring efficient resource utilization	95% throughput for 100 tasks, 46% for 1000 tasks	High computational complexity and scalability issues
Shooli et al. [34]	GSA combined with fuzzy logic	Schedule tasks on existing VMs	Improved results compared to traditional genetic algorithms and GSA without fuzzy logic.	Significant computational resource requirements, limited utility in very large-scale cloud environments
Manavi et al. [35]	Hybrid algorithm integrating genetic algorithms with neural networks	Enhance task scheduling efficiency and promote fairness while minimizing idle time	3.2% improvement in execution time, 13.3% in cost, 12.1% in response time	Scalability issues when applied to larger datasets or complex task dependencies
Abedi et al. [36]	IFA-DSA	Efficiently utilize resources and maximize productivity by balancing workloads across existing virtual machines.	Outperformed ICFA method in makespan criterion by an average of 3%	Rely on heuristic methods for initial population creation may not consistently yield optimal solutions
Selvapandian et al. [37]	Hybrid optimized allocation model combining BOA and PSO algorithm	Minimize energy consumption while meeting task deadlines and optimizing resource allocation time	Allocation time of 47 seconds, minimum energy consumption of 200 kWh	Scalability issues when dealing with larger datasets
Moazeni et al. [38]	AMO-TLBO algorithm	Efficiently allocate resources for fine-grained computational tasks	Resource utilization rate of 80% across 100 tasks	High computational complexity
Gupta et al. [39]	Hybrid approach combining	Optimize resource allocation in cloud computing by reducing	Execution time efficiency of 78%	High host utilization

	ANN with HAS	execution time, task failure counts, and power consumption.		
Du et al. [40]	Cloud computing allocation algorithm based on an enhanced ant colony approach	Identify the shortest response times across resource nodes and select the best available nodes to meet quality requirements.	Execution time of 679 seconds	Low throughput performance
Abouelyazid et al. [41]	Deep-hill algorithm	Enhance cloud resource allocation by accurately predicting SaaS instance configurations.	Accuracy of 96.33%	High-cost consumption
Vhatkar et al. [42]	WR-LA	Optimize container resource allocation in cloud-based microservices	Performance throughput of 67[illegible]	Longer execution times

Despite the existence of optimization algorithms, their limitations highlight the need for further enhancements to address the challenges in cloud resource allocation. A thorough review of these algorithms reveals that techniques such as PSO, IACO, HAS, AMO-TLB, and BAO are not sufficiently effective for addressing the challenges of resource allocation in the cloud without risking SLAs and deadlines. Consequently, this study introduces an improved HC^{2}GOO-based nature-inspired approach that effectively tackles these existing challenges by efficiently allocating incoming requests to resources based on a fitness function. Additionally, the proposed method optimizes key performance indicators while adhering to user-defined deadlines and budget constraints.

III. PROPOSED METHODOLOGY

The proposed methodology for efficient resource allocation in a cloud environment is embodied in the HC^{2}GOO framework. This innovative approach integrates a circle chaotic map to enhance the initialization process, replacing traditional random values during the Osprey population update phase. By introducing the circle chaotic map, this study aims to improve the diversity of initial solutions, thereby fostering a more effective exploration of the solution space. Moreover, during the osprey optimization process, the GA in the HC^{2}GOO framework is intended to preserve a careful balance between exploration and exploitation. This dual focus allows the algorithm to efficiently converge toward the most optimal solution while ensuring that diverse potential solutions are thoroughly investigated. Fig. 1 displays the proposed model workflow diagram.

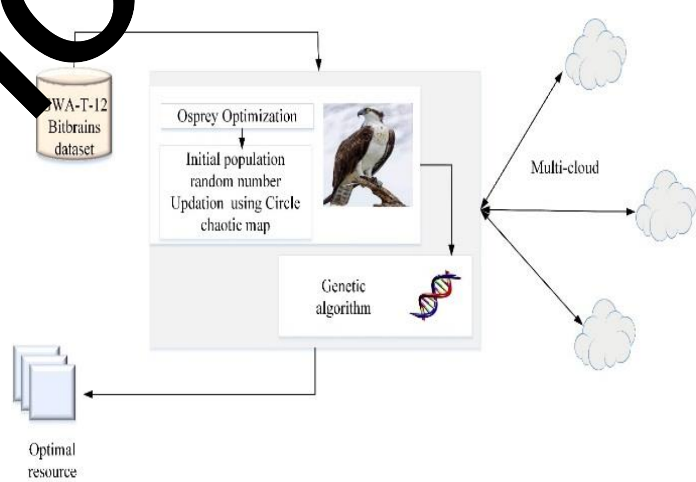


Fig 1. Graphical abstract of the proposed model

A. Osprey Optimization

The osprey is a raptor that preys on fish and is well-known for its wide geographic range and nocturnal habits. It goes by several other names, including sea hawk, river hawk, and fish hawk. With a wingspan of 127–180 cm, these birds weigh between 0.9 and 2.1 kg and measure 50–66 cm in length. Their physical characteristics include:

- Rich glossy brown upperparts and pure white underparts, with irregular brown streaks on their white breast.
- A white head is surrounded by a black facial mask that extends to the neck.
- Light blue translucent nictitating membranes and irises that range in color from golden to brown.
- A black beak with a blue cere and white feet equipped with black claws.
- Short tails and long, slender wings.

As piscivorous birds, ospreys primarily feed on fish, which constitutes about 99% of their diet. Live fish weighing 150–300 g and 25–35 cm long are usual, yet they can catch anything from 2 kg to 50 g. Ospreys can see their underwater prey from 10–40 meters away, due to their extraordinary vision. After identifying a fish, they glide toward it, extend a foot to touch the water, and dive to catch their meal. After catching their meal, ospreys will often take it to a nearby rock to eat [43]. This clever fishing strategy and the behavior of transporting food to a suitable location demonstrates a fascinating instinct that could inspire the development of innovative optimization algorithms.

B. Genetic Algorithm

Charles Darwin's idea of natural selection in which the fittest individuals survive to procreate provided the theoretical foundation for a search strategy known as a genetic algorithm [44]. A fitness function is used to assess the quality of the candidate solutions in the algorithm, and selection, crossover, and mutation are employed to evolve the population towards better solutions. The algorithm iterates through initialization, evaluation, selection, crossover, mutation, and replacement until a closure circumstance is met, such as a extreme quantity of generations. By mimicking the natural selection process, genetic algorithms can effectively search for optimal solutions in complex problem spaces, making them a powerful tool for optimization and search problems.

C. Step involved in the HC²GOO algorithms

The HC^2GOO algorithm is a hybrid optimization algorithm that syndicates the principles of genetic algorithm and osprey optimization. The steps involved in the HC^2GOO algorithm are:

1) Initialization

The O^2A is a population-based approach that iteratively searches for an optimum solution in the problem-solving space. Each osprey in the OOA population represents a potential solution, and its position in the search space is randomly initialized at the beginning of the algorithm. According to equation (1), the population of osprey is described, and equation (2) describes the randomly initialized position of osprey in search space.

$$G = \begin{bmatrix} G_1 \\ \vdots \\ G_P \\ \vdots \\ G_m \end{bmatrix}_{M \times n} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,q} & \cdots & g_{1,n} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ g_{p,1} & \cdots & g_{p,q} & \cdots & g_{p,n} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ g_{m\times 1} & \cdots & g_{m\times q} & \cdots & g_{m\times N} \end{bmatrix}_{M \times N} \tag{1}$$

$$g_{p,q} = a_q + r_{p,q} \cdot (A_q - a_q), \quad p = 1,2,\ldots,M, \quad q = 1,2,\ldots,N, \tag{2}$$

Here, the population matrix of the osprey position is represented as G, the P^{th} position of osprey is G_P with its q^{th} dimension is denoted as $G_{p,q}$. The number of osprey signifies M, the number of problem variables represented as N, and the random number in interval [0, 1] is denoted as $r_{p,q}$.

The improvement of this algorithm is improved by a circle chaotic map in the initialization phase population updating in the original O^2A in equation (2) to increase the performance. The circle chaotic map is a one –one-dimensional map which is a population of a dynamical system on the circle. This map is defined as:

$$g_{p,q} = a_q + r_{0.5,0.2} \cdot (A_q - a_q), \quad p = 1,2,\ldots,M, \quad q = 1,2,\ldots,N \tag{3}$$

Here, equation (3) generated a chaotic number between (0,1) by using $p = 0.5$ and $q = 0.2$. r is taken as a control stricture. The objective function is assessed for every osprey to determine the quality of the solution after the ospreys' positions have been initialized. The objective function value is represented as a vector (equation (4)), and the best and worst solutions are determined based on the objective function value. After each iteration, the position of the ospreys is updated to search for an optimal solution.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_p \\ \vdots \\ F_m \end{bmatrix}_{m\times 1} = \begin{bmatrix} F(G_1) \\ \vdots \\ F(G_p) \\ \vdots \\ F(G_m) \end{bmatrix}_{m\times 1} \quad (4)$$

Where, F and F_p is denoted as the vector of objective function value and p^{th} objective function value.

2) Exploration phase

The exploration phase, in this context, refers to the process by which an osprey identifies and hunts its prey. This phase is characterized by the osprey's keen eyesight, which allows it to spot prey underwater, and its swift diving ability to catch the prey. In this phase, the position of the osprey varies as it searches for prey in its environment. The goal is to improve the osprey's exploration power, enabling it to identify the optimal hunting grounds and avoid getting stuck in suboptimal areas.

Each osprey in the search space aims to have a better objective function than the others. This is achieved by attacking a set of prey, as represented by the equation (5).

$$FN_p = \{G_i \mid i \in \{1,2,\ldots,m\} \wedge F_i < F_p\} \cup \{G_{best}\}, \quad (5)$$

Where, FN_P is denoted as the set of prey's location for p^{th} location, G_{best} is denoted as the best candidate solution. The osprey's position is updated based on its movement towards the prey, as shown in equations (6)-(8).

$$g^{X1}_{p,q} = g_{p,q} + r_{p,q} \cdot (CF_{p,q} - H_{p,q} \cdot g_{p,q}), \quad (6)$$

$$g^{X1}_{p,q} = \begin{cases} g^{X1}_{p,q}, & a_p \le g^{X1}_{p,q} \le A_q; \\ a_q, & g^{X1}_{p,q} < a_q; \\ A_q, & g^{X1}_{p,q} > A_q. \end{cases} \quad (7)$$

$$G_P = \begin{cases} G^{X1}_p, & F^{X1}_P < F_p; \\ G_p, & else \end{cases} \quad (8)$$

Where, the newly updated position of p^{th} osprey is denoted as G^{X1}_p, its q^{th} dimension is represented as $g^{X1}_{p,q}$, and the objective function value is denoted as F^{X1}_P. The selected prey for p^{th} osprey is denoted as CF_p, and its q^{th} dimension is denoted as $CF_{p,q}$, and the random number from set {1, 2} is denoted as $H_{p,q}$.

3) Exploitation phase

The exploitation phase is the second phase of the osprey's hutting process. After catching its prey, the osprey searches for a suitable location to eat. This phase focuses on improving the osprey's ability to find better solutions in the local search space, leading to convergence towards nearby solutions.

The newly updated position of the osprey is determined based on the improvement of the objective function value. This is represented by equation (9),

$$g^{X1}_{p,q} = g_{p,q} + \frac{a_q + r_{p,q} \cdot (A_q - a_q)}{o}, \quad p = 1,2,\ldots,m, \;\; q = 1,2,..,m, \;\; o = 1,2,..O \quad (9)$$

The update process is described by equations (10) and (11).

$$g^{X2}_{p,q} = \begin{cases} g^{X2}_{p,q}, & a_p \le g^{X2}_{p,q} \le A_q; \\ a_q, & g^{X1}_{p,q} < a_q; \\ A_q, & g^{X1}_{p,q} > A_q. \end{cases} \quad (10)$$

$$G_P = \begin{cases} G^{X2}_p, & F^{X2}_P < F_p; \\ G_p, & else \end{cases} \quad (11)$$

Where, the newly updated position of p^{th} osprey is denoted as G_p^{X2}, its q^{th} dimension is represented as $g_{p,q}^{X2}$, and the objective function value is denoted as F_P^{X2}. The count of iterations is o and the whole amount of repetitions is characterized as O. The previous position of the osprey is modified when the objective function value improves, leading to a new position in the search space.

In equation (6), the $r_{p,q}$ plays a crucial role in altering the position of the osprey, which is subsequently used to manage the solution search space of the optimization problem. It is essential to maintain a balance between these two properties. If the solution generated during the osprey's position update does not demonstrate improvement, it suggests an imbalance between exploitation and exploration. This imbalance may hinder the algorithm's ability to effectively navigate the search space, limiting its potential for finding optimal solutions. The proposed approach addresses this issue by incorporating various genetic algorithm operators (selection, mutation, and crossover) aimed at balancing these properties during the osprey's position update phase.

This method is referred to as HC^{2}GOO, which combines Circle Chaotic Osprey and Genetic Algorithm. First, the osprey optimization algorithm and the random numbers for the genetic algorithm are modified. The optimal value is found by analyzing the fitness values of the randomly generated solutions. Then, based on the distance between each value and the optimal value as well as other factors taken into account during the Osprey optimization, a new fitness value is computed.

Consequently, all osprey positions are updated using the newly determined fitness values. The next iteration starts if the updated fitness values indicate improvement; if not, the selection, mutation, and crossover operators of the genetic algorithm are used to improve the optimization process by strengthening both local and global search capabilities.

Applying the genetic algorithm operators requires several technical steps [45-47]. The standard osprey optimization algorithm consists of ospreys, while the standard genetic algorithm employs the concept of genes and chromosomes. To integrate genetic algorithm operators into the osprey optimization framework, the first step is to represent the ospreys as chromosomes in the GA. Each osprey in the O^{2}A corresponds to a chromosome, and collectively, they represent the population's chromosomes. The genes in the created chromosomes are changed and switched in accordance with the mutation and crossover ratios specified in the experimental setup in order to carry out the crossover, mutation, and selection operators. The fitness values of the optimization functions are evaluated after these processes are finished. The process ends if the fitness value of a chromosome meets the required requirements. If not, the procedure runs until either the maximum number of iterations is reached or the termination criteria are met. In the next iteration, the chromosomes are substituted with fireflies. Fig. 2 provides a visual depiction of the flow of the HC2GOO algorithm, highlighting the essential elements and procedures of the technique.

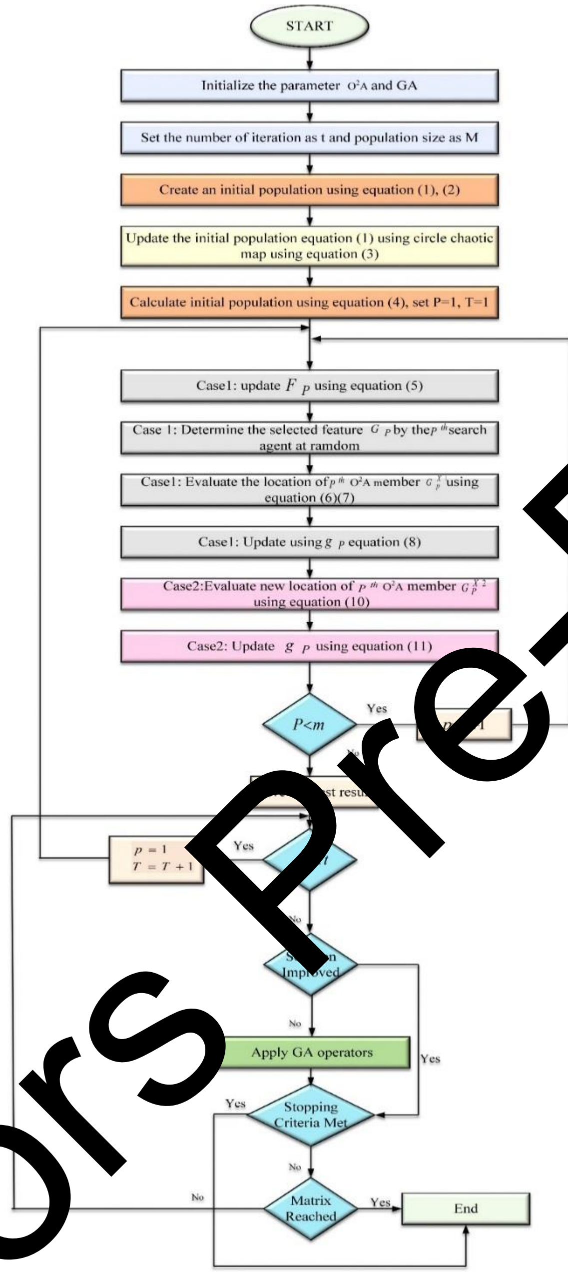


Fig 2. HC^{2}GOO algorithm

The pseudo-code for the HC^{2}GOO algorithm can be found in Table 2.

Table 2. HC^{2}GOO algorithm

Input: Variables, objective function, and constraints.
Let G is population size of osprey and n is the total number of iterations.
Initial population matrix generated using equation (1) and (2).
Update the osprey population using equation (3) circle chaotic map.
The objective function is evaluated using equation (4)

For $q=1$ to n

For $p=1$ to m

Exploration phase:

The prey location is updated for p^{th} osprey using equation (5)

The selected prey is determined by p^{th} osprey randomly.

The updated position of p^{th} osprey is measured using equation (6).
The boundary condition is analyzed for the updated location of osprey using equation (7).

Update p^{th} osprey using equation (8).

Exploitation phase:

The updated location of p^{th} osprey is measured using equation (9).
The boundary condition is analyzed for the updated location of osprey using equation (10).

Update p^{th} osprey using equation (11)
Save the better candidate solution.

End

If solution improved
{
Go to start of the loop
}
Else
{
Apply GA operators
}
p=p+1;
While (Stopping criteria do not meet)
Stop

IV. RESULTS AND DISCUSSION

This section presents a comprehensive experimental analysis of the proposed HC^2GOO algorithm alongside state-of-the-art models, evaluating their performance on the GWA-T-12 Bitbrains dataset for resource allocation in a cloud environment. The performance of the HC^2GOO model is compared to established algorithms, including PSO, Artificial Bee Colony (ABC), Gravitational Search Algorithm (GSA), and Isotropic Markov Mutations with Local Bias (IMMLB) within the same dataset. The hyperparameter details of the HC2GOO algorithm are described in Table 3. The system configurations of this study are presented in Table 4.

Table 3. Hyper-parameter details in HC^2GOO

Parameter	Values
Population size (Number of chromosomes and osprey)	[10,100,100]
Dimension of every osprey	Number tasks
Lower limit	-30
Upper Limit	30
Iteration	200
Search agent	200

Table 4. System configuration of the proposed model

Variables	Specifications
Total no of task	10000
RAM	512 mb
Host parameter	6821 MIPS
Host MIPS	1000000
Task length	1000-3000 mps
Bandwidth	2000MIPS

Cloudlets lengths	[200000 to 500000] in MI
Virtual machine processing rate	[100, 1000] in MIPS
VMs	[1, 2000]
Number of hosts	[1, 40]
DC	1

A. Dataset Description

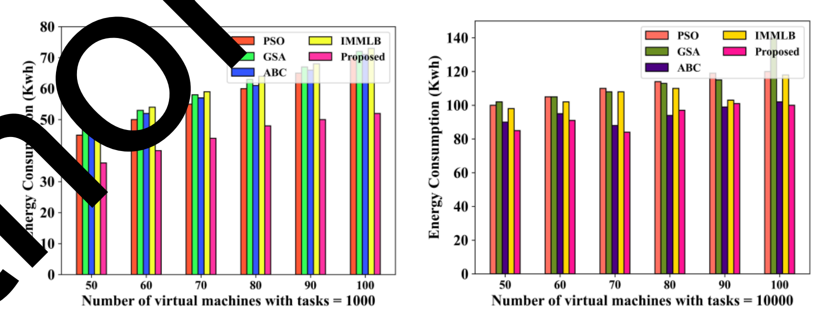
This study focuses on resource allocation in a cloud environment using the GWA-T-12-BitBrains dataset [48-53], a comprehensive collection of VM performance metrics consisting of two distinct subsets: FastStorage and Rnd. The FastStorage subset encompasses 11,221,800 instances, while the Rnd subset includes 12,496,728 instances. This dataset features ten types of metrics that provide a detailed overview of VM performance, including timestamp (measured in milliseconds since January 1, 1970), CPU cores (the number of virtual CPU cores provisioned), CPU capacity provisioned (calculated in MHz as the product of the number of cores and the speed per core), and CPU usage (both in MHz and as a percentage). Additionally, it includes metrics for memory provisioned (in KB), memory usage (in KB), disk read and write throughput (both in KB/s), as well as network received and transmitted throughput (also measured in KB/s). The size of the dataset is 1.16 GB for the FastStorage subset and 1.36 GB for the Rnd subset, highlighting the substantial volume of data captured for effective resource management and performance analysis in cloud environments.

B. Performance Analysis

The competence of the proposed HC^{2}GOO procedure is thoroughly evaluated based on eight key performance metrics: energy consumption (KWh), host utilization (%), SLA violations, average execution time (ms), service cost, task rejection ratio (%), and throughput (m). To provide a comprehensive understanding of the technique's performance, a comparative analysis is conducted against PSO [48], ABC [49], GSA [50], and IMMLB [51]. This analysis takes into account the unique challenges associated with each existing method, including PSO, which can be complex and slow due to high computational demands; ABC may experience longer execution times that affect service responsiveness; GSA can lead to increased costs; and IMMLB may consume too much power, making it less suitable for energy-sensitive environments. The HC^{2}GOO technique aims to address these limitations by combining aspects of various methods, offering reduced complexity, faster execution, lower costs, and improved energy efficiency. The following sections will provide a comparison of these methods, highlighting their strengths and weaknesses across key performance metrics.

1) Energy consumption with varying VMs

The energy consumption is important for evaluating cloud data center performance. High energy usage increases costs and lowers profits. To improve energy efficiency, this study presents the HC^{2}GOO algorithm, which reduces idle and overloaded VM instances. Fig. 3(a) and 3(b) compare energy consumption among different VMs in a cloud environment.



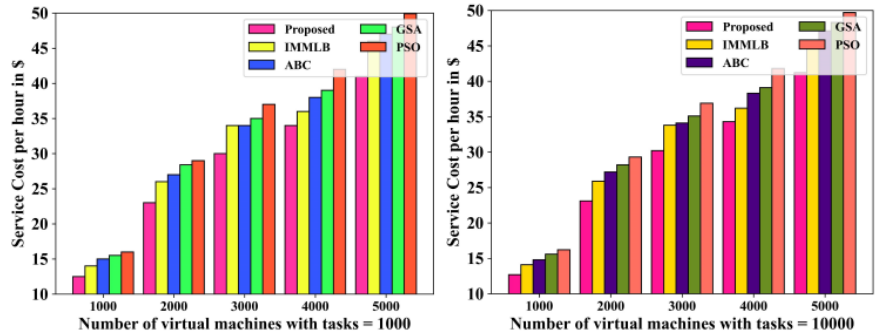
(a) (b)

Fig 3 (a) and (b). Analysis of Energy Consumption

Fig. 3(a) and (b) demonstrate that the energy consumption of different algorithms remains relatively stable as the quantity of VMs ranges from 50 to 100. Notably, HC^{2}GOO algorithms achieved significant energy savings, with energy consumption reduced by 42% in 1,000 tasks and a remarkable 98% in 10,000 tasks. This superior performance can be attributed to the proposed algorithm's innovative approach, which combines Osprey and genetic power to optimize resource allocation in VMs, resulting in substantially less energy consumption compared to other algorithms.

2) Service cost per hour with varying VMs

The service cost per hour in a cloud computing environment varies significantly depending on the number of VMs used. Fig. 4(a) and 4(b) compare service cost per hour among different VMs in cloud computing.



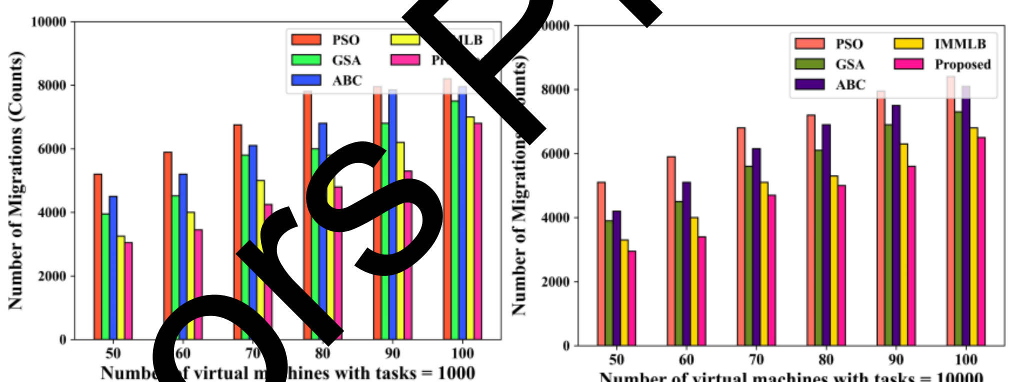
(a) (b)

Fig 4 (a) and (b). Analysis of service cost

Fig. 4(a) and 4(b) demonstrate that the service costs of various algorithms remain relatively stable as the number of VMs increases from 1,000 to 5000. Notably, the HC^{2}GOO algorithms demonstrate a significant reduction in service costs within a cloud environment, achieving a decrease of 7$ for 1,000 tasks and an impressive 9$ for 10,000 tasks. This analysis indicates that the proposed model offers lower service costs per second compared to existing models. The effectiveness of the proposed model stems from its innovative approach, which replaces the traditional population in standard osprey with a circular chaotic map in updated random numbers. As a result of this increased efficiency and speed, the overall cost of running the algorithm is reduced. In simpler terms, the enhancement helps the algorithm work well and faster, which saves money in the long run.

3) Number of Migration with varying VM

The amount of virtual machines (VMs) involved can have a substantial impact on the number of migrations needed. Fig. 5(a) and 5(b) show how the frequency of virtual machine migrations varies in cloud computing settings.



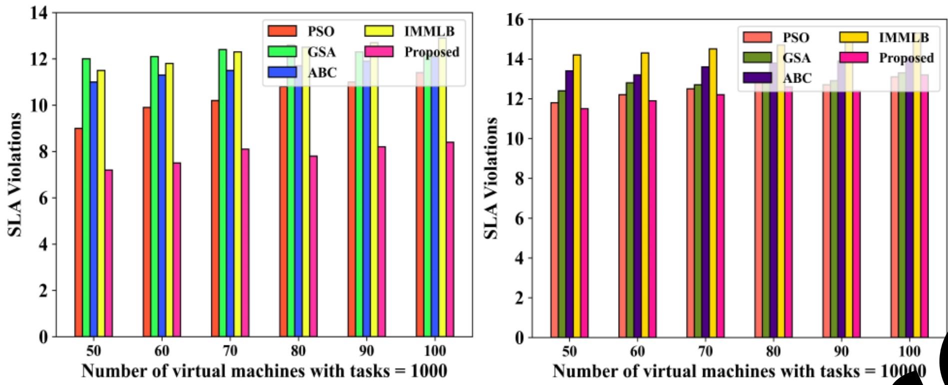
(a) (b)

Fig 5(a) and (b). Analysis of Number of Migrations

To assess the performance metric of VM migration count, the analysis examines the variation in the number of VMs ranging from 50 to 100. The HC^{2}GOO algorithms exhibit a substantial reduction in the number of migrations within a cloud environment, achieving a decrease of [illegible],000 migrations for every 1,000 in 100 tasks, as well as a notable reduction of 6,000 migrations for every 10,000 in 100 tasks. This model effectively demonstrates lower migration counts compared to the existing models, highlighting its efficiency in optimizing VM migrations.

4) SLA violation with varying VM

SLA violations can happen when a supplier fails to provide the specified levels of service, leading to problems like downtime or data loss. With more VMs involved, SLA violations become more likely and have a different impact. Fig. 6(a) and 6(b) compare service cost per hour among different VMs in computing cloud.



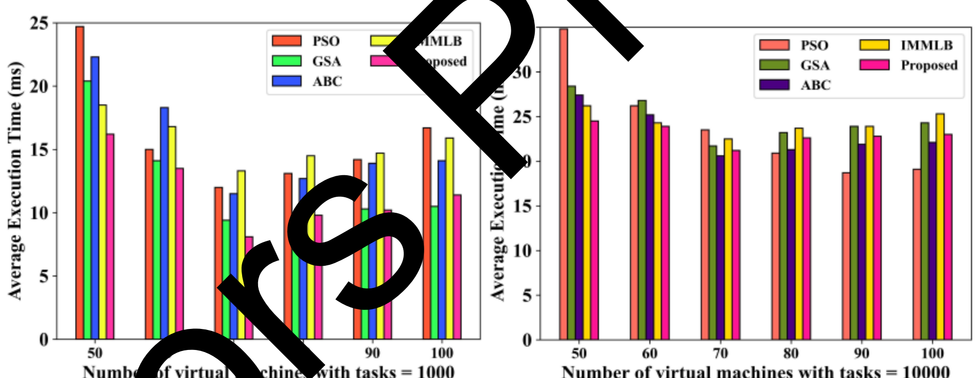
(a) (b)

Fig 6(a) & (b). Analysis of SLA Violations

To evaluate the performance metric of VM SLA violations, the analysis explores variations in the number of VMs, increasing from 50 to 100 in increments of 10. The proposed model shows a significant reduction in SLA violations within a cloud environment, achieving a decrease of 8 violations per 1,000 tasks for 100 tasks and a notable reduction of 12 violations per 10,000 tasks for the same number of tasks. This model outperforms existing models, underscoring its effectiveness in minimizing VM SLA violations.

5) Average Execution Time with VM

The number of VMs participating in a job or application can have a substantial impact on its average execution time. Fig. 7(a) and 7(b) illustrate how different virtual machines' average execution times (ms) vary inside a cloud computing environment.



(a) (b)

Fig 7(a) & (b). Analysis of Average Execution Time

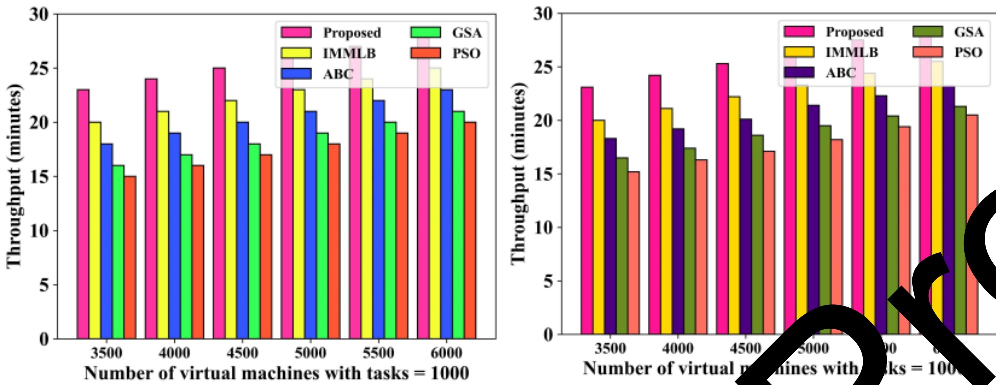
Fig. 7(a) and 7(b) demonstrate that the average execution time of various algorithms remains relatively stable as the number of VMs increases from 1,000 to 5,000. Notably, the proposed model achieves a significant reduction in average execution time within a cloud environment, with a decrease of 11 ms for 1,000 tasks across 100 VMs and an impressive 22 ms for 10,000 tasks across the same number of VMs. This analysis indicates that the HC^2GOO algorithms consistently outperform existing algorithms in terms of average execution time. The effectiveness of the HC^2GOO algorithms is attributed to its innovative approach O^2A. The HC^2GOO algorithms effectively allocate resources across various VMs in the cloud environment, enhancing overall performance and efficiency.

6) Throughput with varying VM

The following equations are used to compute it based on the quantity of applications that are completed in a given amount of time:

$$Throughput = \frac{successfully\ execution\ of\ Tasks}{Total processing\ Time}$$

Throughput is a critical parameter for assessing the performance of the suggested architecture. A high throughput shows the ability to handle a greater number of applications in a shorter period, resulting in improved customer happiness and cloud service quality. Fig. 8(a) and 8(b) compare throughput among different VMs in a cloud environment.



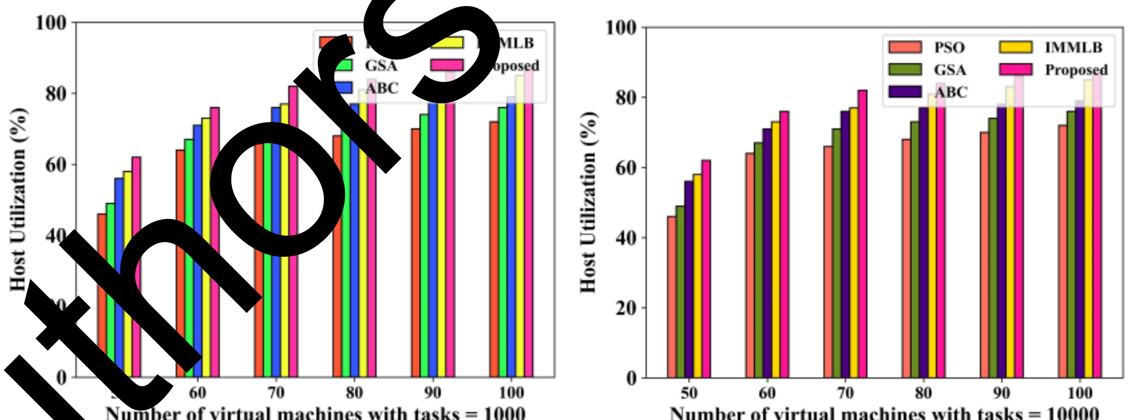
(a) (b)

Fig 8(a) & (b). Analysis of Throughput

Fig. 8(a) and (b) carried out six trials to evaluate the HC^{2}GOO algorithm's performance. Initially, 3,500 tasks were scheduled across 1,000 and 10,000 virtual machines (VMs), with each schedule running a minimum of ten times to obtain the average throughput using both the HC^{2}GOO algorithm and existing algorithms. Additionally, the number of tasks increased by 500 in each schedule, allocated to a fixed number of 1,000 heterogeneous VMs. The throughput of the HC^{2}GOO algorithm significantly outperforms the existing algorithms. Consequently, the improved HC^{2}GOO algorithm outperforms the aforementioned baseline techniques in terms of performance by dynamically assigning the best resources to user requests through an adaptive strategy at runtime.

7) Host utilization with varying VM

Host utilization refers to the percentage of a host's resources (CPU, RAM, storage, and network) being used by VMs. The level of host utilization can significantly impact the performance, reliability, and scalability of a virtualized environment. Fig. 9(a) and 9(b) compare host utilization among different VMs in a cloud environment.



(a) (b)

Fig 9(a) & (b). Analysis of Host Utilization

Fig. 9(a) and 9(b) illustrate that host utilization across different algorithms remains fairly consistent as the number of VMs increases from 50 to 100. Notably, the HC^{2}GOO algorithms significantly reduce host utilization time in a cloud environment, achieving a 65% decrease for 1,000 tasks distributed across 100 VMs and an impressive 90% reduction for 10,000 tasks under the same conditions. The proposed approach effectively optimizes resource allocation among the VMs, leading to enhanced overall performance and efficiency in the cloud environment. Table 5 presents an overall comparison of the HC^{2}GOO and the existing algorithm's performance.

Table 5. Overall resource allocation in cloud environment performance in the HC^2GOO and existing algorithms

Energy Consumption (Kwh)

	Table for 1000 Virtual Machines					Table for 10000 Virtual Machines				
X-tick	PSO	GSA	ABC	IMMLB	Proposed	PSO	GSA	ABC	IMMLB	Proposed
50	45	48	47	49	**36**	100	102	90	98	**85**
60	50	53	52	54	**40**	105	105	95	102	**91**
70	55	58	57	59	**44**	110	108	88	108	**84**
80	60	63	61	64	**48**	114	113	94	110	**97**
90	65	67	66	68	**50**	119	115	[illegible]	103	**101**
100	70	72	71	73	**52**	120	140	102	118	**100**

SLA Violations

	Table for 1000 Virtual Machines					Table for 10000 Virtual Machines				
X-tick	PSO	GSA	ABC	IMMLB	Proposed	PSO	GSA	ABC	IMMLB	Proposed
50	9	12	11	11.15	**7.2**	11.8	[illegible]	13.4	14.2	**11.5**
60	9.9	12.1	11.3	11.8	**7.5**	12.2	12.8	13.2	14.3	**11.9**
70	10.2	12.4	11.5	12.3	**8.1**	12.5	[illegible]	13.6	14.5	**12.2**
80	10.8	12.6	11.7	12.5	**7.8**	[illegible]	13.2	13.8	14.7	**12.6**
90	11	12.3	11.9	12.7	**8.2**	12.7	12.9	13.9	14.9	**12.4**
100	11.4	12.2	12.1	12.9	**[illegible]**	13.1	13.3	14.1	15.3	**13.2**

Average Execution Time (ms)

	Table for 1000 Virtual Machines					Table for 10000 Virtual Machines				
X-tick	PSO	GSA	ABC	IMMLB	Proposed	PSO	GSA	ABC	IMMLB	Proposed
50	24.7	20.4	[illegible]	18.5	**16.2**	34.8	28.4	27.4	26.2	**24.5**
60	15	14.1	[illegible]	16.8	**13.5**	26.2	26.8	25.2	24.3	**23.9**
70	12	9.4	[illegible]	13.3	**8.1**	23.5	21.7	20.6	22.5	**21.2**
80	13.1	[illegible]	12.7	14.5	**9.8**	20.9	23.3	21.3	23.7	**22.6**
90	14.2	10.3	[illegible]	14.7	**10.2**	18.7	23.9	21.9	23.9	**22.8**
100	[illegible]	[illegible]	14.1	15.9	**11.4**	19.1	24.3	22.1	25.3	**23**

Number of Migrations (counts)

	Table for 1000 Virtual Machines					Table for 10000 Virtual Machines				
X-tick	PSO	GSA	ABC	IMMLB	Proposed	PSO	GSA	ABC	IMMLB	Proposed
50	3201	3950	4500	3250	**3050**	5100	3900	4200	3300	**2950**
60	5890	4520	5200	4000	**3450**	5900	4500	5100	4000	**3400**
70	6750	5800	6100	5000	**4250**	6800	5600	6150	5100	**4700**
80	7800	6000	6800	5800	**4800**	7200	6100	6900	5300	**5000**
90	7950	6800	7850	6200	**5300**	7950	6900	7500	6300	**5600**
100	8200	7500	7950	7000	**6800**	8400	7300	8100	6800	**6500**

Service cost per hour is $

Table for 1000 Virtual Machines	Table for 10000 Virtual Machines

X-tick	PSO	GSA	ABC	IMMLB	Proposed	PSO	GSA	ABC	IMMLB	Proposed
1000	16	15.5	15	14	12.5	16.2	15.2	14.8	14.1	**12.7**
2000	29	28.4	27	26	23	29.3	28.2	27.2	25.9	**23.1**
3000	37	35	34	34	30	36.9	35.1	34.1	33.8	**30.2**
4000	42	39	38	38	34	41.8	39.1	38.3	36.2	**34.3**
5000	49.9	48	47	46	41	49.7	48.3	47.1	45.8	**41.2**

Throughput

Table for 1000 Virtual Machines						**Table for 10000 Virtual Machines**				
X-tick	**PSO**	**GSA**	**ABC**	**IMMLB**	**Proposed**	**PSO**	**GSA**	**ABC**	**IMMLB**	**Proposed**
3500	15	16	18	20	**23**	15.2	16.5	18.3	[illegible]	[illegible]
4000	16	17	19	21	**24**	16.3	17.4	[illegible]	[illegible]	**24.2**
4500	17	18	20	22	**25**	17.1	18.6	20.1	22.2	**25.3**
5000	18	19	21	23	**26**	18.2	19.5	[illegible]	23.3	**26.4**
5500	19	20	22	24	**27**	19.4	20.4	22.3	24.4	**27.5**
6000	20	21	23	25	**28**	20.5	21.3	[illegible]	25.5	**28.6**

Host Utilization %

Number of VM 1000		**Number of VM 10000**	
Current unit	**Time in seconds**	**Current unit**	**Time in seconds**
50.83	**13800**	50.83	**13700**
54.54	[illegible]	54.54	**12300**
60.52	**10900**	60.52	**11300**
64.32	**10400**	64.32	**10700**
68.08	**9900**	68.08	**10100**
71.63	**9400**	71.63	**9600**
74.85	[illegible]	74.85	**9300**

Table 5 presents a comparative analysis of the HC^{2}GOO algorithm and the existing algorithm based on several performance metrics. The results denote that the HC^{2}GOO algorithm consistently outperforms the existing algorithms across all these metrics. This superiority suggests that the HC^{2}GOO algorithm effectively addresses the shortcomings of the existing algorithms, resulting in improved resource allocation performance in cloud computing environments.

C. Discussion

In this paper's discussion, the proposed method provides the cloud resource allocation method to allocate the resource VMs with better outcomes compared to the existing method. Table 6 compares the performance of the cloud resource allocation to the existing literature, demonstrating the effectiveness of the HC^{2}GOO model.

Table 6. Execution time comparison of proposed and existing literature work

Author name & References	Technique used	Performances
Devi et al. [33]	GEC-DRP	Energy consumption 121%
Shooli et al. [34]	GSA Combined with Fuzzy logic	Energy consumption 133%
Manavi et al. [35]	Hybrid algorithm integrating genetic algorithms neural network	Energy consumption 152%
Abedi et al.[36]	IFA-DSA	Energy consumption 281%
Selvapandian et al. [37]	BOA and PSO algorithm	Energy consumption 227%
Moazeni et al.[38]	AMO-TLBO	Energy consumption 87%

Gupta et al.[39]	ANN with HAS	Energy consumption 100%
Du et al. [40]	Cloud computing allocation based on an enhanced ant colony approach	Energy consumption 99%
Abouelyazid et al. [41]	Deep-hill algorithm	Energy consumption 152%
Vhatkar et al. [42]	WR-LA	Energy consumption 128%
Proposed	**HC²GOO**	**Energy Consumption 36%**

The HC^{2}GOO has demonstrated exceptional performance in allocating resources in a cloud environment, surpassing existing methods in terms of energy consumption and execution time. By integrating an O^{2}A with a circle chaotic map to enhance population random number generation, the model can generate a more diverse and robust population, leading to a more efficient exploration of the solution space. Furthermore, the GA within the HC^{2}GOO framework is designed to maintain a delicate balance between exploration and exploitation during the osprey optimization process. This dual focus allows the algorithm to efficiently converge toward the most optimal solution while ensuring diverse potential solutions. As a result, the proposed method achieves a significant reduction in energy consumption, with a rate of 36%, compared to existing methods, which range from 87% to 281%. This lower energy use results in financial savings as well as a cloud computing environment that is more ecologically friendly and sustainable. Overall, the HC^{2}GOO model offers a promising solution for cloud resource allocation, addressing the limitations of existing models and providing a more efficient, effective, and sustainable approach.

V. CONCLUSION

The HC^{2}GOO algorithm presents a novel and effective solution for optimal resource allocation in cloud environments. By accurately balancing exploration and exploitation strategies in O^{2}A, along with its robust GA algorithm, the algorithm successfully optimizes resource allocation while minimizing energy consumption. The results from this study highlight the algorithm's superior performance in terms of energy consumption (3[illegible] kWh), host utilization (13,800), SLA violations (7.2), average execution time (16.2 ms), service cost ($12.5), number of migrations (3,[illegible]), and throughput (28.6%) across 100 virtual machines setting compared to existing algorithms. This exceptional performance positions the HC^{2}GOO algorithm as a capable solution for cloud resource allocation, with significant implications for sustainability and reduced operational costs. In future work, explore the applicability of the HC^{2}GOO algorithm in other contexts such as edge and fog computing. Additionally, the algorithm's effectiveness can be enhanced by integrating advanced optimization techniques, including machine learning and deep learning. Its versatility also opens opportunities for addressing other optimization challenges, such as scheduling and resource allocation across various domains.

References:

[1]. A. Belgacem, K. Beghdad-Bey, H. Nacer, and S. Bouznad, "Efficient dynamic resource allocation method for cloud computing environment," Cluster Computing, vol.23, no.4, pp.2871-2889, 2020.

[2]. K. Saidi, and D. Bardou, "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities," Cluster Computing, vol.26, no.5, pp.3069-3087, 2023.

[3]. H. M. T. Gadiyar, M. Bharathrajkumar, and T.K. Sowmya, "Enhanced cipher text-policy attribute-based encryption and serialization on media cloud data," International Journal of Pervasive Computing and Communications, 2022.

[4]. J. Vergara, J. Botero, and L. Fletscher, "A comprehensive survey on resource allocation strategies in fog/cloud environments," Sensors, vol.23, no.9, pp.4413, 2023.

[5]. Y. Gong, J. Huang, B. Liu, J. Xu, B. Wu, and Y. Zhang, "Dynamic resource allocation for virtual machine migration optimization using machine learning," arXiv preprint arXiv:2403, pp.13619, 2024.

[6]. H. M. T. Gadiyar, T. S. Thyagaraju, and R. H. Goudar, "An adaptive approach for preserving privacy in context aware applications for smartphones in cloud computing platform," International Journal of Advanced Computer Science and Applications, vol.13, no.5, 2022.

[7]. [illegible] K. Sam[illegible], "Strategies for efficient resource management in federated cloud environments supporting Infrastructure as a Service (IaaS)," Journal of Engineering Research, vol.12, no.2, pp.101-114, 2024.

[8]. C. O. Kumar, K. Tejaswi, and P. Bhargavi, "A distributed cloud-prevents attacks and preserves user privacy," In 2013 15th International Conference on Advanced Computing Technologies (ICACT), pp. 1-6, 2013. IEEE.

[9]. S. Singh, P. Singh, and S. Tanwar, "Energy aware resource allocation via MS-SLnO in cloud data center," Multimedia Tools and Applications, vol.82, no.29, pp.45541-45563, 2023.

[10]. K. Malathi, R. Anandan, and J. F. Vijay, "Cloud Environment Task Scheduling Optimization of Modified Genetic Algorithm," J. Internet Serv. Inf. Secur., vol.13, no.1, pp.34-43, 2023.

[11]. J. A. Murali, and T. Brindha, "Efficient resource allocation in cloud computing using Hungarian optimization in Aws," 2023.

[12]. M. Kumar, K. Dubey, S. Singh, J. Kumar Samriya, and S. S. Gill, “Experimental performance analysis of cloud resource allocation framework using spider monkey optimization algorithm,” Concurrency and Computation: Practice and Experience, vol.35, no.2, pp.e7469, 2023 .

[13]. A. K. Sangaiah, A. Javadpour, P. Pinto, S. Rezaei, and W. Zhang, “Enhanced resource allocation in distributed cloud using fuzzy meta-heuristics optimization,” Computer Communications, vol.209, pp.14-25, 2023.

[14]. A. K. Singh, S. R. Swain, D. Saxena, and C. N. Lee, “A bio-inspired virtual machine placement toward sustainable cloud resource management,” IEEE Systems Journal, vol.17, no.3, pp.3894-3905, 2023.

[15]. V. Garg, and B. Jindal, “Resource optimization using predictive virtual machine consolidation approach in cloud environment,” Intelligent Decision Technologies, vol.17, no.2, pp.471-484, 2023.

[16]. I. Petrovska, and H. Kuchuk, “Adaptive resource allocation method for data processing and security in cloud environment,” Advanced Information Systems, vol.7, no.3, pp.67-73, 2023.

[17]. T. Alyas, T. M. Ghazal, B. S. Alfurhood, G. F. Issa, O. A. Thawabeh, and Q. Abbas, “Optimizing Resource Allocation Framework for Multi-Cloud Environment,” Computers, Materials and Continua, vol.75, no.2, 2023.

[18]. D. Paulraj, T. Sethukarasi, S. Neelakandan, M. Prakash, and E. Baburaj, “An efficient hybrid job scheduling optimization (EHJSO) approach to enhance resource search using Cuckoo and Grey Wolf Job Optimization for cloud environment,” Plos one, vol.18, no.3, pp.e0282600, 2023.

[19]. J. Jeyaraman, S. V. Bayani, and J. N. A. Malaiyappan, “Optimizing Resource Allocation in Cloud Computing Using Machine Learning,” European Journal of Technology, vol.8, no.3, pp.12-22, 2024.

[20]. V. Ramasamy, and S. Thalavai Pillai, “An effective HPSO-MGA optimization algorithm for dynamic resource allocation in cloud environment,” Cluster Computing, vol.23, pp.1711-1724, 2020.

[21]. A. Rajagopalan, D. R. Modale, and R. Senthilkumar, “Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm,” In Advances in Decision Sciences, Image Processing, Security and Computer Vision: International Conference on Emerging Trends in Engineering (ICETE), Vol. 2, pp. 678-687, 2020. Springer International Publishing.

[22]. V. Jafari, and M. H. Rezvani, “Joint optimization of energy consumption and time delay in IoT-fog-cloud computing environments using NSGA-II metaheuristic algorithm,” Journal of Ambient Intelligence and Humanized Computing, vol.14, no.3, pp.1675-1698, 2023.

[23]. M. Ghobaei-Arani, and A. Shahidinejad, “An efficient resource provisioning approach for analyzing cloud workloads: a metaheuristic-based clustering approach,” The Journal of Supercomputing, vol.77, no.1, pp.711-750, 2021.

[24]. R. K. Kalimuthu, and B. Thomas, “An effective multi-objective task scheduling and resource optimization in cloud environment using hybridized metaheuristic algorithm,” Journal of Intelligent and Fuzzy Systems, vol.42, no.4, pp.4051-4063, 2022.

[25]. H. Singh, S. Tyagi, and P. Kumar, “Scheduling in cloud computing environment using metaheuristic techniques: a survey,” In Emerging technology in modelling and graphics: proceedings of IEM graph 2018, pp. 753-763, 2020. Springer Singapore.

[26]. R. R. Dornala, S. Ponnapalli, K. T. Sai, S. [illegible], R. R. Koteru, and B. Koteru, “Ensemble Resource Allocation using Optimized Particle Swarm Optimization (PSO) in Cloud Computing,” In 2024 3rd International Conference on Sentiment Analysis and Deep Learning (ICSADL), pp. 342-348, 2024. IEEE.

[27]. T. Renugadevi, K. Geetha, K. Muthukumar, and Z. W. Geem, “Energy-Efficient Resource Provisioning Using Adaptive Harmony Search Algorithm for Compute-Intensive Workloads with Load Balancing in Datacenters,” Applied Sciences, vol.10, no.7, pp.2323, 2020.

[28]. S. Achar, “Neural Hill: A Novel Algorithm for Efficient Scheduling IoT-Cloud Resource to Maintain Scalability,” IEEE Access, vol.11, pp.265[illegible]-26511, 2023.

[29]. W. Bi, J. Ma, [illegible] Zhu, W. [illegible]ng, and A. Zhang, “Cloud service selection based on weighted KD tree nearest neighbor search,” Applied Soft Computing, vol.131, pp.109780, 2022.

[30]. P. Devaras[illegible]y, and [illegible] Reddy, “Genetic algorithm for quality of service based resource allocation in cloud computing,” Evolutionary Intelligence, vol.14, pp.381-387, 2021.

[31]. D. Gabi, [illegible] M. Dankolo, A. A. Muslim, A. Abraham, M. U. Joda, A. Zainal, and Z. Zakaria, “Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme,” Neural Computing and Applications, vol.34, no.16, pp.14085-14105, 2022.

[32]. Q. Zhou, “Research on optimization algorithm of cloud computing resource allocation for internet of things engineering based on improved ant colony algorithm,” Mathematical Problems in Engineering, vol.2022, no.1, pp.5632117, 2022.

[33]. K. L. Devi, and S. Valli, “Multi-objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment,” The Journal of Supercomputing, vol.77, no.8, pp.8252-8280, 2021.

[34]. Bhanurangarao, M., & Mahaveerakannan, R. (2024, October). Enhancing Hybrid Object Identification for Instantaneous Healthcare through Lorentz Force. In 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 1365-1368). IEEE.

[35]. M. Manavi, Y. Zhang, and G. Chen, “Resource allocation in cloud computing using genetic algorithm and neural network,” In 2023 IEEE 8th International Conference on Smart Cloud (SmartCloud), pp. 25-32, 2023. IEEE.

[36]. S.Abedi, M. Ghobaei-Arani, E. Khorami, and M. Mojarad, “Dynamic resource allocation using improved firefly optimization algorithm in cloud environment,” Applied Artificial Intelligence, vol.36, no.1, pp.2055394, 2022.

[37]. D. Selvapandian, and R. Santosh, “A hybrid optimized resource allocation model for multi-cloud environment using bat and particle swarm optimization algorithms”, Computer Assisted Methods in Engineering and Science, vol.29, no.1–2, pp.87-103, 2022.

[38]. Yuvarani, R., & Mahaveerakannan, R. (2024, October). Enhanced IoT-based Healthcare Device for Secure Patient Data Management using Hybrid Cryptography Algorithm. In 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 22-28). IEEE.

[39]. P. Gupta, S. Bhagat, and P. Rawat, “Fault aware hybrid harmony search technique for optimal resource allocation in cloud,” Journal of Intelligent and Fuzzy Systems, vol.42, no.4, pp.3677-3689, 2022.

[40]. H. Du, and J. Chen, “An Improved Ant Colony Algorithm for New energy Industry Resource Allocation in Cloud Environment,” Tehnicki vjesnik, vol.30, no.1, pp.153-157, 2023.

[41]. M. Abouelyazid, “Deep-Hill: An Innovative Cloud Resource Optimization Algorithm by Predicting SaaS Instance Configuration using Deep Learning,” IEEE Access, 2024.

[42]. K. N. Vhatkar, and G. P. Bhole, “Optimal container resource allocation in cloud architecture: A new hybrid model,” Journal of King Saud University-Computer and Information Sciences, vol.34, no.5, pp.1906-1918, 2022.

[43]. K. Panneerselvam, P. P. Nayudu, M. S. Banu, and P. M. Rekha, “Multi-objective load balancing based on adaptive osprey optimization algorithm,” International Journal of Information Technology, pp.1-8, 2024.

[44]. G. Portaluri, S. Giordano, D. Kliazovich, and B. Dorronsoro, “A power efficient genetic algorithm for resource allocation in cloud computing data centers,” In 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 58-63, 2014. IEEE.

[45]. Mahaveerakannan, R., Choudhary, S. L., Dixit, R. S., Mylapalli, S., & Kumar, M. S. (2024, October). Enhancing Diagnostic Accuracy and Early Detection Through the Application of Deep Learning Techniques to the Segmentation of Colon Cancer in Histopathological Images. In 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 1809-1815). IEEE..

[46]. S. K. Suman, D. Kumar, L. Bhagyalakshmi, "SINR Pricing in Non-Cooperative Power Control Game for Wireless Ad Hoc Networks," KSII Transactions on Internet and Information Systems, vol. 8, no. 7, pp. 2281-2301, 2014. DOI: 10.3837/tiis.2014.07.005.

[47]. Y. J. Gong, J. Zhang, H. S. H. Chung, W. N. Chen, Z. H. Zhan, Y. Li, and Y. H. Shi, “An efficient resource allocation scheme using particle swarm optimization,” IEEE Transactions on Evolutionary Computation, vol.16, no.6, pp.801-816, 2012.

[48]. R. K, S. K. Suman, U. Rajeswari, S. S, H. Poddar and A. T. S, "Reinforcement Learning Models for Autonomous Decision Making in Sensor Systems," *2024 International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)*, Indore, India, 2024, pp. 1-6, doi: 10.1109/ACROSET62108.2024.10743345

[49]. B. Muthulakshmi, and K. Somasundaram, “A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment,” Cluster Computing, vol.22, no.Suppl 5, pp.10769-10777, 2019.

[50]. L. Bhagyalakshmi, S. K. Suman and K. Murugan, "Corona based clustering with mixed routing and data aggregation to avoid energy hole problem in wireless sensor networks," *2012 Fourth International Conference on Advanced Computing (ICoAC)*, Chennai, India, 2012, pp. 1-8, doi: 10.1109/ICoAC.2012.6416860

[51]. L. Datta, and G. Thippanna, “A [illegible]SA Based Algorithm to Optimize Task Scheduling in Cloud Computing Environment,” COMPUTER, vol 24, no.1, 2024.

[52]. A. Gopu, K. Thirugnanasambandam, A. S. AlGhamdi, S. S. Alshamrani, K. Maharajan, and M. Rashid, “Energy-efficient virtual machine placement in distributed cloud using NSGA-III algorithm,” Journal of Cloud Computing, vol.12, no.1, pp.124, 2023.

[53]. Bhagyalakshmi, L., Suman, S.K. & Sujeethadevi, T. Joint Routing and Resource Allocation for Cluster Based Isolated Nodes in Cognitive Radio Wireless Sensor Networks. *Wireless Pers Commun* **114**, 3477–3488 (2020). https://doi.org/10.1007/s11277-020-0754[illegible]