

Optimizing cloud resource allocation using Gaussian convergence squirrel search algorithm

Sidagouda Basagouda Patil, Mukund Anant Kulkarni

Department of Computer Science (Management Studies), College of Engineering, Bharati Vidyapeeth (Deemed to be University),
Kolhapur, India

Article Info

Article history:

Received Aug 28, 2025

Revised Jul 30, 2026

Accepted Aug 12, 2026

Keywords:

Chaotic map

Cloud computing

Gaussian-based convergence factor

Squirrel search algorithm

Virtual machines

ABSTRACT

Cloud computing has gained significant attention for efficiently executing requests according to users' needs, providing quality services, and optimizing task execution time across virtual machines (VMs). However, resource allocation for various applications is difficult due to dynamic workload conditions and uncertainty in cloud computing. In this research, a Gaussian-based convergence factor (GCF) with the squirrel search algorithm (SSA) is used for allocating resources in a cloud environment for a small finance organization. The proposed GCF along with SSA balances workload and allocates many relevant resources to users' applications while also ensuring deadline constraints. Tent chaotic map and GCF are incorporated in a traditional SSA to improve the search ability and convergence rate of SSA to allocate optimal resources in the cloud for a small finance organization. The selection of appropriate instance types that align with financial organization requirements includes choosing between on-demand, reserved, or spot instances. The GCF with SSA obtained less energy of 0.505 J, lower execution time of 0.472 s, less makespan of 0.723 s, and higher resource utilization of 51% for 100 tasks of 30 VMs, when compared to existing methods.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sidagouda Basagouda Patil

Department of Computer Science (Management Studies), College of Engineering

Bharati Vidyapeeth (Deemed to be University)

Kolhapur, Maharashtra, India

Email: ashidu28@gmail.com

1. INTRODUCTION

Recently, the volume of the data processed in cloud environments has maximized significantly due to the availability of various internet of things (IoT) combinations [1]. The maximization of growth and popularity of IoT leads to the data development in the perception layer of IoT [2]. Cloud management is developed to allocate total server resources efficiently, allowing clients to access the requested data via explicit requests through storage and database on the server [3]–[5]. In the cloud, a huge amount of data increases the risk of potential breaches and hacking through malicious groups [6]. Cloud computing is deployed over companies for accentuating security while computing employment costs through accepting that virtualization is minimized [7]. The process of application and multi operation processed in the same server [8]. The usage of hardware and security is improved through virtualization in the cloud and gives better performance in the process of effectiveness and resiliency [9]. Additionally, virtualization is an efficient and fast cloud technology that enables the system for diverse environments. The partial energy consumption of used servers in data centers is the main reason for high consumption of energy. Consequently, the service providers pay high costs without attaining the required amount of service

usage [10]. The main component of the executing method is the virtualization technique, which transforms physical machines (PMs) to multi-virtual machines (VMs), fulfilling the requirements of several customers and users [11]. As a result, VMs have become a standard phase and a unit to deliver programs. Each VM that is integrated with the PM in a data center has some quantity of resource ability [12]. User requests are treated as a group of tasks in cloud computing and each VM serves every task by available resources [13]. Hence, the resource allocation process in cloud computing is most significant, it enables the user requests to execute automatically without the requirement of direct intervention [14]. The services of cloud computing are generally provided on demand and subjected to time constraints as hours and minutes [15].

Javaheri *et al.* [16] suggested the clipped double deep Q-learning (CDDQL) method and particle swarm optimization (PSO) to allocate resources in cloud architecture. PSO algorithm was utilized for prioritizing tasks, and the CDDQL was utilized to the core of auto-CDDQL for allocating destined VM resources for tasks. The suggested method was introduced in the fog and performed autonomously. Although the suggested method had high scalability, the method was unable to consider the cost of data locality, energy consumption, and processing expense. Singh *et al.* [17] presented the optimization-based resource allocation method, which was increased with fewer costs. Before the allocation, workload clustering was performed by utilizing advanced k-means clustering using lion with advanced mating process (LAAM) fine-tuned a centroid. The task clusters were processed depending on quality of service (QoS) and time of task. Finally, the hybrid method was named moth search adapted sealion optimization (MS-SLnO) to allocate resources by considering power usage effectiveness (PUE), utilization of CPU, and computation time. The presented method efficiently solved the issue of local optima; however, its execution time was high.

Singh *et al.* [18] introduced the flower pollination-non dominated sorting optimization (FP-NSO) that increased resource usage and reduced energy consumption, and minimized carbon emission in data center. Multi-resource constraint measures were integrated with an introduced method that helped identify the most appropriate PMs to deploy VMs in a cloud environment. The introduced method had cost unpredictability and restricted execution because of the difficulties in identifying necessities. Vhatkar and Bhole [19] implemented the whale random update-based lion algorithm (WR-LA) was an integration of the lion algorithm (LA) and whale optimization algorithm (WOA) for optimal resource allocation. The solution of optimal resource allocation was developed through choosing factors such as balanced cluster use, entire network distance, system failure, and threshold distance. The implemented method had high complexity, resource overhead, and less scalability. Thakur and Goraya [20] developed the phasor particle swarm optimization (PPSO) and dragonfly algorithm (DA) for generating the optimum resource allocation to balance load in the cloud. The developed method utilized DA features for distracting the search agent from a space of the worst solution through a novel algorithm. However, the developed method had a high execution time. Hence, scheduling is developed to ensure optimal resource utilization for maximum effectiveness. In small finance organization, the optimization algorithm was employed for optimizing the resource distribution like capital, employees or operational assets effectively and adaptively.

Alatawi [21] presented a reinforcement learning (RL)-driven adaptive resource allocation model to address these challenges. The presented model used Markov decision process (MDP) with dynamic states as resource allocation, which consisted of latency, resource utilization, and energy consumption. The reward function was developed to optimize throughput, latency, and energy efficiency when tenants were ensured. Bharathi *et al.* [22] developed one-to-one-based optimizer with priority and load balancing (O2O-PLB) that enhanced resource allocation in fog-cloud settings. O2O-PLB adopted a priority-based algorithm and also employed tasks in accordance with system drawbacks and available resources, when their load balancing feature ensured consistent distribution of tasks to prevent congestion. Rafique [23] developed a machine learning-based rate-adaptive service provisioning framework (ML-RASPF) for effective service delivery. The ML-RASPF optimized latency and service delivery rate by predictive analysis and adaptive control over a modular edge-cloud model. The developed model formulated task provisioning as a joint optimization issue, which aimed at reducing service latency and increasing delivery throughput.

Cloud computing has become crucial to deliver scalable and cost-effective services, though resource allocation remains a significant challenge because of dynamic workloads, heterogeneous VMs and uncertain user demands. Existing optimization algorithms suffer from high execution time, poor convergence, and over provisioning of resources causes increased energy consumption, minimized resource utilization and high operational costs. To overcome these issues for small organizations, where effective cloud resource management directly impacts cost savings, transaction continuity, and customer satisfaction. The objective of this research is to develop a Gaussian-based convergence factor (GCF) – squirrel search algorithm (SSA) to obtain resource allocation in cloud environments. Particularly, the proposed model aims to enhance convergence rate and search ability of traditional SSA by combining Tent chaotic map and GCF, thereby reducing energy consumption, execution time and makespan while increasing resource utilization. The main aim is to provide effective, adaptive and cost-effective resource allocation strategy for small financial

organizations process under constraints. The contributions of this manuscript are given as follows. The GCF with SSA is developed to allocate optimal resources in the cloud while ensuring deadline constraints. The Tent chaotic map and GCF are included in a traditional SSA-enhanced search capability and convergence rate of SSA to allocate optimal resources in the cloud. The GCF with SSA effectively optimize the cloud resources for a small finance organization, thereby minimizing costs. This ensures a seamless transaction process, which is essential for customer satisfaction and business continuity.

This research paper is organized as follows. Section 2 provides details of a proposed model. Section 3 discusses the outcomes of proposed model. The conclusion of this research work is in section 4.

2. METHOD

The aim of the proposed GCF with SSA method is to assess the resources of users and continuously monitor the status of the whole VM samples for avoiding probability of over- or under-resourcing provisioning. Additionally, the proposed GCF with SSA can make decisions at processing time and select good resources for processing of end-users and enhance the performance of indicator parameters. This method has different components like the cloud layer, resource manager, resource provisioning and de-provisioning, allocation strategy, and mapping PM with VM. By evaluating the organization's workloads, which include transaction volume, peak utilization times, and performance requirements, and by analyzing the organization's transaction volume, peak utilization times and performance requirements, determine the appropriate amount and type of cloud resources required, which develops the basis to develop the cloud infrastructure. The correct instance types that align with the requirements of the organization are selected. This includes deciding between on-demand, reserved or spot instances. Resources like capital and employees are treated as variables to be allocated across various financial products, branches or operational units. Figure 1 represents the architecture of resource allocation in a cloud environment.

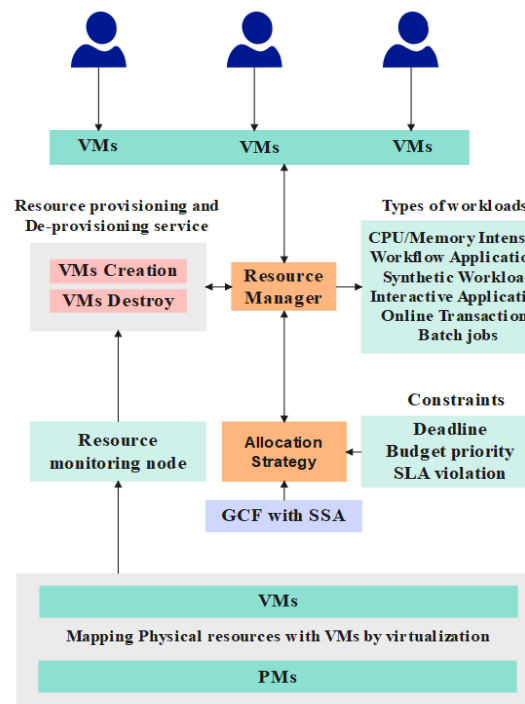


Figure 1. Resource allocation in a cloud environment

The cooperative societies, which was registered in India under 1904 act on initial 5 to 6 years are described: Rajahauli Village Bank, Jorhat, Jorhat Cooperative Town Bank, and Charigaon Village Bank, Jorhat, Assam (1904), Tirur Primary Agricultural Cooperative Bank Ltd., Tamil Nadu (1904), Agricultural Service Cooperative Society Ltd., Devgaon, Piparia MP (1905), Bains Cooperative Thrift and Credit Society Ltd., Punjab (1905) Bilipada Service Cooperative Society Ltd., Orissa (1905), Government of India, Secretariat Cooperative Thrift & Credit Society (1905), Kanginhal Vyvasaya Seva Sahakari Bank Ltd., Karnataka (1905), Kasabe Tadvale Cooperative Multi-Purpose Society, Maharashtra (1905), Premier Urban Credit Society of Calcutta, West Bengal (1905), Chittoor Cooperative Town Bank, Andhra Pradesh (1907),

and Rohika Union of Cooperative Credit Societies Ltd., Bihar (1909). Under this act, numerous non-credit initiatives come like Triplicane society on Madras that run consumer store, weaver credit cooperatives on Dharwar and Hubli that gave credit in a form of yarn.

2.1. Cloud layer

In a cloud layer, a user submits n applications $(\zeta_1, \zeta_2, \zeta_3, \dots, \zeta_n)$ for execution in the server of data center, which includes the PM, which is represented as $PM = \{PM_1, PM_2, PM_3, \dots, PM_N\}$. The PM includes multiple heterogeneous VMs, which is represented as $(\rho_1, \rho_2, \rho_3, \dots, \rho_m)$ for accomplishing the end-users demands. The number of VMs deployed in workstations are based on the rest resource capacity of individual workstation and also the resources needed through applications. Every task ζ_1 having length of $l(\zeta_1)$ and integrated with a deadline $\delta(\zeta_1)$. Each upcoming application is scheduled over heterogeneous resources such as memory (MM_{mem}), processing speed (ρ_s), storage (s_c), number of CPU based on demands. The end-user expects to execute each task with high throughput, less time and financial cost, when service providers try to get high gain from the resources of datacenters [24]. All upcoming applications are analyzed and allocated to a good resource to process, or else the request is rejected. Finally, the proposed GCF with SSA is used to allocate optimal requested tasks to available VMs. The acknowledgement is sent when the process is completed.

2.2. Resource manager

This is a primary element of a cloud resource allocation method. Its objective is to interact with various elements to effectively process end-user tasks while ensuring that constraints such as user budgets and deadlines are met. The resource manager gathers diverse users' requests and saves them in the queue of work to process. It analyzes each request according to its constraints and demands, then forwards it to provisioning and de-provisioning elements to develop or deploy VM instances, thereby ensuring optimal resource selection to satisfy the requirements of the request. The list of resources is fed to the allocation strategy that schedules and allocates resources to diverse workloads by utilizing GCF with SSA and returns to the resource manager.

2.3. Resource provisioning and de-provisioning

VM instances were developed and destroyed through this component depending on users' demand and a status of cloud resources. This retrieves cloud resource status from a node of resource monitoring and gathers end-users demands from a resource manager. Node monitoring plays an important role on a GCF with SSA, which continuously monitors cloud resources and sends data to a resource provisioning and de-provisioning parameters to further significant action. When users' demand is extended in processing time, many VM instances are deployed in PM next for avoiding under-provisioning conditions. If end-user demand decreases during processing, the corresponding VM instances are terminated to prevent over-provisioning in the cloud data center. The provision resources are "right-sized" for present workloads, thereby avoiding over-provisioning to minimize costs. The right-sizing includes selecting optimum instance size and types, storage options, and configurations of the network. The introduction of policies of auto-scaling, which automatically adjusts the number of active instances depending on real-time demand. This ensures that the resource is scaled up in high demand and scaled down in low demand, optimizing performance and costs.

2.4. Mapping the PMs with VMs

The effectiveness of a proposed GCF with SSA is based on mapping among PM and VM instances in a datacenter of cloud. These challenges are critical due to the heterogeneity of both PMs and VMs as well as unpredictable user demands. The virtual machine monitor (VMM) is essential for deploying multi VM instances over PM by virtualization. The goal of VMM is to position VM instances across the most appropriate PM and enhance resource usage.

2.5. Squirrel search algorithm

This algorithm is a swarm intelligence-based approach that mimics the foraging behavior of squirrels, and its effective technique of locomotion is called gliding. In a small finance organization, the algorithm maximizes profits and customer satisfaction and minimizes risk through optimizing the loan portfolio, which are the main objective functions [25]. In SSA, there are four essential phases, which are described as follows.

- i) In the forest, there has n squirrels and n trees, 1 squirrel on 1 tree.
- ii) These n trees are hickory trees and acorn trees $N_{fs} (1 < N_{fs} < n)$ and the remaining trees are normal.
- iii) There are 3 kinds of trees on forest. Hickory trees contain a good food source; acorn trees contain a normal food source and normal trees contain null food.

- iv) Every squirrel searches for food individually and utilizes available food sources using its dynamic foraging behavior. Figure 2 represents the process of GCF with SSA.

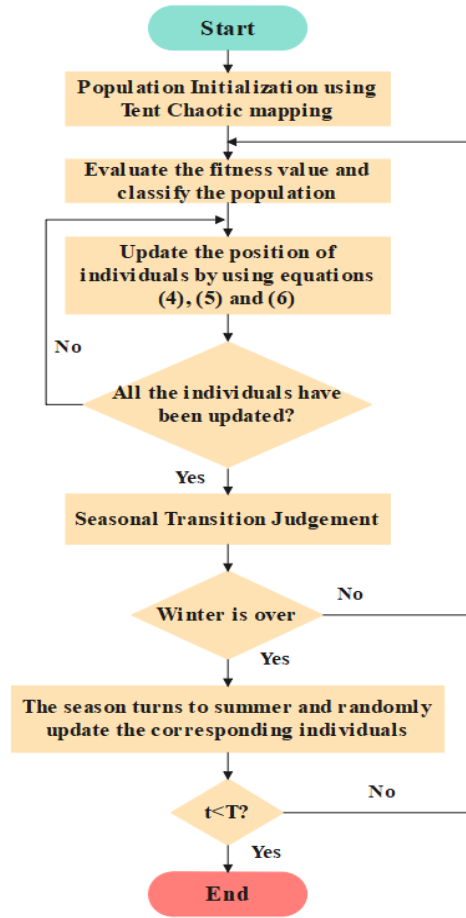


Figure 2. Process of GCF with SSA

2.5.1. Population initialization

The initialization of squirrels is performed by using the Tent chaos of population (30) and maximum iteration (100). The search space has possible configurations of resource allocations. Each squirrel describes various resource allocation to different tasks in the cloud. This research utilized the Tent chaotic map for generating the diverse set of initial solutions, and the mathematical expression is given in (1). This process facilitates the distribution of initial resource allocation strategy throughout the search space, thereby covering a wide range of possibilities. After including the Tent chaotic map, the mathematical expression for the initial population is given in (2).

$$x_{n+1} = \begin{cases} \frac{x_n}{\mu}, & \text{if } x_n < \mu \\ \frac{1-x_n}{1-\mu}, & \text{if } x_n \geq \mu \end{cases} \quad (1)$$

$$FS_i = FS_L + x_{n+1}(1, d) \times (FS_U - FS_L), (i = 1, 2, \dots, n) \quad (2)$$

2.5.2. Population classification

The squirrels' locations are estimated through a fitness function f . Variable for decision (squirrels' position vector) is fed as input to a fitness function and respective outcome $f(FS_i) = f(FS_{i1}, FS_{i2}, \dots, FS_{id})$, ($i = 1, 2, \dots, n$) represents fitness score of i th squirrel. Fitness values describe a food source quality fought through i th squirrel. The squirrels in hickory tree (FS_h) is the one that has less fitness values. Squirrels in acorn tree (FS_a) describes individuals that have fitness value range from $2 - N_{fs} + 1$. The balanced individual squirrels are described as normal tress (FS_n).

2.5.3. Position update

The squirrels' position is updated through gliding for trees of various species. n_1 squirrels which are on normal trees move towards acorn trees, n_2 squirrels move towards hickory trees. n_3 squirrels on acorn trees move towards hickory trees. GCF a is incorporated in the traditional position updated formula of SSA, that improves search capability and convergence rate of SSA in resource allocation. The mathematical expression for GCF is given in (3).

$$a_{GCF} = \begin{cases} \phi \cdot \frac{1}{\sqrt{2\pi}(T_{max}/3)} e^{\frac{t^2}{2(T_{max}/3)^2}}, & t \leq \delta T_{max} \\ \phi \cdot \frac{1}{\sqrt{2\pi}(T_{max}/3)} e^{\frac{t^2}{2(T_{max}/3)^2}}, & \delta T_{max} \leq t < T_{max} \end{cases} \quad (3)$$

Where ϕ represents decreasing function, which changes with the number of iterations, the T_{max} is highest number of iterations, and the t is the present iteration number. The convergence factor a depends on Gaussian distribution that initially allows much exploration and gradually minimizes, which allows smooth transition from exploration to exploitation of the algorithm. The mathematical expression of location update after incorporating the GCF is given in (4) to (6).

$$FS_i^{t+1} = \begin{cases} FS_i^t + a_{GCF} \times d_g \times G_c \times (FS_h^t - FS_i^t), & R \geq P_{dp} \\ \text{random location}, & \text{or else} \end{cases} \quad (4)$$

$$FS_i^{t+1} = \begin{cases} FS_i^t + a_{GCF} \times d_g \times G_c \times (FS_a^t - FS_i^t), & R \geq P_{dp} \\ \text{random location}, & \text{or else} \end{cases} \quad (5)$$

$$FS_i^{t+1} = \begin{cases} FS_i^t + a_{GCF} \times d_g \times G_c \times (FS_h^t - FS_a^t), & R \geq P_{dp} \\ \text{random location}, & \text{or else} \end{cases} \quad (6)$$

Where t represents the present iteration, R represents a random number in $[0,1]$ range and P_{dp} represents the probability of predator presence. The G_c represents the gliding constant of value 1.9, d_g represents the random gliding distance and its mathematical expression is given in (7), where the h_g and sf represent the constant value of 8 and 18 respectively and the $\tan(\varphi)$ is the gliding angle. The mathematical formula for calculating gliding angle is in (8), where the D represents drag force and the L represents lift force. The mathematical expression is given in (9) and (10), where ρ represents the air density, V represents the squirrel's gliding speed, the S represents the surface area, the C_D represents the coefficient of frictional drag and, the C_L represents the random number.

$$d_g = \frac{h_g}{\tan(\varphi) \times sf} \quad (7)$$

$$\tan(\varphi) = \frac{D}{L} \quad (8)$$

$$D = \frac{1}{2\rho V^2 S C_D} \quad (9)$$

$$L = \frac{1}{2\rho V^2 S C_L} \quad (10)$$

2.5.4. Seasonal transition judgement

The condition for seasonal monitoring is used in SSA that supports a method to jump out of local optima. In the initial phase of every iteration, the SSA needed whole population in a winter state, which represents the updated whole individuals. While whole individuals are updated, the mathematical formula for change in season is given in (11) and (12).

$$S_c^t = \sqrt{\sum_{k=1}^d (FS_{h,k}^t - FS_{aj,k}^t)^2} \quad j = 1, 2, \dots, N_{fs} \quad (11)$$

$$S_{min} = \frac{10e^{-6}}{(365)^{t/(T/2.5)}} \quad (12)$$

Where T represents the highest amount of iterations. While seasonal constant is less than ($S_c^t < S_{min}$), season changes from winter to summer. While season changes, whole individuals glide to F_h stayed in

updated locations and whole individuals to F_a without the predator relocating its location and the mathematical formula is given in (13). The mathematical formula for Levy distribution is measured using (14). In (15), the r_a and r_b represents the normally distributed random number $[0,1]$, the β is constant value and mathematical formula for calculate in ξ is in (16). In (15), the $\Gamma(x) = (x - 1)$.

$$FS_i^{new} = FS_L + Levy(n) \times (FS_U - FS_L) \quad (13)$$

$$Levy(x) = 0.01 \times \frac{r_a \times \xi}{|r_b|^{\frac{1}{\beta}}} \quad (14)$$

$$\xi = \frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})^{\frac{1}{\beta}}}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{\frac{\beta-1}{2}}} \quad (15)$$

2.5.5. Objective function

In a small finance organization, it is essential to optimize the loan portfolio through allocating funds across various loans to maximize returns while minimizing risk. The mathematical formula for the objective is given as (16). In (16), i represents various loan products. The developed GCF with SSA allocates the optimal resources in the cloud while ensuring deadline constraints.

$$Maximize \sum(Loan Amount_i \times Interest Rate_i) - \sum(Default Risk_i) \quad (16)$$

The GCF with SSA effectively optimize the cloud resources for small finance organization, thereby minimizing costs. This ensures a seamless transaction process, which is essential for customer satisfaction and business continuity. The GCF with SSA handles multiple types of objective functions like maximizing the profit, minimizing the cost. Algorithm 1 provides GCF-SSA for cloud resource allocation.

Algorithm 1. GCF-SSA for cloud resource allocation

Input: tasks $T = \{t_1, t_2, \dots, t_n\}$, $VM = \{vm_1, vm_2, \dots, vm_m\}$, N population size, maxiter maximum iterations, Predator probability P_{pred} .

Output: optimal task-VM allocation A^*

Initialize squirrel population $S = \{S_1, S_2, \dots, S_N\}$ using Tent chaotic map to improve diversity

Every squirrel represents a candidate task-VM allocation

Set iteration count $iter = 1$

For every squirrel S_i , calculate fitness by multi-objective function

$$Fitness = \alpha E + \beta T_{exec} + \gamma Makespan - \delta Utilization + \lambda Penalty$$

Sort squirrels based on fitness

Employ

Best squirrel $\rightarrow$ Hickory tree

Next best $\rightarrow$ Acorn trees

Remaining $\rightarrow$ Normal trees

Calculate Gaussian convergence factor

$$GCF(iter) = e^{-\frac{iter^2}{2(MaxIter)^2}}$$

Update positions:

Normal $\rightarrow$ move towards acorn

Acorn $\rightarrow$ move toward hickory

Hickory $\rightarrow$ fine local search

If $rand > P_{pred}$ employ gliding motion,

else

perform random walk (Levy flight)

If seasonal condition is met, reallocate squirrels by Levy distribution to escape local optima

Re-evaluate fitness of updated population

Update global best solution A^*

If $iter = MaxIter$ or convergence obtains

Otherwise, set $iter = iter + 1$ and repeat from step 3

Allocate tasks to VMs based on optimal solution A^*

3. RESULTS AND DISCUSSION

The performance of developed GCF with SSA is simulated with CloudSim tool and the configurations used are i5 processor, 8 GB RAM, and Windows 10 OS. The performance metrics used to evaluate the developed GCF with SSA are energy consumption, Makespan, execution time and resource utilization. The analysis of these metrics for the proposed model is described.

3.1. Analysis of energy consumption

In Figures 3 and 4, the energy consumption of GCF with SSA is evaluated based on the number of tasks. In scenario 1, performance of GCF with SSA is analyzed with 30 VMs and in scenario 2, the performance of GCF with SSA is analyzed with 60 VMs. The energy (in Joule) is needed to execute the task with minimum, which results in high network lifetime. The existing algorithms used to evaluate the GCF with SSA are tunicate swarm optimization (TSO), reptile search algorithm (RSA), WOA, and traditional SSA. In scenario 1, the GCF with SSA consumed minimal energy of 0.505 J for 100 tasks, 0.532 J for 150 tasks, 0.579 J for 200 tasks, 0.602 J for 250 tasks and 0.632 J for 300 tasks. In scenario 2, the GCF with SSA consumed minimal energy of 0.541 J for 100 tasks, 0.584 J for 150 tasks, 0.605 J for 200 tasks, 0.628 J for 250 tasks, and 0.664 J for 300 tasks. When compared with existing algorithms, the proposed GCF with SSA consumed minimal energy and showed effective performance.

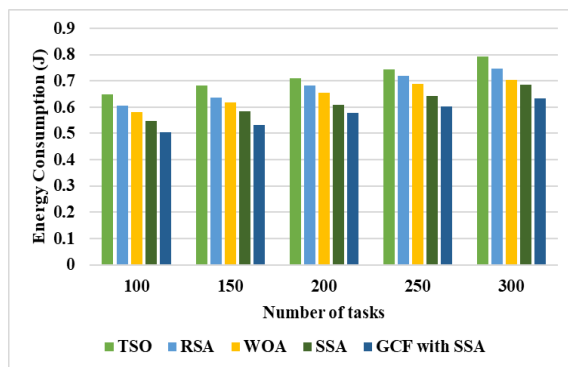


Figure 3. Number of tasks vs energy consumption (VM = 30)

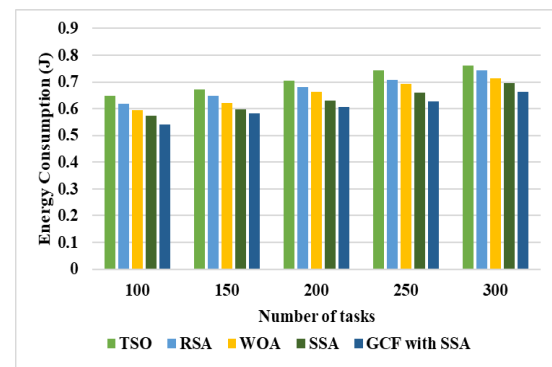


Figure 4. Number of tasks vs energy consumption (VM = 60)

3.2. Analysis of execution time

In Figures 5 and 6, the execution time of GCF with SSA is evaluated based on some tasks. In scenario 1, the performance of GCF with SSA is analyzed with 30 VMs and in scenario 2, performance of GCF with SSA is analyzed with 60 VMs. To enhance the accuracy of PM, VMs are required to process the tasks with less execution time. Existing algorithms used to evaluate the GCF with SSA are TSO, RSA, WOA, and traditional SSA. In scenario 1, the GCF with SSA consumed minimal time for completion of 0.472 s for 100 tasks, 0.432 s for 150 tasks, 0.392 s for 200 tasks, 0.365 s for 250 tasks, and 0.304 s for 300 tasks. In scenario 2, the GCF with SSA consumed minimal completion time of 0.491 s for 100 tasks, 0.446 s for 150 tasks, 0.414 s for 200 tasks, 0.383 s for 250 tasks and 0.343 s for 300 tasks. Compared to existing algorithms, proposed GCF with SSA consumed minimal completion time and showed effective performance.

3.3. Analysis of makespan

In Figures 7 and 8, the makespan of GCF with SSA is evaluated based on several tasks. In scenario 1, performance of GCF with SSA is analyzed with 30 VMs and in scenario 2, performance of GCF with SSA is analyzed with 60 VMs. Makespan represents the job completion time; the positive outcome is minimal makespan. The existing algorithms used to evaluate the GCF with SSA are TSO, RSA, WOA, and traditional SSA. In scenario 1, the GCF with SSA consumed minimal energy of 0.732 s for 100 tasks, 0.692 s for 150 tasks, 0.652 s for 200 tasks, 0.613 s for 250 tasks, and 0.654 s for 300 tasks. In scenario 2, the GCF with SSA consumed minimal energy of 0.808 s for 100 tasks, 0.793 s for 150 tasks, 0.757 s for 200 tasks, 0.718 s for 250 tasks, and 0.693 s for 300 tasks. Compared with existing algorithms, the proposed GCF with SSA consumed minimal makespan and showed effective performance.

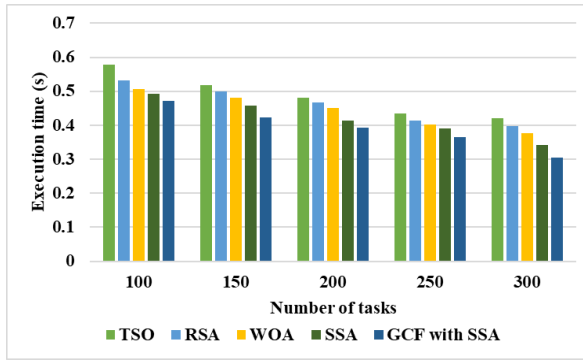


Figure 5. Number of tasks vs. execution time (VM = 30)

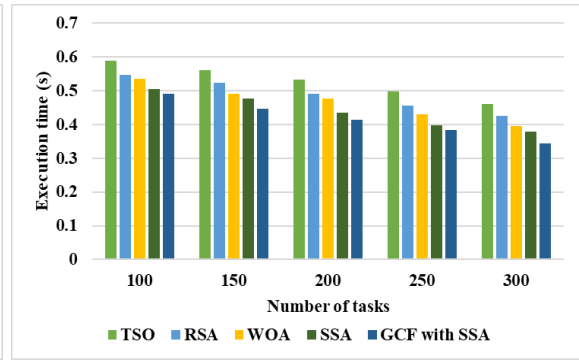


Figure 6. Number of tasks vs. execution time (VM = 60)

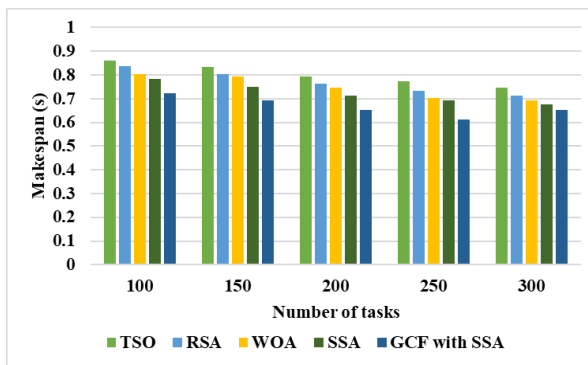


Figure 7. Number of tasks vs. makespan (VM = 30)

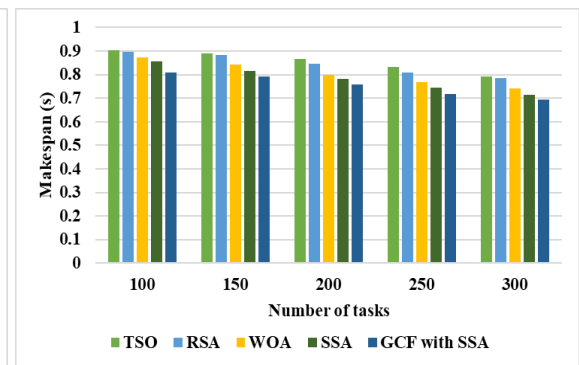


Figure 8. Number of tasks vs. makespan (VM = 60)

3.4. Analysis of resource utilization (%)

In Figures 9 and 10, the resource utilization of GCF with SSA is evaluated based on the number of tasks. In scenario 1, performance of GCF with SSA is analyzed with 30 VMs and in scenario 2, performance of GCF with SSA is analyzed with 60 VMs. The existing algorithms used to evaluate the GCF with SSA are TSO, RSA, WOA, and traditional SSA. In scenario 1, the GCF with SSA reached resource utilization of 51% for 100 tasks, 57% for 150 tasks, 70% for 200 tasks, 79% for 250 tasks, and 85% for 300 tasks. In scenario 2, the GCF with SSA reached resource utilization of 55% for 100 tasks, 61% for 150 tasks, 76% for 200 tasks, 83% for 250 tasks, and 91% for 300 tasks. When compared with existing algorithms, the proposed GCF with SSA reached high resource utilization and showed effective performance.

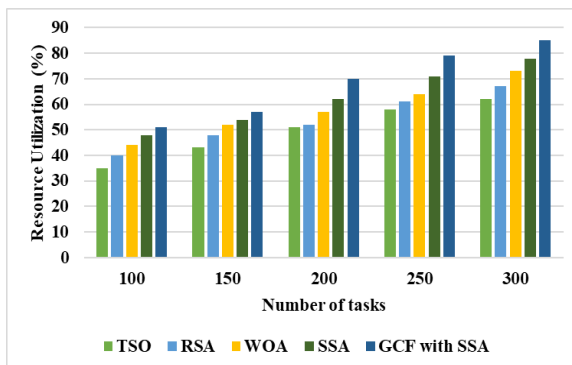


Figure 9. Number of tasks vs. resource utilization (VM = 30)

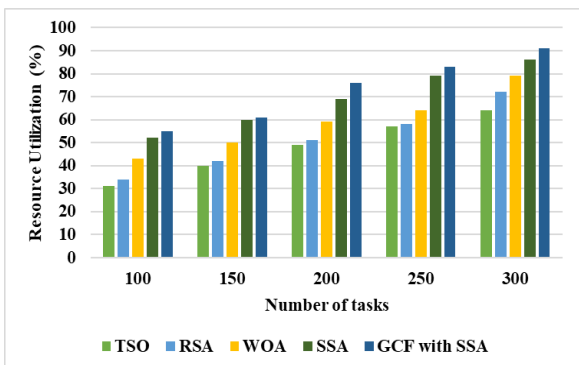


Figure 10. Number of tasks vs. resource utilization (VM = 60)

Table 1 presents a statistical analysis of the proposed GCF with SSA method with conventional models. By integrating GCF and Tent chaotic map, the proposed algorithm accelerates convergence and improves balance between exploration and exploitation resulting in minimized energy consumption of 0.505 J, fast execution time of 0.472 s, and lower makespan of 0.723 s. Additionally, the proposed model obtains superior resource utilization that represents more efficient workload distribution across VMs. The confidence intervals and less standard deviation show consistency across trails, when p-value shows that performance gains due to random variation. These outcomes show robustness, computational efficacy and scalability of GCF with SSA for cloud resource allocation under dynamic workloads.

Table 1. Evaluation of statistical analysis of the proposed model with existing algorithms

Metrics	Methods	Mean $\pm$ SD	Confidence interval	p-value from t-test
Energy consumption (J)	TSO	0.612 $\pm$ 0.019	[0.593, 0.631]	<0.013
	RSA	0.589 $\pm$ 0.017	[0.572, 0.606]	<0.010
	WOA	0.574 $\pm$ 0.021	[0.553, 0.595]	<0.008
	SSA	0.552 $\pm$ 0.018	[0.534, 0.570]	<0.006
	Proposed GCF with SSA	0.505 $\pm$ 0.012	[0.493, 0.517]	<0.003
Execution time (s)	TSO	0.590 $\pm$ 0.015	[0.575, 0.605]	<0.017
	RSA	0.564 $\pm$ 0.014	[0.550, 0.578]	<0.011
	WOA	0.551 $\pm$ 0.016	[0.535, 0.567]	<0.009
	SSA	0.528 $\pm$ 0.012	[0.516, 0.540]	<0.006
	Proposed GCF with SSA	0.472 $\pm$ 0.009	[0.463, 0.481]	<0.004
Makespan (s)	TSO	0.815 $\pm$ 0.020	[0.795, 0.738]	<0.023
	RSA	0.792 $\pm$ 0.019	[0.773, 0.811]	<0.019
	WOA	0.776 $\pm$ 0.018	[0.758, 0.794]	<0.012
	SSA	0.759 $\pm$ 0.017	[0.742, 0.776]	<0.008
	Proposed GCF with SSA	0.723 $\pm$ 0.015	[0.708, 0.738]	<0.005
Resource utilization (%)	TSO	45.5 $\pm$ 1.2	[44.1, 46.9]	<0.027
	RSA	46.2 $\pm$ 1.3	[44.9, 47.5]	<0.018
	WOA	47.1 $\pm$ 1.5	[45.6, 48.6]	<0.013
	SSA	48.3 $\pm$ 1.3	[47.0, 49.6]	<0.007
	Proposed GCF with SSA	51.0 $\pm$ 1.2	[49.8, 52.2]	<0.004

Table 2 represents computational analysis of the GCF with SSA model with existing algorithms like TSO, RSA, WOA, and SSA. The proposed model obtains less training time of 45.3 s and fast inference time of 2.1 ms/task, which ensure fast convergence and real-time adaptability. With less memory usage of 182 MB, minimized floating-point operations (FLOPs) and few parameters, the proposed model determines less computational overhead while maintaining robust optimization. These results shows that GCF with SSA is a lightweight, scalable and practical solution for cloud resource allocation under dynamic workloads.

Table 2. Computational analysis of the proposed model with existing algorithms

Metrics	TSO	RSA	WOA	SSA	GCF with SSA
Training time (s)	64.2	59.8	57.6	61.7	45.3
Inference time (ms/task)	3.8	4.2	3.9	3.6	2.1
Memory usage (MB)	225	240	232	228	182
FLOPs	94	101	97	92	68
Parameters	5,600	6,100	5,800	5,500	4,200

3.5. Comparative analysis

In this section, performance of GCF with SSA is comparing to existing models like AutoCDDQL [16], MS-SLNO [17], and FP-NSO [18] based on task size and a few tasks. In Table 3, performance of GCF with SSA is compared with AutoCDDQL [16] in terms of makespan, response time, resource usage and energy consumption. In Table 4, performance of GCF with SSA is compared with MS-SLNO [17] with metrics of energy consumption, execution time, makespan, and resource utilization. In Table 5, performance of GCF with SSA is compared with FP-NSO [18] with metrics of resource utilization and execution time. When compared with these three existing methods, the proposed GCF with SSA consumed less time and energy and reached the maximum of makespan, response time, and resource utilization. Table 6 represents the comparison of proposed GCF with SSA with deep Q-network (DQN) with RL [21] in terms of latency, throughput and energy efficiency. The proposed model obtains less latency of 105 ms, high throughput of 280 tasks/s, determines its superiority for cloud resource allocation.

Table 3. Comparison of AutoCDDQL

Methods	Task size (kb)	Makespan (s)	Response time (s)	Resource utilization (%)	Energy consumption (kWh)
AutoCDDQL [16]	20	250	2.8	30	75
	40	520	7	65	99
	60	825	15	90	295
	80	1150	19	95	390
Proposed GCF with SSA	20	267	3.5	45	70
	40	545	9	78	84
	60	871	17	95	271
	80	1189	22	97	375

Table 4. Comparison of MS-SLNO

Methods	Number of VMs	Number of tasks	Energy consumption (J)	Execution time (s)	Makespan (s)	Resource utilization
MS-SLNO [17]	30	100	0.740	0.278	0.760	0.740
		200	0.530	0.167	0.123	0.530
		300	0.669	0.275	0.145	0.669
		400	0.605	0.377	0.214	0.605
	60	100	0.868	0.397	0.919	0.868
		200	0.839	0.265	0.123	0.839
		300	0.774	0.209	0.155	0.774
		400	0.782	0.286	0.217	0.782
Proposed GCF with SSA	30	100	0.732	0.256	0.742	0.726
		200	0.521	0.143	0.114	0.515
		300	0.634	0.258	0.126	0.648
		400	0.543	0.361	0.201	0.587
	60	100	0.852	0.372	0.900	0.852
		200	0.817	0.256	0.103	0.814
		300	0.758	0.197	0.137	0.742
		400	0.769	0.263	0.203	0.757

Table 5. Comparison of FP-NSO

Methods	Number of VMs	Resource utilization (%)	Execution time (s)
FP-NSO [18]	100	42.59	3.95
	400	42.47	15.75
	800	42.87	85.57
	1200	42.99	148
Proposed GCF with SSA	100	43.67	3.46

Table 6. Comparison of DQN with RL

Methods	Latency (ms)	Throughput (tasks/s)	Energy efficiency (% improvement)
DQN with RL [21]	120	250	35
Proposed GCF with SSA	105	280	42

3.6. Discussion

The performance outcomes of GCF with SSA are validated with various numbers of tasks and by varying VMs. The results of GCF with SSA are compared with TSO, RSA, WOA, and traditional SSA with metrics of execution time, energy consumption, makespan, and resource utilization. Additionally, it is compared with existing methods like AutoCDDQL [16], MS-SLNO [17], FP-NSO [18], and DQN with RL [21] with different metrics. The GCF with SSA balances workload and assigns many relevant resources to users' applications as well as ensures deadline constraints. Tent chaotic map and GCF are included in traditional SSA, which improved the search capability and convergence rate of SSA to assign optimum resources in the cloud. The performance of GCF with SSA is validated with various measures of execution time, makespan, energy consumption, and resource usage. GCF with SSA handles multiple types of objective functions like maximizing profit and minimizing cost.

4. CONCLUSION

An effective optimization-based resource allocation framework is developed, which safeguards the cloud infrastructure and allocates optimal resources to meet user demands by using a decision-making process. Here, the GCF with SSA is utilized to search and assign good possible resources to requests and ignore the probability of an imbalanced workload. The developed GCF with SSA has a high convergence rate

because it consumes less amount of time to identify optimal resources. The performance of GCF with SSA is validated with various metrics of execution time, makespan, energy consumption, and resource usage. The GCF with SSA obtained lower energy of 0.505 J, lower execution time of 0.472 s, lower makespan of 0.723 s, and high resource usage of 51% for 100 tasks of 30 VMs, which is efficient when compared to existing models. The GCF with SSA handles multiple types of objective functions, like maximizing profit and minimizing cost. In the future, the security of the cloud and resource allocation in the cloud can be performed to further improve the performance of VMs in a cloud environment.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sidagouda Basagouda Patil	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	✓
Mukund Anant Kulkarni		✓				✓		✓	✓	✓	✓	✓		

- C : Conceptualization
- M : Methodology
- So : Software
- Va : Validation
- Fo : Formal analysis
- I : Investigation
- R : Resources
- D : Data Curation
- O : Writing - Original Draft
- E : Writing - Review & Editing
- Vi : Visualization
- Su : Supervision
- P : Project administration
- Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] Z. He, Y. Guo, X. Zhai, M. Zhao, W. Zhou, and K. Li, "Joint computation offloading and resource allocation in mobile-edge cloud computing: a two-layer game approach," *IEEE Transactions on Cloud Computing*, vol. 13, no. 1, pp. 411–428, Jan. 2025, doi: 10.1109/TCC.2025.3538090.

[2] A. Mohammad and Y. Abbas, "Key challenges of cloud computing resource allocation in small and medium enterprises," *Digital*, vol. 4, no. 2, pp. 372–388, Apr. 2024, doi: 10.3390/digital4020018.

[3] S. Gurusamy and R. Selvaraj, "Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network," *Expert Systems with Applications*, vol. 249, Sep. 2024, doi: 10.1016/j.eswa.2024.123554.

[4] Y. He, J. Fang, F. R. Yu, and V. C. Leung, "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: an active inference approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11253–11264, Dec. 2024, doi: 10.1109/TMC.2024.3415661.

[5] Y. Chen, S. Ye, J. Wu, B. Wang, H. Wang, and W. Li, "Fast multi-type resource allocation in local-edge-cloud computing for energy-efficient service provision," *Information Sciences*, vol. 668, May 2024, doi: 10.1016/j.ins.2024.120502.

[6] Y. Zeng *et al.*, "AoI-oriented computation offloading and resource allocation for end-edge-cloud computing systems," *IEEE Internet of Things Journal*, vol. 12, no. 19, pp. 41118–41134, Oct. 2025, doi: 10.1109/JIOT.2025.3591682.

[7] W. Fan, X. Liu, H. Yuan, N. Li, and Y. Liu, "Time-slotted task offloading and resource allocation for cloud-edge-end cooperative computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8225–8241, Aug. 2024, doi: 10.1109/TMC.2024.3349551.

[8] M. E. Ebadi, W. Yu, K. R. Rahmani, and M. Hakimi, "Resource allocation in the cloud environment with supervised machine learning for effective data transmission," *Journal of Computer Science and Technology Studies*, vol. 6, no. 3, pp. 22–34, Jul. 2024, doi: 10.32996/jcsts.2024.6.3.3.

[9] C. Zeng, X. Wang, R. Zeng, Y. Li, J. Shi, and M. Huang, "Joint optimization of multi-dimensional resource allocation and task offloading for QoE enhancement in cloud-edge-end collaboration," *Future Generation Computer Systems*, vol. 155, pp. 121–131, Jun. 2024, doi: 10.1016/j.future.2024.01.025.

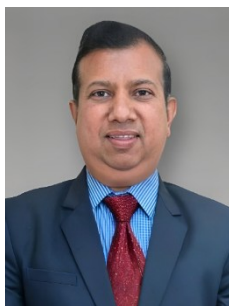
[10] U. M. R. Inkollu and J. K. R. Sastry, "AI-driven reinforced optimal cloud resource allocation (ROCRA) for high-speed satellite imagery data processing," *Earth Science Informatics*, vol. 17, no. 2, pp. 1609–1624, Apr. 2024, doi: 10.1007/s12145-024-01242-5.

- [11] O. Ghandour, S. El Kafhali, and M. Hanini, "Adaptive workload management in cloud computing for service level agreements compliance and resource optimization," *Computers and Electrical Engineering*, vol. 120, Dec. 2024, doi: 10.1016/j.compeleceng.2024.109712.
- [12] J. Anand and B. Karthikeyan, "Dynamic priority-based task scheduling and adaptive resource allocation algorithms for efficient edge computing in healthcare systems," *Results in Engineering*, vol. 25, Mar. 2025, doi: 10.1016/j.rineng.2025.104342.
- [13] H. Zhang, J. Wang, H. Zhang, and C. Bu, "Security computing resource allocation based on deep reinforcement learning in serverless multi-cloud edge computing," *Future Generation Computer Systems*, vol. 151, pp. 152–161, Feb. 2024, doi: 10.1016/j.future.2023.09.016.
- [14] V. Hayyolalam and Ö. Özkasap, "CBWO: a novel multi-objective load balancing technique for cloud computing," *Future Generation Computer Systems*, vol. 164, Mar. 2025, doi: 10.1016/j.future.2024.107561.
- [15] H. Mikram, S. El Kafhali, and Y. Saadi, "HEPGA: a new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment," *Simulation Modelling Practice and Theory*, vol. 130, Jan. 2024, doi: 10.1016/j.simpat.2023.102864.
- [16] S. D. A. Javaheri, R. Ghaemi, and H. M. Naeen, "An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing," *Computing*, vol. 106, no. 2, pp. 371–403, Feb. 2024, doi: 10.1007/s00607-023-01220-7.
- [17] 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, Dec. 2023, doi: 10.1007/s11042-023-15521-8.
- [18] 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, Sep. 2023, doi: 10.1109/JSYST.2023.3248118.
- [19] 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, May 2022, doi: 10.1016/j.jksuci.2019.10.009.
- [20] A. Thakur and M. S. Goraya, "RAFL: a hybrid metaheuristic-based resource allocation framework for load balancing in cloud computing environment," *Simulation Modelling Practice and Theory*, vol. 116, Apr. 2022, doi: 10.1016/j.simpat.2021.102485.
- [21] M. N. Alatawi, "Optimizing multitancy: adaptive resource allocation in serverless cloud environments using reinforcement learning," *Electronics*, vol. 14, no. 15, Jul. 2025, doi: 10.3390/electronics14153004.
- [22] V. C. Bharathi, S. S. Abuthahir, M. Ayyavaraiah, G. Arunkumar, U. Abdurrahman, and S. A. A. Biabani, "O2O-PLB: a one-to-one-based optimizer with priority and load balancing mechanism for resource allocation in fog-cloud environments," *IEEE Access*, vol. 13, pp. 22146–22155, 2025, doi: 10.1109/ACCESS.2025.3536210.
- [23] W. Rafique, "ML-RASPF: a machine learning-based rate-adaptive framework for dynamic resource allocation in smart healthcare IoT," *Algorithms*, vol. 18, no. 6, May 2025, doi: 10.3390/a18060325.
- [24] 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, Jan. 2023, doi: 10.1002/cpe.7469.
- [25] X. Zhang, K. Zhao, L. Wang, Y. Wang, and Y. Niu, "An improved squirrel search algorithm with reproductive behavior," *IEEE Access*, vol. 8, pp. 101118–101132, 2020, doi: 10.1109/ACCESS.2020.2998324.

BIOGRAPHIES OF AUTHORS



Sidagouda Basagouda Patil    currently pursuing his Ph.D. as a research scholar at Bharati Vidyapeeth (Deemed to be University), Institute of Management, Kolhapur. He holds a Master of Computer Applications (MCA) degree and has a strong academic and research inclination toward emerging technologies. His current research focuses on the convergence of cloud computing and image processing, with particular interest in optimizing performance, security, and scalability in cloud-based image analytics. With a solid foundation in computer science, he aims to contribute to advancements in distributed systems and intelligent image processing frameworks. His academic journey is marked by a commitment to interdisciplinary innovation, combining theory with practical applications. He has also participated in various seminars and workshops related to cloud technologies and artificial intelligence. He is passionate about leveraging research to develop efficient, real-world solutions in the field of computing and continues to explore new avenues to expand the impact of his work. He can be contacted at email: ashidu28@gmail.com.



Mukund Anant Kulkarni    distinguished academic and researcher currently serving as an assistant professor and SDE coordinator at Bharati Vidyapeeth Institute of Management, Kolhapur. With a robust educational background, he holds advanced degrees in Engineering, culminating in a Ph.D. that underscores his commitment to academic excellence. He has authored numerous papers published in reputable journals, showcasing his expertise in areas such as machine learning, data analysis, and system optimization. His research not only advances theoretical knowledge but also offers practical applications that benefit industry practices. As a passionate educator, he is dedicated to mentoring students and fostering a collaborative learning environment. His commitment to innovation and research excellence has earned him recognition within the academic community, making him a respected figure and an inspiring role model for aspiring engineers. Through his work, he continues to influence the next generation of thinkers and practitioners in his field. He can be contacted at email: My_mukund@yahoo.com.