The Scientific Temper (2024) Vol. 15 (spl): 45-55
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.06
https://scientifictemper.com/

RESEARCH ARTICLE

Dynamic resource allocation with otpimization techniques for qos in cloud computing

V. Baby Deepa*, R. Jeya

Abstract

Ensuring the quality of service (QoS) in cloud computing environments requires efficient resource allocation mechanisms to manage dynamic workloads and meet user demands. This paper proposes a dynamic resource allocation strategy that integrates gravitational search optimization (GSO) with Harris Hawks optimization (HHO) to optimize resource utilization and maintain QoS in cloud infrastructures. The proposed hybrid approach combines the global search capabilities of GSO, inspired by the law of gravity, with the exploitation and exploration strategies of HHO, mimicking the cooperative hunting behavior of Harris hawks. This synergy enables adaptive and efficient allocation of computational resources based on real-time workload fluctuations, reducing response times, minimizing energy consumption, and preventing Service Level Agreement (SLA) violations. By predicting workload variations and adjusting resource allocation dynamically, the proposed method ensures higher reliability, scalability, and cost-effectiveness compared to traditional resource allocation techniques. Simulation results demonstrate that the GSO-HHO-based approach outperforms conventional optimization algorithms in balancing the trade-offs between performance and resource efficiency, making it a robust solution for maintaining QoS in cloud computing environments.

Keywords: Cloud computing, quality of service, Optimization techniques, Dynamic resource allocation.

Introduction

Resource allocation is a critical component of cloud computing, where computing resources such as CPU, memory, storage, and network bandwidth must be allocated efficiently to meet the demands of users and applications. Cloud computing operates in a highly dynamic environment, where multiple users share infrastructure, and workloads fluctuate in real time. As cloud service providers (CSPs) aim to deliver scalable, flexible, and on-demand services, effective resource allocation becomes crucial to ensure optimal performance, minimize costs, and maintain the quality of service (QoS) commitments laid out in Service Level Agreements (SLAs), Belgacem, A. (2022), Saidi, K., & Bardou, D. (2023), Jawhar, M. M., & Osman, H. M. (2022), Xu, H., Xu, S., Wei, W., & Guo, N. (2023).

Resource allocation in cloud computing involves assigning resources to tasks or virtual machines (VMs) based on the current demand and workload while keeping the infrastructure utilization at optimal levels. The goal is to allocate just enough resources to ensure that all applications receive the necessary resources to function properly without over-provisioning, which can lead to wastage or under-provisioning, which can lead to performance degradation and SLA violations. Efficient resource allocation ensures high availability, low latency, scalability, and energy efficiency while adhering to QoS requirements like response time, throughput, and fault tolerance, Mohamed, Y. A., & Mohamed, A. O. (2022, July), Chen, F., Lu, A., Wu, H., Dou, R., & Wang, X. (2022).

In traditional computing environments, static resource allocation was sufficient, where predefined resources were assigned based on historical workloads. However, cloud environments present unique challenges, such as workload variability, multi-tenancy, and dynamic scaling requirements, making static allocation inefficient. As cloud workloads vary over time, with users running diverse applications that may have unpredictable resource demands, a static allocation

Department of Computer Science, Government Arts College (Autonomous) (Affiliated to Bharathidasan University, Tiruchirappalli), Karur, Tamil Nadu, India.

***Corresponding Author:** V. Baby Deepa, Department of Computer Science, Government Arts College (Autonomous) (Affiliated to Bharathidasan University, Tiruchirappalli), Karur, Tamil Nadu, India, E-Mail: deepamct@gmail.com

How to cite this article: Baby, D.V., Jeya, R. (2024). Dynamic resource allocation with otpimization techniques for qos in cloud computing. The Scientific Temper, **15**(spl):45-55.

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.06

Source of support: Nil

Conflict of interest: None.

Received: 18/09/2024 **Accepted:** 26/09/2024 **Published** : 16/10/2024

approach often results in underutilized resources during periods of low demand or performance bottlenecks during peak loads. Consequently, there is a strong need for dynamic resource allocation mechanisms that can adapt in real time to changing conditions.

Background Study On Resource Allocation

Cloud computing has evolved into a prominent paradigm for delivering on-demand computing services over the internet, enabling users to access and utilize resources such as processing power, storage, and networking without the need for direct infrastructure ownership. The inherent flexibility, scalability, and cost-efficiency of cloud computing make it highly suitable for diverse applications across industries, ranging from web hosting and big data analytics to machine learning and internet of things (IoT) services, Kumar, N., & Kumar, S. (2022), Umer, A., Nazir, B., & Ahmad, Z. (2022).

One of the foundational aspects of cloud computing is the efficient management of resources, which directly impacts performance, cost-effectiveness, and quality of service (QoS). Resource allocation, in particular, is a key process in which available computational resources are dynamically assigned to meet the needs of cloud applications while optimizing for performance, energy consumption, and economic factors, Umer, A., Nazir, B., & Ahmad, Z. (2022), Osypanka, P., & Nawrocki, P. (2022), Shi, F., & Lin, J. (2022).

This background study explores the historical context, resource allocation models, and challenges faced in cloud computing environments, followed by a discussion of existing optimization techniques for dynamic resource allocation.

Resource Allocation Models in Cloud Computing

Resource allocation in cloud computing involves managing several types of resources, including, Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., ... & Zomaya, A. (2016), Naeem, M., Anpalagan, A., Jaseemuddin, M., & Lee, D. C. (2013):

- Compute resources: CPU cycles, memory, and storage.
- Network resources: Bandwidth, latency, and network paths.
- Energy resources: Power consumption and cooling requirements.

Several models have been proposed over the years for resource allocation in cloud computing:

Static Resource Allocation

In this model, resources are allocated to tasks or VMs based on predefined configurations or historical data. Static allocation is easy to implement but suffers from inefficiencies, particularly in highly dynamic cloud environments. This can result in over-provisioning during low-demand periods or under-provisioning during peak demand, leading to SLA violations or resource wastage, Petrovska, I., & Kuchuk, H. (2022).

Dynamic Resource Allocation

Dynamic resource allocation is the process of continuously monitoring workload demands and adjusting the resource allocation in real time. DRA improves resource utilization and efficiency by reallocating resources based on current demand. It is critical for maintaining QoS in cloud environments, as workloads can fluctuate rapidly, making static allocation inadequate, Si Salem, T., Iosifidis, G., & Neglia, G. (2022).

On-Demand Resource Allocation

This model dynamically provisions resources only when they are needed, based on user requests or system demands. This approach underpins the cloud's "pay-as-you-go" billing model, where users are charged for resources consumed on a per-usage basis, Han, H., Bai, X., Hou, Y., & Qiao, J. (2022).

Predictive Resource Allocation

Predictive resource allocation uses machine learning or statistical models to forecast future workload patterns based on historical data. The goal is to proactively adjust resource allocation before demand spikes occur, reducing latency and preventing bottlenecks, Chen, J., Wang, Y., & Liu, T. (2021).

Hybrid Resource Allocation

Hybrid approaches combine multiple models, such as static and dynamic allocation, to achieve an optimal balance between performance and resource utilization. Hybrid models can be particularly useful in cloud environments where different types of workloads coexist (e.g., a mix of long-running batch jobs and latency-sensitive real-time tasks), Teekaraman, Y., Manoharan, H., Basha, A. R., & Manoharan, A. (2023).

Gravitational Search Optimization Approach

Gravitational search optimization (GSO) is a population-based metaheuristic algorithm inspired by Newton's law of gravity and motion. First introduced by Esmat Rashedi in 2009, GSO models individuals (or agents) in the search space as objects that attract one another based on their masses, with the force of attraction being proportional to their fitness. The stronger an agent's fitness (mass), the greater its gravitational pull, which influences the movement of other agents toward it. This enables GSO to explore and exploit the search space in an efficient manner, making it well-suited for solving complex optimization problems, including dynamic resource allocation in cloud computing environments, Hashemi, A., Dowlatshahi, M. B., & Nezamabadi-Pour, H. (2021), Ahmadabadi, J. Z., Mood, S. E., & Souri, A. (2023).

Key Concepts

Agents

Each agent represents a potential solution in the search space. The performance of an agent is evaluated using a fitness function.

Mass

Each agent is assigned a mass based on its fitness; better solutions have higher mass.

Gravity

The gravitational force attracts agents toward better solutions.

Basic Steps of GSO

Initialization

Randomly initialize the positions and masses of the agents.

Fitness evaluation

Calculate the fitness of each agent based on the objective function.

Gravitational force calculation

For each agent, calculate the gravitational force exerted by other agents.

Update positions

Update the positions of the agents based on the calculated gravitational forces.

Iteration

Repeat the fitness evaluation and position update for a predetermined number of iterations or until convergence.

Mathematical Formulation

Gravitational force

The gravitational force between two agents i and j is given by:

$$F_{ij} = G.\frac{m_i.m_j}{d_{ij}^2} \quad (1)$$

Where F_{ij} is the gravitational force between agents i and j. G is the gravitational constant. $m_i\ amd\ m_j$ are the masses of agents i and j. d_{ij} is the distance between agents i and j.

Distance calculation

The distance between two agents i and j in a multi-dimensional space can be calculated as:

$$d_{ij} = \sqrt{\sum_{k=1}^{D}(x_{ik} - x_{jk})^2} \quad (2)$$

Where D is the number of dimensions. x_{ik} and x_{jk} are the coordinates of agents iii and j in dimension k.

Net gravitational force

The net gravitational force acting on agent i from all other agents is calculated as:

$$F_i = \sum_{j=1, j\neq 1}^{N} F_{ij} \quad (3)$$

Where N is the total number of agents.

Acceleration

The acceleration of agent i due to the net gravitational force is given by:

$$a_i = \frac{F_i}{m_i} \quad (4)$$

Position update

The position of agent i is updated using the acceleration:

$$x_i(t+1) = x_i(t) + a_i\Delta t \quad (5)$$

Where $x_i(t)$ is the current position of agent i. Δt is the time step.

Mass assignment

The mass of each agent can be updated based on its fitness as follows:

$$m_i = \frac{f_{best} - f_i}{f_{best} - f_{worst}} \quad (6)$$

Where f_{best} is the fitness of the best agent, f_{worst} is the fitness of the worst agent, and f_i is the fitness of agent i.

Algorithm Steps

Step 1: Initialize the population of agents.
Step 2: Evaluate the fitness of each agent.
Step 3: Calculate the mass for each agent based on fitness.
Step 4: Compute the gravitational forces and update the positions.
Step 5: Repeat steps 2-4 until convergence criteria are met.

Harris Hawks Optimization Approach

Harris Hawks optimization (HHO) is a metaheuristic algorithm inspired by the cooperative hunting strategy of Harris Hawks. It is widely applied in optimization problems, and its dynamic and adaptive nature makes it suitable for complex problems.

HHO mimics the predatory behavior of Harris hawks, particularly their surprise pounce mechanism. The algorithm alternates between exploration (searching for prey) and exploitation (attacking the prey). In cloud computing, HHO can be used to allocate resources (like CPU, memory, and bandwidth) dynamically by optimizing performance metrics such as response time, cost, and energy efficiency, Zivkovic, M., Bezdan, T., Strumberger, I., Bacanin, N., & Venkatachalam, K. (2021), Alabool, H. M., Alarabiat, D., Abualigah, L., & Heidari, A. A. (2021).

Key Components for Dynamic Resource Allocation

Hawks (Agents)

Each hawk represents a potential solution, such as a specific resource allocation configuration.

Prey (Best Solution)

The best resource allocation solution (in terms of fitness, such as cost minimization or performance maximization) is the target for all hawks.

Energy level (Exploration vs Exploitation)

Hawks adapt their behavior based on the "energy" of the system, switching between exploration (searching for a better resource allocation) and exploitation (fine-tuning the current allocation).

HHO Algorithm Phases

Exploration phase

The hawks search for prey by randomly adjusting resource allocation configurations to explore the solution space. The goal is to avoid local optima and find diverse potential configurations.

Transition to exploitation

Based on the hawk's energy, the algorithm dynamically adjusts to move from exploration to exploitation. In cloud computing, this transition corresponds to refining the best-found resource allocation solutions.

Exploitation phase

The hawks attack the prey by narrowing down the solution space and fine-tuning resource allocations. In dynamic resource allocation, this involves optimizing resource usage according to the current workload and performance constraints.

Mathematical Function

Hawk's position update

The position of hawks, representing resource allocation configurations, is updated based on the current prey (best solution) and energy levels.

During exploration

$$X_i(t+1) = X_{random}(t) - r_1 . |X_{random}(t) - 2r_2 . X_i(t)| \quad (7)$$

Where $X_i(t)$ is the current position (resource allocation) of hawk i, $X_{random}(t)$ is a randomly selected position (another resource configuration), r_1 and r_2 are random numbers in [0,1], controlling randomness.

During exploitation (Soft besiege strategy)

$$X_i(t+1) = \Delta X(t) - E . |J . X_p(t) - X_i(t)| \quad (8)$$

Where $X_p(t)$ is the position of the prey (best resource configuration), $\Delta X(t)$ is the difference between current hawk position and prey, and E and J are constants controlling energy dissipation and jump strength.

Energy level (E)

The energy level decreases over time, controlling the transition between exploration and exploitation.

$$E = 2E_0\left(1 - \frac{t}{T}\right) \quad (9)$$

Where E_0 is the initial energy (random between -1 and 1), t is the current iteration and T is the maximum number of iterations.

Escape energy and attack mode

The algorithm switches between different pounce strategies based on the prey's escape energy E.

- When : Exploration is emphasized.
- When : Exploitation is emphasized.

Fitness function

In cloud computing, the fitness function evaluates the performance of resource allocation. It can be formulated based on parameters such as:

- Cost (e.g., cost of virtual machines).
- Energy consumption.
- Response time (time taken to allocate resources).
- Service level agreement (SLA) violations.

Proposed GSO-HHO Based Resource Allocation (GSO-HHO-RA) Approach

The GSO-HHO Based Resource Allocation (GSO-HHO-RA) approach is a hybrid optimization strategy combining gravitational search optimization (GSO) and Harris Hawks optimization (HHO) for efficient resource allocation in cloud computing. By leveraging the strengths of both algorithms, GSO-HHO-RA aims to enhance the performance, cost-efficiency, and energy optimization in dynamic cloud environments where resource demands fluctuate frequently.

Resource allocation is a critical issue in cloud computing environments due to the dynamic and unpredictable nature of workloads. The allocation process involves assigning VMs and other computational resources to user tasks efficiently to meet service level agreements (SLAs) while minimizing costs and energy consumption.

While GSO is excellent in exploring the search space and avoiding local optima, HHO excels in exploitation fine-tuning solutions. By combining these two algorithms, GSO-HHO-RA provides a powerful, dynamic mechanism for allocating resources in cloud environments. The hybrid approach benefits from GSO's ability to handle complex solution spaces and HHO's exploitation power to ensure optimal or near-optimal solutions.

Dynamic nature of cloud workloads

The unpredictable variation in user requests and workloads demands a flexible and adaptive resource allocation approach.

Resource efficiency

Cloud providers need to maximize resource utilization while minimizing operational costs, which requires robust optimization techniques.

Energy consumption

Over-provisioning resources can lead to wasted energy, while under-provisioning can result in performance degradation. A balanced resource allocation scheme is crucial for energy-efficient cloud operations.

The proposed GSO-HHO-RA method integrates the GSO's global search capability with HHO's local search capability. It first uses GSO to perform global exploration of the solution space, searching for promising regions where the best resource allocation configurations might lie. After this, HHO refines the best-found solutions from GSO by performing a local search to further optimize resource allocation.

Procedure for Proposed GSO-HHO-RA

Step 1: Initialize Cloud Environment and Resources

- Define the cloud infrastructure, including the available resources such as VMs, CPU, memory, storage, and bandwidth.
- Identify user workloads or tasks that need resource allocation. These tasks may vary in resource requirements over time.

Agent representation

Each agent represents a candidate resource allocation solution. A candidate solution consists of how the available cloud resources are distributed among the incoming tasks or workloads.

Solution encoding

Each agent's position corresponds to a set of parameters such as:

- Number of VMs allocated to each task.
- CPU and memory distribution.
- Network bandwidth allocation.

Step 2: Define the Fitness Function

The fitness function evaluates the quality of each resource allocation solution. The function should consider multiple objectives, such as:

- Cost: The cost of using VMs, CPUs, storage, and other resources.
- Performance: Metrics such as response time, task execution time, and throughput.
- Energy Efficiency: The energy consumption of the allocated resources.
- SLA Violations: Penalties for not meeting service-level agreements (e.g., performance guarantees).

$F=\alpha.Cost+\beta.Response\ Time+\gamma.Energy\ Consumption+\delta.SLA\ Violations$

Where $\alpha,\beta,\gamma,\delta$ are weighting factors for each objective.

Step 3: Initialization of GSO

Randomly initialize the positions of agents (resource configurations). These positions correspond to the initial resource allocation strategies.

Mass calculation

For each agent, calculate the mass based on its fitness. Agents with better fitness (lower cost, faster response time, etc.) are assigned higher masses. The mass for agent i can be computed with equation (6).

Step 4: Gravitational Search Optimization (GSO) Phase - Global Exploration

Gravitational Force Calculation

Calculate the gravitational force between agents. The force between agent i and agent j is given by equation (1).

Update Positions

Update the positions (resource configurations) of agents based on the forces acting on them. The new position of agent i is computed with equation (5).

Evaluate fitness

After updating positions, evaluate the fitness of the new resource configurations.

Repeat GSO

Continue iterating through the GSO process for a defined number of iterations or until a convergence criterion (e.g., minimal improvement in fitness) is met.

Step 5: Transition to Harris Hawks Optimization (HHO) Phase - Local Exploitation

After the GSO phase, select the best solutions (prey) based on fitness. These are the most promising resource allocation configurations found by GSO.

Initialize hawks

The hawks in HHO are initialized around the best solutions (prey). Hawks represent candidate resource allocation refinements.

Step 6: Exploration and Exploitation in HHO

Energy Calculation

Calculate the energy level E for each hawk based on the iteration count t using equation (9).

Exploration phase

If the hawk's energy $|E| \geq 1$, perform exploration. Hawks explore new positions by adjusting resource allocations randomly around the prey. The position update during exploration is given by equation (7).

Exploitation phase

If $|E| < 1$, switch to exploitation. Hawks focus on refining the best solutions (prey). The position update during exploitation is given by equation (8).

Energy-driven strategy

Depending on the energy level, different pounce strategies (soft besiege or hard besiege) are applied to fine-tune the solutions.

Step 7: Update Fitness and Select Best Resource Allocation

- After each iteration of HHO, evaluate the fitness of the new solutions and compare them with the current best solution (prey).
- If a better solution is found, update the prey (best resource allocation configuration).
- Repeat the HHO process for a specified number of iterations or until convergence is achieved.

Step 8: Final Resource Allocation Decision

- After completing both the GSO and HHO phases, the best resource allocation configuration (prey) is selected as the final solution.
- This configuration specifies how resources (VMs, CPU, memory, bandwidth) will be dynamically allocated to tasks in the cloud environment.

Step 9: Termination

- The algorithm terminates when the predefined stopping criteria are met (e.g., a maximum number of iterations convergence to a specific fitness level).
- The final solution is applied to allocate cloud resources in real time to the incoming tasks.

Result And Discussion

Response time (RT)

Response time is the time taken from the submission of a task to the allocation of resources and task execution.

RT = Task completion time–Task submission time.

Table 1 depicts the Response Time (in Milliseconds) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO with a number of hosts = 400 and a number of VMs 100 with varying numbers of tasks starting from 100 to 900.

From Table 1, GSO-HHO-RA (Proposed) consistently achieves the lowest response times, indicating that the hybrid approach effectively balances global exploration and local exploitation, leading to more efficient resource allocation and faster task execution. GSO performs better than PSO and slightly worse than HHO. It struggles with local exploitation, resulting in longer response times as the number of tasks increases. HHO performs better than PSO due to its effective local exploitation strategy but still falls behind GSO-HHO-RA because it lacks the global search capability that GSO provides. PSO shows the highest response times across all task loads. While PSO is a well-known optimization algorithm, it doesn't perform as well in highly dynamic environments with multiple tasks and resource constraints.

Table 2 depicts the response time (in Milliseconds) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO with a number of hosts = 800 and number of VMs 200 with varying numbers of tasks starting from 100 to 900.

From Table 2, GSO-HHO-RA (Proposed) continues to demonstrate the lowest response times, showing its strong adaptability to increasing resource availability (800 hosts and 200 VMs) and varying workloads. The hybrid GSO-HHO mechanism maintains efficient resource allocation even as the number of tasks increases. GSO performs well but experiences slightly higher response times as the number of tasks increases. The algorithm's performance is less optimal than GSO-HHO-RA because of its slower convergence to the best resource allocation solution. HHO also achieves better performance than PSO, particularly in handling smaller workloads, but begins to show limitations with larger task sizes, indicating that it lacks the balance between global exploration and local exploitation provided by the hybrid GSO-HHO-RA. PSO consistently shows the highest response times across all tasks. It struggles to manage larger workloads, which causes inefficiencies in resource allocation, resulting in higher response times.

Execution Time (ET)

The total time required to execute a given workload after resource allocation.

ET = End time start time of the task execution

Table 3 depicts the Execution Time (ET) (in milliseconds) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 400 and the number of VMs is 100.

GSO-HHO-RA (Proposed) again demonstrates the lowest execution time for all task counts, proving its efficiency in handling resource allocation and task execution in a constrained environment with fewer hosts (400) and VMs (100). GSO exhibits a slightly higher execution time than GSO-HHO-RA but performs better than PSO and marginally worse than HHO. Its slower convergence leads to higher execution times as the number of tasks increases. HHO performs well, with execution times lower than PSO and close to GSO but still higher than the hybrid GSO-HHO-RA. HHO benefits from strong local exploitation but lacks GSO's

Table 1: Response time (in Milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with Number of hosts = 400 and Number of VMs = 100

Number of tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	320	380	350	400
200	335	395	365	420
300	345	410	375	435
400	360	430	395	455
500	375	445	410	470
600	390	460	425	485
700	405	475	440	500
800	420	490	455	515
900	435	505	470	530

Table 2: Response time (in Milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with number of hosts = 800 and Number of VMs = 200

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	290	350	320	370
200	305	365	335	390
300	315	380	350	405
400	330	395	370	420
500	345	410	385	440
600	360	425	400	455
700	375	440	415	470
800	390	455	430	485
900	405	470	445	500

global exploration capabilities. PSO consistently shows the highest execution times due to its slower resource allocation process, especially with a higher number of tasks, resulting in less efficient execution overall.

Table 4 depicts the execution time (ET) (in milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 800 and the number of VMs is 200.

GSO-HHO-RA (Proposed) achieves the lowest execution time across all task sizes, indicating that it is highly efficient in handling task scheduling and resource allocation, ensuring that tasks are executed quickly and efficiently. GSO performs better than PSO but slightly worse than HHO. While it is a strong optimization method, it takes more time to converge to the optimal resource allocation, resulting in slightly higher execution times compared to GSO-HHO-RA. HHO performs well, showing a clear advantage over PSO due to its effective exploitation phase, which helps reduce execution time. However, it lacks the hybrid mechanism of GSO-HHO-RA, which further optimizes resource allocation. PSO consistently shows the highest execution times among all approaches, struggling to allocate resources efficiently as the number of tasks increases. Its slower convergence to the optimal solution results in higher execution times.

Energy Consumption (EC)

The total energy consumed by cloud resources (e.g., servers, virtual machines) during task execution.

Table 5 depicts the Energy Consumption (EC) (in milliseconds) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 400 and the number of VMs is 100.

GSO-HHO-RA (Proposed) consistently achieves the lowest energy consumption, indicating that the hybrid approach is highly energy-efficient. The combined strengths of GSO's global exploration and HHO's local exploitation ensure that resources are allocated optimally, minimizing unnecessary energy usage. GSO performs better than PSO but consumes slightly more energy compared to HHO. It is less energy-efficient than GSO-HHO-RA due to its slower convergence and higher number of iterations for finding the optimal solution. HHO performs well, with energy consumption lower than GSO and PSO, but it is still higher than GSO-HHO-RA. HHO's local exploitation helps in reducing energy consumption, but it lacks the global optimization capability that makes GSO-HHO-RA more energy-efficient. PSO consistently exhibits the highest energy consumption. The slower convergence and less efficient resource allocation lead to higher energy usage, especially as the number of tasks increases.

Table 6 depicts the ET (in milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 800, and the number of VMs is 200.

GSO-HHO-RA (Proposed) shows the lowest energy consumption across all task counts, indicating that the hybrid optimization approach effectively reduces

Table 3: Execution time (in Milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with number of hosts = 400 and Number of VMs = 100

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	580	640	610	665
200	600	660	630	685
300	620	680	650	705
400	640	700	670	725
500	660	720	690	745
600	680	740	710	765
700	700	760	730	785
800	720	780	750	805
900	740	800	770	825

Table 4: Execution time (in Milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with Number of hosts = 800 and Number of VMs = 200

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	550	610	580	645
200	570	630	600	665
300	590	650	620	685
400	610	670	640	705
500	630	690	660	725
600	650	710	680	745
700	670	730	700	765
800	690	750	720	785
900	710	770	740	805

Table 5: Energy consumption (in kWh) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with Number of hosts = 400 and number of VMs = 100

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	1.85	2.1	2	2.2
200	1.95	2.2	2.1	2.3
300	2.05	2.3	2.2	2.4
400	2.15	2.4	2.3	2.5
500	2.25	2.5	2.4	2.6
600	2.35	2.6	2.5	2.7
700	2.45	2.7	2.6	2.8
800	2.55	2.8	2.7	2.9
900	2.65	2.9	2.8	3

unnecessary energy usage. The combination of global exploration from GSO and local exploitation from HHO contributes to optimal resource allocation and energy efficiency. GSO demonstrates improved energy efficiency compared to PSO but consumes slightly more energy than HHO. Its performance is hindered by a longer convergence time, resulting in increased energy consumption as the number of tasks grows. HHO performs well, with energy consumption lower than GSO and PSO, due to its strong local optimization capabilities. However, it still cannot match the energy efficiency of the GSO-HHO-RA approach. PSO consistently exhibits the highest energy consumption, as its slower resource allocation leads to inefficient use of energy resources. This inefficiency becomes more pronounced with an increasing number of tasks.

Resource Utilization (RU)

The percentage of cloud resources utilized during task execution relative to the total available resources.

Table 7 depicts the Resource Utilization (in %) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO

Table 6: Execution time (in Milliseconds) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with Number of hosts = 800 and Number of VMs = 200

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	1.5	1.8	1.7	1.9
200	1.6	1.9	1.8	2
300	1.7	2	1.9	2.1
400	1.8	2.1	2	2.2
500	1.9	2.2	2.1	2.3
600	2	2.3	2.2	2.4
700	2.1	2.4	2.3	2.5
800	2.2	2.5	2.4	2.6
900	2.3	2.6	2.5	2.7

approaches, where the number of hosts is 400 and the number of VMs is 100.

GSO-HHO-RA (Proposed) achieves the highest resource utilization across all task counts, demonstrating its efficiency in allocating resources effectively. This indicates that the hybrid approach optimally uses available resources, maximizing overall system performance. GSO performs well but shows lower utilization rates compared to GSO-HHO-RA. Its effectiveness decreases as the number of tasks increases, likely due to less optimal resource allocation strategies. HHO provides better resource utilization than PSO but does not reach the levels achieved by the proposed hybrid approach. While it excels in local optimization, it lacks the broader exploration capability that GSO-HHO-RA provides. PSO consistently exhibits the lowest resource utilization among all approaches. Its inefficiencies in resource allocation led to underutilization of the available VMs and hosts, especially as the number of tasks increased.

Table 8 depicts the resource utilization (RU) (in %) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 800 and the number of VMs is 200.

GSO-HHO-RA (Proposed) achieves the highest resource utilization across all task counts, indicating its ability to effectively allocate resources and maximize performance in a larger setup with 800 hosts and 200 VMs. GSO shows good resource utilization but does not reach the levels achieved by GSO-HHO-RA. Its lower performance indicates that it may struggle with optimal resource allocation as the number of tasks increases. HHO performs better than PSO but still falls short of the utilization rates achieved by the proposed hybrid approach. While it provides decent local optimization, it lacks the comprehensive global exploration offered by GSO-HHO-RA. PSO consistently exhibits the lowest resource utilization among all approaches. Its inefficiencies in resource allocation lead to underutilization, particularly as the number of tasks grows.

Table 7: Resource utilization (RU) (in %) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with Number of hosts = 400 and Number of VMs = 100

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	85	80	82	78
200	87	82	84	79
300	88	83	85	80
400	89	84	86	81
500	90	85	87	82
600	91	86	88	83
700	92	87	89	84
800	93	88	90	85
900	94	89	91	86

Table 8: Resource utilization (RU) (in %) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with number of hosts = 800 and number of VMs = 200

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	88	82	85	80
200	90	84	87	81
300	91	85	88	82
400	92	86	89	83
500	93	87	90	84
600	94	88	91	85
700	95	89	92	86
800	96	90	93	87
900	97	91	94	88

Service-Level Agreement (SLA) Violation Rate (SVR)

The percentage of tasks that fail to meet the performance guarantees outlined in the service-level agreements (SLAs).

Table 9 depicts the SLA (in %) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 400 and the number of VMs is 100.

GSO-HHO-RA (Proposed) achieves the lowest SLA violation rate across all task counts, indicating its effectiveness in meeting service level agreements (SLAs) and ensuring timely task completion. The hybrid approach efficiently allocates resources to minimize SLA violations. GSO shows a higher SLA violation rate than GSO-HHO-RA, reflecting its challenges in optimizing resource allocation effectively as the number of tasks increases. HHO performs better than PSO but still has a higher SLA violation rate compared to the proposed hybrid approach. Its local optimization capabilities help reduce violations, but it does not reach the same level of performance as GSO-HHO-RA. PSO consistently exhibits the highest SLA violation rate among all approaches, indicating significant inefficiencies in meeting SLAs, particularly as the number of tasks increases.

Table 9: SLA violation rate (SVR) (in %) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with number of hosts = 400 and number of VMs = 100

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	2.5	5	4	6
200	3	5.5	4.5	6.5
300	3.5	6	5	7
400	4	6.5	5.5	7.5
500	4.5	7	6	8
600	5	7.5	6.5	8.5
700	5.5	8	7	9
800	6	8.5	7.5	9.5
900	6.5	9	8	10

Table 10 depicts the SLA violation rate (SVR) (in %) obtained by the Proposed GSO-HHO-RA, GSO, HHO, and PSO approaches, where the number of hosts is 800, and the number of VMs is 200.

GSO-HHO-RA (Proposed) achieves the lowest SLA violation rate across all task counts, demonstrating its effectiveness in meeting service level agreements and ensuring timely completion of tasks. This highlights the hybrid approach's capability to optimize resource allocation effectively. GSO shows a higher SLA violation rate than GSO-HHO-RA, indicating challenges in maintaining SLAs as the number of tasks increases. While it performs reasonably well, it does not match the efficiency of the proposed method. HHO performs better than PSO but still has a higher SLA violation rate compared to GSO-HHO-RA. Although HHO is effective in local optimization, it falls short in overall performance compared to the hybrid approach. PSO consistently exhibits the highest SLA violation rate among all approaches, reflecting significant inefficiencies in task scheduling and resource allocation, particularly as the number of tasks increases.

Table 10: SLA violation rate (SVR) (in %) obtained by the proposed GSO-HHO-RA, GSO, HHO, and PSO with number of hosts = 800 and number of VMs = 200

Number of Tasks	*GSO-HHO-RA (Proposed)*	*GSO*	*HHO*	*PSO*
100	1.8	4	3.5	5.5
200	2.2	4.5	3.9	6
300	2.6	5	4.3	6.5
400	3	5.5	4.8	7
500	3.4	6	5.2	7.5
600	3.8	6.5	5.7	8
700	4.2	7	6	8.5
800	4.6	7.5	6.5	9
900	5	8	7	9.5

Conclusion

The proposed gravitational search optimization - Harris Hawks optimization resource allocation (GSO-HHO-RA) approach demonstrates significant advantages over traditional resource allocation techniques (GSO, HHO, and PSO) in cloud computing environments. The results obtained across various performance metrics, including response time (RT), execution time (ET), energy consumption (EC), resource utilization (RU), and SLA violation rate (SVR), highlight the effectiveness and efficiency of the GSO-HHO-RA method.

Improved performance

The GSO-HHO-RA approach consistently achieved the best results in response and execution times, indicating its capability to allocate resources quickly and effectively while minimizing latency in task processing.

Energy efficiency

The hybrid approach exhibited the lowest energy consumption rates across different scenarios, showcasing its ability to optimize resource utilization and reduce operational costs in cloud environments.

Maximized resource utilization

The GSO-HHO-RA method demonstrated superior resource utilization, ensuring that available resources were effectively allocated and used to their full potential. This resulted in higher system performance and efficiency.

Minimized SLA violations

With the lowest SLA violation rates, the proposed approach ensured adherence to service level agreements, providing reliability and quality in cloud service delivery. This is particularly crucial for maintaining customer satisfaction in dynamic and competitive environments.

Overall, the GSO-HHO-RA approach proves to be a robust solution for dynamic resource allocation in cloud computing. By effectively combining the strengths of gravitational search and Harris Hawk's optimization techniques, this hybrid method not only enhances performance metrics but also aligns with the growing demand for energy-efficient and high-quality cloud services. The findings emphasize its suitability for modern cloud applications, making it a promising choice for organizations looking to optimize their resource management strategies while ensuring compliance with service commitments.

References

Ahmadabadi, J. Z., Mood, S. E., & Souri, A. (2023). Star-quake: A new operator in multi-objective gravitational search algorithm for task scheduling in IoT based cloud-fog computing system. *IEEE Transactions on Consumer Electronics*.

Alabool, H. M., Alarabiat, D., Abualigah, L., & Heidari, A. A. (2021). Harris hawks optimization: a comprehensive review of recent variants and applications. *Neural computing and applications*, 33, 8939-8980.

Belgacem, A. (2022). Dynamic resource allocation in cloud computing: analysis and taxonomies. *Computing*, 104(3), 681-710.

Chen, F., Lu, A., Wu, H., Dou, R., & Wang, X. (2022). Optimal strategies on pricing and resource allocation for cloud services with service guarantees. *Computers & Industrial Engineering*, 165, 107957.

Chen, J., Wang, Y., & Liu, T. (2021). A proactive resource allocation method based on adaptive prediction of resource requests in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2021(1), 24.

Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., ... & Zomaya, A. (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98, 751-774.

Han, H., Bai, X., Hou, Y., & Qiao, J. (2022). Multitask particle swarm optimization with dynamic on-demand allocation. *IEEE Transactions on Evolutionary Computation*, 27(4), 1015-1026.

Hashemi, A., Dowlatshahi, M. B., & Nezamabadi-Pour, H. (2021). Gravitational search algorithm: Theory, literature review, and applications. *Handbook of AI-based Metaheuristics*, 119-150.

Jawhar, M. M., & Osman, H. M. (2022). Quality of Service and Load Balancing in Cloud Computing: A Review. *AL-Rafidain Journal of Computer Sciences and Mathematics*, 16(1), 15-22.

Kumar, N., & Kumar, S. (2022). A Salp Swarm Optimization for Dynamic Resource Management to Improve Quality of Service in Cloud Computing and IoT Environment. *International Journal of Sensors Wireless Communications and Control*, 12(1), 88-94.

Mohamed, Y. A., & Mohamed, A. O. (2022, July). An Approach to Enhance Quality of Services Aware Resource Allocation in Cloud Computing. *In International Conference on Information Systems and Intelligent Applications* (pp. 623-637). Cham: Springer International Publishing.

Naeem, M., Anpalagan, A., Jaseemuddin, M., & Lee, D. C. (2013). Resource allocation techniques in cooperative cognitive radio networks. *IEEE Communications surveys & tutorials*, 16(2), 729-744.

Osypanka, P., & Nawrocki, P. (2022). QoS-aware cloud resource prediction for computing services. *IEEE Transactions on Services Computing*, 16(2), 1346-1357.

Petrovska, I., & Kuchuk, H. (2022). Static allocation method in a cloud environment with a service model IAAS. *Advanced Information Systems*, 6(3), 99-106.

Saidi, K., & Bardou, D. (2023). Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities. *Cluster Computing*, 26(5), 3069-3087.

Shi, F., & Lin, J. (2022). Virtual machine resource allocation optimization in cloud computing based on multiobjective genetic algorithm. *Computational Intelligence and Neuroscience*, 2022(1), 7873131.

Si Salem, T., Iosifidis, G., & Neglia, G. (2022). Enabling long-term fairness in dynamic resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(3), 1-36.

Teekaraman, Y., Manoharan, H., Basha, A. R., & Manoharan, A. (2023). Hybrid optimization algorithms for resource allocation in heterogeneous cognitive radio networks. *Neural Processing Letters*, 55(4), 3813-3826.

Umer, A., Nazir, B., & Ahmad, Z. (2022). Adaptive market-oriented combinatorial double auction resource allocation model in cloud computing. *The Journal of Supercomputing*, 78(1), 1244-1286.

Xu, H., Xu, S., Wei, W., & Guo, N. (2023). Fault tolerance and quality of service aware virtual machine scheduling algorithm in cloud data centers. *The Journal of Supercomputing*, 79(3), 2603-2625.

Zivkovic, M., Bezdan, T., Strumberger, I., Bacanin, N., & Venkatachalam, K. (2021). Improved harris hawks optimization algorithm for workflow scheduling challenge in cloud–edge environment. *In Computer networks, big data and IoT: proceedings of ICCBI 2020* (pp. 87-102). Springer Singapore.