International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/221896 Volume 10, Issue 3, May – June (2023)

RESEARCH ARTICLE

Modified Fire Hawks Gazelle Optimization (MFHGO) Algorithm Based Optimized Approach to Improve the QoS Provisioning in Cloud Computing Environment

Manila Gupta
Department of Computer Science and Engineering, IFTM University, Moradabad, Uttar Pradesh, India.
guptamanila6@gmail.com

Devendra Singh
Department of Computer Science and Engineering, IFTM University, Moradabad, Uttar Pradesh, India.
dev625g@gmail.com

Bhumika Gupta
Department of Computer Science and Engineering, Govind Ballabh Pant Institute of Engineering and Technology, Pauri Garhwal, Uttarakhand, India.
bhumikamit6@gmail.com

Received: 16 April 2023 / Revised: 01 June 2023 / Accepted: 06 June 2023 / Published: 30 June 2023

Abstract – This work introduces a method that focuses on enhancing resource allocation in cloud computing environments by considering Quality of Service (QoS) factors. Since resource allocation plays a crucial role in determining the QoS of cloud services, it is important to consider indicators like response time, throughput, waiting time, and makespan. The primary difficulty in cloud computing lies in resource allocation, which can be tackled by proposing a novel algorithm known as Modified Fire Hawks Gazelle Optimization (MFHGO). The proposed approach involves the hybridization of the modified fire hawks algorithm with gazelle optimization to facilitate efficient resource allocation. It aims to optimize several objectives, such as resource utilization, degree of imbalance, completion time, throughput, relative error, and response time. To achieve this, an optimal resource allocation is achieved using the Partitioning around K-medoids (PAKM) clustering approach. The proposed model extends the K-means clustering method. For simulation purposes, the GWA-T-12 Bitbrains dataset is utilized, while the JAVA tool is employed for exploratory analysis. The effectiveness of the proposed resource allocation and clustering approach is demonstrated by comparing it with existing schemes. The proposed work's makespan is 1.45 seconds for 50 tasks, 3.6 seconds for 100 tasks, 3.67 seconds for 150 tasks, and 5.34 seconds for 200 jobs. As a result, the proposed model achieves the smallest makespan value when compared to the previous approaches. The proposed work yielded response times of 105ms for a task length of 100, 376ms for 200, 555ms for 300, 624ms for 400, and 1014ms for 500. These results indicate that the proposed model outperforms current approaches by achieving a faster response time and also attains a bandwidth utilization of 0.80%, 0.90%, and 0.97% for 4, 6, and 16 tasks, respectively, indicating better bandwidth utilization than the other approaches.

Index Terms – Cloud Computing, Resource Allocation, Throughput, Response Time, Bandwidth Utilization, Time Consumption, Resource Utilization.

1. INTRODUCTION

A cloud environment is a broad term that refers to services provided to businesses to improve their functionality, Infrastructure, platforms, and software capacity. It increases the hardware's available storage capacity and processing power to help the cloud service provider build a centralized and powerful computing network accessible via the Internet. As a result, cloud computing has become the new norm in which organizations can considerably benefit and proliferate by utilizing its possibilities. The cloud environment may help achieve client satisfaction and enhance business profits to have a reliable cloud environment and appropriate use resources. Resource allocation is becoming more of a problem as people and corporations maintain more and more of their data in the Cloud. This wide variety of capabilities presents several resource allocations in a cloud environment, such as Distributed Resource Allocation, Dynamic Resource Allocation, Virtual Machine Resource allocation, Energy

Efficiency Resource Allocation, Utility Resource Allocation, and SLA Resource Allocation.

Nowadays, cloud computing has become one of, if not the, most important areas in the IT sector. One of the resources available in distributed computing is capacity servers, along with data set servers and computation servers [1]. These services are provided to clients via the cloud through a pay-as-you-go billing scheme. In general, cloud computing is a PC platform that provides on-demand network access to a larger pool of system administration, handling, and capacity resources outside of the internet [2]. Due to the fact that the hardware is not required for the specific operations of the client, this type of computing allows for cost minimization [3]. Customers face difficulty in selecting appropriate resources due to the unique nature of the resources offered by cloud service providers, such as on-demand services and wide network access.Customers only purchase computing resources from a cloud service provider when they have a need for cloud services [4]. The increasing popularity of web applications and cloud deployment brings about various concerns for cloud service providers, such as lengthy execution times and escalating costs [5] [6]. Resource allocation tailored to the specific needs of customer applications poses a significant challenge in the cloud environment. Consequently, multiple algorithms are employed to address the resource allocation problem and provide optimal solutions. Amazon Web Services offers a range of services, including database, networking, computing, and storage, each with its own pricing structure. This makes it challenging for customers to select resources that fit within their budget while also considering quality of service (QoS) requirements [7]. A fundamental component of distributed computing is the asset cloud, which controls how resources like processing power, memory, storage, and organised data transport are spread among various tasks or customers. In cloud computing, the main goal of the asset part is to ensure efficient and effective use of the resources that are available while meeting the demands and requirements of the customers. Many aspects, such as responsibility qualities, asset accessibility, client demands, and nature of administration (QoS) requirements, have an influence on asset portion in distributed computing. Asset distribution is streamlined via the use of asset assignment formulas and methodologies, which vary depending on the specific goals and requirements of the framework.

New computing concepts of cloud computing to offer end-users trustworthy customized and QoS (Quality of Service) ensured dynamic computing environments. The categories of Static/Dynamic Allocation of resources should be determined depending on the application prerequisites to properly use the resources without violating SLAs and meeting quality of service criteria. Over- and under-provisioning of resources needs to be controlled. Another critical constraint is the use of electricity. Power consumption, dissipation, and VM placement should all be minimized. A methodology for avoiding excessive cost use might need to be developed. As a result, a cloud user's ultimate goal is to allocate resources at the minimum cost. In spite of this, the ultimate objective of a cloud service provider is to maximize profit by effectively allocating available resources. Cloud Service Providers, frequently referred to as CSPs (such as Google, Microsoft, and Amazon), are third parties that offer their customers the facilities of cloud computing resources, applications, and services. These facilities are utilized dynamically based on the customer's demand and are consistent with a specific business model. The companies can be accessible online through the Internet by using a web browser, and the information and software applications are saved on cloud servers placed in data centers according to various pay-as-you-go subscription models.

Some commonly used resource allocation techniques in cloud computing include:

- Static Allocation: In this approach, resources are allocated to users or applications based on a fixed allocation policy. This method works well when the workload is stable and predictable. Static allocation in cloud computing is a method of allocating computing resources, such as virtual machines (VMs) or containers, to specific applications or users in a fixed manner. With static allocation, the allocation decisions are made in advance and remain unchanged throughout the runtime of the applications or services. This approach involves determining the resource requirements of each application or user beforehand and provisioning the necessary resources accordingly. Once the resources are allocated, they are exclusively dedicated to the assigned applications or users, irrespective of their actual usage levels. Static allocation offers stability and predictability since resources are reserved ahead of time, guaranteeing their availability for the assigned applications. However, it may lead to resource underutilization if the assigned applications fail to fully utilize the allocated resources.
- Dynamic Allocation: In this approach, resources are allocated based on the current demand and workload. This approach works well in situations where the workload is unpredictable and changes frequently. Dynamic allocation in cloud computing involves a strategy of allocating and reallocating computing resources, such as virtual machines (VMs) or containers, based on real-time demand. Unlike static allocation, which has predetermined and fixed resource assignments, dynamic allocation offers flexibility and optimizes resource utilization. In this approach, resources are assigned to applications or users according to their current requirements and can be adjusted as demand fluctuates. The dynamic allocation process

considers factors like workload, performance needs, and resource availability to make real-time allocation decisions. Techniques like auto-scaling and load balancing are commonly employed to manage resource allocation dynamically. By dynamically allocating resources, cloud providers ensure that applications or users receive the necessary resources when required, while any unused resources can be reclaimed and reallocated to other workloads. This approach promotes scalability, responsiveness, and cost-effectiveness within cloud computing environments.

- Load Balancing: To prevent any server from becoming overburdened, this strategy involves distributing the workload equally across several servers. Calculations for load adjustment can be used to enhance asset designation across the servers.
- Virtualization: Virtualization allows multiple users or applications to share a single physical resource, such as a server or storage device. Virtualization enables flexible and efficient resource allocation by creating virtual machines that can be allocated resources as needed. Virtualization is a component of cloud computing that enables users to access their services from any location using any interface. Instead of coming from visible entities, the resources it required came from the cloud. You can accomplish everything you want if you have a laptop or mobile phone with internet connectivity. Users can get or distribute it securely easily at any time and location. Users can do work that can only be done on multiple computers simultaneously.
- Hybrid Approaches: Some cloud providers use a combination of static and dynamic allocation techniques to optimize resource allocation based on workload characteristics and user demands.

1.1. Challenges in Resource Allocation

Resource allocation is a critical task in cloud computing that involves distributing resources among users or applications in a way that maximizes efficiency, performance, and availability while minimizing costs. However, there are several challenges that cloud providers must address to ensure that resource allocation is done effectively. Some of the key challenges include:

- Heterogeneity: Cloud computing infrastructure is typically composed of heterogeneous resources such as CPUs, GPUs, memory, storage, and network bandwidth. Allocating these resources optimally is challenging because they have different performance characteristics, costs, and constraints. Resource allocation algorithms must be able to account for these differences and allocate resources appropriately.
- Dynamicity: Cloud computing workloads are highly dynamic, and resource demands can vary rapidly over time. Resource allocation algorithms must be able to adapt to changing demand patterns and allocate resources in real-time to ensure that applications are running smoothly.
- Scalability: Considering that cloud computing frameworks are designed to be incredibly flexible, they should be able to handle huge numbers of customers and tasks. To ensure that assets are allocated effectively and realistically as the responsibility grows, asset allocation calculations should have the flexibility to scale along with the framework.
- Cost optimization: Cloud computing resources are expensive, and optimizing resource allocation to minimize costs is a key challenge. Resource allocation algorithms must balance cost optimization with performance and availability requirements to ensure that customers are getting the best value for their money.
- Security and compliance: Cloud computing systems must comply with a range of security and compliance regulations, which can make resource allocation more challenging. Resource allocation algorithms must take into account these regulations and ensure that resources are allocated in a way that meets security and compliance requirements.
- Multi-tenancy: Cloud computing infrastructure is typically shared among multiple tenants, which can make resource allocation more challenging. Resource allocation algorithms must ensure that resources are allocated fairly among tenants and that no one tenant is hogging resources at the expense of others.

1.2. Problem Statement

In cloud computing, resource allocation is a critical issue that needs to be addressed to ensure that cloud providers can efficiently and effectively allocate resources to meet the varying demands of their customers' applications. The problem arises from the fact that cloud resources are finite, and must be allocated in a way that maximizes efficiency and minimizes costs, while also ensuring that performance is not compromised. This requires the development of sophisticated resource allocation algorithms and techniques that can dynamically allocate resources to meet changing demand, while also accounting for the complex interactions between different types of resources, such as CPU, memory, storage, and network bandwidth. Additionally, cloud providers need to consider factors such as data security, privacy, and regulatory compliance when allocating resources, adding further complexity to the problem. Therefore, in cloud computing, the resource allocation problem is a multifaceted and challenging issue that requires careful consideration and innovative solutions to ensure that cloud computing continues to provide value to businesses and organizations.

In brief, the paper's remaining content can be summarized as follows: Section 2 examines previous studies that utilize different resource allocation methods. Section 3 outlines the suggested methodology. The outcomes and analysis of the suggested methodology are provided in Section 4, with Section 5 serving as the conclusion of the paper.

2. LITERATURE SURVEY

This section provides a survey of high-level classification research papers published in resource allocation approaches has been offered as a result of the discussion that took place before. In addition to providing a summary of the chosen articles and placing them under the appropriate headings, this page also discusses an evolution in the methods used to allocate resources over the years.

In addition, it outlines several exciting and potentially fruitful future possibilities in the subject of RA in cloud computing. On the other hand, additional options need to be studied further to build more cost-effective allocation methods. The main objectives of RA strategies should focus on enhancing security, ensuring performance isolation, facilitating smooth virtual machine migration, promoting interoperability, building resilience against failures, enabling graceful recovery, and optimizing cost savings in data center operations. It is expected that cloud computing services will soon become an integral part of diverse information systems, spanning various types and sizes.

A game-hypothetical approach for equitable asset division in distributed computing administrations has been offered by the creators in [8]. The technique takes into account the preferences and financial goals of the clients as well as the costs and restrictions of the cloud assets. The architects support a Nash bargaining system that increases social government aid while ensuring fairness in the allocation of assets. They evaluate the suggested strategy through reenactment exams and demonstrate that it outperforms existing designation strategies in terms of decency and efficacy. The study concludes that a fair and effective asset designation component for cloud computing administrations may be provided by the game-hypothetical approach.

Processing resources required by vehicles can be obtained through computation offloading services, with the main focus of earlier research being on cloud computing or mobile edge computing. For vehicle organisations that employ both portable edge processing and distributed computing, the developers have suggested a cooperative arrangement [9]. It is currently feasible to offload cooperative calculations for NP-hard and non-curved problems. The results demonstrate how a cooperative arrangement might improve the performance of the proposed structure, but its crucial transmission time is its primary drawback.

In [10] the authors have presented a priority-based dynamic resource allocation approach for cloud computing. The approach is designed to allocate resources to different users based on their priorities, which are determined by their QoS requirements and service level agreements (SLAs). The authors propose a priority-based queuing model and a dynamic resource allocation algorithm that takes into account the changing workload and resource availability in the cloud environment. They evaluate the proposed approach using simulation experiments and show that it can provide better QoS and resource utilization compared to traditional approaches. The paper concludes that the priority-based dynamic resource allocation approach can improve the performance and efficiency of cloud computing services while meeting the diverse QoS requirements of different users.

A novel asset part model has been put up by the authors in [11] to fulfil client asset requirements while enhancing distributed computing's feasibility and reducing transmission delays. To save costs and build a season of virtual machines, the developers used the dispersing multi-objective insect lion calculation (S-MOAL), a multi-objective inquiry calculation. Using the S-MOAL method, the energy consumption and response to internal failure were also examined. The S-MOAL algorithm was not very effective in virtual machine tuning or undertaking determination, the inventors noted.

Asset portion, load adjustment, and asset booking are essential in cloud computing for improving the nature of administration (QoS). A review's enhancement technique in [12] suggested combining the fake honey bee province (ABC) model and the recreated tempering (SA) strategy in order to choose the best asset. In a cloud environment, this method increases booking productivity by taking into account the need for solicitation, the volume of work, and the distance between the hubs and the server. To improve asset allocation, the ABC computation makes use of demand approval, virtual machine approval, and simplified booking techniques. The outcomes of the recreation reveal that this improvement strategy is incredibly practical for planning frameworks and improving distributed computing execution.

In order to tackle interference problems associated with evaluating the effectiveness of resource allocation methods, the authors have introduced a robust algorithm [13] that enables dynamic resource allocation based on user requests within virtual machines. This approach utilizes a feature extraction algorithm to analyze task requirements from the user pool, extracting pertinent features related to both the user's tasks and the cloud server. By employing a modified PCA technique to reduce the feature set, followed by resource allocation through an optimization process based on HPSO-MGA. The approach described offers an effective and efficient resource allocation solution specifically designed for cloud computing environments.

The authors have introduced a hierarchical multi-agent optimization (HMAO) technique [14] with the objective of enhancing resource utilization and minimizing bandwidth expenses. This approach combines multi-agent optimization with genetic algorithms (GA) to identify service nodes with optimal resource utilization for task delivery. The HMAO approach employs decentralized-based MAO to minimize bandwidth costs. The effectiveness of this model is compared to conventional methods.

A work titled energy-effective asset fraction issue in cloud settings has been proposed by the authors in [15], with the aim of reducing energy consumption while still satisfying the Nature of Administration (QoS) requirements of cloud customers. In light of the Bug Monkey Enhancement (SMO) calculation, which is motivated by the behaviour of arachnid monkeys in search of food, the inventors suggest a novel technique. The suggested strategy calls for upgrading virtual machines (VMs) to physical hosts in the cloud environment. To solve the advancement problem, which entails determining the optimal number of VMs and allocating them to actual hosts, the SMO computation is used. The goal of streamlining is to fulfil the QoS requirements of cloud clients while reducing the energy consumption of the cloud foundation. By using recreations, the designers evaluate how the suggested strategy is presented. The replications are focused on a cloud environment with a fluctuating number of real hosts and virtual machines. The results demonstrate that the suggested technique is effective in reducing energy consumption while meeting the QoS requirements of cloud customers.

The development of a hybrid metaheurisctic-based resource allocation framework called RAFL is presented by the authors in [16] for load balancing in cloud computing environments. The framework incorporates three distinct metaheurisctic methods, namely FA, GWO and PSO, to effectively achieve load balancing and resource allocation. The RAFL runs in two stages. At the first stage, the framework group's virtual machines (VMs) according to their CPU and memory use using a modified version of the K-Means clustering method. The hybrid metaheurisctic algorithms are used in the second phase to evenly and effectively distribute the virtual machines (VMs) across the physical machines (PMs).

The authors used CloudSim, a cloud simulation tool, to conduct tests to gauge RAFL's performance. The trials included a range of situations with various workload intensities, VM and PM counts, and performance indicators. The findings demonstrate that, in terms of a variety of performance parameters, including resource usage, reaction time, and throughput, RAFL surpasses other cutting-edge resource allocation algorithms. The authors demonstrate that the combination of the different metaheurisctic algorithms utilised in RAFL leads in greater performance by comparing the performance of each method.

In [17], the authors put out a brand-new load balancing technique for cloud computing that was motivated by krill herd behaviour. The suggested technique uses a multi-objective optimization strategy to balance workloads among cloud-based virtual machines while consuming the least amount of energy and making the best use of available resources. By imitating krill's swarming activity in the search space, the algorithm inspired by krill herd behaviour operates. The algorithm employs a number of strategies, including crossover, mutation, and selection, to generate fresh answers and modify the search space. . Three metrics—makespan, energy use, and load balance—are used to evaluate the proposed algorithm's performance. Makespan indicates how long it takes for all activities to be completed, energy consumption represents how much energy the cloud environment uses, and load balance assesses how evenly the burden is spread among virtual machines. The experimental findings demonstrate that, while generating equivalent makespan outcomes, the suggested algorithm outperforms current state-of-the-art algorithms in terms of load balancing and energy usage. The study comes to the conclusion that the algorithm inspired by krill herd behaviour is a useful technique for load balancing in cloud computing systems, resulting in better resource usage and energy consumption.

In [18], the authors introduce the Hybrid Particle Swarm Optimization (HPSO)-Multiple Genetic Algorithm (MGA) technique as a means of optimizing dynamic resource allocation in cloud systems. The main objective of this algorithm is to minimize energy consumption within the cloud environment while ensuring the QoS requirements of cloud users are met. By combining the strengths of both Particle Swarm Optimization (PSO) and GA, the HPSO-MGA algorithm offers a powerful optimization approach. PSO enables efficient exploration of the search space to identify the best possible solution, while GA enhances the solution's accuracy and convergence speed. The results demonstrate that, despite achieving comparable levels of QoS satisfaction, the HPSO-MGA algorithm surpasses other techniques in terms of energy efficiency.

In [19], a new method is introduced for resource allocation in cloud computing environments. This method integrates a hierarchical resource allocation strategy with a clustering model, a hybrid Capuchin Search algorithm with multiple objectives, and a genetic algorithm (GA). The main aim of this method is to enhance energy efficiency and reduce response time by optimizing the distribution of virtual machines (VMs) on cloud nodes. The suggested approach is used to find the best location for VMs while taking energy use and response time into account. The resource allocation method in each node is optimized using the GA. To balance energy usage and reaction time, the GA is specifically utilised to optimize the CPU, memory, and bandwidth allocation in each node. The authors provide a hierarchical resource allocation plan with a clustering model to further boost the performance of the suggested strategy. The cloud nodes are divided into clusters by the hierarchical structure, and each cluster has its own method for allocating resources. The nodes are grouped using the clustering model according to their resource needs and communication styles. The suggested method is tested in a cloud computing simulation environment, and the findings demonstrate that it performs better than the standard approaches in terms of energy usage and reaction time. Specifically, the proposed approach achieves a 30% reduction in energy consumption and a 25% reduction in response time compared to existing methods.

3. PROPOSED METHODOLOGY

To enhance network services and subsequently QoS in cloud computing, we are focusing on resource allocation and associated aspects. Because fault-tolerant systems are connected to dependable systems, throughput, makespan time, response time, time consumption, utilisation percent for different jobs, and waiting time are metrics that would affect resource allocation. Dependability includes a number of important requirements for the fault tolerance system. In the proposed approach, we outline a mathematical justification for resource allocation in our system model. Our primary goal is to optimize the utilization of resources and minimize the expenses associated with bandwidth in the context of cloud computing. Furthermore, we consider various physical limitations and constraints to ensure a comprehensive approach. It's important to note that in this post, we don't delve into network resource restrictions such as routers and switches. To ensure uninterrupted service delivery when implementing the recommended method in a real cloud environment, we deploy multiple redundant service nodes. These redundant nodes are often in a dormant or powered-off state to reduce operational expenses. However, they can be swiftly activated when a few active service nodes are unable to handle their dynamic workloads. Our proposed approach also proven effective in maintaining service delivery in the event of service node failures, as dynamic workloads from failing nodes can be seamlessly transferred to redundant service nodes.

As shown in Figure 1, the task at hand undergoes an initial division into multiple subtasks. This division allows for a more manageable and organized approach to handling the overall task. Next, the partitioning around K-medoids method is employed to create clusters of these subtasks. This method considers the characteristics and relationships between the subtasks to group them accordingly, potentially improving efficiency and effectiveness. Once the clustering is completed, the scheduling of these subtasks comes into play. The proposed approach utilizes the Modified Fire Hawks Gazelle Optimization algorithm, which is specifically designed to optimize scheduling in scenarios with multiple objectives or criteria. This algorithm determines the most efficient sequence and allocation of resources for executing the subtasks. To ensure the appropriate allocation of resources, a resource manager is employed. The resource manager takes the schedule generated by the Modified Fire Hawks Gazelle Optimization algorithm and allocates the necessary resources, such as computing power, memory, or personnel, to each subtask. This resource allocation process ensures that the required resources are available and properly assigned to facilitate the execution of the tasks according to the optimized schedule.

The proposed approach combines task division, clustering using the partitioning around K-medoids method, scheduling using the Modified Fire Hawks Gazelle Optimization algorithm, and resource allocation managed by a resource manager. Together, these steps aim to enhance the efficiency and effectiveness of task execution in a complex and resource-intensive environment.

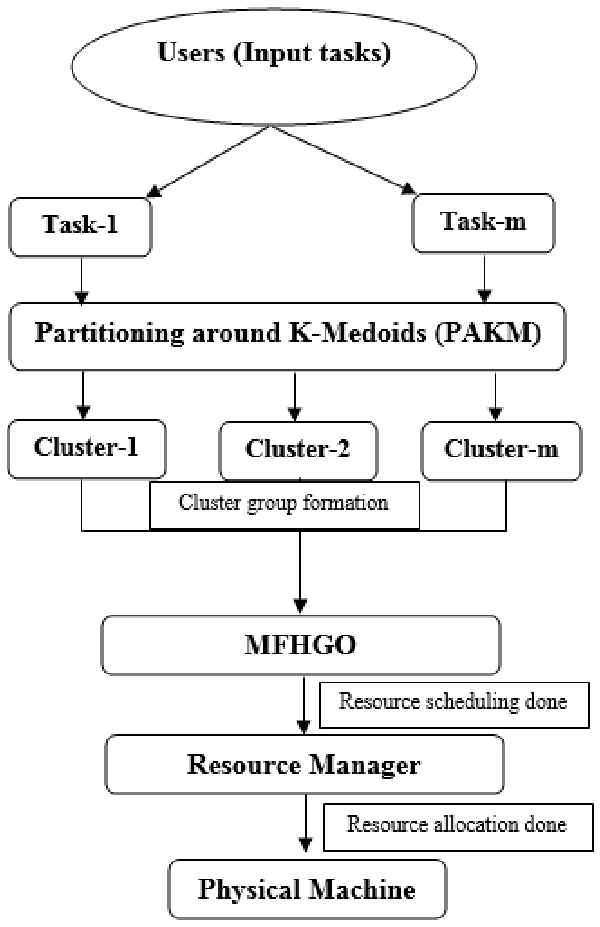

Figure 1 Proposed Methodology

3.1. K-Medoids Algorithm

The k-medoid algorithm can be applied to resource allocation problems to cluster resources based on their similarity and allocate them to appropriate tasks or entities. Here's a general outline of how the k-medoid algorithm can be used for resource allocation:

- Define similarity measure: Determine a suitable similarity measure to quantify the similarity or dissimilarity between resources. This measure should capture relevant characteristics or attributes of the resources that are important for allocation.
- Prepare the data: Collect or extract the necessary data on the resources to be allocated. This data could include attributes such as resource type, capacity, availability, cost, or any other relevant factors.
- Determine the number of clusters: Decide on the number of clusters (k) that you want to create for resource allocation. This number should reflect the desired level of granularity in the allocation process.
- Initialize medoids: Randomly select k data points from the resource dataset as the initial medoids. These medoids will represent the cluster centers.
- Assign resources to clusters: Assign each resource to the cluster represented by the closest medoid based on the similarity measure. This step involves computing the dissimilarity between each resource and each medoid and assigning the resource to the cluster with the minimum dissimilarity.
- Update medoids: For each cluster, calculate the total dissimilarity between all resources in the cluster and each candidate medoid. Select the data point with the minimum total dissimilarity as the new medoid for that cluster.
- Repeat steps 5 and 6: Iterate the assignment and medoid update steps until convergence. Convergence occurs when the medoids no longer change or when a predefined number of iterations is reached.
- Allocate resources: Once the algorithm converges, each resource will be assigned to a specific cluster represented by its respective medoid. You can then allocate the resources within each cluster to appropriate tasks or entities based on their similarity and specific allocation criteria.
- Monitor and adjust: Monitor the resource allocation and assess its effectiveness. If necessary, you can re-run the k-medoid algorithm with different parameters or update the similarity measure to further optimize the allocation process.

By utilizing the k-medoid algorithm for resource allocation, you can effectively cluster resources based on their similarities and allocate them in a manner that maximizes utilization and meets specific allocation objectives.

Here is how the K-medoids algorithm operates:

1. Choose K randomly chosen points from the dataset to form the first medoids, where K is the number of clusters (K).
2. Connect every data point with the closest medoid (using a distance metric such as Euclidean distance).
3. If doing so minimises the overall distance between the medoid and the other points in the cluster, choose the data point closest to the medoid for each cluster and substitute it for the medoid.
4. Continue doing steps 2 and 3 until the medoids stop changing or a certain number of iterations has been achieved.

3.2. PAKM Clustering Approach

Partitioning around K-Medoids (PAKM) is a clustering algorithm used to group data points into K clusters. It is an extension of the K-Means algorithm that uses medoids instead of means to represent the cluster center. Medoids are data points that are closest to the center of the cluster. PAKM works by selecting K medoids randomly, assigning each data point to the nearest medoid, and then replacing each medoid with the data point that minimizes the sum of the distances to all other points in the cluster. This process is repeated until the medoids no longer change. PAKM has been shown to be effective in optimizing resource allocation in cloud computing. Algorithm 1 shows the working of PAKM method.

```
Input:
  • Dataset D
  • Number of clusters K
Output:
  • K clusters C_1, C_2... C_K with medoids m_1, m_2... m_K
  1. Initialize K medoids randomly from the dataset
  2. Assign each data point to the nearest medoid to form K clusters
  3. While medoids are changing do the following:
     3.1. For each cluster C_i and non-medoid point p in C_i,
          swap m_i and p and compute the cost of the resulting cluster configuration
```

3.2 Select the medoid p with the minimum cost and swap it with its corresponding medoid

3.3 Assign each data point to the nearest medoid to form K clusters

4. Return the final K clusters C_1, C_2... C_K with medoids m_1, m_2... m_K

Note: The cost function can be defined as the sum of the distances between each point in a cluster and its medoid.

Algorithm 1 Partitioning Around K-Medoids (PAKM)

3.3. Modified Fire Hawks Gazelle Optimization (MFHGO) Algorithm

The Modified Fire Hawks Gazelle Optimization (MFHGO) [20] [21] algorithm is a nature-inspired optimization technique that takes inspiration from the hunting behavior of two animals: the fire hawk and the gazelle. In 2017, Seyedali Mirjalili and Seyed Mohammad Mirjalili introduced this algorithm that combines the exceptional vision and hunting skills of the fire hawk with the speed and agility of the gazelle. In the MFHGO algorithm, the solutions are represented as fire hawks, while the objective function values are represented as gazelles. The aim of the algorithm is to improve the positions of the fire hawks iteratively to catch the gazelles with the highest objective function values. The algorithm employs three primary operators: the search operator, the catch operator, and the escape operator. The search operator explores the search space by moving the fire hawks randomly. The catch operator catches the gazelles with the highest objective function values. The escape operator enables the fire hawks to escape local optima by moving them away from their current position. While the MFHGO algorithm has demonstrated promising results when tested on various benchmark functions, its performance may vary depending on the problem and parameter settings.

In the proposed work, the resource allocation problem is handled by the Modified Fire Hawks Gazelle Optimization (MFHGO) algorithm. This algorithm is a hybridized version of the recently introduced metaheurisctic such as modified fire hawks algorithm and gazelle optimization algorithm. The proposed work opted these algorithms as these are highly efficient in terms of convergence compared to other algorithms. Moreover, the flexibilities provided by these algorithms made them most suitable for resource allocation as the considered scenario consists of several constraints. The proposed algorithm is multi- objective and it deals with the following major objectives: resource utilization, degree of imbalance, completion time, throughput, relative error and response time. The proposed algorithm is also iteration-based and for each iteration, the algorithm evaluates the entire population in terms of the fitness function. Based on fitness evaluations, the most optimal solution for resource allocation is identified. In the proposed algorithm, the modified fire hawks algorithm is considered where the opposition-based learning (OBL) strategy is considered to improve the algorithm. This strategy is mainly followed to enhance the overall efficiency in identifying the candidate solutions. Also, this strategy identifies the opposite solutions of the candidates and compares the fitness values of both the current and opposite solutions. This helps the algorithm to accurately identify the global optimal solution within a minimum number of iterations. The sequential steps followed in MFHGO algorithm are explained below and pseudo code is shown in Algorithm 2:

Step 1: The algorithmic parameters are defined and the input of the algorithm is provided such as the virtual machines, input tasks and objective function.

Step 2: Using the gazelle optimization algorithm, the initialization procedure is carried out. The problem dimension and problem space is defined and the matrix representation is followed to initialize the input values.

Step 3: For each input candidate solution, the objective values are defined based on the fitness function. After evaluating the fitness of each solution, the best solution identified is considered to construct the elite matrix of the algorithm. This matrix is updated at the end of each iteration to keep track of the most optimal solutions.

Step 3.1: The Brownian motion is defined for the algorithm using the step length to regulate the exploration process of the algorithm.

Step 3.2: Using the levy flight strategy, the random walk of the search agents is defined using the levy flight distribution. Also, the levy stable process is defined to generate stable motion in the algorithm.

Step 4: The exploratory behavior of the algorithm takes place when a predator is sighted in the problem space. Both the search agent and the predator runs at the maximum speed and the levy flight strategy is adopted to characterize the movement of the search agent.

Step 5: The fitness functions of the solutions after the exploratory behavior are re-evaluated to identify the optimal solutions. Moreover, the exploration behavior of gazelle optimization algorithm prevents the solutions to get trapped in local optima.

Step 6: Again the elite matrix constructed at the initial stage is updated with the solutions obtained from the exploration behavior. This helps to identify the most optimal solutions that are fit to be passed to the next iteration.

Step 7: After evaluating the fitness of the population and sorting them accordingly, the exploitation behavior of the modified fire hawks algorithm is followed to identify the global optimal solution.

Step 8: In case of the fire hawks algorithm, both the fire hawks and the prey are considered to be candidate solutions and the distance value is determined between the prey and fire hawks initially.

Step 9: The fire hawk territory in this algorithm is assumed to be a circular area and the overall distance is dependent on both the prey and the hawk.

Step 10: After identifying the territory of fire hawk, the exploitation process is carried out for both the original and opposite solutions. For each solution obtained in the exploration process, the opposite solution is generated using the OBL strategy.

Step 11: Then, the fitness functions of the original and opposite solutions are calculated and the most optimal solution is identified among the two solution sets. The update formulation is finally followed for the optimal solution and this solution is considered as the output of the resource allocation process.

Step 12: Return the obtained optimal solution as the output.

Input:

- Number of agents N
- Max_iter (Maximum number of iterations)
- Objective function F (Objective Function)
- Lb and Ub (Lower and upper bounds of variables)
- Number of preys Prey_num
- Number of fires Fire_num
- Number of hawks Hawk_num
- Fire range R

Output:

- The best solution x* and the corresponding function value F(x*)

1. Initialize the agents' positions and velocities randomly within the search space.
2. Evaluate the objective function F for each agent.
3. Set the best agent x* as the agent with the lowest function value.
4. For t = 1 to Max_iter do the following:

4.1. Sort the agents based on their function values.

4.2. Update the position and velocity of each agent using the following equations: -

$$\text{Velocity update: } v\_i(t+1) = wv\_i(t) + c1r1*(pbest\_i - x\_i) + c2r2(gbest - x\_i)$$

$$-\text{Position update: } x\_i(t+1) = x\_i(t) + v\_i(t+1)$$

4.3. Clip the agents' positions to the bounds Lb and Ub.

4.4. Evaluate the objective function F for each agent.

4.5. Update the best agent x* if there is an agent with a lower function value.

4.6. For each prey in the population, do the following:

4.6.1. Select a random fire and a random hawk.

4.6.2. If the distance between the prey and the fire is less than R, update the prey's position using the fire's position.

4.6.3. If the distance between the prey and the hawk is less than R, update the prey's position using the hawk's position.

4.7. For each fire in the population, do the following:

4.7.1. Calculate the average distance between the fire and the preys. 4.7.2. Update the fire's position using the following equation:

$$f\_i\ (t+1) = f\_i\ (t) + alpha*(avg\_dist\_i - dist\_i)*rand\ (1, D)$$

4.8. For each hawk in the population, do the following:

4.8.1. Select a random prey and a random fire.

4.8.2. If the distance between the hawk and the prey is less than R, update the hawk's position using the prey's position.

4.8.3. If the distance between the hawk and the fire is less than R, update the hawk's position using the fire's position.

4.9. Update the population by replacing the worst agents with new ones randomly generated within the search space.

5. Return the best agent x* and the corresponding function value F(x*).

Note: r1, r2, and alpha are constants representing the cognitive, social, and fire step sizes, respectively. D is the number of dimensions in the search space.

Algorithm 2 Sequential Steps Followed in MFHGO

4. RESULTS AND DISCUSSION

This section outlines the simulation methodology employed to assess the proposed model, which was implemented using a Java tool. The performance evaluation encompassed various metrics, including time consumption, throughput waiting time, response time, makespan, and resource utilization. The experimental setup featured five types of virtual machines, classified into low-performance and high-performance groups based on their MIPS values. Specifically, VMs 1, 2, and 3 were classified as low performers, while VMs 4 and 5 were categorized as high performers. The simulation process utilized the GWA-T-12 Bitbrains dataset, which comprises performance statistics for 1750 virtual machines within a Bitbrains data center. Bitbrains serves as a service provider that manages computing operations for businesses. The dataset contained performance statistics files for each virtual machine, organized according to traces of Rnd and fast storage. In the Rnd trace, 1,250 VMs were connected to a fast Storage Area Network (SAN) device, while in the Rnd trace, 500 VMs were linked to either Network Attached Storage (NAS) or fast SAN storage systems. These performance statistics files contained relevant information regarding VM resource utilization and performance, enabling the evaluation of resource allocation and scheduling algorithms' effectiveness in cloud computing. The dataset plays a crucial role as a valuable asset for researchers and practitioners who aim to create and evaluate innovative algorithms for resource allocation and scheduling within cloud computing environments.

4.1. Makespan

The makespan represents the total time required for completing all tasks across all resources. It measures the difference between the starting and ending points of a task. A minimal makespan value indicates that the allocator employs effective planning strategies for resource scheduling, while a large makespan value suggests inefficient planning techniques for resource allocation.

Figure 2 depicts the evaluation of makespan using the proposed methodology, comparing it to existing strategies such as FCFS, PSO, KPSHOW, KMPS, and MHCSGA. The makespan is assessed using four different task lengths of 50, 100, 150, and 200. The proposed model surpasses other existing techniques by consistently reducing the makespan across all problem levels.

The makespan of the suggested model over other existing approaches is seen in the above figure 2. The suggested model demonstrates a makespan of 1.45 seconds for 50 tasks, 3.6 seconds for 100 tasks, 3.67 seconds for 150 tasks, and 5.34 seconds for 200 jobs. Consequently, the proposed model outperforms previous approaches by attaining the lowest makespan value.

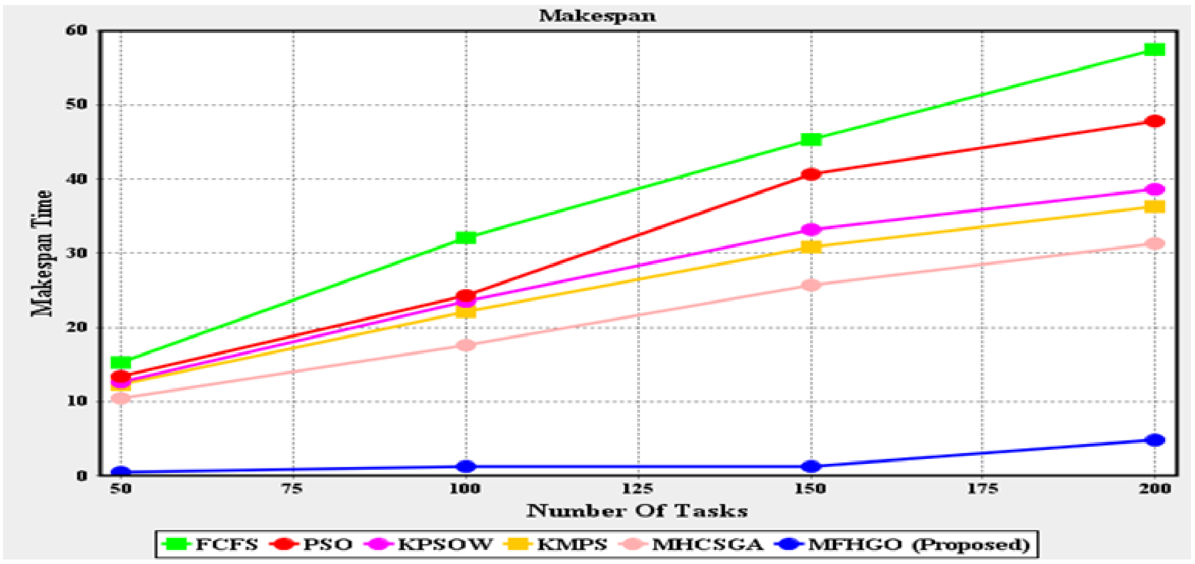

Figure 2 Makespan vs Number of Tasks

4.2. Virtual Machines Utilization

Utilization of virtual machines refers to the extent to which the resources of a virtual machine (VM) are being used effectively. This includes factors such as CPU usage, memory usage, network usage, and storage usage. The goal of VM utilization is to ensure that resources are being allocated optimally and efficiently, so that workloads can be completed in a timely manner and with minimal waste. The evaluation of virtual machine utilization was conducted across a range of process numbers, spanning from 50 to 200, as illustrated in Figure 3-6. The proposed model's virtual machine utilization was compared to that of existing methods, including FCFS, PSO, KPSHOW, KMPS and MHCSGA. According to the assumption, virtual machines with lower performance should handle less complex tasks, while those with higher performance should handle more complex ones. Hence, in the case of virtual machines with lower performance, the utilization rate of VM1 should be comparatively lower than that of VM2 and VM3. Similarly, when dealing with virtual machines exhibiting higher performance, the utilization rate of VM4 should be relatively lower than that of VM5.

Additionally, the entire makespan time should be reduced. The proposed MFHGO approach outperformed the existing methods by accurately controlling virtual machine jobs with a shorter makespan.

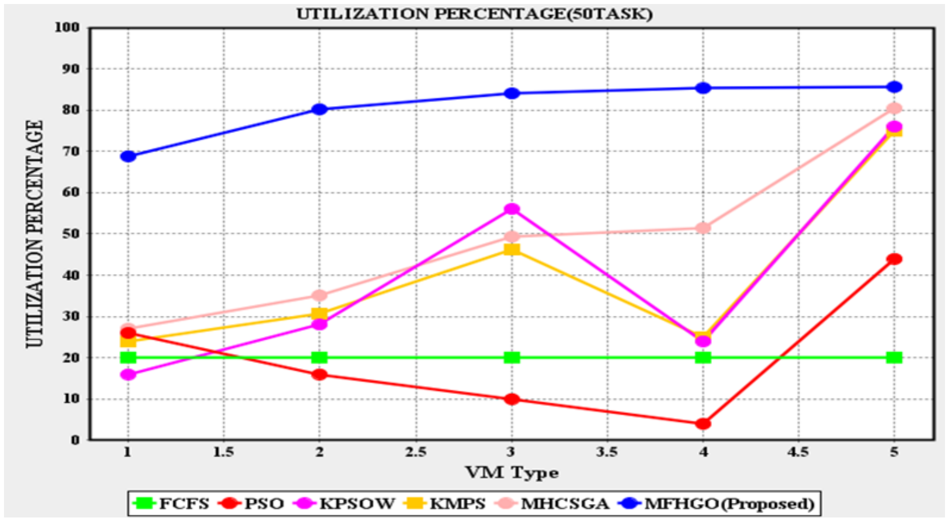

Figure 3 Utilization vs VM Types at 50 Tasks

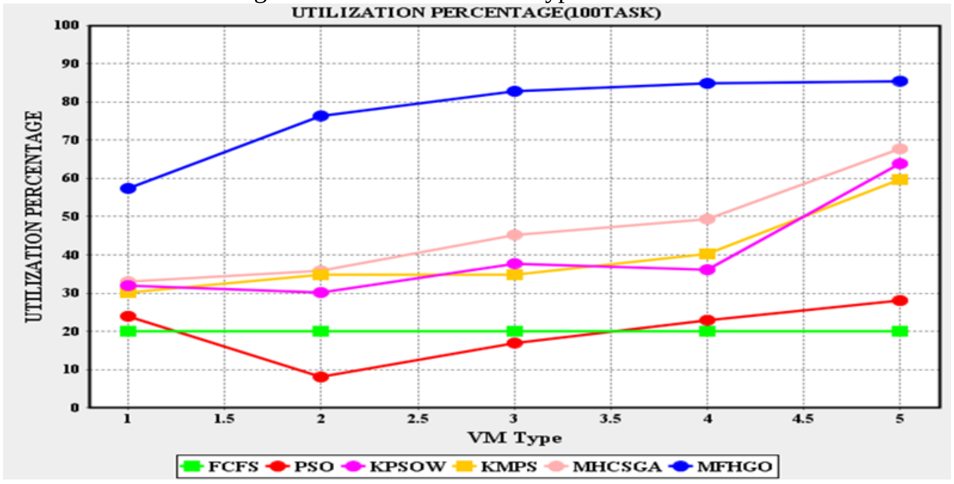

Figure 4 Utilization vs VM Types at 100 Tasks

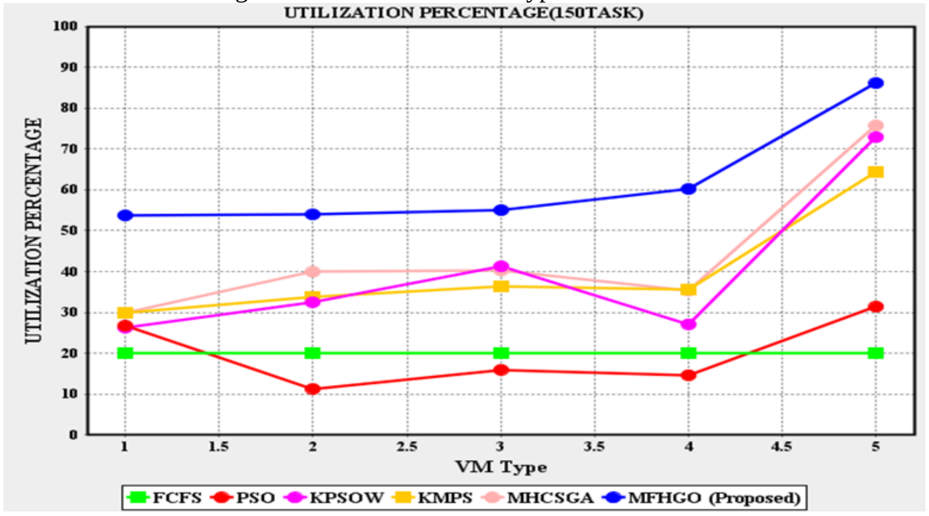

Figure 5 Utilization vs VM Types at 150 Tasks

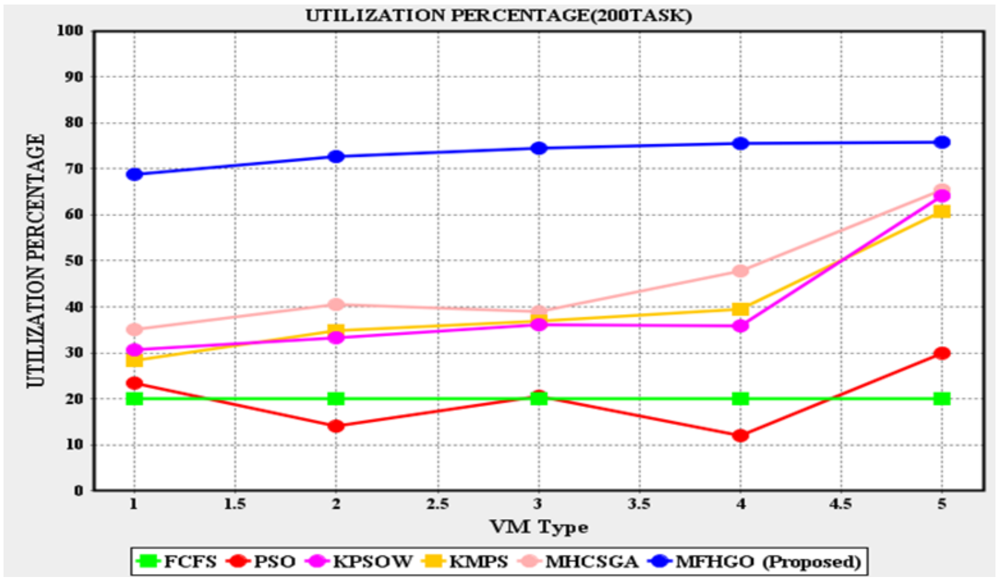

Figure 6 Utilization vs VM Types at 200 Tasks

4.3. Waiting Time

The average waiting time, which refers to the duration jobs spend in the queue of each assigned virtual machine (VM), is a crucial metric. Mathematically, this typical waiting time can be represented by Equation 1.

$$Aw_t = \frac{\sum Tw_t}{m} \qquad (1)$$

where Aw_t is the average waiting time, Tw_t is the waiting time for tasks to be executed, m is the total number of tasks, and is the average waiting time. Depending on how many requests there are, the waiting period changes. There are 100 tasks total, with 500 being the last challenge. The proposed approach in the study demonstrates superior performance in terms of waiting time when compared to existing techniques. The experimental analysis conducted in the study provides evidence that the suggested model effectively reduces waiting time for job allocation on the cloud server, outperforming the methods currently in use. The reduction in waiting time is a crucial factor in improving the efficiency and responsiveness of cloud services. By minimizing the time jobs spend in the queue before being allocated to resources, the proposed model ensures faster processing and improved user experience. The experimental evaluation of the suggested model validates its effectiveness in reducing waiting time and highlights its superiority over the existing techniques. The results clearly demonstrate that the proposed approach offers a more efficient solution for job allocation in cloud computing environments.

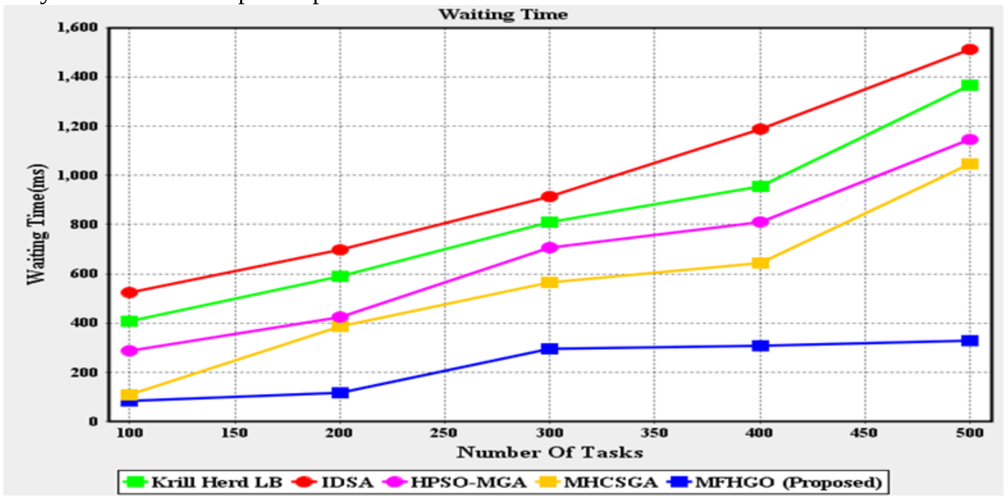

Figure 7 Waiting Time vs Number of Tasks

4.4. Response Time

The response time, which measures the duration required to handle a service request, is calculated by adding the waiting time and service time. Figure 8 in the paper presents a comparative analysis of response times between the proposed MFHGO approach and other existing methods, including krill herd load balancing (LB), HPSO-MGA, IDSA, and MHCSGA. The results clearly indicate the superiority of the proposed approach, as it achieves faster response times for jobs of varying lengths in comparison to the existing methods. The simulation study provides compelling evidence supporting the improved performance of the proposed approach over the current methods. Hence, it can be concluded that the proposed methodology significantly enhances resource allocation and scheduling in cloud computing.

Response time was measured for a range of task lengths from 100 to 500. The response times obtained by the proposed model were 105ms for a task length of 100, 376ms for 200, 555ms for 300, 624ms for 400, and 1014ms for 500. These findings clearly demonstrate that the proposed model surpasses existing approaches by delivering quicker response times.

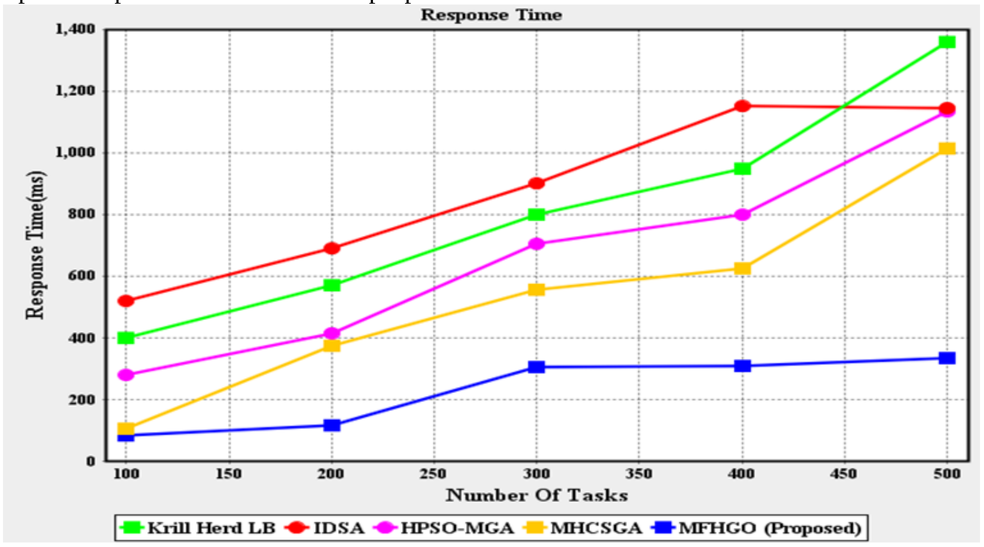

Figure 8 Response Time vs Number of Tasks

4.5. Throughput

Throughput is a metric that quantifies the total number of tasks or processes completed within a given time period. It measures the efficiency of a system in terms of task completion rate. In the context of cloud computing, throughput is an important factor as it directly impacts the overall system performance and user experience. The primary objective of throughput optimization is to minimize the time required to finish a process. This means that a higher throughput value indicates that the system can handle a larger number of tasks in a given period, resulting in faster task completion.

On the other hand, a lower throughput value indicates that the system is slower in processing tasks, potentially leading to delays and longer completion times. The level of throughput performance achieved by a particular method is influenced by the number of requests or tasks being processed. Generally, as the number of requests increases, the throughput performance becomes more critical. The system needs to efficiently handle a larger volume of tasks to ensure timely completion.

In Figure 9, the proposed approach for resource allocation is compared to existing methods such as IDSA, HPSO-MGA, Krill herd LB, and MHCSGA in terms of throughput performance. The purpose of this comparison is to evaluate how effectively each method can process and complete tasks within a specific time frame. Figure 9 presents a comparative analysis of the proposed approach's throughput performance with the existing methods. By evaluating and comparing the throughput values, it is possible to determine which approach is more effective and efficient in terms of task completion speed. The aim is to identify the approach that can achieve higher throughput, indicating better system performance in terms of task processing and completion. The main goal of throughput is to reduce the time needed to complete a process. The effectiveness of throughput can vary depending on the number of tasks involved, which can range from 100 to 500. The unit of measurement for throughput time is milliseconds. The presented illustration demonstrates that the proposed model surpasses existing methods by requiring less time. Hence, it can be inferred that the proposed model exhibits superior throughput performance compared to other approaches.

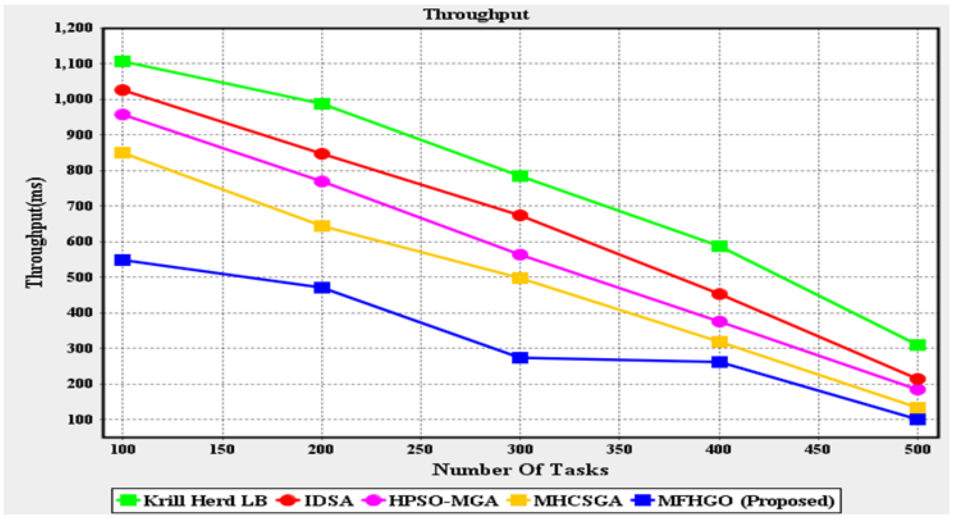

Figure 9 Throughput vs Number of Tasks

4.6. Time Consumption

Time consumption in cloud computing refers to the amount of time it takes to complete a task or workload in a cloud computing environment. This includes factors such as network latency, processing time, and data transfer times. The time consumption can vary depending on the size and complexity of the workload, as well as the availability and allocation of resources in the cloud. This can involve optimizing resource allocation, balancing workloads across multiple servers, and minimizing network latency and data transfer times. Effective time consumption management can help organizations to improve productivity, reduce costs, and enhance the overall performance and reliability of their cloud computing infrastructure. Figure 10 presents a comparison of the proposed MFHGO approach with other methods, including krill herd LB, IDSA, HPSO-MGA, and MHCSGA, in terms of time consumption. The simulation results show that the proposed approach achieves a lower rate of time consumption compared to the other methods. This implies that the proposed approach in resource allocation and scheduling within cloud computing is more efficient and requires less time compared to other methods.

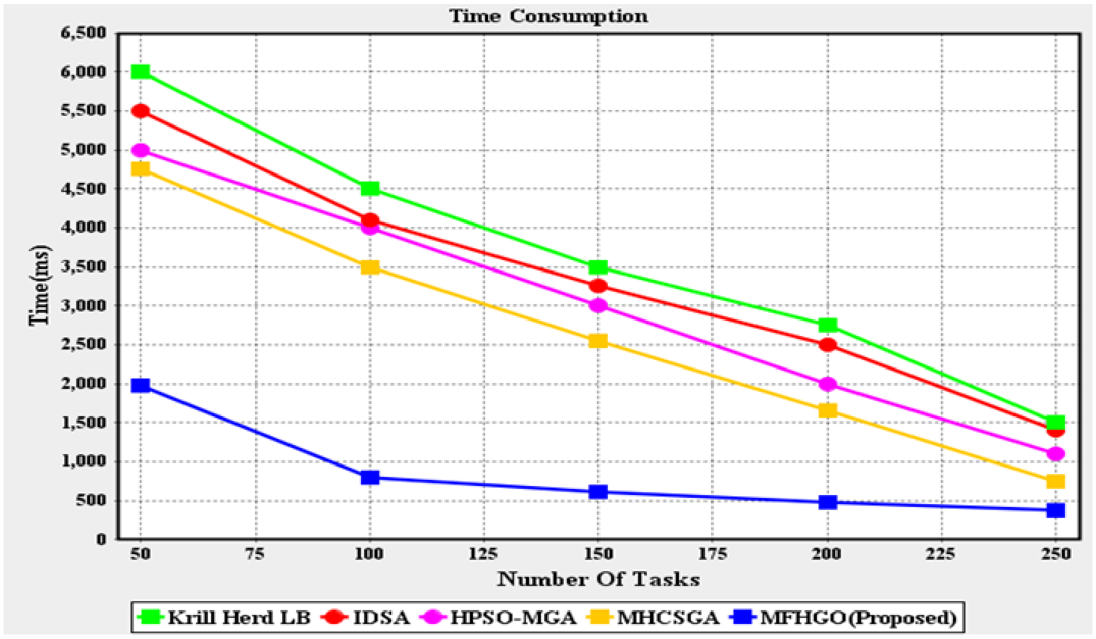

Figure 10 Time Consumption vs Number of Tasks

4.7. Resource Utilization

In the paper, Figure 11 presents a comparison of resource utilization between the proposed model and alternative techniques (HMAO, GA, NSGA-II, and MHCSGA) across varying task sizes ranging from 4 to 16. The proposed model exhibits better resource utilization for each task than the existing methods, which suffer from low convergence rate, reduced stability, and poor quality, leading to increased optimization problem rate and make it ineffective for allocating resources optimally. To overcome these issues, this work proposes the MFHGO approach to achieve effective resource utilization. The proposed model attains a bandwidth utilization of 0.80%, 0.90%, and 0.97% for 4, 6, and 16 tasks, respectively, indicating better bandwidth utilization than the other approaches.

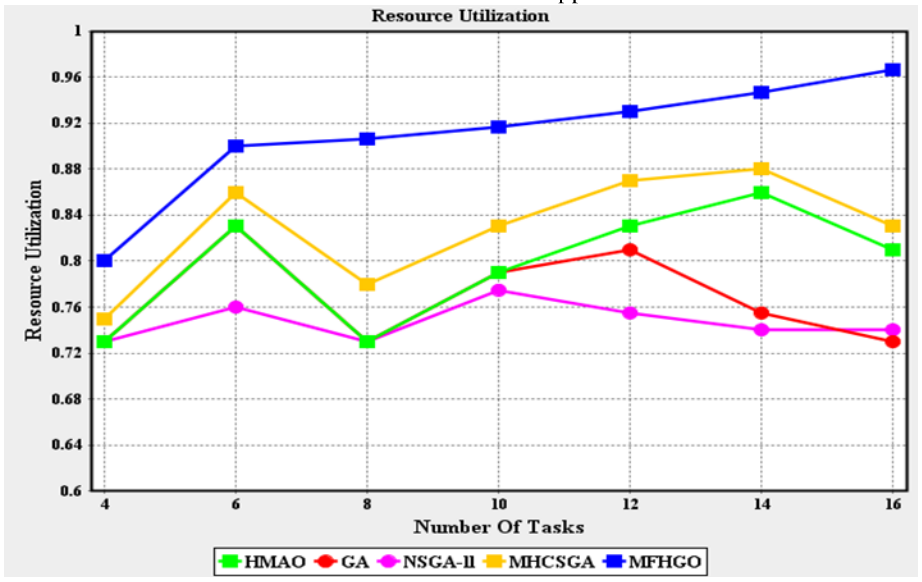

Figure 11 Resource Utilization vs Number of Tasks

4.8. Bandwidth Utilization

Bandwidth utilization pertains to the amount of data that can be transmitted within a communication channel during a given period. Meanwhile, the number of tasks indicates the number of independent jobs or processes that require completion. Figure 12, which is being discussed, appears to demonstrate the comparison of bandwidth utilization between the proposed model and various existing techniques (HMAO, GA, NSGA-II, and MHCSGA). The proposed model outperforms the other methods in terms of bandwidth utilization for tasks ranging from 4 to 16. This improvement is attributed to the proposed model's utilization of the PAKM clustering approach to group tasks, which enhances stability and results in optimal bandwidth utilization.

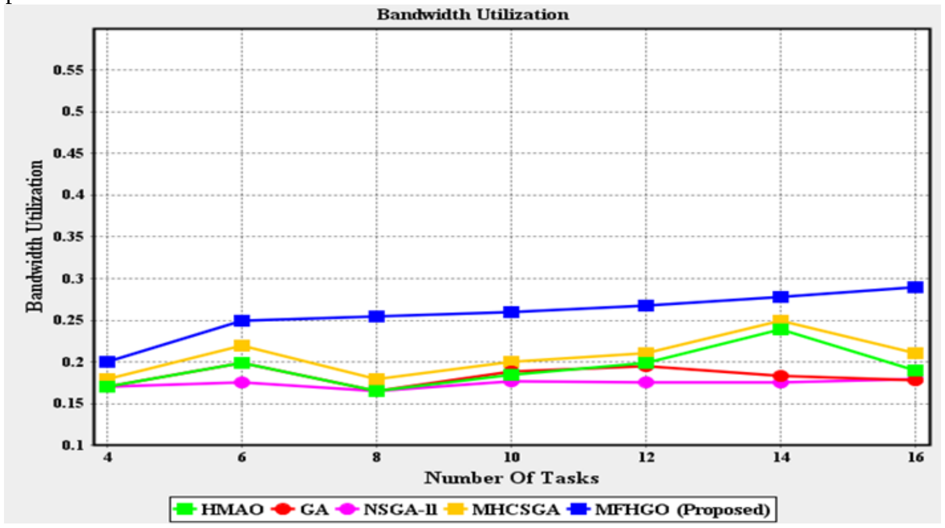

Figure 12 Bandwidth Utilization vs Number of Tasks

5. CONCLUSION

The computing environment of the cloud has discovered an increasing number of uses in commercial and business environments. Within the framework of the cloud, it is necessary to employ a method that is both effective and efficient in allocating resources to satisfy users' needs and increase the revenue of cloud service providers. Energy Efficiency Resource allocation, and SLA resource allocation, some strategies for allocating resources are discussed; however, these strategies lack certain aspects. This work offered new taxonomies based on parameters, algorithms, and optimization strategies based on a comprehensive study of relevant approaches in the literature based on their benefits and limitations. This analysis was based on the results of our search for relevant techniques. The assessment was based on comparing the various approaches' merits and deficiencies, both in terms of their strengths and weaknesses. The improved strategy based on Modified Fire Hawks Gazelle Optimization (MFHGO) algorithm offers a practical means of enhancing Quality of Service (QoS) provisioning in cloud computing environment. To maximize resource allocation and enhance provisioning of QoS, the strategy makes use of the advantages of both modified fire hawks algorithm and gazelle optimization. The suggested method optimizes resource allocation to fulfil multiple QoS requirements while guaranteeing resource efficiency. These requirements include reaction time, availability, and throughput. The experimental findings show that, in terms of QoS performance measures, the suggested technique performs better than alternative approaches that are already in use. Cloud service providers may use the suggested method to optimize resource allocation and enhance QoS provisioning, resulting in higher customer satisfaction and better overall cloud computing environment performance. In this work, the improved strategy based on Modified Fire Hawks Gazelle Optimization (MFHGO) algorithm offers a practical means of enhancing Quality of Service (QoS) provisioning in cloud computing environment. To maximise resource allocation and enhance provisioning of QoS, the strategy makes use of the advantages of both modified fire hawks algorithm and gazelle optimization. The suggested method optimises resource allocation to fulfil multiple QoS requirements while guaranteeing resource efficiency. These requirements include reaction time, availability, and throughput. The experimental findings show that, in terms of QoS performance measures, the suggested technique performs better than alternative approaches that are already in use. Cloud service providers may use the suggested method to optimise resource allocation and enhance QoS provisioning, resulting in higher customer satisfaction and better overall cloud computing environment performance. The MFHGO algorithm offers a practical solution for enhancing QoS provisioning in the cloud computing environment. By leveraging the strengths of the modified fire hawks algorithm and gazelle optimization, the strategy optimizes resource allocation, fulfills multiple QoS requirements, and outperforms existing approaches. Its implementation can result in improved customer satisfaction and overall performance in the cloud computing environment.

5.1. Future Enhancement

This research and analysis consider different RA approaches in cloud computing environments. It has a significant number of advantages. Resource allocation has many benefits when changing cloud computing, independent of the size of the company's business markets. However, there are several restrictions, like reaction time, security, computation, reliability, and less. In addition, the review area includes the user rating system. Different topologies are represented in the network virtual machines used to connect. When consolidating workloads on servers, the primary focus is distributing identical sorts of work across all servers. The servers do not consider the various applications being run on virtual machines. If the distribution of resources is not carried out efficiently, then a significant number of process migrations will occur. Consequently, additional effort is required to install CPUs on the same or closely placed servers. The application interface might also provide end users with varying degrees of performance and allocate resources efficiently concerning energy consumption. As a result, service-aware resource allocation policies' quality has a significant place in cloud computing. The business model of cloud computing needs to consider the user's priorities regarding the availability and distribution of resources. Furthermore, additional research is required to investigate various aspects independent of the quantity of power, bandwidth, or prominent utilization used. Moreover, the experts advise conducting other research on different topics. Furthermore, based on the utility resource allocation, a collection of workloads running on VMs or PMs must estimate the computational and network resource consumption needed for performance maximization. It is required to meet the utility resource allocation standards. It is because it is critical to share these resources. SLA compliance involves the computation of the cost of resource consumption, selecting resources suitable for meeting SLA requirements, and evaluating the resources necessary. There is a need for research on allocating resources in an energy-conscious manner and conducting computation in an energy-efficient manner for the future generation of cloud computing infrastructure. As the cloud expands as a global infrastructure platform, new resource allocation mechanisms will be required to meet service level agreements (SLAs). Mobile cloud computing has swiftly become the industry standard for cloud-based application deployment environments in cloud computing. Cloud computing is a business strategy in which each cloud provider aims to increase revenues while minimizing costs (such as those connected with energy

RESEARCH ARTICLE

consumption, temperature management, and data storage, among other things). Cloud clients are constantly seeking ways to improve the performance of their services while saving money and effort.

REFERENCES

[1] Pradhan, Pandaba, Prafulla Ku. Behera, and B.N.B. Ray. "Modified Round Robin Algorithm for Resource Allocation in Cloud Computing." Procedia Computer Science 85 (2016): 878–90. https://doi.org/10.1016/j.procs.2016.05.278.

[2] Kinger, Kushagra, Ajeet Singh, and Sanjaya Kumar Panda. "Priority-Aware Resource Allocation Algorithm for Cloud Computing." Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing, 2022. https://doi.org/10.1145/3549206.3549236.

[3] Ashawa, Moses, Oyakhire Douglas, Jude Osamor, and Riley Jackie. "Improving Cloud Efficiency through Optimized Resource Allocation Technique for Load Balancing Using LSTM Machine Learning Algorithm." Journal of Cloud Computing 11, no. 1 (2022). https://doi.org/10.1186/s13677-022-00362-x.

[4] Akintoye, Samson Busuyi, and Antoine Bagula. "Improving quality-of-service in cloud/fog computing through efficient resource allocation." Sensors 19, no. 6 (2019): 1267.

[5] Vaibhav Sharma, Gola, K.K. (2016). ASCCS: Architecture for Secure Communication Using Cloud Services. In: Pant, M., Deep, K., Bansal, J., Nagar, A., Das, K. (eds) Proceedings of Fifth International Conference on Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing, vol 437. Springer, Singapore. https://doi.org/10.1007/978-981-10-0451-3_3

[6] Devarasetty, Prasad, and Satyananda Reddy. "Genetic algorithm for quality of service based resource allocation in cloud computing." Evolutionary Intelligence 14, no. 2 (2021): 381-387.

[7] Shrimali, B., & Patel, H. (2020). Multi-objective optimization oriented policy for performance and energy efficient resource allocation in Cloud environment. Journal of King Saud University-Computer and Information Sciences, 32(7), 860-869.

[8] Wei, G., Vasilakos, A.V., Zheng, Y. et al. A game-theoretic method of fair resource allocation for cloud computing services. J Supercomput 54, 252–269 (2010). https://doi.org/10.1007/s11227-009-0318-1

[9] Zhao, Junhui, Qiuping Li, Yi Gong, and Ke Zhang. "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks." IEEE Transactions on Vehicular Technology 68, no. 8 (2019): 7944-7956.

[10] C. S. Pawar and R. B. Wagh, "Priority Based Dynamic Resource Allocation in Cloud Computing," *2012 International Symposium on Cloud and Services Computing*, Mangalore, India, 2012, pp. 1-6, doi: 10.1109/ISCOS.2012.14.

[11] Belgacem, Ali, Kadda Beghdad-Bey, Hassina Nacer, and Sofiane Bouznad. "Efficient dynamic resource allocation method for cloud computing environment." Cluster Computing 23, no. 4 (2020): 2871-2889.

[12] Muthulakshmi, B., and Krishnan Somasundaram. "A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment." Cluster Computing 22, no. 5 (2019): 10769-10777.

[13] Ramasamy, Vadivel, and SudalaiMuthu Thalavai Pillai. "An effective HPSO-MGA optimization algorithm for dynamic resource allocation in cloud environment." Cluster Computing 23, no. 3 (2020): 1711-1724.

[14] Gao, Xiangqiang, Rongke Liu, and Aryan Kaushik. "Hierarchical multi-agent optimization for resource allocation in cloud computing." IEEE Transactions on Parallel and Distributed Systems 32, no. 3 (2020): 692-707.

[15] Samriya, J. K. ., & Kumar, N. (2022). Spider Monkey Optimization based Energy-Efficient Resource Allocation in Cloud Environment. Trends in Sciences, 19(1), 1710. https://doi.org/10.48048/tis.2022.1710

[16] 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, p. 102485, 2022.

[17] Raed Abdulkareem HASAN*, Muamer N. MOHAMMED, A Krill Herd Behaviour Inspired Load Balancing of Tasks in Cloud Computing, Studies in Informatics and Control, ISSN 1220-1766, vol. 26(4), pp. 413-424, 2017.

[18] Ramasamy, V., Thalavai Pillai, S. An effective HPSO-MGA optimization algorithm for dynamic resource allocation in cloud environment. Cluster Comput 23, 1711–1724 (2020). https://doi.org/10.1007/s10586-020-03118-x.

[19] K. K. Gola, B. M. Singh, B. Gupta, N. Chaurasia, and S. Arya, "multi-objective hybrid capuchin search with genetic algorithm based hierarchical resource allocation scheme with Clustering Model in cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 35, no. 7, 2023.

[20] Heidari, Ali Asghar, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen. "Harris Hawks Optimization: Algorithm and Applications." Future Generation Computer Systems 97 (2019): 849–72. https://doi.org/10.1016/j.future.2019.02.028.

[21] Agushaka, Jeffrey O., Absalom E. Ezugwu, and Laith Abualigah. "Gazelle Optimization Algorithm: A Novel Nature-Inspired Metaheuristic Optimizer." Neural Computing and Applications 35, no. 5 (2022): 4099–4131. https://doi.org/10.1007/s00521-022-07854-6.

Authors

Manila Gupta, received a B. Tech degree in Computer Science & Engineering from (U. P. Technical University), UP, India in 2009. She Completed the M. Tech degree in Information Technology from Mahamaya Technical University, Noida, India. She is currently pursuing Ph.D. from I.F.T.M University, Moradabad. She had a working experience of 4.5 years in an Engineering College. Her research interest includes signal processing, networking, machine learning and cloud computing.

Dr. Devendra Singh received a B. Tech degree in Computer Science & Engineering from (U.P.T.U), UP, India in 2005. He completed the M. Tech degree (Wireless networking) in Computer Science from PEC University of Technology (Deemed University), Chandigarh in June 2009, and a Ph.D. degree in Ad hoc Networks from the IFTM University (State University), Moradabad in 2016. He is currently working as Associate Professor and Head of Department at IFTM University, Moradabad with a teaching experience of 17.1 years in the area of academic/research/administration at IFTM University, Moradabad (since Aug2005). His area of interest broadly includes intrusion Detection Systems, Wireless Networks, Ad-Hoc Networks Security, and Machine Learning.

Dr. Bhumika Gupta received a B. Tech degree in Computer Science & Engineering from (U.P.T.U), UP, India in 2005. She completed the M. Tech degree Information Technology from Guru Govind Singh Indraprastha University, Delhi 2010, and a Ph.D. degree from the IFTM University (State University), Moradabad in 2015. Her thesis title is "Iterated Function Systems (IFS) in Chaos and Fractals. She is currently working as an Associate Professor in Department of Computer Science & Engineering, G. B. Pant Institute of Engineering &Technology, Pauri-Garhwal with a teaching experience of 16 years in the area of academic/research/administration. Her area of interest broadly includes Machine learning, Artificial Intelligence, Fuzzy Logic.

RESEARCH ARTICLE

How to cite this article:

Manila Gupta, Devendra Singh, Bhumika Gupta, "Modified Fire Hawks Gazelle Optimization (MFHGO) Algorithm Based Optimized Approach to Improve the QoS Provisioning in Cloud Computing Environment", International Journal of Computer Networks and Applications (IJCNA), 10(3), PP: 383-400, 2023, DOI: 10.22247/ijcna/2023/221896.