Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.046516

ARTICLE

Maximizing Resource Efficiency in Cloud Data Centers through Knowledge-Based Flower Pollination Algorithm (KB-FPA)

Nidhika Chauhan[1], Navneet Kaur[2], Kamaljit Singh Saini[3], Sahil Verma[3], Kavita[3], Ruba Abu Khurma[4,5] and Pedro A. Castillo[6,*]

[1]University Institute of Computing Department, Chandigarh University, Punjab, 140413, India

[2]Department of Computer Science and Engineering, Chandigarh University, Punjab, 140413, India

[3]Universidade Federal do Piauí, Teresina, Piauí, 64049-550, Brazil

[4]MEU Research Unit, Faculty of Information Technology, Middle East University, Amman, 11831, Jordan

[5]Applied Science Research Center, Applied Science Private University, Amman, 11931, Jordan

[6]Department of Computer Engineering, Automatics and Robotics, University of Granada, Granada, 18071, Spain

*Corresponding Author: Pedro A. Castillo. Email: pacv@ugr.es

Received: 05 October 2023 Accepted: 26 March 2024 Published: 20 June 2024

ABSTRACT

Cloud computing is a dynamic and rapidly evolving field, where the demand for resources fluctuates continuously. This paper delves into the imperative need for adaptability in the allocation of resources to applications and services within cloud computing environments. The motivation stems from the pressing issue of accommodating fluctuating levels of user demand efficiently. By adhering to the proposed resource allocation method, we aim to achieve a substantial reduction in energy consumption. This reduction hinges on the precise and efficient allocation of resources to the tasks that require those most, aligning with the broader goal of sustainable and eco-friendly cloud computing systems. To enhance the resource allocation process, we introduce a novel knowledge-based optimization algorithm. In this study, we rigorously evaluate its efficacy by comparing it to existing algorithms, including the Flower Pollination Algorithm (FPA), Spark Lion Whale Optimization (SLWO), and Firefly Algorithm. Our findings reveal that our proposed algorithm, Knowledge Based Flower Pollination Algorithm (KB-FPA), consistently outperforms these conventional methods in both resource allocation efficiency and energy consumption reduction. This paper underscores the profound significance of resource allocation in the realm of cloud computing. By addressing the critical issue of adaptability and energy efficiency, it lays the groundwork for a more sustainable future in cloud computing systems. Our contribution to the field lies in the introduction of a new resource allocation strategy, offering the potential for significantly improved efficiency and sustainability within cloud computing infrastructures.

KEYWORDS

Cloud computing; resource allocation; energy consumption; optimization algorithm; flower pollination algorithm

1 Introduction

Cloud computing is a computing paradigm that has evolved from various other paradigms, such as virtualization, distributed computing, grid computing, and utility computing. It has gained prominence results leading to a computing paradigm for providing on-demand internet services. Cloud computing and data centers have evolved into an important component of our everyday lives as a consequence of multiple internet-dependent services, that we have grown accustomed to using frequently [1]. The data center is the foundation of Information Technology (IT) operations, and it also supports computational features for online services. It pools a cluster of resources such as Central Processing Unit (CPU), Random-Access Memory (RAM), and memory, via networks, cloud services, and communication links. The information technology infrastructure and the electrical and cooling infrastructure are the two major subsystems of the data center [2,3]. The data center's functionality and attributes are critical for optimizing cloud services. Power consumption in data centers has increased rapidly with the demand for increased cloud services. Power management, as well as CO_2 emissions and overloading, has become a significant challenge for cloud data centers [4]. Given that energy consumption is a significant source of global warming, investigating cloud data center energy concerns and reducing its consumption are considered to be promising strategies for energy-efficient management in the cloud environment [5].

Data centers, integral to modern IT operations and online services, play a pivotal role in the digital age by acting as centralized hubs for data storage and processing. However, their operation can contribute to carbon dioxide (CO_2) emissions due to several key factors [6]. The primary driver is the substantial energy consumption required to power and cool the servers, networking equipment, and infrastructure within these facilities. Often, this energy is generated from fossil fuels, such as coal and natural gas, which release CO_2 when burned. In addition to the direct energy use, cooling systems in data centers, including air conditioning and chillers, also consume significant electricity, further amplifying emissions. Inefficiencies in data center operations, including underutilized servers and poor hardware optimization, exacerbate the issue by leading to higher energy consumption. The rapid expansion of data centers to meet the growing demand for cloud services has intensified their carbon footprint. Outdated and inefficient equipment within data centers, along with their reluctance to adopt renewable energy sources, further compound CO_2 emissions [7]. While data centers are essential for the digital ecosystem, mitigating their environmental impact requires improving energy efficiency, optimizing hardware utilization, and transiting to cleaner energy sources [8].

Additionally, saving energy not only helps the environment but also saves money. Therefore, it becomes essential to reduce the energy usage of data centers and cloud computing systems [6]. Multiple problems with energy consumption directly affect the functionality and usefulness of the cloud network. The usage of resources, which directly impacts energy consumption, is, therefore, a major concern. When resources are not managed properly, the efficiency of the cloud environment suffers [7]. The amount of heat produced by data center servers has increased along with the demand for cloud computing services [8]. The electricity required to run and cool these data center servers comes at the expense of the highest possible energy consumption. It has been discovered that the cost of energy consumption accounts for the majority of the total cost [9]. To solve the problem, various methods such as VM virtualization, VM migration, resource allocation, and scheduling are used. A resource allocation system is a crucial component of the cloud environment. When assigning resources to services, cloud service providers should do it in a way that uses less energy while complying with the Quality of Service (QoS) requirements of the end users [10]. Resource allocation is a technique for finding, choosing, deploying, and managing resources to complete the hosted application while adhering to the quality-of-service requirements and accomplishing service providers' goals like better resource usage, decreased energy expenditures, and decreased carbon footprints. There are two types of resource allocation: Static and Dynamic Allocation. Static/Dynamic allocation of resources must be determined depending on the application demands to efficiently utilize the resources and satisfy the QoS standards [11,12].

In addition, over and under-provisioning of resources must be avoided. The main benefits of efficient resource allocation are:

•Scalability: The ability of cloud computing to flexibly scale up and down in response to changes in resource demand is one of its key features [13].

•Speed: Without an IT expert, users may swiftly spin up many devices according to their needs [14].

•Savings: Users may substantially reduce costs by adopting the pay-as-you-go approach, which is achieved by allocating or removing resources in response to user demand [15]. The resource allocation methods in cloud computing can also be divided into reactive and proactive resource allocation [16], as shown in Fig. 1. Reactive resource allocation methods are a kind of dynamic allocation method where resources are allocated based on tasks. It includes meta-heuristic or optimization approaches along with other traditional allocation mechanisms. On the other hand, proactive methods are table-driven methods that use machine learning approaches and create a knowledge base for their sources [17]. Proactive resource allocation develops a database just once, but if any new resources are added or current resources are deleted or damaged, the database should be regenerated for optimal allocation [18].

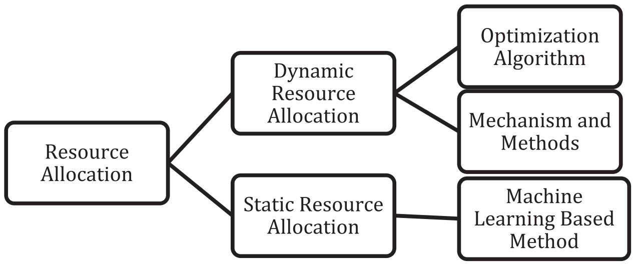
Resource Allocation
Dynamic Resource Allocation
Static Resource Allocation
Optimization Algorithm
Mechanism and Methods
Machine Learning Based Method

Figure 1: Resource allocation methods in cloud computing

1.1 Research Problems

The data center, a pivotal component in IT operations, is crucial for supporting the computational requirements of online services. It functions as a resource hub, pooling CPU, RAM, and memory through networks, cloud services, and communication links. Within the data center, two major subsystems, the information technology infrastructure, and the electrical and cooling infrastructure, play a defining role in its functionality [2,3]. Yet, as data centers have evolved into critical components of cloud service delivery, the surge in power consumption has emerged as a pressing challenge. This increase in power consumption encompasses power management, CO_2 emissions, and the potential for overloading in cloud data centers, adding complexity to their operation [4].

1.2 The Gaps

In light of the critical role that energy consumption plays in global warming, the exploration of energy concerns within cloud data centers, along with strategies to curtail consumption, has garnered significant attention [5]. The importance of this issue extends beyond environmental concerns; energy efficiency also translates into cost savings. Consequently, the imperative to reduce energy usage in data centers and cloud computing systems has grown [6].

Reactive Resource Allocation: Reactive adaptation is based on a remedial action that is hard-coded and predetermined and is activated when a certain event occurs, such as CPU utilization or CPU energy consumption surpassing a specific threshold. The reactive allocation's efficacy is determined by its ability to recognize variations. Resources allotted to the applications must be modified to reflect changes in workload if the considerable deviation from the QoS performance targets is to be avoided. Reactive techniques do not need a thorough grasp of an application's real-time behavior [19]. Proactive Resource Allocation: Proactive adaptation is a popular technique. It determines the control action that, under certain constraints, maximizes a cost function to forecast the behavior of the system over a specific time. When adequate data on resource performance and workload is available, and historic data follows some distribution, then the proactive approach can be employed [20]. Hybrid Resource Allocation: The hybrid adaption technique uses proactive and reactive resource allocation methods to meet (i) Service Level Agreement (SLA), (ii) save energy, and (iii) lower provisioning costs. Managing proactive and reactive strategies can maximize energy efficiency while retaining performance [21]. Recent years have witnessed tremendous growth in the cloud computing market as more and more businesses and organizations use the technology to meet their growing computational needs [22]. However, with this expansion comes the difficulty of allocating and managing resources effectively to ensure optimal performance and cost savings [23]. This study proposes a hybrid strategy that combines static and dynamic resource allocation techniques to address this issue.

1.3 Proposed Approaches

Energy consumption challenges are intrinsically tied to resource management issues. Proper resource allocation stands as a vital strategy in addressing these challenges. Inefficient resource utilization can lead to energy wastage, directly impacting the overall performance of cloud environments [7].

The proposed hybrid approach offers a promising solution to the challenges faced by cloud computing providers, and future research could investigate methods to improve resource allocation techniques further to meet the changing demands of cloud computing. Our research is fueled by several significant challenges and shortcomings we have identified in current cloud computing studies. Cloud computing, which has become integral to our digital lives, presents a range of issues as it has matured. Firstly, we are concerned about the environmental impact of cloud computing. Data centers that power cloud services are consuming more energy than ever before, leading to environmental worries due to increased carbon emissions. Current solutions to this problem often falls short of achieving substantial reductions in data center carbon footprints. Another pressing issue is how resources are allocated within cloud environments. Finding the right balance between allocating resources optimally, controlling costs, and ensuring high-quality service is complex. Existing methods struggle to adapt to the dynamic workloads typical in cloud computing. The lack of standardized resource allocation practices in the cloud industry hinders interoperability and complicates assessing services based on energy efficiency. This absence of common guidelines has made it challenging to develop a unified and efficient approach to resource allocation.

Our work introduces a novel approach to resource allocation in cloud computing, leveraging the Knowledge-Based Flower Pollination Algorithm (KB-FPA). By integrating machine learning intelligence and optimization techniques, our method aims to enhance the efficiency of cloud resource allocation. We address energy consumption and Quality of Service (QoS) concerns through a combination of decision tree-based knowledge acquisition and flower pollination optimization [11]. In the subsequent sections, we delve into the details of our approach, experimental results, and comparative analysis. Through our research, we aim to contribute to the advancement of resource allocation strategies in cloud computing. In the domain of resource optimization algorithms, we see untapped innovation potential. New approaches, especially those leveraging machine learning, offer promising ways to improve efficiency and adaptability in the ever-changing cloud landscape. Lastly, we face the challenge of bridging the gap between theoretical research and practical implementation. Many proposed resource allocation solutions remain largely theoretical or lack robust testing in real-world cloud environments. These complex challenges, combined with the continuous growth of cloud computing, motivate our research to reshape resource allocation practices. We aim to contribute to the development of sustainable, efficient, and adaptable solutions that address these critical issues and limitations.

The increase in heat production from data center servers, driven by the burgeoning demand for cloud computing services, compounds energy consumption as servers require electricity for operation and cooling [8]. It has been revealed that the majority of the total operational cost is attributed to energy consumption [9].

Following is the structure of this paper: In Section 2, the literature review highlights the static, dynamic, and hybrid approaches to resource allocation in cloud computing. Static approaches allocate resources based on predetermined principles and assumptions, whereas dynamic approaches modify resource allocation based on real-time demand. This study proposes a hybrid approach that combines the benefits of both static and dynamic approaches, allowing for greater flexibility and adaptability to changes in load and resources. In Section 3, the proposed hybrid approach employs an FPA algorithm and Decision Tree to optimize resource allocation based on workload. This strategy allows for the efficient use of resources, resulting in improved performance and cost savings. It also demonstrates the ability to adapt to shifting resource and load requirements, making it a promising solution for cloud computing service providers. In Section 4, the effectiveness of the proposed method is evaluated and compared to existing methods. The hybrid approach proposed outperforms traditional approaches in terms of energy efficiency and cost savings. This demonstrates that the proposed technique for optimizing cloud computing resource allocation and utilization is effective. Section 5 of the paper concludes with a summary of the findings and suggestions for future research.

2 Literature Review

Over the past decade, researchers have increasingly focused on optimizing cloud performance by addressing the crucial aspect of energy-efficient cloud resource allocation. This emphasis has led to the development of various resource allocation models, each offering unique approaches to enhancing the efficiency of resource allocation in cloud environments.

Belgacem et al. [1] introduced the Intelligent Multi-Agent System and Reinforcement Learning Method (IMARM). IMARM represents a significant innovation in cloud resource allocation as it integrates multi-agent capabilities with the Q-learning method. This integration allows IMARM to dynamically allocate and release resources based on the characteristics of multi-agent systems, adapting effectively to evolving customer requirements. IMARM leverages reinforcement learning to guide virtual machines toward optimal states, aligning with the surrounding environment. Notably, experimental results have demonstrated IMARM's superiority over complex fault tolerance and energy-based algorithms, particularly when accounting for load balancing and execution time considerations. The study conducted by Moparthia et al. [2] focused on examining strategies for enhancing network response times and reducing energy consumption in Internet of Things (IoT) systems. The authors' analysis offers valuable perspectives on these areas, uncovering potential strategies for enhancing the efficiency and efficacy of IoT networks. The load balancer algorithm proposed by Moparthia et al. is widely recognized as a significant advancement in the industry. The algorithm has been found to have positive effects in terms of optimizing reaction time, improving energy efficiency, and reducing infrastructure expenses.

In pursuit of efficient resource allocation, Abbas et al. [3] explored the use of Artificial Neural Networks (ANNs). ANNs were found to excel in estimating costs for both clients and service providers, crucial for achieving optimal resource allocation. The accuracy of the ANN model in cost estimation surpasses that of state-of-the-art techniques. However, it is essential to acknowledge the computational demands associated with training ANN models, especially when considering their implementation in large-scale cloud environments. Samriya et al. [4] introduced the Spider Monkey Optimization (SMO) approach, aiming to optimize various parameters influencing cloud resource allocation, including application time, migration time, and resource utilization, with a specific focus on energy consumption. Their work employed the Green Cloud Scheduling Model (GCSM) to assess energy efficiency and other performance metrics. The SMO approach adopts a holistic perspective, considering multiple performance aspects. Results from simulations have demonstrated the superiority of this methodology, showcasing improvements in response time, makespan, energy usage, and resource utility. However, practitioners should be mindful of the computational resources required for the effective implementation of the comprehensive SMO approach. In the context of cloud computing, Al-Wesabi et al. [5] proposed the Hybrid Metaheuristics technique for Energy-Efficient Resource Allocation (HMEERA). This innovative model leverages feature extraction based on client work demands and incorporates Principal Component Analysis (PCA) for feature reduction. To optimize resource allocation, HMEERA combines the Group Teaching Optimization Algorithm (GTOA) and Rat Swarm Optimizer (RSO). This amalgamation of algorithms enhances resource allocation efficiency among virtual machines (VMs) in cloud data centers, leading to superior performance and throughput compared to alternative models. While HMEERA has demonstrated its potential, real-world implementation challenges and scalability considerations should be addressed. Josilo et al. [23] tackled the intricate issue of resource management in fog computing, particularly as the number of offloading options escalates with an increasing number of devices in fog environments. They introduced a model founded on game theory and inequality theory to allocate computational tasks on devices in proximity. The proposed algorithm allows devices to make informed choices between offloading computation to nearby devices or the edge cloud. While the experimental results highlighted good system performance, it is important to note that this model primarily considers average system parameters, leaving room for further exploration of additional performance metrics such as energy efficiency and makespan.

The study conducted by Rostami et al. [24] focuses on enhancing cloud data center (DC) performance in response to rising computing and storage demands in cloud computing (CC). Effective task scheduling is crucial for optimal resource use, minimizing energy consumption (EC), reducing response time, and maximizing overall efficiency. The research introduces a novel migration approach for virtual machines (VMs) using the Capuchin Search Algorithm (CapSA). This hybrid framework, combining CapSA and Inverted Ant Colony Optimization (IACO) algorithms, dynamically selects the best algorithm for tasks based on prevailing conditions. The proposed approach outperforms prior methods, achieving 15%–20% improvements in key metrics like EC, execution time (ET), and load balancing. Signifying a significant advancement, this research demonstrates the potential of the CapSA-IACO hybrid for optimizing cloud DC performance in response to escalating service demands.

Tarahomi et al. [25] examined the growing interest in distributed models for addressing challenges in Cloud computing environments, specifically resource allocation. This exploration focuses on two main aspects: Task scheduling, which involves the allocation of tasks to Virtual Machines (VMs) by Cloud providers, and VM-to Physical Machine mapping.

These aspects are closely linked to the important matter of energy consumption in Cloud computing. The work examines current challenges and identifies emerging opportunities for future research. It serves as a valuable resource for researchers to develop new contributions or improve existing ones. The primary objective is to drive progress in resource allocation in Cloud computing environments, thereby contributing to the continuous evolution and innovation in this domain. Cloud computing's rapid growth has facilitated the seamless transfer and hosting of applications for organizations [26]. Efficient task scheduling is necessary to effectively manage diverse user requests across multiple cloud resources. Inadequate scheduling can lead to resource underutilization or overutilization, thereby impacting the efficiency of cloud resources and service performance negatively. Swarm intelligence metaheuristics are gaining popularity in addressing the complex task scheduling challenges in cloud environments. This review offers a thorough analysis of swarm intelligence optimization techniques employed in task scheduling for cloud computing. Fig. 2 shows various resource allocation models and outcome examines various algorithms and performs a comparative analysis using important performance metrics. This study evaluates simulation tools, identifies challenges, and proposes directions for future research. This review examines the potential of advanced swarm-based algorithms in enhancing resource allocation, system performance, and overall utilization of cloud resources.

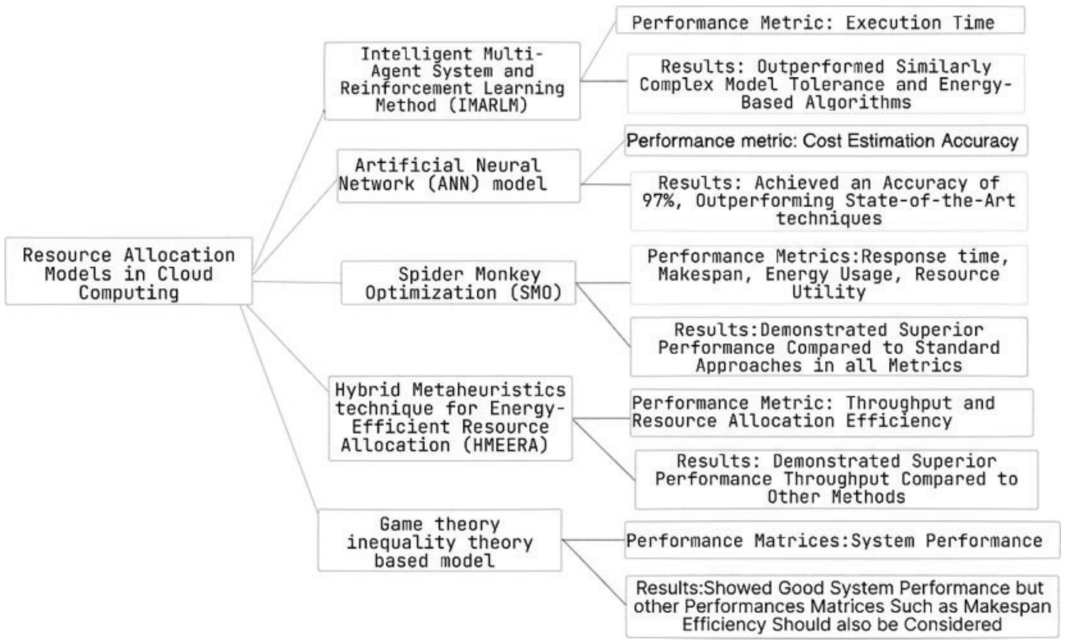
Resource Allocation Models in Cloud Computing
Intelligent Multi-Agent System and Reinforcement Learning Method (IMARLM)
Performance Metric: Execution Time
Results: Outperformed Similarly Complex Model Tolerance and Energy-Based Algorithms
Artificial Neural Network (ANN) model
Performance metric: Cost Estimation Accuracy
Results: Achieved an Accuracy of 97%, Outperforming State-of-the-Art techniques
Spider Monkey Optimization (SMO)
Performance Metrics:Response time, Makespan, Energy Usage, Resource Utility
Results:Demonstrated Superior Performance Compared to Standard Approaches in all Metrics
Hybrid Metaheuristics technique for Energy-Efficient Resource Allocation (HMEERA)
Performance Metric: Throughput and Resource Allocation Efficiency
Results: Demonstrated Superior Performance Throughput Compared to Other Methods
Game theory inequality theory based model
Performance Matrices:System Performance
Results:Showed Good System Performance but other Performances Matrices Such as Makespan Efficiency Should also be Considered

Figure 2: Resource allocation models, performance metrics and outcome

2.1 Decision Tree Algorithm

The decision tree method is a type of supervised learning approach, as described in [27]. This method can be used for both classification and regression problems. The decision tree operates by using a tree-like structure, as shown in Fig. 3. Each Leaf Node (LN) in the tree corresponds to a specific class label, while the interior nodes represent the characteristics used to answer the problem. The root of the decision tree is initially set as the entire training set. It is important to note that feature values in this method must be categorical [24,25].

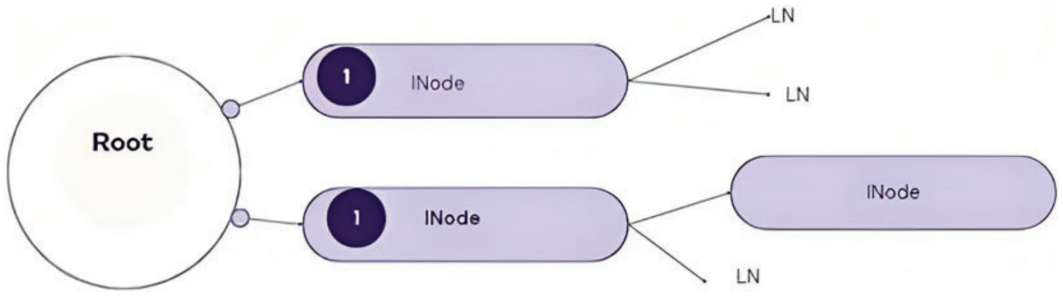

Figure 3: Decision tree

If the data is continuous, it needs to be segmented before model development. Records are distributed recursively based on the values of their attributes. The system employs 4 statistical techniques to arrange characteristics such as the Internal Node (Inode) or the root. The initial iteration of the experiment employs four variables: R, K, Ri, and values(K). The undefined variable R is assumed to be a collection of data instances. Ri is a subset of R in which K has a particular value; values(K) represent the attribute's possible values. Entropy and Igain are introduced in the second iteration. Igain (information gain) measures the quantity of information a given attribute contributes to determining the class of a data instance, whereas entropy measures the impurity of a data set. The same four variables from the initial iteration, but R is now explicitly defined as an instance collection. In the third iteration of model development, additional variables are included in the experiment. R, K, Ri, values(K), entropy, and Igain continue to exist, whereas HI, H(Ri), p(i), and Info (R, K) are introduce I(R) is the entropy of the entire dataset R, whereas H(Ri) is the entropy of the subset Ri. p(i) is the ratio of the number of Ri instances to the total number of R instances. Info (R, K) quantifies an attribute's classification utility in R. These iterations define a model for decision trees that can be applied to classification and prediction tasks in machine learning.

By introducing new variables and refining the definitions of existing variables, the model's understanding of the decision tree algorithm is enhanced. Each iteration builds upon the previous one, adding more variables and calculations as needed. The variables include R (the set of instances), K (the attribute being evaluated), Ri (the subset of R with K = i), and values(K) (the possible values of K). Other variables include entropy, information gain (IGain), H(R) (the entropy of the entire set), H(Ri) (the entropy of each subset), p(i) (the probability of each value of K), and Info (R, K) (the information content of K in R). The exact variables used may vary depending on the level of detail required in the analysis.

$$IGain\left(R, K\right) = Entropy\left(R\right) - \sum\nolimits_{i \in values(K)} \frac{|R_i|}{|R|}.Entropy\left(R_i\right) \tag{1}$$

Eq. (1) defines Entropy as a metric for measuring the uncertainty of a random variable, which helps in determining the impurity of a set of random instances [26,27]. More information is contained when Entropy is higher. The steps to build a decision tree using information gain are as follows:

1) The first node in the decision tree should represent all training instances connected to the root node.
2) Determine which attribute to assign to each node by calculating the information gain.
3) Ensure that no root-to-leaf path has more than one instance of a discrete attribute.
4) Recursively create each subtree on the subset of training cases categorized down that tree branch.
5) If a node has just positive or only negative training examples, respectively, label it with "Yes" or "No".
6) If the label is "No", then the bulk of the training cases are present at that node.
7) The Gini Index is a metric that measures the probability of incorrectly identifying a randomly chosen element. Choose the attribute with a lower Gini Index. Decision trees offer several advantages, including their ability to generate clear rules, conduct categorization without requiring much computation, handle both continuous and categorical variables, and highlight the critical fields for categorization or prediction [28].

However, decision trees have some drawbacks. They are less suitable for estimating situations where the objective is to predict the value of a continuous characteristic and are prone to errors in classification problems with multiple classes and limited training samples. Additionally, the training of decision trees can be computationally expensive [29]. The growth of a decision tree requires extensive computing work, with each potential splitting field at each node needing to be sorted to determine the optimal split. Some algorithms use combinations of fields, which require finding the best-combining weights. Pruning algorithms can also be costly due to the need to create and evaluate multiple candidate sub-trees [30].

2.2 *Flower Pollination Algorithm (FPA)*

The FPA approach is based on the principles of natural pollination, which involves propagating the fittest flowers of a species through reproduction [29]. In FPA, each potential solution is represented by a flower or pollen (assuming that each plant produces only one flower and one pollen). During the optimization process, the search space is explored through "biotic and cross-pollination, with pollen movement modeled using Levy flight". Levy flight is a random walk that involves selecting a random step from the Levy distribution [28]. Levy flying is more valuable than walking because the intervals between big and tiny random steps are consistent. The Levy distribution is utilized to represent anomalous diffusion, which causes considerably longer migration from its current position and has infinite mean and infinite variance.

The distribution has strong power-law tails. Consequently, it is more effective in exploring the search space. The intensification stage is represented by abiotic self-pollination [22]. FPA combines both exploration and exploitation to ensure high-quality search, and randomly alternates between them. "The pseudo-code for FPA presented in FPA relies on the following idealized concepts" [11]: P1: Biotic, Levy flight-based cross-pollination serves as a worldwide search mechanism. P2: Abiotic and self-pollination processes are used to do local searches. P3: Because two flowers are identical, floral constancy may be implicated.

```
Initialize parameters with switching probability p ∈ [0, 1];
Generate the initial population of flowers randomly;
Evaluate the initial population and find the current best solution gbest;
while (stopping criterion not satisfied) do
For each flower
if rand () <p
        Global pollination: $x_i^{t+1} = x_i^t + L(x_i^t - gbest)$; // Based on Lévy step
else
         Select two random solutions $x_j^t$ and $x_k^t$;
          Local pollination: $x_i^{t+1} = x_i^t + \epsilon\ (x_j^t - x_k^t)$;
end if
    Evaluate new solutions;
     Update solutions with better new ones;
end for
  Keep the current best solution;
end while
```

P4: A random probability between [0, 1] is used to switch between local and global search.

The method begins by randomly creating an initial population that is then analyzed to identify the current best option. Before generating a new solution in FPA, a pollination type should be selected based on a predefined probability p (P4).

If a random number r is generated between 0 and 1, the global pollination and flower constancy (P1 and P3) can occur if r is less than p, as follows:

$$x_i^{t+1} = x_i^t + \gamma \mathrm{L}(x_i^t - g_{best}) \tag{2}$$

In Eq. (2): x_i^t–solution i at time tg_{best}–current best solution γ–scaling factor.

In Eq. (3): L-step size was drawn from Levy flight as:

$$L(s, a) \sim \frac{\lambda \Upsilon(\lambda) sin(\frac{\pi\lambda}{2})}{\pi} \frac{a}{s^{1-\lambda}}, |s| \to \alpha \tag{3}$$

Here, $\Gamma(\lambda)$–standard gamma Function. a is used as a control parameter which can be assigned 1 and s is given by Eq. (4).

$$s = \frac{U}{|V|^{\lambda-1}} \tag{4}$$

Here, U and V are two random samples drawn from a Gaussian normal distribution with mean = 0, and standard deviation = σ_u and σ_v given by Eqs. (5) and (6).

$$U \sim \left(0, \sigma_u^2\right), V \sim \left(0, \sigma_v^2\right) \tag{5}$$

$$\sigma_u = \left[\frac{\Gamma(1+\lambda)}{\lambda \Gamma \frac{(1+\lambda)}{2}} \cdot \frac{sin\left(\frac{\pi(\lambda)}{2}\right)}{2^{\frac{(\lambda-1)}{2}}}\right]^{\frac{1}{\lambda}}, \sigma_u = 1 \tag{6}$$

However, if r > p, then the local pollination and the flower constancy (P2 and P3) performed as shown in Eq. (7).

$$x_i^{t+1} = x_i^t + \epsilon \left(x_j^t - x_k^t\right) \tag{7}$$

Here, x_j^t and x_k^t–two randomly selected solutions. $\epsilon \in [0, 1]$–random number.

The current best is modified, and the search iterations continue until the termination requirements are met. FPA, a nature inspired algorithm based on the biological process of pollination, is considered to be one of the most advanced optimization techniques [22]. It has been widely applied in engineering [22] and various research domains, such as Cuckoo Search Algorithm (CSA), and has shown significant advantages over other popular algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithm (GA), Non-dominated Sorting Genetic Algorithm II (NSGA II), and Clonal Selection Algorithm.

FPA's performance and effectiveness have been verified using several well-known benchmark problems, and its ability to handle optimization tasks has been demonstrated [10,13,14]. The FPA algorithm is adaptable to solving a variety of problems in different NP-hard situations, and the evolution process shows that it is prepared, flexible, and capable. The extensive and varied size of the IaaS cloud and the diverse resource management needs make it challenging to employ FPA directly for resource allocation problems. The solution space is extensive, and it may require a long time to locate an efficient solution. To address this challenge, a new search operator strategy, including decision trees, must be integrated, depending on the problem's characteristics [29].

3 Proposed Knowledge-Based Flower Pollination Algorithm (KB-FPA)

The proposed research focuses on resource allocation methods that employ a hybrid approach. Metaheuristic techniques are classified as reactive allocation, whereas machine learning is classified as proactive allocation. For this study, the Flower Pollination Algorithm [14,29] is combined with a decision tree algorithm [28]. This indicates that the study deployed proactive resource allocation strategies to generate resource tables, which were then used to develop a knowledge-based approach. When a certain event, such as task allocation to resources, exceeded a specified limit, the reactive allocation technique was used. In case a task needs higher memory for its execution and there are no available resources. Fig. 4 explains the design approach suggested in the study. The framework represents a system for resource allocation and scheduling in a computing environment. It involves multiple hosts, virtual machines (VMs), a scheduling strategy, a resource allocation model, the Flower Pollination Algorithm, and the Decision Tree.

1. **Hosts and VMs (Virtual Machines):** The system consists of multiple hosts, each of which has a set of VMs associated with it. VMs are virtualized computing instances that run on physical hosts.
2. **Scheduling Strategy:** The "Scheduling Strategy" component represents the approach or algorithm used to schedule tasks and allocate resources in this computing environment. It involves deciding which VM should execute which tasks.
3. **Resource Allocation Model:** The "Resource Allocation Model" is a critical part of the system. It defines how resources, such as CPU, memory, and storage, are allocated to VMs or tasks.
4. **Decision Tree:** The "Decision Tree" is another algorithm or method used for making decisions, related to task and resource allocation based on knowledge.

5. **Flower Pollination Algorithm:** The "Flower Pollination Algorithm" is a specific algorithm used for resource allocation. It is responsible for optimizing resource allocation based on memory and RAM utilization criteria.
6. **VM Pool:** The "VM Pool" is a collection of available VMs that can be assigned to execute tasks.

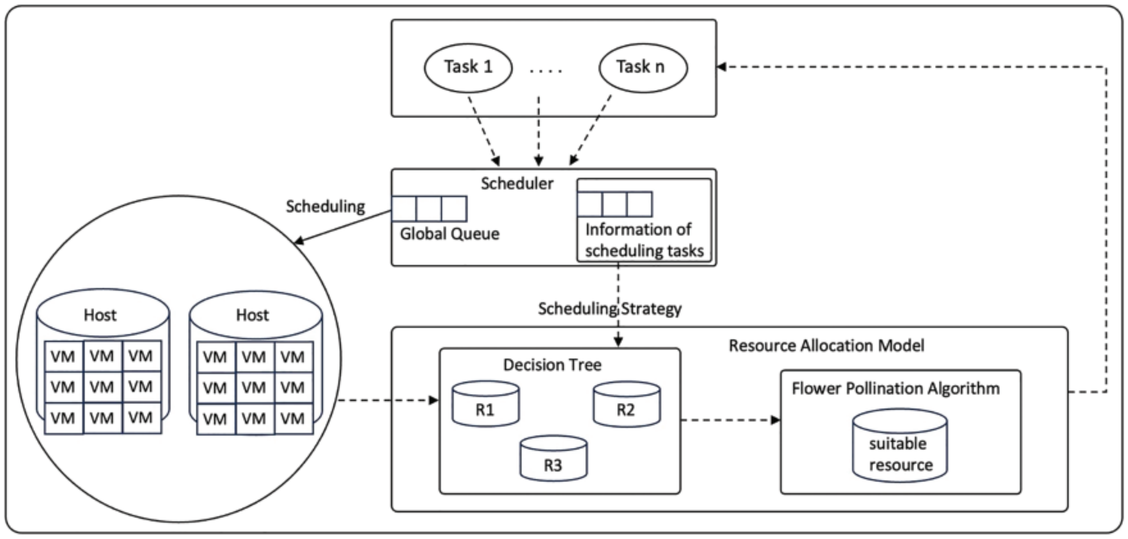

Figure 4: Conceptual design of KB FPA

The system combines both proactive and reactive resource allocation strategies. The Flower Pollination Algorithm is used for proactive allocation. This means it is used to make resource allocation decisions in advance and the system employs a knowledge-based approach to resource allocation. This involves using historical data or patterns obtained through a Decision Tree. The ultimate goal of the system is likely to efficiently allocate resources to tasks or VMs to optimize performance and resource utilization.

3.1 Design Model

This section proposes a system that allocates resources efficiently to the tasks assigned by the cloud users. In this proposed system, when the user assigns a task to the cloud scheduler, it allocates the best-matched VM to the task, and this selection uses the proposed KB-FPA. This work presents the Decision Tree algorithm and Flower Pollination algorithm to construct a hybrid approach for resource allocation.

The proposed KB-FPA initializes the resources of the cloud and creates a knowledge base (KB) using a decision tree algorithm. Then the machine selection is made using 6 the flower pollination algorithm. The flow of this proposed KB-FPA is as described below:

Step 1: To create a KB, first, it retrieves the list of cloud resources with its attributes, including RAM and Storage (in GB) of the virtual machines (VMs) associated with the resources.

Step 2: Construction of Decision Tree (DT): The first step is to decide which attribute becomes the root of the DT.

Step 3: As mentioned in the previous section, the information gain and Gini Index can decide the root node. Here in this work, Information Gain (IGain) is used.

Step 4: To find out the attribute with the highest information gain.

If IGain (RAM) > IGain (Storage), then, the Decision Tree will find the attribute with the highest information gain

ELSE the Decision Tree will find the attribute with non-highest information gain

Step 5: When a user assigns a task to the cloud, it follows the decision tree as per their requirement.

For example, in the leaf node, there may be only one resource or maybe more than one.

If the number of VMs == 1 The task is given to that VM

ELSE VM Selection is made using FPA.

Step 6: The pseudo-code of FPA for VM selection is given:

Algorithm 1: Flower Pollination Algorithm for VM Allocation

```
Require: Resources obtained after applying DT ($R_i$), Virtual Machines ($VM_i$), Workload ($W_i$)
Ensure: Allocate $VM_i$ to $W_i$
Begin
1: Initialize a population of k flowers/pollen gamete with random solutions;
2: Find the best solution b ∗ in the initial population;
3: Define switch probability p;
4: while t < Max iteration do
 5: for i = 1: K do
 6: if rand < p then
 7: Draw a step vector (d-dimensional) that obeys Levy distribution (L);
8: Perform Global Pollination;
9: else
10: Draw ϵ from the uniform distribution;
 11: Randomly select m and n among all selected solutions;
12: Do Local Pollination;
13: end if
14: Evaluate new solutions;
15: if new solutions are better, update them in the population then
16: end if
17: end for
18: Find the current best b ∗;
19: end while
 End
```

3.2 *Proposed System Architecture*

Fig. 5 presents the proposed KB FPA system architecture. Here, the data center of the cloud has various resources, and each resource has its VMs for which the knowledge base is generated using DT and the scheduler uses this KB for resource allocation. DT is suitable for prediction. A model is generated using decision tree algorithms, which learn uncomplicated decision rules from data features to anticipate the value of a target variable. Further, FPA helps in resource allocation based on the task or workload assigned by the users. The Decision Tree deals with the selection of resources from the pool of resources. The resources are selected considering the parameters like RAM and storage. The selection of the root node for the decision tree is done through the "Information Gain and Gini Index". Based on this, we can divide the proposed system architecture into two phases:

(i) Resource Knowledge Base generation.

(ii) Enhanced Resource Allocation.

In Phase 1, the knowledge base is generated for virtual machines where a decision tree algorithm is used. This work uses two main attributes, RAM and Storage. A decision tree is generated using these attributes, one of which is treated as the attribute for the root node based on which a further tree is created.

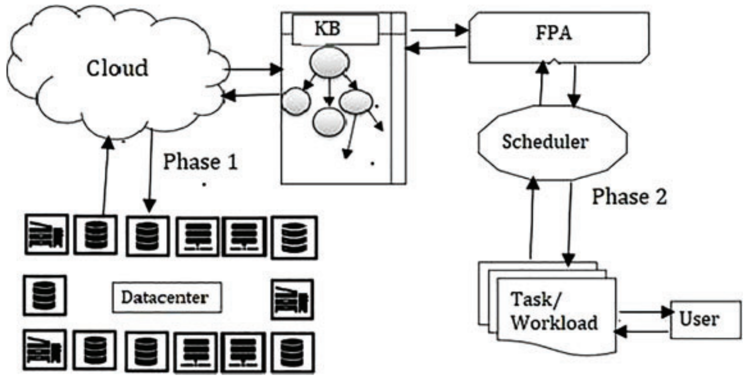

Figure 5: Proposed system architecture

As per Fig. 6, the minimum workload requirement for RAM is 90 GB, and for storage, it is 1400 GB. Based on this requirement, KB provides the resource list RL2 that fulfills the required configuration and moves to Phase 2. Outcome: Out of a pool of resources it will only select those resources that are likely to fit user requirements. It selects some of the possible solutions. Advantage: Rather than assigning random resources there is a systematic process to assign resources.

Fig. 6. Resource Allocation based on KB Phase 2-Enhanced Resource Allocation After Phase 1, when provided with a list of resources from the KB for the assigned workload, FPA is used to select a suitable VM for finding a feasible solution. Instead of random initialization, the work employs an efficient resource allocation objective function to minimize energy consumption. Until the objective function is achieved the whole process is iterated multiple times.

Outcome: Hence, the final allocation is done to the assigned workload. Similarly, resources are allocated to all the assigned tasks in the cloud system. Finds the improved solution from multiple solutions.

Advantage: Reduce resource wastage by selecting the appropriate resource as per task. As the number of resources is less due to the screening process done in Phase 1, the time complexity is also reduced. In the earlier model, the Dynamic Switching Probability (DSP) strategy was used with FPA for finding and balancing the exploration and exploitation of local and global search which was helpful in resource provisioning but the system has not been considered for a multi-objective approach. The environment used for simulation is homogeneous in nature, thus lacking to meet the dynamic user needs.

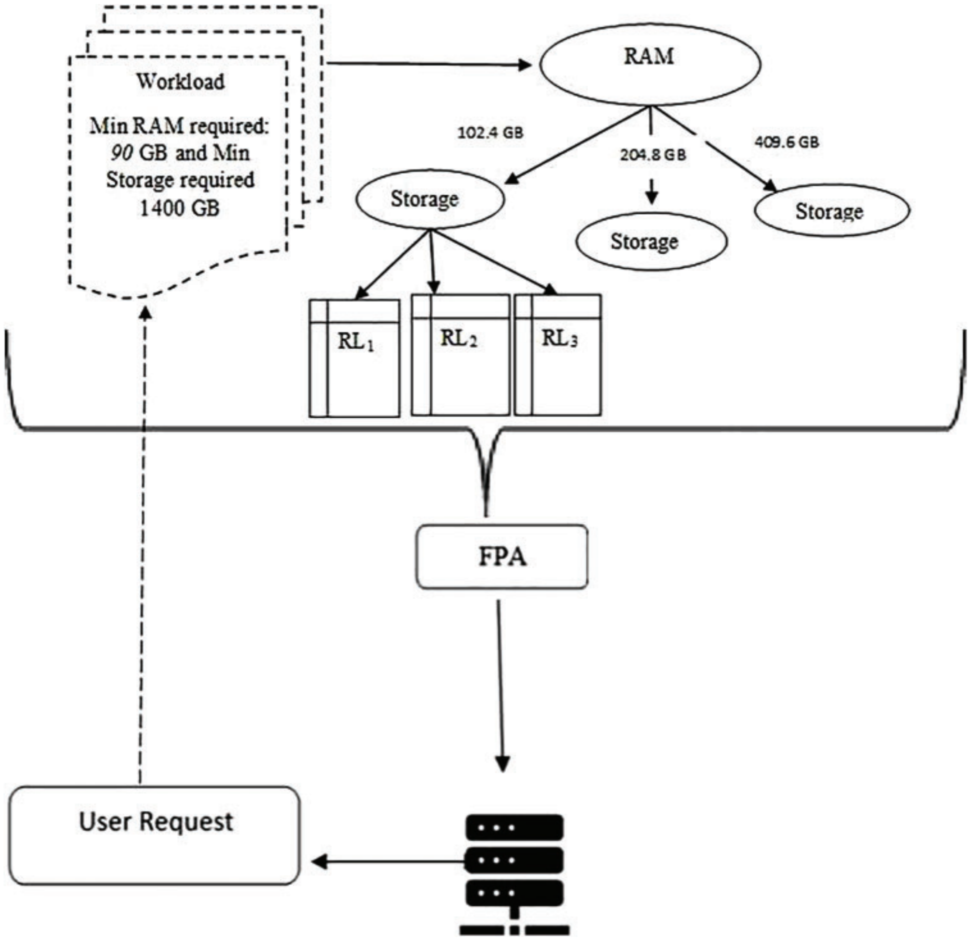

Figure 6: Resource allocation based on KB

4 Results and Discussion

This work proposed an efficient resource allocation approach that utilizes reactive and proactive methods for assigning resources to the tasks given by the cloud servers. For efficient resource handling, the knowledge base of the resources is generated using a decision tree algorithm, and the final resource allocation is based on the flower pollination algorithm. To analyze the performance of this proposed approach, the local cloud environment is implemented using Visual Studio, and performance is computed based on different performance metrics, including resource utilization and energy consumption.

4.1 Simulation Setup

The performance of the proposed approach is analyzed on a local cloud environment generated using Visual Studio. In this environment, a total of 800 physical machines (PMs) are added, and each machine has its own configuration, which is randomly assigned to it. In these 800 physical machines, 2680 Virtual Machines (VMs) are created based on their configurations, where each PM has a maximum of 5 VMs. The number of PMs and VMs is kept similar to E-FPA. Nevertheless, in the proposed work, a heterogeneous environment is considered to handle dynamic user requests. The proposed work considers a meta-object approach for resources in the data center.

The simulation parameters for implementation along with the parameters for the decision tree and FPA are given in Table 1 along with the parameters of the algorithms used in the proposed architecture, and their values are decided based on experimentation and other requirements. The testing of the proposed approach is done with the given workloads and using the machines available for simulation. Table 1 gives the details of the configuration of the PMs and VMs used in this simulation. As E-FPA worked on a homogeneous environment and considered RAM to be 204.8 GB proposed approach used a heterogeneous environment with RAM values as 102.4, 204.8 and 409.6 GB which is equal to, less than, and more than the previously used RAM size.

Table 1: PM and VM simulation setup parameter

Parameter	Value
Data center	1
PM	800
VM	2680
RAM (VMs)	102.4, 204.8, 409.6 GB
Storage	1000, 1500, 2000 GB
Bandwidth	0.1 GBs-1
MIPS	367 MHz
Number of workloads	1000, 3000, 5000
Simulation parameters for KB-FPA	
Max generations	1000
Number of flower/pollens	25
Number of branches in DT	3
Level of DT	2

For the dynamic environment, the storage in the proposed work was taken to be 1000, 1500 and 2000 GB whereas in E-PFA storage was 1000 GB. We have considered storage from 1000 GB because for large user requests storage below that would have been unsuitable. Storage below 1000 GB would have been unsuitable for large user requests because it may not be enough to store the required data, resulting in insufficient space and potential loss of data. Additionally, as the number and size of user requests increase, a larger storage capacity may be needed to accommodate them. Therefore, considering storage capacities of 1000, 1500, and 2000 GB in the proposed work allows for more flexibility and scalability in handling varying user requests and data storage needs.

4.2 Performance Evaluation

Various performance metrics are used to evaluate the effectiveness of the proposed approach, including resource allocation and energy consumption resource utilization, defines the utilization of the available resources and energy consumption, and gives the values for energy consumed by the resources for the execution of the tasks or workload assigned to it. The performance evaluation of the proposed KB-FPA approach is conducted by comparing it to other approaches, namely E-FPA, SLWO, and Firefly. These approaches are evaluated using the same simulation setup to ensure a fair comparison. The physical machine size is chosen between 1–800. The workload 8-unit size considered is 1000, 3000, and 5000 with different resource utilization, varying execution time, and energy consumption.

Resource Utilization: Resource utilization defines the utilization of the machines by the cloud server for the execution of tasks. It is computed based on the number of resources used for the execution of tasks given to the cloud environment. In Figs. 7 and 8, comparison of existing algorithms such as Spark Lion Whale Optimization (SLWO), Firefly, and Flower Pollination [19] is done with the proposed algorithm. For the comparison, resource utilization of VMs and PMs of servers with tasks assigned to cloud platforms as 1000, 3000 and 5000 was considered. Figs. 7 and 8 display the cloud server's resource utilization for both virtual machines (VMs) and physical machines (PMs) in percentage to represent its performance.

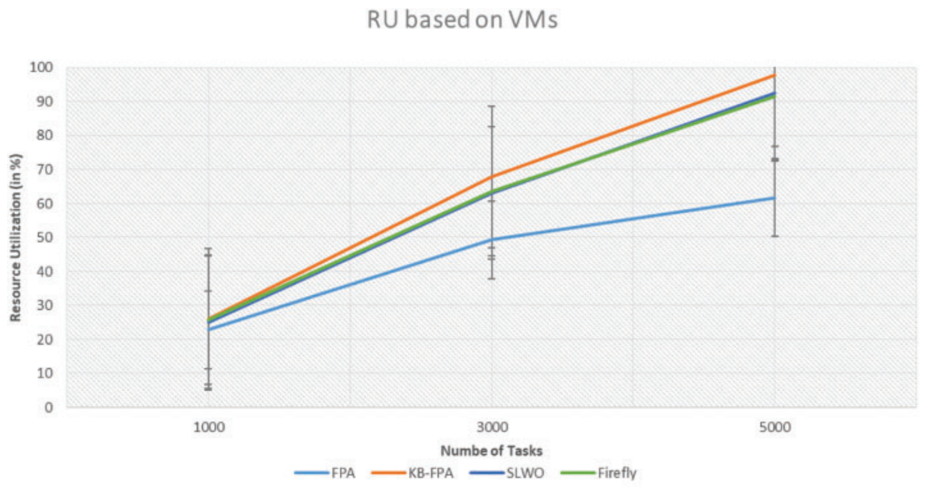

Figure 7: Resource utilization (for VMs)

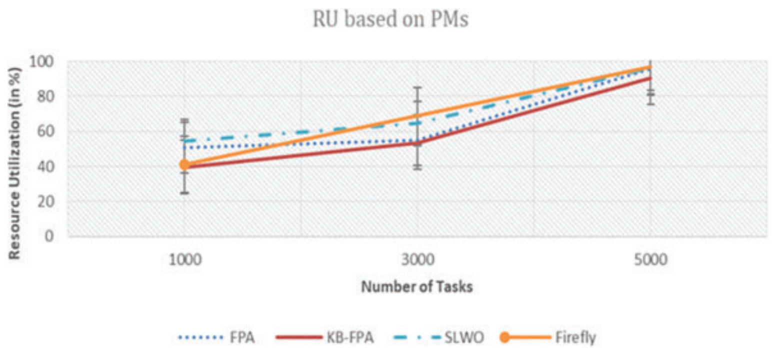

Figure 8: Resource utilization (for PMs)

According to Fig. 7, as the number of tasks increases, the utilization of virtual machines (VMs) also increases. The proposed KB-FPA algorithm outperformed traditional methods such as FPA, SLWO, and Firefly. VM utilization is maximum in the case of KB-FPA with an increase in workload. On the other hand, architecture with FPA uses fewer resources and most resources remain unutilized and some resources are overburdened due to random allocation. The results also indicate that in the case of KB-FPA when the workload is less, there is a slight change in the resource utilization but as the workload increases from 1000 to 3000 and from 3000 to 5000 a significant increase in resource utilization on VM is observed. This is because KB-FPA uses the KB strategy which generates the resource list as per user demands. Further, this resource list is used by FPA to select and assign the resource. It was observed that the KB-FPA scheme outperforms E-FPA, SLWO, and Firefly because KB-FPA uses a DT strategy that enhances FPA's efficiency by allocating resources properly.

Fig. 8 displays the resource utilization for PMs, and it has been seen that the proposed system uses 39.43% PMs when 1000 tasks were considered, which is slightly lesser than other traditional approaches such as FPA, SLWO, and Firefly. When the workload was increased to 3000 then resource utilization for KB FPA was 53.32% and in the case of 5000 workloads then resource utilization was 90.31%. This change was observed due to the use of the DT strategy. The strategy enhances the allocation process by assigning appropriate resources.

Figs. 9 and 10 represent the simulation results observed for 1000 tasks. It was observed that the proposed scheme performed better than traditional methods such as FPA, SLWO, and Firefly. This was due to the incorporation of a KB strategy that improved the efficiency of VM utilization for KB-PFA by 87.673%, compared to 79.56%, 76.64%, and 73.98% for FPA, SLWO, and Firefly, respectively. When analyzing the utilization of CPU, memory, and storage resources for PMs in a data center with 1000 tasks, it was revealed that KB-FPA utilized 39.73 of PMs. On the other hand, FPA, SLWO, and Firefly resulted in 40.54%, 40.76%, and 41% resource utilization, respectively, which led to a 15.3% increase in IaaS resource utilization. This enhancement in the average utilization of PMs can be attributed to the implementation of the KB strategy, which improved the global convergence of KB-FPA.

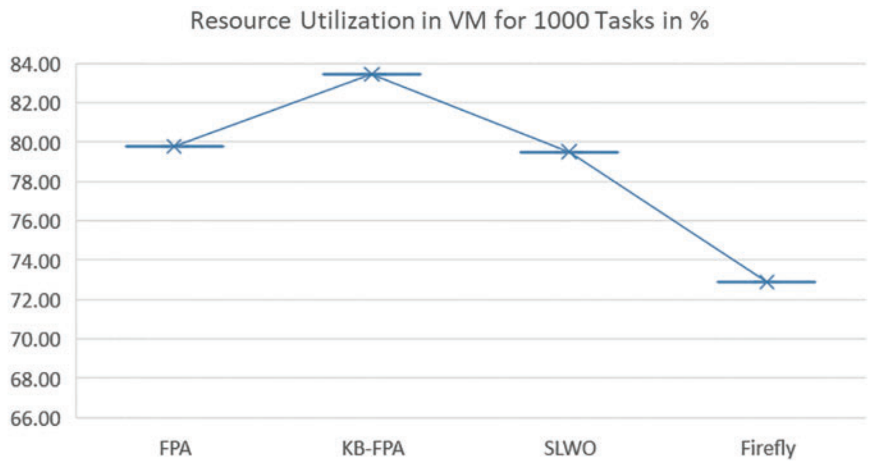

Figure 9: Resource utilization of VM for 1000 tasks

The proposed approach assigns VMs to the chosen PM. In summary, these findings demonstrate the benefits of using the KB strategy in the proposed approach, leading to an improvement in resource utilization for both PMs and VMs.

Figs. 11 and 12 represent the simulation results observed for 3000 tasks. The proposed scheme was found to outperform traditional methods, such as FPA, SLWO, and Firefly, due to the implementation of a KB strategy. This strategy enhances the efficiency of VM utilization for KB-PFA by 67.64%, which is a significant improvement compared to the efficiencies of 59.46%, 63.32%, and 61.28% achieved by

FPA, SLWO, and Firefly, respectively. The KB strategy involves the use of knowledge-based techniques to optimize the allocation of virtual machines (VMs) to tasks. This approach considers factors such as the available resources, the workload, and the priority of the tasks when making allocation decisions. By using this strategy, the proposed scheme can achieve higher efficiency in VM utilization, resulting in better performance overall.

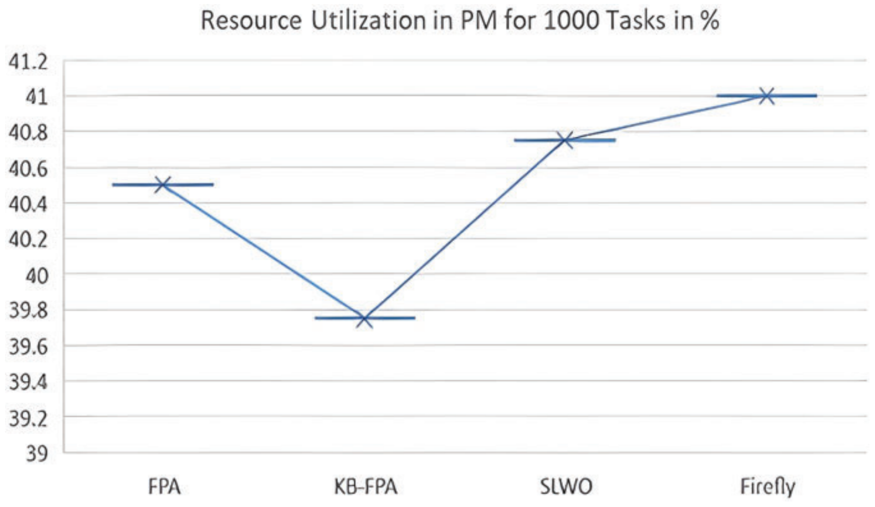

Figure 10: Resource utilization of PM for 1000 tasks

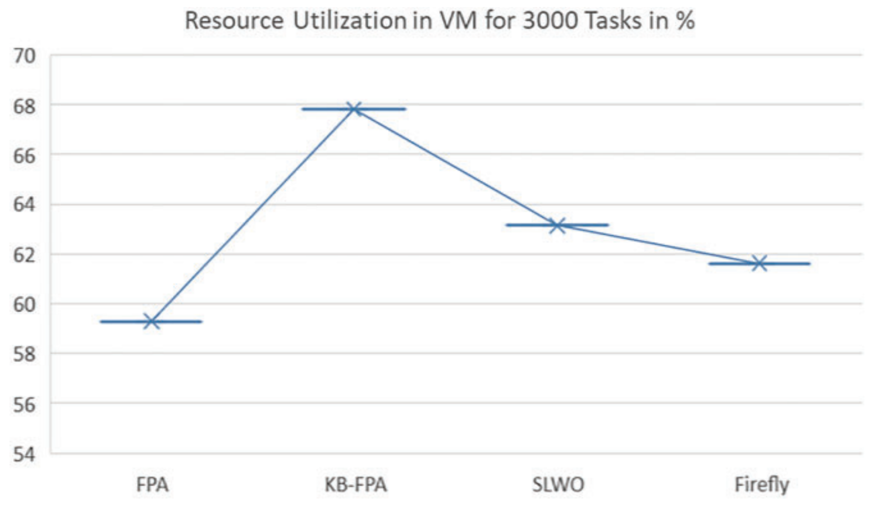

Figure 11: Resource utilization of VM for 3000 tasks

Resource utilization of CPU, memory, and storage for PM was observed in the data center for 3000 tasks. According to the results, KB-FPA utilized 67.12% of physical machines (PMs), while FPA, SLWO, and Firefly utilized 69%, 68.54%, and 69.1% of resources, respectively, resulting in a 37.3% increase in IaaS resource utilization.

This improvement in PM utilization is attributed to the incorporation of a KB strategy that enhanced the global convergence of KB-FPA, allowing for more efficient allocation of virtual machines (VMs) on selected PMs. Overall, these results demonstrate the benefits of using a KB strategy in the proposed scheme. Hence, it is concluded that Resource Utilization has improved both in terms of PMs and VMs.

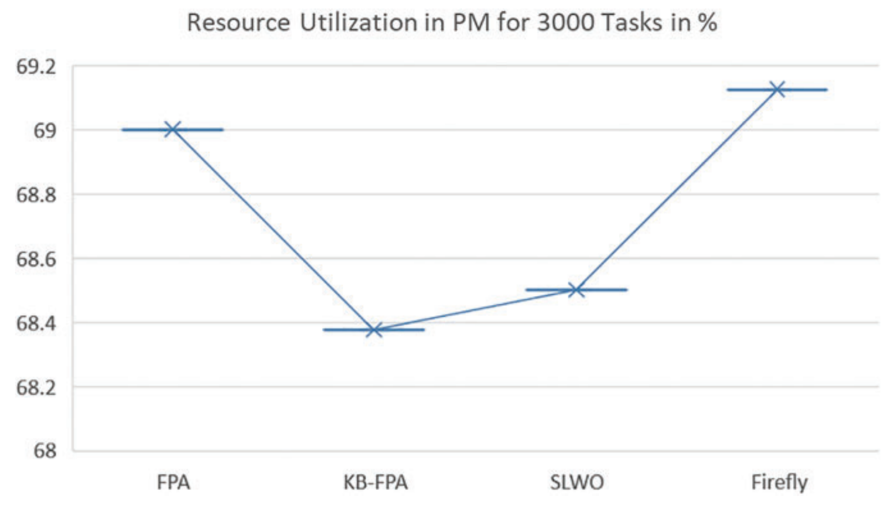

Figure 12: Resource utilization of PM for 3000 tasks

When 5000 tasks were given, the results were observed for both VMs and PMs for KB FPA. VM utilization for KB-PFA was 82.44, whereas for FPA, SLWO, and Firefly VM utilization was 81.61%, 80.50%, and 81.48%, respectively.

Figs. 12 and 13 represent the simulation results observed for 5000 tasks. The results demonstrated that the proposed scheme performed better than traditional methods such as FPA, SLWO, and Firefly. This improvement is due to the utilization of a KB strategy, which enhances the efficiency of VM utilization for KB-PFA by 82.44%. In comparison, the efficiency of FPA, SLWO, and Firefly is 81.61%, 80.50%, and 81.48%, respectively.

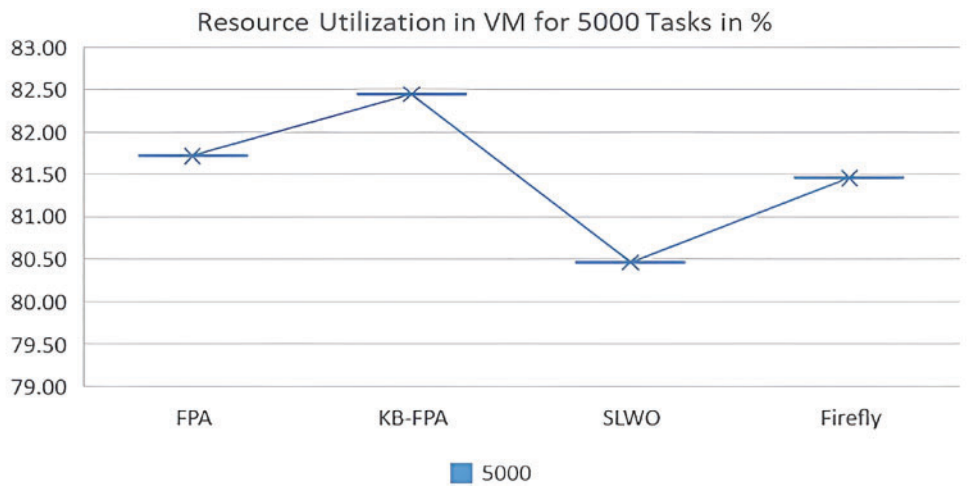

Figure 13: Resource utilization of VM for 5000 tasks

Resource utilization of CPU, memory, and storage for PM was observed in Fig. 14 the data center for 5000 tasks. The results show that KB-FPA uses 84.42% PMs, whereas FPA, SLWO, and Firefly showed 85.74%, 86%, and 86.61% resource use, producing a 32.3% increase in IaaS resource utilization. This increase in average PM usage is the result of the KB strategy's inclusion, which increased the global convergence of KB FPA. The suggested approach distributes VM to the chosen PM. Overall, the aforementioned findings support the benefits of utilizing the KB method in the suggested scheme. Hence it is concluded that Resource Utilization has improved both in terms of PMs and VMs.

Energy Consumption: Energy consumption is another important aspect of the cloud that affects its overall performance. Its computation depends upon the number of resources, type of resources, number of tasks, and type of tasks executed by the cloud server. For this study, energy consumption is computed for each set of tasks where the sums of the energy consumed by the task are taken, and after calculating the total energy used, the overall energy consumption for executing various task sets is displayed in the following Fig. 15, the energy consumption of KB-FPA, E-FPA, SLWO, and Firefly in a cloud data center is presented for various numbers of user requests. As the number of VM demands increases, all four schemes experience different degrees of energy utilization growth. However, KB-FPA demonstrates the lowest energy consumption compared to E-FPA, SLWO, and Firefly.

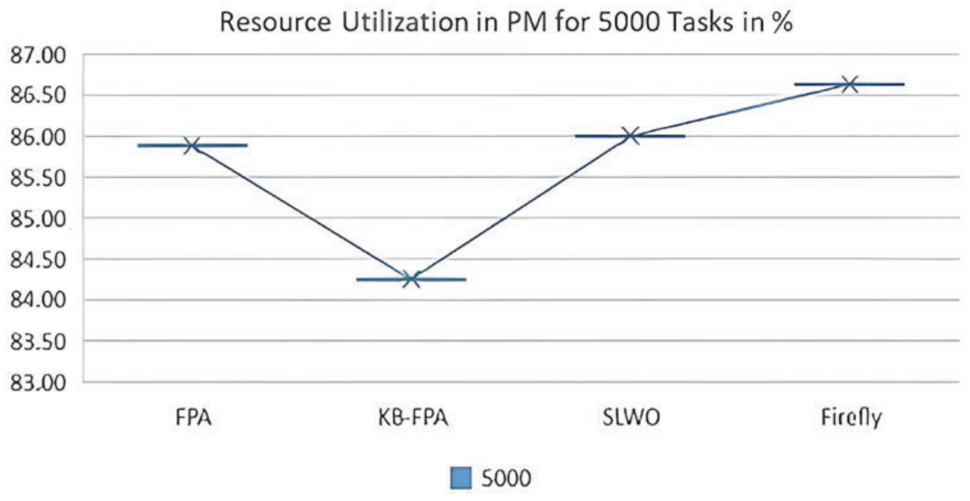

Figure 14: Resource utilization of PM for 5000 tasks

The results indicate that as the number of VM requests increases, energy utilization becomes higher and higher resulting in more PM occupancies and more energy consumption. Fig. 15 demonstrates that energy consumption increases with the number of tasks, because the more the number of tasks, the more resources are utilized, and hence it consumes more energy. Furthermore, it was observed that the performance of KB-FPA outperforms E-FPA, SLWO, and Firefly, as the proposed approach effectively explores the solution space and obtains solutions with fewer PMs. Consequently, the overall efficiency is improved in the KB-FPA scheme. According to Fig. 15, the maximum energy consumption for 1000 tasks is 46.32 kW for KB-FPA, whereas, for FPA, SLWO, and Firefly, it is 61.53 kW, 59.372 kW, and 58.372 kW hence indicating that KB-FPA has the lowest energy consumption among the four schemes. As the workload increases to 3000, energy consumption KB-FPA increases to 79.982 kW while the energy consumption for FPA, SLWO, and Firefly is 96.54 kW, 83.745 kW, and 82.484 kW respectively. At a workload of 5000, it became evident that KB-FPA consumed 98.92 kW of energy, while FPA, SLWO, and Firefly consumed 108.4 kW, 120.312 kW, and 129.330 kW of energy. As a result, the overall data center infrastructure energy consumption was reduced by 26.5.

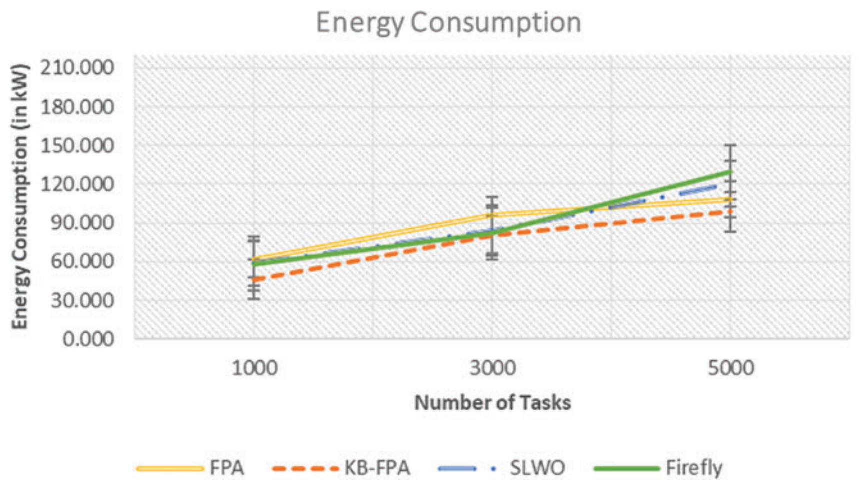

Figure 15: Energy consumption (in kW)

5 Conclusion

The work has presented a Knowledge Based-Flower Pollination Algorithm for cloud computing to achieve efficient resource allocation. This proposed approach utilized the intelligence of machine learning and an optimization algorithm to allocate resources to the given tasks. In this, the decision tree algorithm is initialized with the resources and provides a knowledge base to the scheduler, reducing the time for the final selection. Moreover, the final resource selection is made using flower pollination optimization. This proposed combination achieved better results which are computed based on the experimentation. Three performance metrics, Resource Utilization, Execution Time, and Energy Consumption are calculated with the simulations and the results.

The overall performance of KB-FPA outperforms FPA. The present study differs from previous studies in two distinct ways. First, most of the studies have taken FPA for resource scheduling but in this study, FPA is used for allocating best-matched resources to the task. Secondly, most of the studies have taken a small set of workloads. The knowledge-based flower pollination algorithm (KB-FPA) is a promising method for improving the efficiency of resource utilization in cloud computing. By incorporating a knowledge-based strategy, the KB-FPA algorithm was able to make more informed decisions about the allocation of VMs to PMs, resulting in higher overall efficiency and performance. Through experimentation and analysis, the proposed KB-FPA scheme outperformed traditional methods such as FPA, SLWO, and Firefly.

This improvement was attributed to the KB strategy, which improved the global convergence of KB-FPA and enhanced the efficiency of VM utilization by a significant margin. The results of this study indicate that integrating a knowledge-based approach into optimization algorithms has the potential to improve performance and efficiency in cloud computing systems. Overall, the findings of this study have significant implications for the design and optimization of cloud computing systems.

The proposed work looks at cloud computing from the perspective of resource allocation and resolves problems with energy usage and QoS. Energy-efficient resource management faces major obstacles from other resource management techniques like resource scheduling. It would be interesting to look into how collocating affects performance and total energy use.

6 Limitations

While the Knowledge-Based Flower Pollination Algorithm (KB-FPA) presents a promising approach for improving resource allocation efficiency in cloud computing, certain limitations need to be acknowledged:

Scope of Workloads: The current study primarily focuses on specific workloads, and the results are based on experimentation with these workloads. Extending the research to encompass a wider range of workloads with varying characteristics could provide a more comprehensive understanding of the algorithm's performance in diverse scenarios.

Scalability: The scalability of KB-FPA in large-scale cloud environments remains an open question. Further investigation is needed to assess the algorithm's performance as the size of the cloud infrastructure and the number of tasks and resources increase.

Real-World Implementation: The presented work is primarily based on simulations and experiments. To assess the practical feasibility of KB-FPA, real-world implementations and testing in production cloud environments are necessary.

Complexity: The incorporation of a knowledge-based strategy into the optimization algorithm may introduce computational complexity. A more in-depth analysis of the algorithm's computational requirements, especially for larger datasets, is essential.

6.1 Future Work

The presented work opens avenues for future research and improvements in the field of cloud computing resource allocation. Some potential directions for further investigation include:

Collocation Effects: Exploring the impact of collocating virtual machines (VMs) on performance and energy usage is a promising area of research. Investigating how different VM placement strategies influence resource utilization, energy efficiency, and Quality of Service (QoS) could lead to valuable insights.

Dynamic Resource Allocation: Research into dynamic resource allocation strategies that adapt to changing workloads and resource demands can provide more efficient and responsive cloud systems. Machine learning and reinforcement learning techniques may be applied to enhance the adaptability of resource allocation.

Real-World Validation: Conducting real-world experiments and case studies in actual cloud environments to validate the effectiveness of KB-FPA and to identify any operational challenges.

Security and Privacy: The security and privacy implications of resource allocation algorithms should also be considered. Investigating how KB-FPA or similar approaches impact data security, privacy, and compliance with data protection regulations is crucial.

Multi-Objective Optimization: Extending the optimization criteria beyond Resource Utilization, Execution Time, and Energy Consumption to consider a broader range of objectives, such as cost optimization, fault tolerance, and performance guarantees.

Hybrid Approaches: Combining KB-FPA with other resource allocation techniques, such as predictive analytics, reinforcement learning, or genetic algorithms, to create hybrid strategies that leverage the strengths of multiple approaches.

Addressing these limitations and exploring these future directions will contribute to the advancement of resource allocation techniques in cloud computing, resulting in more efficient and effective cloud systems.

Acknowledgement: This work was supported by the Ministerio Español de Ciencia e Innovación under Project Number PID2020-115570GB-C22 MCIN/AEI/10.13039/501100011033 and by the Cátedra de Empresa Tecnología para las Personas (UGR-Fujitsu).

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: N. Chauhan and N. Kaur; data collection: N. Chauhan, S. Verma, Kavita and K. S. Saini; analysis and interpretation of results: N Chauhan; draft manuscript preparation: N. Chauhan, R. A. Khurma and P. A. Castillo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The experiments conducted in this study utilized real data obtained from Planet Lab, encompassing information from over a thousand servers, including their corresponding components utilizations. The data forms the foundation for the simulations carried out to investigate "Maximizing Resource Efficiency in Cloud Data Centers through Knowledge-Based Flower Pollination Algorithm". All Published SPEC SPECpower_ssj2008 Results, Data Sets | PlanetLab (princeton.edu).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Belgacem, S. Mahmoudi, and M. Kihl, "Intelligent multi-agent reinforcement learning model for resources allocation in cloud computing," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 2, pp. 1198–1206, Feb. 2022. doi: 10.1016/j.jksuci.2022.03.016.

[2] N. R. Moparthi *et al.*, "An improved energy-efficient cloud-optimized load balancing for IoT Frameworks," *Heliyon*, vol. 9, no. 12, pp. e21947, Dec. 2023. doi: 10.1016/j.heliyon.2023.e21947.

[3] M. A. Abbas *et al.*, "An ANN based bidding strategy for resource allocation in cloud computing using IoT double auction algorithm," *Sustain. Energy Technol. Assess.*, vol. 52, pp. 102358, Feb. 2022.

[4] J. K. Samriya and N. Kumar, "Spider monkey optimization based energy-efficient resource allocation in cloud environment," *Trends Sci.*, vol. 19, no. 1, pp. 1710, Jan. 2022. doi: 10.48048/tis.2022.1710.

[5] S. Bharany *et al.*, "A systematic survey on energy-efficient techniques in sustainable cloud computing," *Sustainability*, vol. 14, no. 10, pp. 6256, May 2022. doi: 10.3390/su14106256.

[6] P. J. Kuehn, "Energy effciency and performance of cloud data centers," in *Proc. 5th Int. Workshop Energy Efficient Data Centres*, Jun. 2016.

[7] P. Vinothiyalakshmi and R. Anitha, "Enhanced multi-attribute combinative double auction (EMCDA) for resource allocation in cloud computing," *Wirel. Pers. Commun.*, vol. 122, no. 4, pp. 3833–3857, 2022. doi: 10.1007/s11277-021-09113-8.

[8] I. Batra, S. Verma, and K. Verma, "Performance analysis of data mining techniques in IoT," in *2018 4th Int. Conf. Comput. Sci. (ICCS)*, Jalandhar, India, 2018, pp. 194–199. doi: 10.1109/ICCS.2018.00039.

[9] R. K. Puri, R. K. Challa, and N. K. Sehgal, "Energy-efficient delay-aware preemptive variable-length time slot allocation scheme for WBASN (eDPVT)," in *2nd Int. Conf. Commun., Comput. Netw.*, Chandigarh, India, 2018, pp. 183–194.

[10] L. Chen and H. Shen, "Towards resource-efficient cloud systems: Avoiding over-provisioning in demand—prediction based resource provisioning," in *IEEE Int. Conf. Big Data (Big Data)*, 2016, pp. 697–705.
[11] Z. A. Alkareem Alyasseri *et al.*, "A hybrid flower pollination with-hill climbing algorithm for global optimization," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 8, pp. 4821–4835, 2022.
[12] P. Devarasetty and S. Reddy, "Genetic algorithm for quality of service: Based resource allocation in cloud computing," *Evol. Intell.*, vol. 14, no. 2, pp. 381–387, Jun. 2021. doi: 10.1007/s12065-019-00233-6.
[13] D. Garg, S. Rani, N. Herencsar, S. Verma, M. Wozniak and M. F. Ijaz, "Hybrid technique for cyber-physical security in cloud-based smart industries," *Sensors*, vol. 22, no. 12, pp. 4630, 2022. doi: 10.3390/s22124630.
[14] K. M. Ong, P. Ong, and C. K. Sia, "A new flower pollination algorithm with improved convergence and its application to engineering optimization," *Decis. Analyt. J.*, vol. 5, no. 3, pp. 100144, 2022. doi: 10.1016/j.dajour.2022.100144.
[15] N. Chauhan, N. Kaur, and K. S. Saini, "Energy efficient resource allocation in cloud data center: A comparative analysis," in *2022 Int. Conf. Comput. Modell., Simul. Optim. (ICCMSO)*, Pathum Thani, Thailand, Apr. 27, 2022, pp. 201–206. doi: 10.1109/ICCMSO58359.2022.00049.
[16] S. Sharma, S. Verma, and K. Jyoti, "Kavita hybrid bat algorithm for balancing load in cloud computing," *Int. J. Eng. Technol.*, vol. 7, no. 4.12, pp. 26–29, 2018. doi: 10.14419/ijet.v7i4.12.20986.
[17] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, S. Urushidani and E. Oki, "Robust optimization model for primary and backup resource allocation in cloud providers," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2920–2935, 2021. doi: 10.1109/TCC.2021.3051018.
[18] J. Singh and M. S. Goraya, "An Autonomous multi-agent framework using quality of service to prevent service level agreement violations in cloud environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, 2023. doi: 10.14569/IJACSA.2023.0140325.
[19] R. A. Khurma, I. Aljarah, A. Sharieh, and S. Mirjalili, "Evolopy-FS: An open-source nature-inspired optimization framework in python for feature selection," *Algorithms Intell. Syst.*, vol. 17, no. 4, pp. 131–173, Nov. 2019. doi: 10.1007/978-981-32-9990-0_8.
[20] A. Mohamed *et al.*, "Software-defined networks for resource allocation in cloud computing: A Survey," *Comput. Netw.*, vol. 195, no. 10.07, pp. 108151, 2021. doi: 10.1016/j.comnet.2021.108151.
[21] A. Hameed *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016. doi: 10.1007/s00607-014-0407-8.
[22] H. H. Patel and P. Prajapati, "Study and analysis of decision tree based classification algorithms," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 74–78, 2018. doi: 10.26438/ijcse/v6i10.7478.
[23] S. Josilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 85–97, Feb. 2019. doi: 10.1109/TNET.2018.2880874.
[24] S. Rostami, A. Broumandnia, and A. Khademzadeh, "An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm," *J. Supercomput.*, vol. 80, no. 6, pp. 7812–7848, 2023. doi: 10.1007/s11227-023-05725-y.
[25] M. Tarahomi, M. Izadi, and M. Ghobaei-Arani, "An efficient power-aware VM allocation mechanism in cloud data centers: A micro genetic-based approach," *Cluster Comput.*, vol. 24, no. 2, pp. 919–934, 2021. doi: 10.1007/s10586-020-03152-9.
[26] F. S. Prity, K. M. A. Uddin, and N. Nath, "Exploring swarm intelligence optimization techniques for task scheduling in cloud computing: Algorithms, performance analysis, and future prospects," *Iran J. Comput Sci.*, vol. 2, no. 1, pp. 129, 2023. doi: 10.1007/s42044-023-00163-8.
[27] H. Zhao *et al.*, "VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment," *J. Parallel Distr. Comput.*, vol. 176, no. 8, pp. 17–27, 2023. doi: 10.1016/j.jpdc.2023.02.003.
[28] M. Kumar *et al.*, "A smart privacy preserving framework for industrial IoT using hybrid meta-heuristic algorithm," *Sci. Rep.*, vol. 13, no. 1, pp. 5372, 2023. doi: 10.1038/s41598-023-32098-2.

[29] P. A. Harrison *et al.*, "Selecting methods for ecosystem service assessment: A decision tree approach," *Ecosyst. Serv.*, vol. 29, no. 2, pp. 481–498, 2018. doi: 10.1016/j.ecoser.2017.09.016.

[30] D. Li and S. Chen, "A novel splitting criterion inspired by geometric mean metric learning for decision tree," in *2022 26th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2022.