Modified Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environments

Qian LI*, Xue WANG
College of Electrical Information-School of Changchun, Guanghua University, Changchun 130000, China

***Abstract*—Task scheduling in cloud computing is a complex optimization problem influenced by the ever-changing user requirements and the different architectures of cloud systems. Efficiently distributing workloads across Virtual Machines (VMs) is critical to mitigate the negative consequences of inadequate and excessive workloads, such as higher power consumption and possible machine malfunctions. This paper presents a novel method for dynamic load balancing using a Modified Artificial Bee Colony (MABC) algorithm. The ABC algorithm has exceptional competence in solving complex nonlinear optimization problems based on bee colonies' foraging behavior. Nevertheless, the traditional version of the ABC algorithm cannot effectively use resources, resulting in a rapid decline in population diversity and an ineffective spread of knowledge about the best solution between generations. To address these limitations, this study integrates a genetic model into the algorithm, enhancing population diversity through crossover and mutation operators. The developed algorithm is compared with the prevailing algorithms to confirm its effectiveness. The results of the proposed MABC algorithm for the load balancing method are compared with the current ones, and it is observed that this algorithm is more beneficial in terms of cost and energy as well as resource utilization.**

Keywords—Resource utilization; cloud computing; task scheduling; Artificial Bee Colony; genetic algorithm

I. Introduction

Cloud computing is a recently emerged computing paradigm that provides a wide variety of services using the resources of hardware and software systems available in the data centers through the Internet [1]. These services are pay-as-you-go, meaning users can acquire computing resources, storage, applications, and services on demand. Opting for cloud services presents many benefits to the users, including scalability, global accessibility, reliability, flexibility, and reduced costs for businesses. It permits organizations to quickly extend and reduce their IT infrastructure; thus, resources can be delivered and regained at a minimum cost [2].

A. Context

Cloud computing services are offered in three primary ways: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), with four deployment strategies: private, public, community, and hybrid [3]. SaaS enables consumers to effortlessly utilize cloud providers' programs without the requirement to acquire, install, and manage the software on their servers. This solution eliminates the need to manage the cloud infrastructure and platform used by the software [4]. PaaS allows consumers to utilize computational assets and applications via the Internet without requiring them to handle the underlying infrastructure personally [5]. Instances of this platform category encompass GoGrid, Aptana, EMC Atmos, and Amazon Elastic Cloud. These platforms enable customers to acquire the necessary technologies without incurring real hardware and software costs. IaaS provides many capabilities, offerings, and assets for developing an on-demand virtual computing framework [6]. Such suppliers include Google Base, IBM, Savvis, Rackspace, and Amazon Web Services.

Third-party vendors offer public cloud services over the Internet. The term "public cloud" does not imply that a user's data is accessible or useable by the general public. Public cloud services often provide consumers with an access control method [7]. Private clouds are defined as data and process management for a company and are protected from external network bandwidth constraints, security concerns, or legislative provisions in cloud-based data sharing. Private cloud services improve security and operational resilience by limiting user access and networks while having unmodified infrastructure control for providers and users [8].

In the community cloud, various organizations share and manage cloud computing infrastructure for a special interest group, specific security needs, or a similar mission [9]. From a security perspective, this type of cloud is better than a public cloud, as it allows the community to maintain its security environment and apply additional requirements. This allows each member to share resources and store them and use applications without sharing the same physical environment. A hybrid cloud is a cloud service model that includes two or more public, private, and community clouds connected to one another by standard or private technology. This method gives cloud users more power over their data and application use while remaining independent entities [10].

B. Problem Statement

The cost savings associated with cloud services are attracting small and medium-sized businesses. Cloud service suppliers provide services to clients on a rental basis. Providing cloud services to users is extremely complex, as users can access virtual cloud resources easily. Computing resources are made available to customers via Virtual Machines (VMs) running on physical machines in the cloud. Virtual machines resemble physical computers in terms of capability [11]. As a guest program, a VM mimics the functionality of a physical machine. Optimizing server usage is achieved by dynamically allocating resources based on application requirements [12]. This approach operates dynamically to distribute non-preemptive workloads evenly. Load balancing is a challenging optimization issue in cloud computing that falls under Non-Polynomial-Hard (NP-

Hard) problems [13]. Therefore, researchers have devoted more attention to load balancing, which is found to improve system performance.

Load balancing facilitates the equitable distribution of workloads across available resources. The objective is to ensure uninterrupted operation in the event of a service component failure by managing the allocation and maintenance of application instances and optimizing resource usage [14]. Furthermore, load balancing aims to minimize task reaction time and enhance resource allocation, enhancing system performance while reducing expenses [15]. Moreover, load balancing aims to enhance the scalability and adaptability of applications that may expand in magnitude in the future. It also prioritizes tasks that demand immediate execution above other tasks [16]. To improve data center efficiency and reduce system reaction time, it is imperative to distribute the workloads evenly among physical hosts in the cloud environment, thereby enhancing throughput. Given several physical hosts in a data center, it is crucial to implement load balancing by migrating VMs across these hosts [17]. This is essential for ensuring the delivery of resilient and high-performing services. It is important to prioritize fault tolerance to provide dependable services for load balancing during the migration of VMs.

C. Motivation

Traditionally, load balancing has been accomplished through static and dynamic load balancing strategies [18]. The present status of the system has no effect on static load balancing strategies. Prior knowledge of the system is necessary. During the compilation phase, static load balancing techniques allocate tasks to processors before the commencement of program execution. The scheduling approach relies on preexisting knowledge about node characteristics and capabilities, including execution time, CPU resources, memory, and storage capacity, which are assumed to be known throughout the compilation process. These algorithms are suitable for stable situations with little load fluctuations but cannot adjust to load changes while in operation. In contrast, dynamic approaches consider the system's status and present situation, enabling them to handle varying load circumstances effectively. These solutions employ dynamic procedures to manage users' requests efficiently. While dynamic approaches provide superior performance than static approaches, formulating an algorithm for a dynamic cloud environment poses significant challenges [19].

The cloud platform efficiently manages task scheduling and allocates substantial virtual resources, no matter the duration needed [20]. Therefore, the cloud platform's effectiveness depends entirely on the selected technique for scheduling task resources. Furthermore, it is vital to have a streamlined and productive approach to assigning cloud resources to fulfill users' incoming tasks. Also, the user's tasks must be promptly, efficiently, and dependably processed [21]. Efficient load balancing and minimal resource usage are essential for executing user operations in the cloud computing environment. Nevertheless, the complex structure of the cloud environment and the stringent demands for managing the task scheduling process provide significant challenges in developing and carrying out optimization models [22].

Furthermore, the cloud task scheduling process is highly uncertain because of multiple factors triggering the unpredictable cloud environment, including network connectivity [23], resource usage [24], peak network demands [25], and web service performance inherent to service models of the cloud [26]. Artificial intelligence and machine learning techniques offer intelligent and adaptive solutions by analyzing patterns and predicting future demands, leading to proactive load management [27, 28]. Inspired by natural processes, meta-heuristic algorithms excel at solving complex, nonlinear optimization problems by offering robust and scalable solutions [29]. Their ability to explore and exploit the search space effectively helps achieve balanced load distribution across virtual machines. Integrating these technologies enhances the performance and resilience of cloud systems, enabling them to meet dynamic user demands while minimizing operational costs and resource waste [30].

D. Contribution

This study presents a new method for scheduling tasks in cloud computing using the Artificial Bee Colony (ABC) algorithm. The traditional ABC algorithm is well-suited for exploration but tends to ignore exploitation due to its intrinsic operating strategy. In the conventional ABC algorithm, the solutions produced in each generation are acquired by random search, resulting in the algorithm's inherent vulnerability to exploitation. Furthermore, the method fails to properly exploit the rich information in the best solution during execution. Moreover, the utilization of random neighborhood search results in a fast decline in population variety, rendering the algorithm susceptible to early convergence and trapped in local optimization. To solve these drawbacks, the ABC algorithm is enhanced by implementing two modifications.

The neighborhood search operator incorporates the location information of the global optimal solution to assist the bee colony in finding food. This approach effectively utilizes the information from the previous generation's optimal solution and improves the accuracy of the algorithm's exploitability search. Furthermore, to solve the issue of insufficient population variety, the algorithm integrates a genetic model that enhances population diversity by employing crossover and mutation operators. These two enhancements successfully reconcile the conflict between the ABC algorithm's research and application, strengthening the method's optimization accuracy and speed. The efficacy of the enhanced ABC method is validated by a set of commonly employed numerical functions. In summary, the study made the following contributions:

- Enhanced ABC algorithm for task scheduling: A novel approach is introduced for task scheduling in cloud computing, leveraging the ABC algorithm. Two crucial adjustments are suggested to overcome the shortcomings of the classic ABC algorithm in terms of exploitation.

- Improved exploitation through global neighborhood search: The study integrates a global neighborhood search operator to improve exploitation. By using the location information of the global optimal solution, this modification helps the bee colony to find food efficiently. This ensures better utilization of the information from the previous generation's optimal solution and improves exploitability search accuracy.

- A genetic model for population diversity: To tackle the issue of insufficient population variety and potential premature convergence, a genetic model is integrated into the algorithm. This model introduces crossover and mutation operators, enhancing population diversity. These additions successfully balance the conflict between the algorithm's research and practical application, leading to improved optimization accuracy and speed.

The rest of the paper is organized as follows. Section II provides a comprehensive review of the existing literature and identifies gaps and limitations of current approaches. Section III precisely formulates the task scheduling problem. The proposed task scheduling technique is discussed in Section IV. Section V examines the theoretical underpinnings and practical implications of the proposed algorithm for load balancing. Section VI presents the empirical evaluation of the proposed method and compares its performance with existing methods. Section VII concludes the study by summarizing the main contributions, highlighting results, and suggesting possible avenues for future research in Section VIII.

II. Related Work

This section offers an overview of the current body of literature about task scheduling in cloud computing. Our objective is to uncover fundamental insights, techniques, and advancements by analyzing various algorithms developed to tackle particular challenges in cloud settings.

Wei [31] suggested an approach for optimizing task scheduling in cloud infrastructure using a modified version of the Ant Colony Optimization (ACO) algorithm. The scheduling model utilizes an improved ACO algorithm to prevent the optimization strategy from being stuck in local optimization under cloud computing task scheduling principles. The task scheduling satisfaction function is created by integrating the three objectives of minimizing waiting time, optimizing resource load distribution, and lowering task completion cost to identify the most efficient task scheduling solution. Ultimately, introducing the reward and punishment coefficient enhances the optimization of the pheromone update rules in the ACO algorithm, resulting in an accelerated solution speed. Furthermore, the volatility coefficient is dynamically updated to improve the overall performance of this method. According to the test findings, the suggested algorithm outperforms previous approaches regarding convergence speed, completion time, load balancing, and consumption of virtual machine resources.

Abualigah and Diabat [32] presented a novel hybrid Antlion optimization algorithm that combines elite-based differential evolution to address multi-objective task scheduling challenges in cloud computing environments. The challenge is multi-objective since it requires minimizing the makespan and maximizing resource consumption simultaneously. The Antlion optimization algorithm is enhanced by including elite-based differential expansion as a local search approach to increase its capacity to explore the search space and prevent being stuck in suboptimal solutions. Two tests were conducted using the CloudSim toolbox, one on synthetic datasets and the other on actual trace datasets. The findings indicated that the suggested algorithm exhibited a more rapid convergence rate than alternative methods, rendering it well-suited for extensive planning scenarios.

Ben Alla, et al. [33] proposed a novel approach to prioritizing customer demands and supplier resources. They introduced a highly effective method for scheduling tasks called MCPTS, which involves adjusting the priority depending on four task factors: duration, delay, deadline, and burst time. The MCPTS structure has three components: task priority, task queue priority, and resource priority. A new strategy is suggested to assess and establish task priorities, utilizing an integrated Multi-criteria Decision-Making (MCDM) technique known as ELECTRE III and a metaheuristic algorithm named Differential Evolution. Furthermore, a unique dynamic priority queuing algorithm derived from the queuing model is presented. Moreover, the allocation of resources is dynamically modified according to the task priority model to establish an effective and adaptable connection between resource and task categories. The experimental findings demonstrate the superiority of the MCPTS algorithm in comparison to other current algorithms. Furthermore, it shows the efficacy of the suggested approach in delivering commendable system performance, fulfilling user demands and QoS prerequisites, enhancing load distribution, and optimizing resource usage.

Malti, et al. [34] offer a highly effective task scheduling method that leverages flower pollination behavior. This algorithm incorporates the Pareto optimality principle and the TOPSIS approach to enhance the quality of the solutions achieved. Both single and multi-objective optimization variations are analyzed. In the second scenario, three optimization criteria are considered: decreasing the duration or schedule length, reducing the execution cost, and optimizing the overall dependability of task distribution. The study examined several test bench situations and Quality of Service (QoS) measures. The acquired findings validate the advantages of the suggested method.

Mangalampalli, et al. [35] proposed a Multi-Objective Task Scheduling Gray Wolf Optimization (MOTSGWO). This algorithm is capable of making scheduling decisions in real-time by considering the current state of cloud resources and future workload demands. Moreover, the suggested method distributes resources according to the end users' financial constraints and the tasks' importance. The MOTSGWO technique is applied using the Cloudsim toolbox, and the workload is created by building datasets with various task densities and workload patterns. The comprehensive studies demonstrate that the suggested MOTSGWO strategy surpasses previous baseline strategies and enhances the crucial metrics.

Saravanan, et al. [36] proposed the enhanced Wild Horse Optimization (IWHO) algorithm to tackle the issues of lengthy scheduling time, excessive cost consumption, and high use of virtual machines. Initially, a model for scheduling and distributing cloud computing tasks is constructed, considering the primary aspects of time, cost, and virtual machines. To enhance the local search capability and minimize premature convergence, the IWHO algorithm employs the inertia weight technique to effectively identify the ideal individual. The IWHO method is augmented with the Levy-Flight algorithm to optimize task scheduling in cloud computing. The effectiveness of the proposed hybrid algorithm is verified, and the outcomes are assessed utilizing several methodologies. The simulation results demonstrated that the suggested approach surpassed others in various scenarios.

Behera and Sobhanayak [37] suggested a hybrid approach that integrates the GWO algorithm with the Genetic Algorithm (GA). GWO-GA optimizes multi-objective task scheduling in cloud computing by minimizing processing time, energy usage, and cost. The enhancements to GWO-GA involve incorporating the crossover and mutation operator from the genetic algorithm. Moreover, the accelerated convergence of the GA-based GWO method is a benefit when dealing with extensive planning problems. The suggested algorithm performs better than existing GWO, GA, and PSO algorithms in terms of makespan, cost, and energy usage. It achieves reductions of 19%, 21%, and 15% correspondingly, compared to each of these approaches. In addition, it leads to energy conservation rates of 17%, 19%, and 23% when compared to GWO, GA, and PSO, respectively. Simultaneously, it reduces the total design expenditure by 13%, 17%, and 22%, respectively. The findings illustrate the efficacy of the suggested approach in addressing the task scheduling issue in cloud computing settings.

Pabitha, et al. [38] proposed the Chameleon and Remora Search Optimization (CRSO) algorithm to enhance the scheduling procedure by investigating the influence of MIPS and network bandwidth on virtual machine performance. Furthermore, the study considers the uncertainty aspects of task completion rates, distribution of loads, cost, and makespan concurrently during the scheduling process. The formulation of an optimization model with multiple objectives for cloud task scheduling involves the integration of the advantages of Chameleon Search Algorithm (CSA) and Remora Search Optimization Algorithm (RSOA) utilizing a greedy technique to mimic the actual process of cloud computing task scheduling. Simulation findings confirm that the proposed CRSOA technique substantially decreases the time required for task completion and efficiently manages the workload distribution across the available virtual machines, surpassing other competing algorithms.

III. Problem Definition

The assignment of incoming jobs to free VMs located in cloud data centers is a major problem. This scenario involves a collection of uniform and diverse tools where individual servers run many virtual machines. Virtualization enables users to utilize virtual environments' flexible computing resources. The data center broker controls the scheduling system, supervising the allocation and monitoring of user tasks. Fig. 1 depicts the schematic layout of the scheduling procedure. Initially, cloud users enter tasks kept in the Task Manager module. This module monitors the arrival of tasks and provides relevant information to the corresponding individuals. These task submissions are transmitted to the cloud scheduling system by the task manager. A number of jobs are allocated to VMs according to the MABC algorithm. The cloud information repository is used to gather details about VMs and tasks.

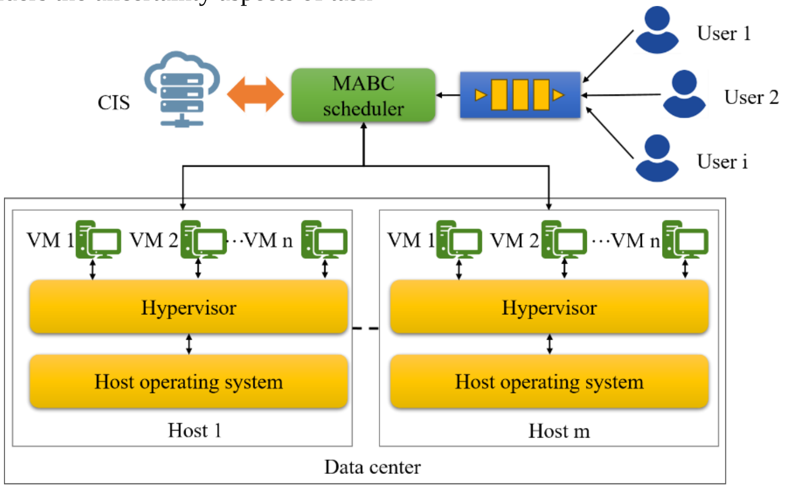

Fig. 1. Scheduling process.

Several data centers are involved in the public cloud system paradigm in order to meet resource demands. Consider a collection of data centers $(dc_1, dc_2, \dots, dc_p)$. A data center comprises several Physical Hosts (PHs). For instance, the data center dc_r includes k PHs labeled $(PH_{r1}, PH_{r2}, \dots, PH_{rk})$. Each PH possesses distinct characteristics, including the computational capability of a processor determined by the number of cores measured in Millions of Instructions Per Second (MIPS). Each PH is equipped with bandwidth, memory, storage capacity, and a VM Manager (VMM). The VMM installed on PHs has a vital function in keeping track of all VMs located on that specific physical host. It guarantees the effective distribution and usage of resources for the virtual machines operating on the host. Each PH inside a data center can handle a specific number of VMs, represented as $(VM_1, VM_2, \dots, VM_m)$. Each VM possesses its own distinct settings, outlined as follows:

- Number of cores: This parameter defines the VM's capacity for handling several tasks simultaneously while simultaneously processing.

- Processing power: This is expressed in MIPS, which denotes the computational capacity for processing commands and performing tasks.
- Storage: The allocated capability for storing data and files particular to virtual machines.
- Main memory: Designated for the storage of data and the execution of applications within the virtual machine.

Furthermore, each VM is associated with an hourly rate representing the expense incurred for utilizing that specific VM for every utilization. Virtual machine configurations and the time spent using them determine how much users are billed for resources. Within the realm of cloud computing, individuals or organizations utilizing cloud services may effortlessly forward their individual requests to the service supplier for processing with no extensive knowledge of its underpinnings. These assignments include distinct criteria regarding the duration of the job and the assets needed. Users enter a set of *n* jobs labeled $(t_1, t_2, \dots, t_n)$. Tasks are given a distinct duration (l_i), quantified in Millions of Instructions (MI). The process starts with determining the duration of the i^{th} job on the j^{th} virtual machine, outlined in Eq. (1).

$$ET(l_i, vm_j) = \frac{l_i}{total_{MIPS}(vm_j)} \quad (1)$$

The scheduler evaluates the VM's energy usage during task execution and the cost of processing tasks within the VM. The primary goal is to lower the total costs associated with task completion by finding the VM with the most favorable costs that meets the individual criteria for the task. Due to the varying processing capabilities of multiple VMs, the execution time and cost of performing a specific function on distinct VMs are inconsistent. As a result, a problem with multiple objectives develops, which seeks to reduce the time it takes to complete a task, the amount of energy used, and the cost incurred while maximizing resource efficiency. Thus, this study aims to tackle the complex challenge of solving the multi-objective scheduling issue using the suggested MABC algorithm.

Cloud computing services are delivered on a two-actor model, encompassing cloud service suppliers and consumers. Service suppliers provide clients with resources to accomplish jobs. Cloud users place a high premium on application efficiency, preferring rapid and efficient computation capabilities. In contrast, cloud service suppliers focus on optimizing resource utilization to achieve optimal financial returns. The purposes may be categorized into client demands, which include the cost of execution and the duration of the schedule, and supplier demands, which encompass energy consumption and resource utilization. Within the cloud computing domain, execution cost signifies the total expenditure accrued during the operation of a given application. It serves as a quantifiable indicator to evaluate financial outlays. However, it is important to specify the costs associated with the assets used. Clients seek to decrease both expenses and schedule duration. The computation of expenses associated with the execution can be summarized as follows:

$$EC = \sum_{j=1}^{m} ET_j \times price_j \quad (2)$$

ET_j denotes the length of time that the j^{th} VM is allocated for executing a task, following the completion of the final task. The schedule length is crucial for assessing the quality of scheduling and is determined by the longest time it takes for any task to be completed, either among all submitted tasks or by the time the final processing virtual machine is complete. The scheduler's efficiency can be accurately assessed using this essential measure. A shorter schedule length indicates an improved scheduling procedure in which tasks are distributed optimally to appropriate resources. In contrast, a longer schedule length signifies a less efficient scheduling method. Eq. (3) can determine the value of the schedule length.

$$SL = \max(\sum_{i=1}^{n} ET(l_i, vm_j) x_{ij}) \forall vm_j \quad (3)$$

Eq. (3) represents the allocation decision variable, x_{ij}, which indicates whether task *i* is assigned to the j^{th} VM. The variable is binary, with a value of 1 indicating that task *i* is assigned to VM *j*, and 0 stating otherwise. Maximizing resource efficiency is crucial for cloud service providers. Their main goal is to maximize the utilization of resources in order to enhance profitability. Providers attempt to optimize their usage given the constraints of limited resources. Eq. (4) clearly describes how to calculate the average resource utilization.

$$average\ RU = \frac{\sum_{i=1}^{m} ET_i}{O_1} \quad (4)$$

The variable O_1 in Eq. (4) denotes the minimum schedule length, which serves as a measure of the desired service quality. In this situation, efficient use of resources implies the optimal exploitation of available VMs to handle tasks. Data center energy usage involves several components, including CPU, network interfaces, and storage devices. Out of all these resources, the CPU is generally the most power-intensive. When evaluating the energy usage of a VM, it is divided into two categories: energy consumption during times of idle and energy consumption during active phases. The overall energy usage takes into account both the idle and active modes of the VM. The energy spent during periods of idle is approximately 60% of the energy consumed by a fully functional VM. The remaining 40% represents the energy consumed by the VM during active calculations. This energy expenditure is dependent on the processing speed of the VM, calculated by Eq. (5).

$$EC = 10^{-8} \times (vm_{i_{mips}})^2 \frac{J}{MI} and\ IE_i = 0.6 \times EC_i \quad (5)$$

EC_i denotes the energy consumed when the VM is in an active state IE_i represents the energy consumed when the VM is idle. The model's energy usage can be defined according to Eq. (6).

$$TE_i = IE_i + EC_i \quad (6)$$

IV. Modified ABC Algorithm for Task Scheduling

The ABC algorithm applies a kind of bionic intelligence inspired by honey bee foraging behavior. In this algorithm, food source position indicates a potential optimization solution, while nectar quality indicates the quantity of nectar [39]. An optimization problem is addressed by three types of bees: worker, onlooker, and scout. Employed bees are equal to food sources. So, each worker bee has access to a food source around the hive. An onlooker bee continuously watches and selects food resources under the activities of employed bees. A scout bee uncovers new sources of food not found by employed bees by searching randomly. Fig. 2 illustrates how the ABC algorithm finds an optimal solution for the optimization problem. Initially, nectar sources are produced in a random manner using Eq. (7).

$$x_{i,j} = x_j^{min} + rand(0,1).(x_j^{max} - x_j^{min}) \quad (7)$$

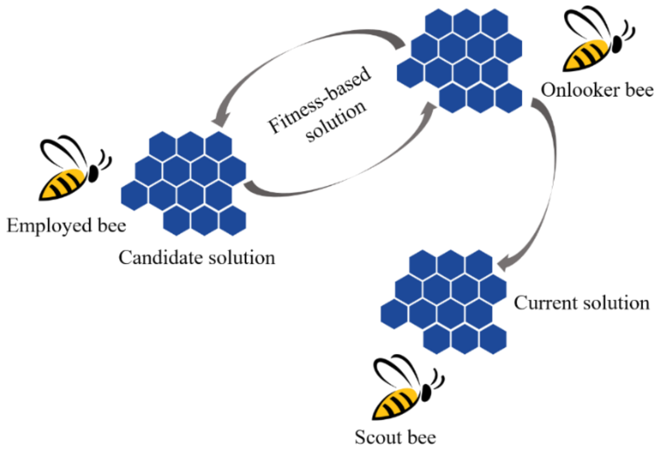

Fig. 2. Workflow of ABC algorithm.

where, $x_{i,j} \in [x_j^{max}, x_j^{min}]$ denotes the j^{th} dimension boundary of the optimization problem, and $rand(0, 1)$ is a random number ranging from 0 to 1. The ABC optimization process consists of three separate stages: the employed bee stage, the onlooker bee stage, and the scout bee stage.

The employed bees have the ability to scan the whole optimization problem space for new sources of nectar. Eq. (8) updates the position of the nectar source simultaneously.

$$v_{i,j} = x_{i,j} + \phi_{i,j}.(x_{i,j} - x_{k,j}) \quad (8)$$

where, x_i represents the initial nectar source, v_i reflects a recently discovered nectar source, $\phi_{i,j} \epsilon [-1,1]$ is a random number chosen uniformly, x_k is a nectar source taken randomly from the population, and x_k is not equal to x_i. It should be noted that j is a dimension that is chosen without any specific criteria, and x_i and v_i vary simply in this particular dimension. If the amount of nectar in v_i is more than that in x_i, then v_i will replace x_i in the subsequent round. Alternatively, x_i stays unaltered.

ffigOnlooker bees will select optimal nectar sources for exploitation depending on the quantity of nectar available, using information provided by employed bees. Additionally, it explores new nectar sources by employing the solution search equation given in Eq. (8). The fitness value of an individual is determined by the quantity of nectar from the nectar supply. This is estimated using the Eq. (9).

$$fitness_i \begin{cases} \frac{1}{1+f(X_i)} & f(X_i) \geq 0 \\ 1 + |f(X_i)| & f(X_i) < 0 \end{cases} \quad (9)$$

Where $fitness_i$ refers to the fitness value of the nectar source, and $(\cdot)$ indicates the objective function value. A higher quantity of nectar in a nectar source increases the likelihood of onlooker bees choosing that source. The probability of selection is computed using the Eq. (10).

$$p_i = \frac{f(X_i)}{\sum_{j=1}^{SN} f(X_i)} \quad (10)$$

After determining the probability of selecting each nectar source, the onlooker bees will employ the roulette method. If the nectar supply linked to the employed bees is not refreshed over a specified threshold $limit$, we assume that the nectar source has been exhausted. In this scenario, a novel nectar source is introduced at random using Eq. (7) to substitute Xi.

The conventional ABC algorithm fails to fully exploit the location details of the optimal solution in each iteration, a valuable piece of information. This study introduces a global neighborhood search operator (as given in Eq. (11)). With this operator, the bee colony locates food sources using the positional data from the global optimal solution, X_{Gbestj}. By using this approach, the honey source $X_{Gbest,j}$ can be fully exploited and utilized. Furthermore, exploring the vicinity of the optimal solution accelerates the convergence of the algorithm. The variable β is given a random number from 0 to 1.

$$New_{X_{i,j}} = X_{i,j} + \phi_{i,j}(X_{Gbest,j} - X_{k,j}) + \beta(X_{Gbest,j} - X_{i,j}) \quad (11)$$

While the current generation's optimal solution information is incorporated into the neighborhood search operator, the random neighborhood search approach remains unchanged. Eq. (11) facilitates algorithm convergence and enhances exploitation capabilities. Nevertheless, it may lead to local optimization by steering the colony towards local extremes. In order to address this constraint and enhance the algorithm's ability to consistently generate new viable solutions, a genetic model is utilized. Each iteration retains half of the ideal solution. Afterward, the preserved solutions undergo recombination according to Eq. (12). Ultimately, the new solution passes the mutation and crossover procedures, as shown in Fig. 3. By employing this evolutionary process, the search range expands, and the variety of viable solutions is enhanced to avoid local optimization.

$$X_{i,j} = X_{k1,j} + \gamma(X_{k2,j} - X_{k3,j}) \quad (12)$$

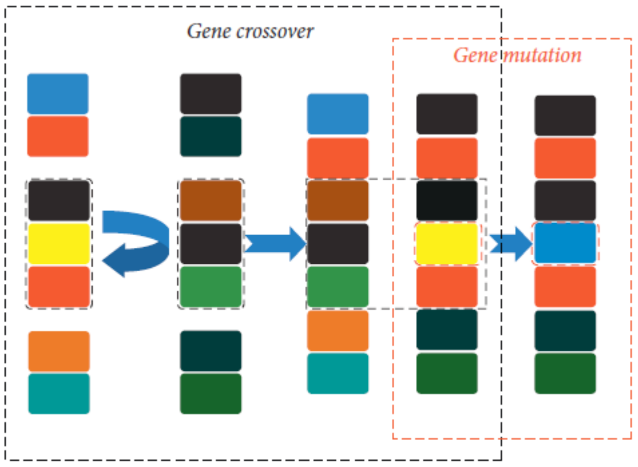

Fig. 3. An illustration of mutations and crossovers.

Three solutions, X_{k1}, X_{k2}, and X_{k3}, are chosen from the preserved feasible solutions through random probability selection. γ is a randomly generated number within the range [0, 1], and j denotes the position of an array element in the plane array.

V. Discussion

This section examines the theoretical underpinnings and practical implications of the MABC algorithm for load balancing in cloud computing environments. The MABC algorithm enhances the ABC algorithm by solving its main limitations, specifically the lack of exploitation capability and rapid loss of population diversity. The traditional ABC algorithm is augmented by including a mechanism for retaining optimal solution information within the neighborhood search operator. By modifying the algorithm, best practices are preserved and utilized effectively in subsequent iterations. Furthermore, the incorporation of a genetic evolution mechanism fosters a balance between exploratory and exploitative behaviors within the scheduling process.

Among the primary advantages of the MABC algorithm is its ability to balance load across VMs in a cloud computing environment. MABC algorithm adapts to dynamic workloads and varying user demands by maintaining a diverse population and effectively utilizing optimal solution information. The result is more efficient resource utilization, reduced energy consumption, and a shorter time to complete tasks. The modifications to the ABC algorithm result in faster convergence rates, making the MABC algorithm well-suited to real-time load balancing applications. While the MABC algorithm demonstrates significant improvements over traditional methods, potential limitations must be acknowledged. The higher complexity due to the retention of optimal solutions and the genetic evolution mechanism may lead to additional computational overhead. This could impact the algorithm's performance under resource-constrained environments. Furthermore, the effectiveness of the MABC algorithm depends on the proper tuning of its parameters. Inappropriate parameter settings may produce suboptimal performance or excessive computational costs.

VI. Simulation Results

In this section, the suggested algorithm is benchmarked against recent swarm-based algorithms (GA, Harris Hawks Optimizer (HHO), ACO, and traditional ABC). Simulations were conducted with the CloudSim 3.0.3 simulator on a Windows 10 laptop powered by 16 GB of RAM. Table I outlines the specifications of the virtual cloud computing environment. Table II provides a summary of the VM parameters involved in the experiment. The synthetic workload is created using an even distribution, guaranteeing an equal spread of tasks in different dimensions. The assessment examines the High-Performance Computing Centre (HPC2N) workload, which is commonly acknowledged as a benchmark for evaluating the performance of distributed systems.

Results of resource utilization for the different methods, including MABC, ABC, GA, HHO, and ACO, using the HPC2N real-word dataset are illustrated in Fig. 4 and 5. Fig. 4 compares different load balancing algorithms applied to 40 VMs. The MABC algorithm demonstrates superior resource utilization across all task quantities. This is because MABC considers resource usage when scheduling tasks, ensuring tasks are allocated to the most appropriate VMs. VMs are thus utilized more efficiently, resulting in enhanced overall performance and reduced idle times compared to ABC, GA, HHO, and ACO. Fig. 5 shows the resource utilization of different algorithms when 80 VMs are used. Similar to the results for 40 VMs, the MABC algorithm consistently outperforms the others. This performance is due to MABC’s advanced scheduling mechanism, which dynamically adjusts to the available resources, thereby maximizing VM efficiency and minimizing resource wastage.

TABLE I. Datacenter and Host Configurations

Cloud component	Feature	Value
Host	Storage	2 TB
	RAM	F GB
	Bandwidth	5 GB
Datacenter	User count	1
	Host count	2
	Datacenter count	1 2

TABLE II. VMs Configurations

Characteristic	Value
VM count	20-100
MIPS	500-1000
Bandwidth	0.5 Gb/S
VMM	Xen
Size	100 MB

Fig. 6 compares the energy consumption of various load balancing algorithms under HPC2N workloads with 40 VMs. The energy conservation performance of each algorithm is evaluated as task counts increase. Traditional algorithms such as ABC, ACO, GA, and HHO exhibit linear rises in energy consumption with increasing task counts, but the MABC algorithm shows a more gradual rise. Under varying workloads, MABC conserves energy efficiently, which makes it a suitable solution for energy-efficient cloud task scheduling. Fig. 7 compares the energy consumption of different load balancing algorithms when applied to 80 VMs. Similar to Fig. 6, the energy consumption patterns of various algorithms are analyzed under a variety of task counts. According to the results, MABC is more effective than traditional algorithms at conserving energy and optimizing resource allocation compared to traditional algorithms. Fig. 8 illustrates the energy consumption across different load balancing algorithms when applied to a synthetic workload with 120 VMs.

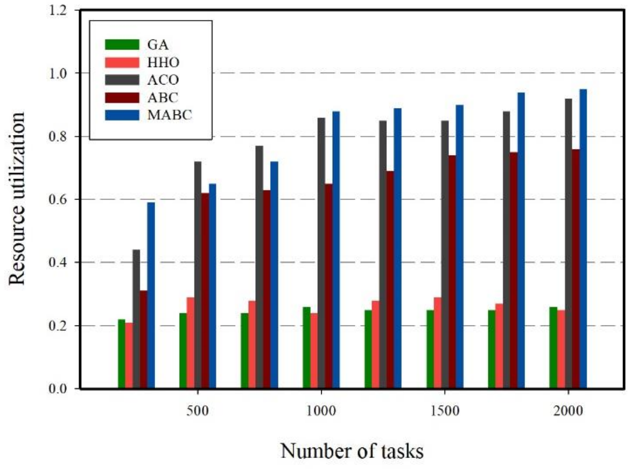

Fig. 4. Resource utilization for HPC2N tasks involving 40 virtual machines.

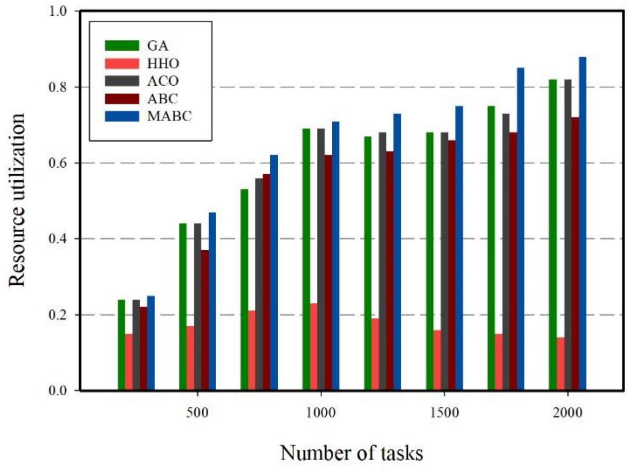

Fig. 5. Resource utilization for HPC2N task involving 80 virtual machines.

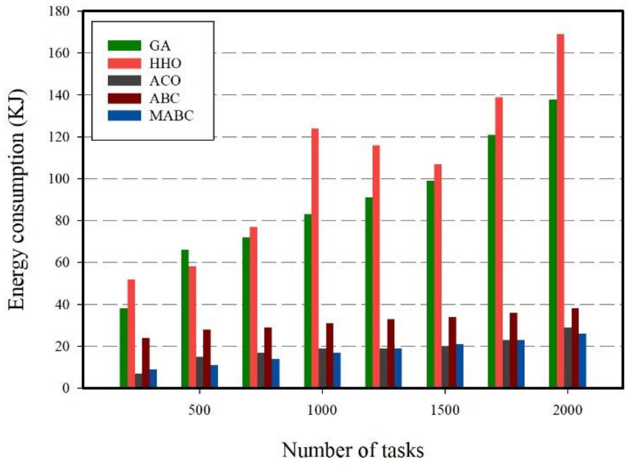

Fig. 6. Energy consumption for HPC2N tasks involving 40 virtual machines.

Fig. 9 to 11 offer a comparative evaluation of various algorithms concerning execution costs. This analysis encompasses both synthetic and HPC2N workloads, highlighting the potential impact of task duration and VM selection on execution costs. The results indicate that MABC outperforms the traditional ABC algorithm across varying task numbers for synthetic and HPC2N workloads. As shown in Fig. 9 to Fig. 11, MABC demonstrates its cost-effectiveness by consistently reducing execution costs across a variety of scenarios. When applied to real-world tasks ranging from 250 to 2000 units, MABC exhibits an average cost reduction of 11% to 43% compared to the ABC algorithm. This advantage extends to synthetic workloads as well, with MABC achieving average cost reductions of 9% to 60% for tasks between 500 and 2000 units. These findings highlight MABC's ability to optimize resource utilization and minimize execution costs across diverse task types and workloads.

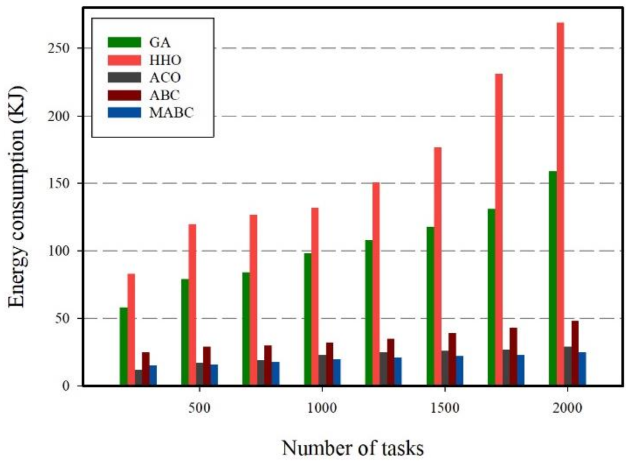

Fig. 7. Energy consumption for HPC2N tasks involving 80 virtual machines.

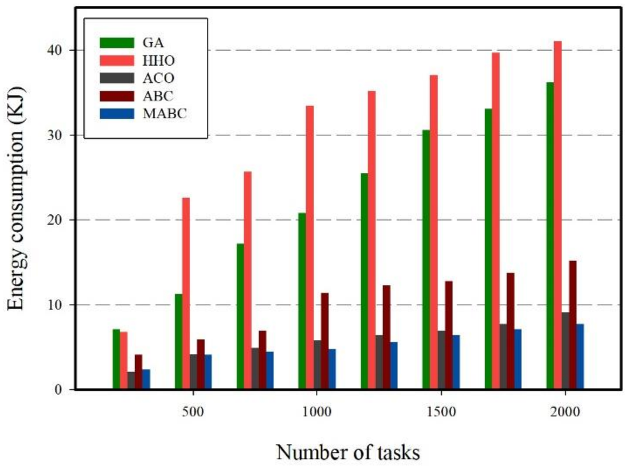

Fig. 8. Energy consumption for synthetic tasks involving 120 virtual machines.

Table III shows numerical functions used to prove the efficiency of the MABC algorithm. An array of benchmark functions is provided, with tests f1-f4 covering unimodal and tests f5-f7 covering multimodal continuous functions. The range of values for the parameters and the lowest possible numerical function value are listed in Table IV. The genetic, GABC, ABC, and MABC algorithms are employed for optimizing seven numerical functions. The algorithm was executed autonomously 20 times. The algorithm's benefits and drawbacks were assessed by utilizing statistical measures such as the average and standard deviation. A performance analysis of algorithms with 30 dimensions and 3000 iterations is presented in Table IV. This table reveals that the MABC algorithm outperforms other algorithms in terms of both the average and standard deviation of its results.

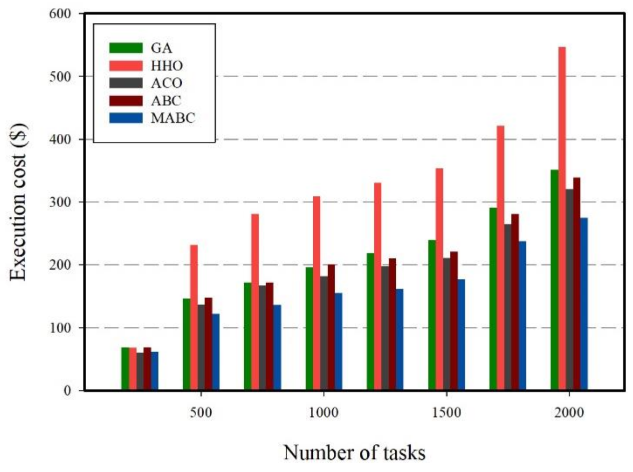

Fig. 9. Execution cost for HPC2N tasks involving 40 virtual machines.

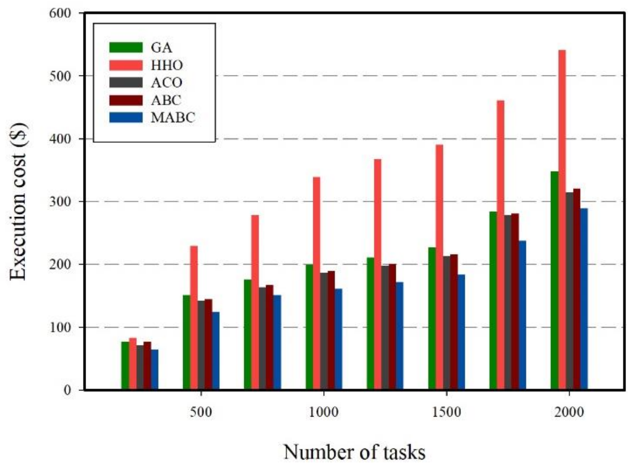

Fig. 10. Execution cost for HPC2N tasks involving 80 virtual machines.

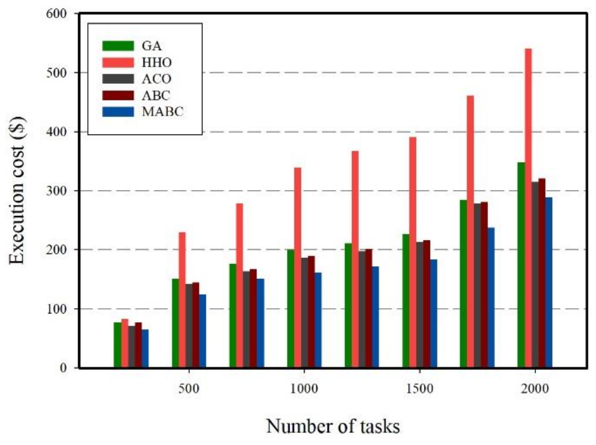

Fig. 11. Execution cost for synthetic tasks involving 120 virtual machines.

TABLE III. NUMERICAL FUNCTIONS

Function	Expression	Range	Minimum value
Exponential	$f_1(x) = \exp(0.5 \times \sum_{i=1}^{D} x_i)$	$[-10,10]^D$	0
SumSquare	$f_2(x) = \sum_{i=1}^{D} ix_i^2$	$[-10,10]^D$	0
Elliptic	$f_3(x) = \sum_{i=1}^{D} (10^6)^{i-1/D-1} x_i^2$	$[-100,100]^D$	0
Sphere	$f_4(x) = \sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	0
Himmelblau	$f_5(x) = 1/D \sum_{i=1}^{D} [x_i^4 - 16x_i^2 + 5x_i]$	$[-5,5]^D$	-78.33
Rastrigin	$f_6(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.11,5.11]^D$	0
Rosenbrock	$f_7(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$	$[-5,10]^D$	0

TABLE IV. ALGORITHMS COMPARISON

Function		MABC	GA	GABC	ABC
f_1	Mean	0	0	7.18e-23	7.18e-21
	Std	0	0	7.07e-23	7.21e-21
f_2	Mean	3.57e-20	8.11e-11	5.253-15	7.33e-15
	Std	6.92e-20	7.81e-11	6.18e-15	8.19e-15
f_3	Mean	4.98e-20	4.47e-12	4.19e-16	4.53e-8
	Std	1.21e-20	5.77e-12	4.25e-16	4.83e-8
f_4	Mean	3.73e-23	1.23e-13	5.12e-16	2.42e-15
	Std	4.16e-23	1.63e-13	4.35e-17	3.2e-15
f_5	Mean	-78.332	-78.332	-78.332	-78.332
	Std	0	1.097e-14	3.13e-15	0
f_6	Mean	0	0	0	1.35e-13
	Std	0	0	0	198e-13
f_7	Mean	1.92e-07	4.15e-05	9.71e-02	4.75e-01
	Std	2.11e-07	5.01e-05	1.01e-01	5.81e-01

VII. CONCLUSION

The process of task scheduling within cloud computing paradigms presents a multi-objective optimization challenge. The dynamic context and varying tasks also pose a challenge to finding an equilibrium between QoS requirements, energy consumption, and resource utilization. This paper proposed MABC algorithm for task scheduling. The proposed modification to the ABC algorithm leverages the intelligent foraging behavior of bee colonies to enhance its competence in solving complex nonlinear optimization problems. The traditional ABC algorithm, while effective, faces limitations in resource utilization, leading to a rapid decline in population diversity and inadequate dissemination of optimal solution knowledge across generations. The introduced modifications to the ABC algorithm effectively addressed these limitations. By retaining optimal solution information within the neighborhood search function and incorporating a genetic evolution process, the MABC algorithm achieved a more balanced exploration-exploitation trade-off, enriching population diversity. Comparative analysis of the MABC algorithm versus established scheduling techniques demonstrated its efficacy in producing a trifecta of desirable outcomes: lower execution costs, diminished energy consumption, and improved resource utilization.

VIII. Future Work

Future research will prioritize task scheduling difficulties that closely resemble real-world cloud computing settings. This also involves taking into account the priority constraint connections among tasks. Moreover, when considering the situation objectively, cost emerges as a significant determinant impacting work scheduling in real-life situations. Users seeking to optimize task completion time must allocate more money toward getting cloud computing services. Hence, we aim to devise a task scheduling algorithm that achieves a harmonious equilibrium among three pivotal factors: job completion time, cost, and load distribution. By developing innovative approaches that prioritize both efficiency and cost-effectiveness, we aim to improve cloud computing systems' efficiency and flexibility in real-world applications. Additionally, we envisage investigating the integration of emerging technologies, such as machine learning and edge computing, to further optimize task scheduling processes and adapt to evolving user demands and system dynamics. Through these future research endeavors, we aim to make a substantial contribution to the ongoing evolution of cloud computing technologies. This pursuit seeks to address the dynamic challenges confronting both cloud service providers and their consumers.

Acknowledgment

This work was supported by project of Jilin Provincial Department of Education Scientific Research Technology. (No. JJKH20231459KJ).

References

[1] H. N. Alshareef, "Current Development, Challenges, and Future Trends in Cloud Computing: A Survey," International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, 2023.

[2] B. Guha, S. Moore, and J. M. Huyghe, "Conceptualizing data-driven closed loop production systems for lean manufacturing of complex biomedical devices—a cyber-physical system approach," Journal of Engineering and Applied Science, vol. 70, no. 1, p. 50, 2023.

[3] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[4] M. Saleem, M. Warsi, and S. Islam, "Secure information processing for multimedia forensics using zero-trust security model for large scale data analytics in SaaS cloud computing environment," Journal of Information Security and Applications, vol. 72, p. 103389, 2023.

[5] U. Gupta and R. Sharma, "Comparison of Different Cloud Computing Platforms for Data Analytics," in Doctoral Symposium on Computational Intelligence, 2023: Springer, pp. 67-78.

[6] A. K. Samha, "Strategies for efficient resource management in federated cloud environments supporting Infrastructure as a Service (IaaS)," Journal of Engineering Research, 2023.

[7] L. Pons et al., "Cloud white: Detecting and estimating qos degradation of latency-critical workloads in the public cloud," Future Generation Computer Systems, vol. 138, pp. 13-25, 2023.

[8] S. Ahmadi, "Security And Privacy Challenges in Cloud-Based Data Warehousing: A Comprehensive Review," International Journal of Computer Science Trends and Technology (IJCST)–Volume, vol. 11, 2023.

[9] U. M. Ismail and S. Islam, "A unified framework for cloud security transparency and audit," Journal of Information Security and Applications, vol. 54, p. 102594, 2020.

[10] V. Hayyolalam, B. Pourghebleh, A. A. P. Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[11] G. Tricomi, G. Merlino, A. Panarello, and A. Puliafito, "Optimal selection techniques for Cloud service providers," IEEE Access, vol. 8, pp. 203591-203618, 2020.

[12] Y. Sun, J. Li, X. Fu, H. Wang, and H. Li, "Application research based on improved genetic algorithm in cloud task scheduling," Journal of Intelligent & Fuzzy Systems, vol. 38, no. 1, pp. 239-246, 2020.

[13] K. J. Naik, "An Adaptive Push-Pull for Disseminating Dynamic Workload and Virtual Machine Live Migration in Cloud Computing," International Journal of Grid and High Performance Computing (IJGHPC), vol. 14, no. 1, pp. 1-25, 2022.

[14] D. A. Shafiq, N. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 7, pp. 3910-3933, 2022.

[15] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," Journal of King Saud University-Computer and Information Sciences, vol. 32, no. 2, pp. 149-158, 2020.

[16] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, pp. 1-21, 2019.

[17] F. Hasan, M. Imran, M. Shahid, F. Ahmad, and M. Sajid, "Load balancing strategy for workflow tasks using stochastic fractal search (SFS) in Cloud Computing," Procedia Computer Science, vol. 215, pp. 815-823, 2022.

[18] V. Mohammadian, N. Jafari Navimipour, M. Hosseinzadeh, and A. Darwesh, "Comprehensive and systematic study on the fault tolerance architectures in the cloud computing," Journal of Circuits, Systems and Computers, 2020.

[19] M. Hamdan et al., "A comprehensive survey of load balancing techniques in software-defined network," Journal of Network and Computer Applications, vol. 174, p. 102856, 2021.

[20] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," Software: Practice and Experience, vol. 44, no. 2, pp. 163-174, 2014.

[21] Y. Yu and Y. Su, "Cloud task scheduling algorithm based on three queues and dynamic priority," in 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), 2019: IEEE, pp. 278-282.

[22] S. Mangalampalli et al., "Fault-Tolerant Trust-Based Task Scheduling Algorithm Using Harris Hawks Optimization in Cloud Computing," Sensors, vol. 23, no. 18, p. 8009, 2023.

[23] W. Anupong et al., "Deep learning algorithms were used to generate photovoltaic renewable energy in saline water analysis via an oxidation process," Water Reuse, vol. 13, no. 1, pp. 68-81, 2023.

[24] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Water Reuse, vol. 13, no. 1, pp. 51-67, 2023.

[25] S. Chekuri et al., "Integrated digital library system for long documents and their elements," in 2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2023: IEEE, pp. 13-24.

[26] Y. Kumar, S. Kaul, and Y.-C. Hu, "Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey," Sustainable Computing: Informatics and Systems, vol. 36, p. 100780, 2022.

[27] M. Hajihosseinlou, A. Maghsoudi, and R. Ghezelbash, "A comprehensive evaluation of OPTICS, GMM and K-means clustering methodologies for geochemical anomaly detection connected with sample catchment basins," Geochemistry, p. 126094, 2024.

[28] K. Xu, J. Lyu, and S. Manoochehri, "In situ process monitoring using acoustic emission and laser scanning techniques based on machine learning models," Journal of Manufacturing Processes, vol. 84, pp. 357-374, 2022.

[29] P. Gholami and H. Seferoglu, "DIGEST: Fast and Communication Efficient Decentralized Learning with Local Updates," IEEE Transactions on Machine Learning in Communications and Networking, 2024.

[30] S. R. Abdul Samad et al., "Analysis of the performance impact of fine-tuned machine learning model for phishing URL detection," Electronics, vol. 12, no. 7, p. 1642, 2023.

[31] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," Journal of Ambient Intelligence and Humanized Computing, pp. 1-12, 2020.

[32] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," Cluster Computing, vol. 24, pp. 205-223, 2021.

[33] H. Ben Alla, S. Ben Alla, A. Ezzati, and A. Touhafi, "A novel multiclass priority algorithm for task scheduling in cloud computing," The Journal of Supercomputing, vol. 77, no. 10, pp. 11514-11555, 2021.

[34] A. N. Malti, M. Hakem, and B. Benmammar, "Multi-objective task scheduling in cloud computing," Concurrency and Computation: Practice and Experience, vol. 34, no. 25, p. e7252, 2022.

[35] S. Mangalampalli, G. R. Karri, and M. Kumar, "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization," Cluster Computing, vol. 26, no. 6, pp. 3803-3822, 2023.

[36] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, p. 24, 2023.

[37] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," Journal of Parallel and Distributed Computing, vol. 183, p. 104766, 2024.

[38] P. Pabitha, K. Nivitha, C. Gunavathi, and B. Panjavarnam, "A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing," Sustainable Computing: Informatics and Systems, vol. 41, p. 100944, 2024.

[39] N. Rahnema and F. S. Gharehchopogh, "An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering," Multimedia Tools and Applications, vol. 79, no. 43-44, pp. 32169-32194, 2020.