Labyrinth: Fayoum Journal of Science and Interdisciplinary Studies 3 (2025) 1; 71-80

Fayoum University

Contents lists available at Egyptian Knowledge Bank

Labyrinth: Fayoum Journal of Science and Interdisciplinary Studies

Journal homepage: https://lfjsis.journals.ekb.eg/

Labyrinth Journal

An Enhanced Cloud Allocation Approach Based on Metaheuristics Algorithm

Mohammed E. M. Shaaban [a], Sayed Abdel Gaber [b], Rasha M. Badry [a,*]

[a] Information Systems Department, Faculty of Computers and Artificial Intelligence, Fayoum University, El Fayoum 63514, Egypt.
[b] Information Systems Department, Faculty of Computers and Artificial Intelligence, Helwan University, Helwan 11795, Egypt.

A R T I C L E I N F O

Keywords:
Data placement,
Big Data,
Cloud Computing,
Metaheuristics,
Marine Predators Algorithm

A B S T R A C T

In cloud computing environments, effective data placement is critical for optimizing system performance and resource utilization. This research introduces an innovative framework: Enhanced Cloud Data Placement Strategy Using Marine Predator Optimization. To address this challenge, this algorithm leverages the Marine Predators Algorithm (MPA), a nature-inspired metaheuristic inspired by marine predators' hunting behavior, balancing exploration and exploitation for efficient optimization. The proposed framework leverages MPA's exploration and exploitation capabilities to reduce data movement between data centers and enhance resource allocation efficiency. Through simulation in a controlled environment using the CloudSim toolkit, we evaluated the performance of MPA in comparison with other state-of-the-art metaheuristic algorithms, including the Gaining Sharing Knowledge-based Algorithm (GSKA), War Strategy Optimization (WarSO Generalized Whale Optimization Algorithm combined with Whale Optimization Algorithm (GWO_WOA), and Success History Intelligent Optimizer (SHIO). Experimental results demonstrate that MPA outperforms these algorithms regarding runtime and overall resource utilization. Further tests, including scalability evaluations with increasing dataset sizes and data center numbers, revealed MPA's robustness and adaptability for large-scale cloud infrastructures. The performance comparison indicates that applying MPA to solve the proposed problems consistently yields lower makespan and runtime, positioning it as a promising solution for dynamic and heterogeneous cloud environments.

1. Introduction

Each detail of data is significant and may guide the decision makers to change their minds and make the right decision, in all business fields, research, social media, and even governmental data. The data generated daily is huge and critical for decision-makers. Data analysis is a very useful and decision-driven process done over big data. It helps in problem-solving, decision-making, manufacturing, and research, and in finding different patterns to deal with the routine process in all aspects. Big data storage is a very critical field as we want to reuse these amounts of detailed information; different approaches are required than the usual data storage approaches [1, 2]. The decision-making process for big data requires a lot of effort, from the process of collecting data until the best results are achieved. Much research has been conducted by multidisciplinary and industrial experts to study and produce the approaches, methods, and tools that can be applied to big data processing [3, 4]. The large size of the data might contain a complex type of data set, which means that the data set has multiple types of criteria and has an imbalance, and inconsistent data values. Data veracity indicates that big data has very high uncertainty and inconsistent data. Big data can be categorized into four phases: data generation, data acquisition, data storage, and data analysis before we can access and analyze the data; which is the main goal of big data access, we have to store the data suitably to make it easy to access it later [5-7].

Big data researchers claim that traditional analytical tools often have difficulties in managing big data; however, storage technologies have not changed the fundamental conventional business intelligence approaches in ordinary business organizations, as different types of organizations may not be the same [8,9]. Before cloud computing, grid computing and sector computing were proposed as solutions for big data storage, but the cloud computing concept is one of the best solutions for big data storage. Cloud computing is an easy-access resource-sharing approach to solve big data storage and processing problems, especially since cloud resources are available anytime, anywhere, and as needed, and also the cost is as used [10,11].

Efficient data placement in distributed computing environments, such as cloud systems, has grown increasingly challenging due to the rapid expansion in both data volume and variety. Traditional approaches, like deterministic algorithms, often fall short in adapting to the dynamic and heterogeneous nature of these environments. This can lead to inefficient use of resources and reduced system performance. As data diversity and

* Corresponding author.
E-mail address: rmb01@fayoum.edu.eg (R. Badry); Tel.: +2010026722120

DOI: 10.21608/ifjsis.2025.369477.1110
Received 18 March 2025; Received in revised form 8 May 2025; Accepted 9 May 2025
Available online 19 May 2025

workloads increase, finding flexible and scalable data placement strategies is essential to ensure optimal resource utilization and system responsiveness in cloud computing [12,13]. Cloud data placement is a well-known combinatorial optimization challenge, where the goal is to efficiently distribute datasets across multiple data centers. Each dataset has specific storage and computational requirements, while data centers have limited storage capacities but can handle computations concurrently. The problem lies in minimizing data scheduling operations during task execution while ensuring that storage and computational resource constraints are met. Efficient data placement can significantly enhance performance by reducing latency and improving the overall system's responsiveness and resource utilization [1].

In recent years, metaheuristic algorithms inspired by nature have become one of the hot topics in solving many problems. Genetic algorithms, particle swarm optimization, and simulated annealing are some examples that draw inspiration from the domains of biological and physical phenomena to tackle complex optimization issues arising in various domains [14]. For example, genetic algorithms, which simulate the process of natural selection, have been used to optimize data placement by exploring a wide search space and evolving solutions over successive generations [15]. Similarly, ant colony optimization, based on the foraging behavior of ants, has been applied to optimize data placement strategies by enabling agents to discover near-optimal solutions through indirect communication [16]. Moreover, particle swarm optimization, inspired by the social behavior of bird flocks, has shown promise in solving high-dimensional data placement problems. This method allows for the exploration of complex search spaces and has been used to optimize data distribution in cloud storage systems, improving both performance and cost-efficiency [17]. These nature-inspired algorithms offer an advantage in managing the uncertainties and dynamic nature of cloud computing environments.

The problem can be formulated as follows: "How can big data be efficiently allocated across a cloud environment while minimizing time, optimizing costs, reducing network cost, and ensuring a balanced workload?" The primary objective of this research is to propose a framework that implements a metaheuristic algorithm that can effectively allocate the data and address these multiple constraints simultaneously. Metaheuristic algorithms are particularly well-suited for solving nondeterministic polynomial hard (NP-hard) and combinatorial problems like data placement, where finding optimal solutions using traditional methods within a reasonable amount of time is challenging [14].

Recent studies have highlighted the growing importance of metaheuristic approaches in addressing such complex optimization problems in cloud computing. For instance, the use of metaheuristics like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) has been explored in data placement to improve resource utilization. These methods, although effective, still have limitations in balancing time, cost, and network constraints, which necessitates the exploration of newer and more efficient algorithms [18-20]. It is important to revisit and define the key terminologies and concepts of this study to provide a comprehensive understanding of it. Cloud computing is a technology that delivers various services such as storage, computing power, and networking over the internet. Rather than owning or maintaining physical data centers, organizations can rent cloud services. This paradigm shift enables businesses and individuals to use computing resources as needed without upfront investment in hardware [21]. Cloud computing is primarily categorized into three models: public cloud. In the public cloud, third-party cloud service providers offer resources such as virtual machines, storage, and applications over the internet [22]. The second model is a private cloud, a cloud infrastructure that is used by a single organization. Private clouds offer more control over data, security, and compliance [23]. The last one is the hybrid cloud, which integrates both public and private cloud models. Hybrid clouds provide a balance between cost efficiency and data security, making them a popular choice for enterprises [24-26]. Managing large, diverse datasets presents challenges, particularly in cloud computing. Surveys highlight issues like storage, data privacy, latency, and fault tolerance[27]. The varied nature of big data, including structured, semi-structured, and unstructured formats, necessitates tailored storage and processing methods. Organizations must adopt specialized techniques like data integration, preprocessing, indexing, and advanced algorithms for efficient and safe data management and analysis in the cloud [28-30].

Cloud computing systems have many categories of heterogeneous processors, meaning that computing nodes can vary in processing capabilities based on factors such as clock speed, core count, cache size, and memory bandwidth. The costs associated with executing tasks within these environments fluctuate depending on the specific processor and network bandwidth available at each cloud node. Therefore, careful planning and optimization are essential to ensure efficient big data storage and processing, maintaining both high performance and cost-effectiveness[31,32]. The data placement problem in data centers represents a complex combinatorial optimization challenge, where the primary objective is to strategically distribute datasets across multiple centers to optimize resource utilization and computational efficiency. One of the desired goals is to ensure that each dataset is stored in only one data center, maximizing the use of storage without exceeding capacity limits [33-36]. Metaheuristic algorithms are powerful, flexible methods for tackling complex optimization problems that are difficult to solve using traditional methods. These algorithms, inspired by natural and behavioral phenomena, aim to find near-optimal solutions efficiently. By balancing exploration (seeking diverse solutions) and exploitation (optimizing known good solutions), they can navigate vast solution spaces without getting trapped in local optima. In cloud computing, metaheuristics have proven effective in resource allocation and data management [37-39].

The MPA is an emerging metaheuristic that models the hunting behaviors of ocean predators like sharks and whales. MPA alternates between phases resembling aggressive searching and patient waiting, allowing it to explore solutions comprehensively while avoiding local optima. In cloud systems, MPA has shown great promise in optimizing data placement, effectively improving resource utilization, and processing speed. Key attributes of the MPA include Exploration-Exploitation Balance: MPA excels in balancing the search for diverse solutions with refining known optimal areas, helping to avoid local optima and reach near-optimal solutions. Adaptive Parameter Adjustment: By adjusting parameters in response to specific problem characteristics, MPA enhances its performance across varied scenarios. Efficient Search Mechanism: MPA systematically explores the solution space, focusing on promising regions to maximize efficiency while minimizing unnecessary calculations [40,41].

In Alboaneen et al. [42], the authors proposed a hybrid metaheuristic approach that addresses data placement over virtual machines (VMs) as a combined problem, optimizing resource use and lowering energy costs by reducing makespan and allocation expenses. Simulations show promising gains in energy efficiency, Quality of Service (QoS), and resource utilization. However, additional analysis of computational overhead would further clarify the approach's practicality in larger data centers. In Prabhu et al. [39], an advanced, nature-inspired data placement method for cloud environments using the Firefly Algorithm is introduced. The proposed approach works to optimize data placement by enhancing the attraction function and implementing local search to achieve fast convergence, reduce time, and data retrieval. While promising, the study could improve by conducting some experiments over large-scale datasets to make valid comparisons and analyze the performance over different scenarios. In Rajashekar et al. [43], a hybrid algorithm is proposed that combines Sine Cosine-based Elephant Herding Optimization (SCEHO) with Improved Particle Swarm Optimization (IPSO) to enhance cloud scheduling. This hybrid approach integrates the exploration ability of SCEHO with the fine-tuning of IPSO for efficient resource allocation, load balancing, and latency minimization. Experimental results show improved performance in execution time, memory efficiency, and latency. While effective, SCEHO-IPSO faces challenges. The increased complexity of the algorithm may raise the overhead and affect the performance. In Hai et al. [44], present some optimization techniques for the efficient task-scheduling process in a cloud environment. It examines some traditional algorithms, such as Heterogeneous Earliest-Finish-Time (HEFT) and Critical Path on Processor (CPOP), for communication cost minimization and task heterogeneity handling, introducing enhanced versions of these approaches that take security into account to protect data during processing. While these improvements offer a balanced approach to scheduling, the study's analysis is limited by a lack of scalability testing on large workloads.

In Yahia et al. [45], present a comprehensive examination of how nature-inspired algorithms can enhance scheduling in cloud computing environments. Some key algorithms have been focused on Ant Colony Optimization (ACO), Ginatic Algorithm (GA), Particle Swarm Optimization (PSO), and Cuckoo Search (CS). Such algorithms would be assessed concerning their capabilities to handle vital issues like load balancing, minimization of makespan, energy consumption, and resource allocation. Inspired by biological procedures for exploration and exploitation, these algorithms find a highly suitable application in dynamic and distributed cloud computing systems. However, there are no detailed performance metrics. In Algamdi [46], a hybrid methodology has been investigated to improve load balancing in cloud computing frameworks. The authors implement neural networks with binary particle swarm optimization to improve the efficiency of resource allocation. In this integration of Artificial Neural Network ANN, the binary particle swarm optimization (BPSO) algorithm is enhanced to guarantee better optimization of the scheduling of tasks across virtual machines. The research objectives were minimizing makespan, enhancing resource utilization, and achieving effective load balancing. Experimental results show that the ANN-based BPSO model outperforms other techniques in optimizing resource allocation and for better processing time, and effective load distribution. The one major disadvantage of this approach is the computational overhead. Besides, the combination of ANN and BPSO introduces complexity that might be hard to implement and maintain.

In Table 1, the reviewed studies highlight the significant advancements in data placement in cloud computing, leveraging hybrid and nature-inspired metaheuristic approaches. These methods consistently demonstrate improvements in reducing makespan, enhancing resource utilization, and optimizing energy efficiency. However, challenges such as computational overhead, scalability testing, and real-time applicability remain prevalent. Future research should focus on addressing these gaps through adaptive hybrid frameworks, large-scale dataset experiments, and practical implementations to further refine and validate these innovative solutions.

Table 1: Summary of related research papers

Approach	Focus Area	Key Contributions	Limitations	Reference
Hybrid metaheuristic	Data placement & task allocation	Optimizes resource use, reduces energy cost, and makes	Lacks computational overhead analysis for large-scale data centers	[42]
Firefly Algorithm	Data placement in the cloud	Fast convergence, time, and data retrieval optimization	Needs large-scale experiments and scenario comparisons	[39]
SCEHO + IPSO (Hybrid)	Cloud Scheduling	Combines exploration & fine-tuning for efficient allocation, load balancing, and latency	Increased complexity and algorithmic overhead	[43]
Enhanced HEFT, CPOP	Task scheduling & security	Security-aware improvements in traditional task-scheduling algorithms	Lacks scalability tests on large workloads	[44]
ACO, GA, PSO, CS	Scheduling & resource allocation	Addresses load balancing, energy, makespan, and allocation using bio-inspired strategies	No detailed performance metrics	[45]
ANN + BPSO (Hybrid)	Load balancing & scheduling	Improved resource utilization, processing time, and load distribution	Computational overhead and implementation complexity	[46]

In this research, we aim to implement recent and novel metaheuristic-based algorithm to solve the data placement problems, comparing them with other well-established metaheuristic algorithms from various categories. The goal is to minimize the total number of data scheduling operations and reduce the need for inter-data center communication. Performance evaluations will validate the model's effectiveness using simulations and real-world big data workloads. The rest of this paper is organized as follows: Section 2 explains the materials and methodology, focusing on the proposed approach and its development. Section 3 presents the experimental setup and results, including a comparison of the proposed framework with other algorithms. Section 4 provides a discussion of the findings and key insights. Finally, Section 5 concludes the paper and suggests future research.

2. Materials and Methods

In cloud computing environments, there are many challenges in data placement operations because of the large-scale resource complexity and workloads [47]. In this research, we introduce the Enhanced Cloud Data Placement Strategy Using Marine Predator Optimization, applying a metaheuristic algorithm called MPA. The proposed framework is intended to be an innovation-based solution targeting optimizing the data allocation process over the cloud, which is categorized as a combinatorial optimization problem. With NP-hard problems like this problem, the traditional algorithms often fail to find optimal solutions within a suitable time, which makes metaheuristic algorithms a more suitable alternative. The main objective of this study is to enhance performance and reduce overall runtime within cloud systems [38,40]. The proposed framework focuses on identifying and applying a metaheuristic algorithm for which there is no investigation for data placement. The goal is to utilize and examine new methods over existing metaheuristics from various categories, developing an enhanced framework that guarantees resource utilization.

2.1 The mathematical representation of the combinatorial optimization problem

To accurately define and solve the data placement challenge in cloud computing, it is essential to formulate the problem mathematically. This section presents a formal model that captures the key objectives and constraints of the problem, allowing it to be approached as a combinatorial optimization task. Given that data placement is NP-hard, this formulation will guide the development and evaluation of our proposed metaheuristic solution. The goal of the data placement problem is to assign data blocks to virtual machines (VMs) or storage nodes in a cloud environment such that certain objectives (e.g., minimizing cost, latency, or data movement) are optimized while respecting system constraints (e.g., storage capacity, data locality, bandwidth, etc.) [48].

Notation:

- $D = \{d_1, d_2, \ldots, d_n\}$ be the set of data blocks
- $V = \{v_1, v_2, \ldots, v_m\}$ be the set of virtual machines
- $x_{ij} \in \{0, 1\}$:
  - $x_{ij} = 1$ if data item d_i is assigned to VM v_j
  - $x_{ij} = 0$ otherwise
- c_{ij}: the cost (e.g., time, energy, or bandwidth) of placing d_i on v_j
- s_j: the storage capacity of VM v_j
- $size_i$: the size of data item d_i

Objective Function:

Minimize total placement cost: $\min \sum_1^n \sum_1^m c_{ij} * x_{ij}$

Constraints:

Each data item must be placed on exactly one VM: $\sum_1^m x_{ij} = 1 \ \forall\ i = 1, \ldots, n$

Storage capacity constraint of each VM: $\sum_1^n size_i * x_{ij} \le s_j \ \forall\ j = 1, \ldots, m$

Binary assignment variables: $x_{ij} \in \{0, 1\} \ \forall\ i = 1, \ldots, n;\ j = 1, \ldots, m$

2.2 The Proposed Algorithm: Enhanced Cloud Data Placement Strategy Using Marine Predator Optimization

The proposed data placement strategy employs the Marine Predators Algorithm (MPA), a nature-inspired metaheuristic, to optimize dataset allocation across distributed cloud data centers. The algorithm addresses challenges in minimizing data access latency while adhering to storage capacity constraints. Input parameters include datasets (defined by size), data centers (with storage limits), computations (task requirements), and an access matrix mapping dataset-task dependencies as mentioned in Fig. 1. Initial solutions are generated as "predators" exploring the solution space, representing diverse dataset-to-data-center assignments. These solutions evolve through exploration and exploitation phases to identify optimal placements.

```
Initialize Parameters:
    Set number of datasets `n = 5`
    Generate random sizes of datasets `s = [s1, s2, ..., sn]` in GB
    Set number of data centers `l = 2`
    Generate random capacities of data centers `C = [C1, C2]` in GB
    Set number of computations `m = 5`
    Generate random execution frequencies `μ = [μ1, μ2, ..., μm]`
    Generate a binary matrix `α` of size (m, n) representing dataset requirements for each computation
Define DataPlacement Class:
    Class DataPlacement(Problem):
        Constructor:
            Initialize the data (`s`, `C`, `μ`, `α`)
            Call superclass constructor
        Method: `Constraints(solution)`:
            Initialize `data_center_capacities = [0, 0]`
            For each dataset `i = 0` to `n-1`:
                Assign dataset `i` to the data center specified in `solution[i]`
                Update `data_center_capacities` accordingly
            Initialize `penalty = 0`
            For each data center `k = 0` to `l-1`:
                If `data_center_capacities[k] > C[k]`:
                    Add the excess to `penalty`
            Return `-penalty`
        Method: `obj_func(solution)`:
            Decode solution: determine which data center each dataset is placed in
            Initialize `beta = zeros matrix (n, l)`  // tracks dataset-to-data center mapping
            For each dataset `i = 0` to `n-1`:
                Set `beta[i, assigned_data_center] = 1`
            Compute `z = α.dot(beta)`  // Matrix multiplication
            Create binary array `u = (z > 0)` indicating access of data for computations
            Calculate `constraint = Constraints(solution)`
            Initialize `data_placements = 0`
            For each computation `i = 0` to `m-1`:
                Update `data_placements += μ[i] * (sum(u[i]) - 1)`
            If `data_placements == 0`:
                Return `1 + constraint`
            Return `(1 / data_placements) + constraint`
Define Variable Bounds:
    Set lower bounds `bounds_lower = [1, 1, 1, 1, 1]`
    Set upper bounds `bounds_upper = [l, l, l, l, l]`  // where l is the number of data centers
Initialize Problem Instance:
    Create an instance of DataPlacement with `bounds = (bounds_lower, bounds_upper)` and other data
Configure Marine Predators Algorithm (MPA):
    Set parameters: `epoch = 100`, `pop_size = 100`
    Create the MPA model with the given parameters
Solve the Problem:
    Run the MPA algorithm: `best_solution = model.solve(problem_instance)`
Output the Results:
    Print the best solution (dataset-to-data center assignments)
    Print the best fitness value as `1 / best_solution.target.fitness`
End Algorithm
```

Fig. 1. Enhanced Cloud Data Placement Strategy Using Marine Predator Optimization Algorithm

During exploration, datasets are randomly distributed to maximize solution diversity. In the exploitation phase, high-performing solutions are refined to reduce data scheduling costs—the total number of data accesses required to fulfill computation requests. The fitness function prioritizes placements that minimize this cost while respecting capacity limits. The algorithm iterates until convergence, yielding an optimized placement matrix. This approach balances global exploration and local refinement, ensuring scalability and efficiency in dynamic cloud environments. It aligns with trends in metaheuristic-driven resource management, offering a robust solution for large-scale data-intensive applications [40, 41].

The algorithm proposes to optimize data placement by applying MPA and balancing exploration and exploitation within the search space to ensure minimal data movement between data centers. By placing datasets adjacent to the virtual machines (VMs) doing the associated tasks, the suggested method aims to reduce communication across data centers. This closeness improves system performance by lowering latency. By comparing the proposed framework applying MPA with GSKA: a nature-inspired metaheuristic that simulates human interactions and knowledge exchange to solve optimization problems by balancing exploration and exploitation through two stages: knowledge acquisition and knowledge sharing, WarSo: a nature-inspired metaheuristic that mimics military tactics to solve complex optimization problems by dynamically balancing exploration and exploitation phases. It has been proposed for applications in cloud computing, such as resource allocation and task scheduling. and GWA_WOA: combines the Grey Wolf Optimizer (GWA) and Whale Optimization Algorithm (WOA) to leverage their complementary strengths in exploration and exploitation, creating a hybrid metaheuristic for solving complex optimization problems [49-51], this study aims to provide valuable insights into the robustness and scalability of MPA in real-world cloud scenarios, especially when managing large-scale datasets.

Algorithm inputs are the datasets, data centers, computations, and expected access requirements: A matrix representing the association between datasets and computations, indicating which datasets are needed for which computations. The optimal solution, an optimized placement matrix that allocates datasets to data centers in a way that minimizes data scheduling while adhering to capacity restrictions, is produced by MPA after iterating through the exploration and exploitation phases. Fig. 2 displays the key elements of the algorithm and also shows how the MPA algorithm is utilized.

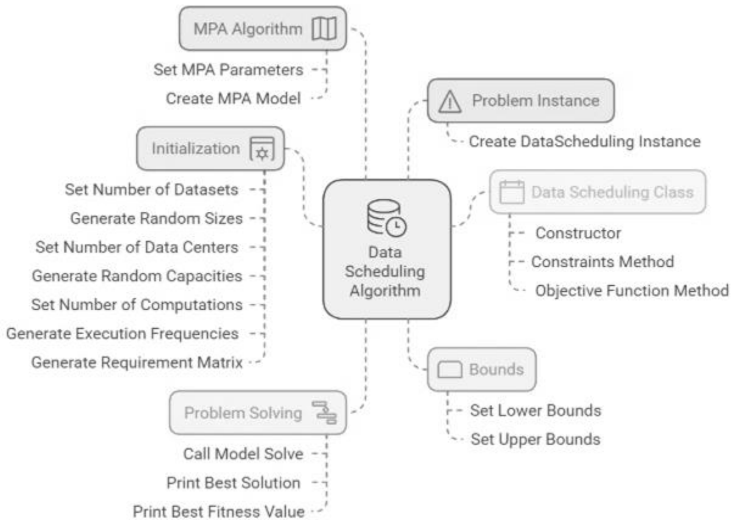

Fig. 2 Main Components of the Proposed Algorithm

The following Fig. 3 shows the algorithm workflow as follows: the first step is initializing all the necessary parameters for datasets, data centers, computations, and their requirements, the second step is to define the core functionality of the data placement problem, including constraint checking and objective function evaluation. Step three is to set the bounds for each dataset's assignment. The next step is creating the problem instance based on the initialized data and bounds. Then the configuration of the MPA with the necessary parameters, solving the problem using and finding the near-optimal solution. Finally outputs the best solution and its corresponding fitness value.

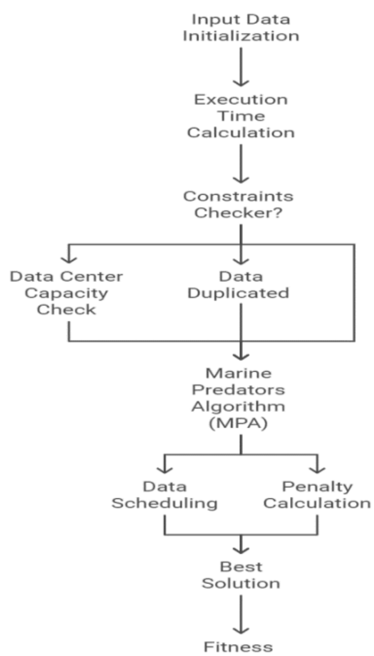

Fig. 3 Data Flow in the Proposed Algorithm

3. Results

3.1 Experiment Setup

The experimental settings and techniques utilized to assess the suggested framework's performance in data placement frameworks are described in this section. We performed controlled experiments to assess the efficiency of the suggested framework for data placement. A Windows computer with an Intel Core i7-8550U CPU running at 1.80 GHz with a turbo boost capacity of up to 2.00 GHz and 12 GB of RAM made up the testing setup. Python was used to implement the algorithms. The absence of defined benchmarks or simulated environments created especially for evaluating data placement issues, especially with metaheuristic techniques, was one major obstacle we faced. We developed our benchmark dataset to get around this problem. With the use of this unique dataset, we were able to thoroughly assess the suggested framework's performance using MPA in comparison to one math-based algorithm, SHIO, and two human-inspired metaheuristic algorithms, GSKA and WarSO. Through the use of this dataset, we made sure that the various optimization techniques were fairly compared, enabling a thorough evaluation of their efficacy in various data placement scenarios.

CloudSim, which is a popular simulation toolkit made for cloud computing environments, was used to assess the proposed algorithms. We created a simulation of a cloud infrastructure with two data centers and two hosts in each. Using a time-shared virtual machine scheduling mechanism, each host has 20 GB of RAM, 1 TB of storage, and a bandwidth of 10 GB. The virtual machines (VMs) had processing speeds ranging from 100 to 1000 MIPS and were set up with a 1 GB picture size, 0.5 GB of memory, and a 1 GB bandwidth. The cloudlet lengths were generated following a normal distribution, with a mean of 10,000 and a standard deviation of 5,000. The proposed framework was implemented using Python and tested with CloudSim [52].

3.2 Performance Comparison

We used CloudSim to simulate and evaluate a variety of metaheuristic algorithms, such as the MPA, GSKA, which is a human-based metaheuristic, and GWO_WOA; both are hybrid metaheuristic algorithms, to compare the performance in data placement. The key performance indicators (KPIs), including average performance, runtime, and the best result, were the focus of the evaluation. Due to their stochastic nature, it was insufficient to run a single run to evaluate the performance of these algorithms. Each algorithm's solution with the least amount of data scheduling operations is reflected in the best result measure. While execution time was used to assess computational efficiency, the average performance metric gave a more comprehensive view of algorithm performance by summarizing the results of all 100 executions. We changed the number of data centers and datasets to better evaluate scalability. We employed 15 data centers with capacities ranging from 10,000 to 20,000 GB and 100 datasets with sizes ranging from 500 to 2000 GB in our simulation. Furthermore, we randomly produced 30 calculations with execution frequencies ranging from 10 to 30.

Table 2 shows how well the suggested framework with MPA performs in comparison to GSKA and GWO_WOA, providing information on how well it works to optimize data placement in distributed cloud systems, especially concerning scalability and adaptability. The suggested algorithm performed better than the alternative approaches in important measures, as shown in the next section, Discussion.

Table 2: Data placement solution quality (no. of accesses) and runtime in seconds

	MPA	GSKA	GWO_WOA	MPA	GSKA	GWO_WOA
	Solution quality (No. of accesses)			Runtime in seconds		
Best	4878	6853	5768	13.5 s	14.9 s	13.3 s
Average	5667.8	7278.44	6462.72	17 s	18.4 s	14.16 s
Worst	7003	7523	7060	23.2 s	27.4 s	15.5 s

As mentioned in the following charts, Fig. 4 and Fig. 5, The fitness function assesses how well each solution performs based on 2 aspects: data scheduling which is defined as the total number of data scheduling operations needed to fulfill computation requests based on the placement of datasets, and the algorithm calculates the total number of data accesses across all computations for each solution.

The experimental evaluation demonstrates that the proposed Enhanced Cloud Data Placement Strategy using the Marine Predators Algorithm (MPA) consistently outperforms other benchmark algorithms (GSKA, GWO_WOA, SHIO, and WarSO) in both solution quality and runtime. MPA achieved the lowest number of data accesses across best, average, and worst scenarios, indicating more efficient and balanced data placement. Additionally, it maintained competitive execution times, confirming its practical suitability for real-time and large-scale cloud environments. These outcomes highlight MPA's effectiveness in delivering high-quality solutions with reliable performance under varying workloads and conditions. The results show that the proposed framework applying MPA provides a robust approach for solving the data placement problem with a strong balance between performance and efficiency across multiple runs.

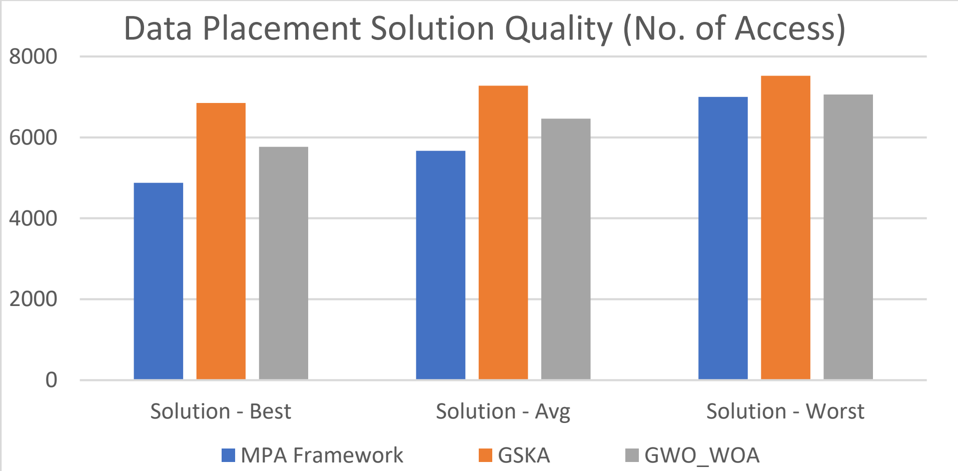

Fig. 4 Solution Quality Chart

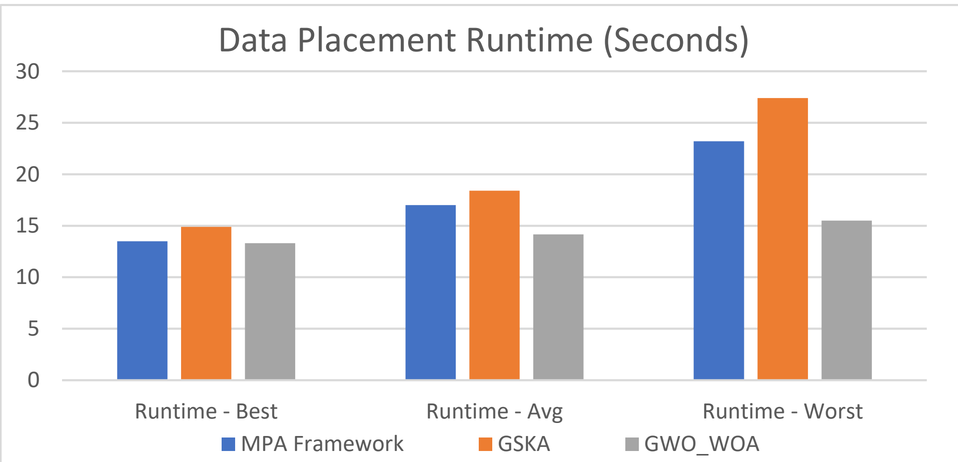

Fig. 5 Chart of Data Placement Runtime

4. Discussion

This section presents a detailed analysis of the results obtained from the experiments, focusing on the performance of the Enhanced Cloud Data Placement Strategy using the MPA. For comparative purposes, several nature-inspired optimization algorithms were evaluated, including GSKA, WarSO, GWO_WOA, and SHIO [49-51]. These algorithms were chosen based on their relevance to cloud resource scheduling and optimization challenges. Solution quality: The suggested framework achieved the best solution with the lowest data scheduling operation cost of 4878. In contrast, GSKA recorded the highest cost at 6853, followed by GWO_WOA with 5768. These values indicate that the suggested framework consistently outperformed both GSKA and GWO_WOA in generating efficient placement solutions. The average solution value across 100 runs further supports this observation, where the suggested framework recorded 5667.8, compared to GWO_WOA with 6462.72 and GSKA with 7278.44. Even in the worst-case scenario, the suggested framework remained superior, achieving a cost of 7003, which is better than GWO_WOA at 7060 and GSKA at 7523. These results demonstrate the robustness and reliability of the suggested framework under varying conditions. Additional comparative insights include WarSO, which recorded a solution cost of 5580, and SHIO, which resulted in 6021. Although both showed competitive results, they did not match the consistent performance of the proposed framework.

Runtime Performance: Regarding execution time, the suggested framework recorded a best-case runtime of 13.5 seconds. GWO_WOA achieved the fastest runtime at 13.3 seconds, while GSKA and SHIO recorded 14.9 and 14.5 seconds, respectively. WarSO showed slightly slower performance with 15.1 seconds. On average, the suggested framework achieved a runtime of 17 seconds, remaining competitive against GWO_WOA's 14.16 seconds, GSKA's 18.4 seconds, SHIO's 17.5 seconds, and WarSO's 18.1 seconds. Despite GWO_WOA being marginally faster on average, the suggested framework offered a stronger balance and trad-off between runtime and solution quality. In the worst-case scenario, the suggested framework recorded a runtime of 23.2 seconds, while GSKA reached 27.4 seconds, SHIO 24.8 seconds, and WarSO 26.7 seconds. GWO_WOA maintained a faster worst-case time of 15.5 seconds.

To contextualize the efficacy of the proposed MPA-based framework, it's instructive to examine similar studies employing nature-inspired algorithms for cloud resource scheduling. In Panesar and Chadha [52], introduce an optimization strategy that combines deep reinforcement learning (DRL) with a cloud-based adaptive multi-agent framework. The study is targeted at enhancing the efficiency of VM migration in cloud computing environments. The proposed technique combines the multi-agent system with DDPG, a class of DRL algorithms that enables dynamic and adaptive decision-making for VM placement. This hybrid approach is designed to minimize migration costs, network latency, and makespan while optimizing resource allocation across the cloud infrastructure. It results from the experiment that the proposed technique has outperformed other scheduling and migration methods, showing improved performance, especially in decreasing computation overhead and balancing the loads. Enhanced Whale Optimization Algorithm (EWOA): Zhang and Wang (2024) introduced EWOA, integrating Lévy flight and adaptive crossover strategies to enhance exploration and convergence in task scheduling. Their simulations using CloudSim demonstrated that EWOA outperformed traditional WOA, ACO, and GA in resource utilization, energy consumption, and execution cost [54]. Hybrid Grey Wolf and Whale Optimization (HGWWO): Ababneh (2021) developed HGWWO to tackle cloud task scheduling, aiming to minimize costs, energy consumption, and execution time. The hybrid algorithm demonstrated superior performance over standalone GWO and WOA algorithms in simulations conducted using CloudSim [55]. In conclusion, the suggested framework demonstrated an excellent trade-off between solution quality and runtime performance. It consistently delivered competitive computing efficiency while producing optimal or near-optimal outcomes. Compared to other metaheuristic algorithms such as GSKA, GWO_WOA, SHIO, and WarSO, the proposed solution maintained superior performance across all evaluation metrics, emphasizing its suitability for practical deployment in cloud data placement environments.

5. Conclusion

In this study, we used the MPA to present an effective framework designed for cloud computing environments called Enhanced Cloud Data Placement Strategy Using Marine Predator Optimization. Our experimental results show that the suggested algorithms perform better than other techniques in terms of makespan reduction, resource allocation optimization, and system performance enhancement. Because of this, they are promising solutions for large-scale, dynamic cloud systems with a variety of resource configurations and activities of different complexity. Overall, this study highlights the potential of nature-inspired optimization algorithms like MPA in tackling complex scheduling challenges in cloud computing, paving the way for further research and development in this field. Additionally, our study demonstrates the effectiveness of MPA for optimizing both data placement and task scheduling within cloud environments. By comparing MPA with other state-of-the-art algorithms such as GSKA, WarSO, SHIO, and GWA_WOA, we illustrate its capability to achieve significant improvements in makespan, resource utilization, and runtime. This study opens the door for more research and development in this area: nature-inspired algorithms in addressing challenging scheduling problems in cloud computing. Furthermore, our study shows how well MPA works in cloud systems to optimize job scheduling and data location. This approach could be expanded in future studies to incorporate multi-objective optimization, which would handle aspects like fault tolerance and energy usage.

Acknowledgment

The authors would like to thank Fayoum University for supporting the publication of this work.

Author Contributions

All authors contributed to this work. Shaaban collected the dataset and completed the experimental results and evaluations. Shaaban completed the paper writing. Both Shaaban and Badry followed the paper writing, analyzing the data, validation, and performance of the results, Badry, Shaaban and Abdel Gaber followed the revision and submission of the manuscript for publication.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Berisha, E. Mëziu, I. Shabani, Big data analytics in Cloud computing: an overview, J. Cloud Comput. 11(1) (2022) 24. https://doi.org/10.1186/s13677-022-00301-w.

[2] A. Sandhu, Big data with cloud computing: Discussions and challenges, Big Data Min. Anal. 5(1) (2022) 32–40. https://doi.org/10.26599/BDMA.2021.9020016.

[3] C. Torre, G.M. Guazzo, V. Çekani, V. Bacco, The relationship between Big Data and decision making: A systematic literature review, J. Serv. Sci. Manag. 15(2) (2022) 89–107. https://doi.org/10.4236/jssm.2022.152007.

[4] S. Jeble, S. Kumari, Y. Patil, Role of Big Data in Decision Making, Oper. Supply Chain Manag. Int. J. (2018) 36–44. https://doi.org/10.31387/oscm0300198.

[5] M. Mohamad, A. Selamat, O. Krejcar, H. Fujita, T. Wu, An analysis on new hybrid parameter selection model performance over the big data set, Knowl.-Based Syst. 192 (2020) 105441. https://doi.org/10.1016/j.knosys.2019.105441.

[6] T. Tekdogan, A. Cakmak, Benchmarking Apache Spark and Hadoop MapReduce on Big Data classification, in: Proc. 5th Int. Conf. Cloud Big Data Comput. (ICCBDC), Liverpool, UK, 2021, pp. 15–20. https://doi.org/10.1145/3481646.3481649.

[7] M. Younas, Research challenges of big data, Serv. Oriented Comput. Appl. 13(2) (2019) 105–107. https://doi.org/10.1007/s11761-019-00265-x.

[8] A. Banik, S.K. Bandyopadhyay, Big Data – A Review on Analysing 3Vs, 2016, p. 5.

[9] R.S. Kalan, M.O. Unalir, Leveraging big data technology for small and medium-sized enterprises (SMEs), in: Proc. 6th Int. Conf. Comput. Knowl. Eng. (ICCKE), Mashhad, Iran, 2016, pp. 1–6. https://doi.org/10.1109/ICCKE.2016.7802106.

[10] Y. Mao, D. Gan, D.S. Mwakapesa, Y.A. Nanehkaran, T. Tao, X. Huang, A MapReduce-based K-means clustering algorithm, J. Supercomput. 78(4) (2022) 5181–5202. https://doi.org/10.1007/s11227-021-04078-8.

[11] M. Reis, A.F. de Araújo, ArchaDIA: An architecture for big data as a service in private cloud, in: Proc. 9th Int. Conf. Cloud Comput. Serv. Sci., Crete, Greece, 2019, pp. 187–197. https://doi.org/10.5220/0007787801870197.
[12] A.Q. Khan, N. Nikolov, M. Matskin, R. Prodan, D. Roman, B. Sahin, C. Bussler, A. Soylu, Smart data placement using Storage-as-a-Service model for big data pipelines, Sensors 23(2) (2023) 564. https://doi.org/10.3390/s23020564.
[13] B. Lin, F. Zhu, J. Zhang, J. Chen, X. Chen, N. Xiong, J. Lloret, A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing, IEEE Trans. Ind. Inform. (2019) 1–1. https://doi.org/10.1109/TII.2019.2905659.
[14] I. Fister, X.-S. Yang, D. Fister, J. Brest, A comprehensive review of nature-inspired algorithms in cloud computing: Taxonomy and open challenges, Future Gener. Comput. Syst. 129 (2022) 1–18.
[15] K. Srinivas, B. Kalavathi, Genetic algorithm-based approach for optimizing data placement in cloud environments, J. Cloud Comput. 10(1) (2021) 1–14.
[16] X. Su, C. Liu, Y. Gu, Ant colony optimization-based data placement strategy for cloud storage, J. Netw. Comput. Appl. 152 (2020) 102–123.
[17] Y. Zhao, X. Xu, Y. Wang, Particle swarm optimization for data placement in heterogeneous cloud environments: A performance evaluation, IEEE Trans. Cloud Comput. 11(2) (2023) 325–337.
[18] A. Shah, S. Khan, M. Ali, Metaheuristic approaches for task scheduling in cloud environments: A comprehensive review, J. Cloud Comput. Res. 11(2) (2023) 45–62.
[19] J. Li, Y. Wang, Z. Luo, Optimizing resource utilization in cloud computing with genetic algorithms, Cloud Syst. Manag. 7(1) (2022) 12–29.
[20] H. Zhang, F. Yang, A Grey Wolf Optimizer-based approach for efficient data placement and task scheduling, J. Comput. Optim. 9(3) (2021) 200–215.
[21] P. Mell, T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication, National Institute of Standards and Technology, 2011.
[22] R. Buyya, Mastering Cloud Computing, 2nd ed., McGraw Hill, New Delhi, India, 2024.
[23] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, M. Badger, D. Leaf, NIST Cloud Computing Reference Architecture, NIST SP 500-292, Gaithersburg, MD, 2011. https://doi.org/10.6028/NIST.SP.500-292.
[24] M.G. Avram, Advantages and challenges of adopting cloud computing from an enterprise perspective, Procedia Technol. 12 (2014) 529–534.
[25] M.B. Mollah, M.S. Islam, S. Islam, Data security and privacy in cloud computing: Challenges and solutions, IEEE Access 9 (2022) 29699–29720.
[26] L. Wang, R. Ranjan, et al., Cloud resource management: State of the art and future trends, J. Cloud Comput. (2019).
[27] M. Megahed, M. Badry, S. AbdelGaber, Survey on big data and cloud computing: Storage challenges and open issues, in: Proc. 4th Int. Conf. Commun. Inf. Electron. Energy Syst. (CIEES), Plovdiv, Bulgaria, 2023, pp. 1–6. https://doi.org/10.1109/CIEES58940.2023.10378822.
[28] M. Dawood, S. Tu, C. Xiao, H. Alasmary, M. Waqas, S.U. Rehman, Cyberattacks and security of cloud computing: A complete guideline, Symmetry 15 (2023) 1981. https://doi.org/10.3390/sym15111981.
[29] M.V. Kushala, B.S. Shylaja, Recent trends on security issues in multi-cloud computing: A survey, in: Proc. Int. Conf. Smart Electron. Commun. (ICOSEC), Trichy, India, 2020, pp. 777–781.
[30] M. Assuncao, R. Calheiros, S. Bianchi, M. Netto, R. Buyya, Big Data computing and clouds: Trends and future directions, J. Parallel Distrib. Comput. (2014).
[31] M. Andronie, et al., Big data management algorithms, deep learning-based object detection technologies, and geospatial simulation and sensor fusion tools in the internet of robotic things, ISPRS Int. J. Geo-Inf. 12(2) (2023) 35.
[32] J. Zhang, et al., Hybrid edge-cloud collaborator resource scheduling approach based on deep reinforcement learning and multi-objective optimization, IEEE Trans. Comput. (2023).
[33] S.U. Mushtaq, S. Sheikh, S.M. Idrees, et al., In-depth analysis of fault-tolerant approaches integrated with load balancing and task scheduling, Peer-to-Peer Netw. Appl. 17 (2024) 4303–4337. https://doi.org/10.1007/s12083-024-01798-5.
[34] R.T. Venkatesh, D.K. Chandrashekar, P.B.S. Rao, R. Sridhar, S. Rajanna, Systematic approaches to data placement, replication and migration in heterogeneous edge-cloud computing systems: A comprehensive literature review, Ingénierie des Systèmes d'Information 28(3) (2023) 751–759. https://doi.org/10.18280/isi.280326.
[35] V. Nivethitha, G. Aghil, Improved Harris Hawks Optimization Algorithm based data placement strategy for integrated cloud and edge computing, Intell. Autom. Soft Comput. 37(1) (2023) 887–904.
[36] Y. Zhang, B. Pranggono, H. Tianfield, A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers, Future Gener. Comput. Syst. 115 (2021) 1–15. https://doi.org/10.1016/j.future.2020.08.036.
[37] H.H. Essam, A.G. Gad, Y.M. Wazery, P.N. Suganthan, Task scheduling in cloud computing based on metaheuristics: Review, taxonomy, open challenges, and future trends, Swarm Evol. Comput. 62 (2021) 100841.
[38] M.N. Aktan, H. Bulut, Metaheuristic task scheduling algorithms for cloud computing environments, Concurrency Comput. 34(9) (2022) e6513.
[39] B. Prabhu Shankar, H. Najmusher, N. Rajkumar, R. Jayavadivel, C. Viji, S.M. Nandha Gopal, Nature-inspired data placement strategy in distributed cloud environment using improved firefly algorithm, J. Cloud Comput. (2022).
[40] A. Faramarzi, M. Heidarinejad, S. Mirjalili, A.H. Gandomi, Marine Predators Algorithm: A nature-inspired metaheuristic, Expert Syst. Appl. 152 (2020) 113377. https://doi.org/10.1016/j.eswa.2020.113377.
[41] E.H. Houssein, A.G. Gad, Y.M. Wazery, P.N. Suganthan, Task scheduling in cloud computing based on metaheuristics: Review, taxonomy, open challenges, and future trends, Swarm Evol. Comput. 62 (2021) 100841.
[42] D. Alboaneen, H. Tianfield, Y. Zhang, B. Pranggono, A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers, Future Gener. Comput. Syst. 115 (2021) 201–212.
[43] S. Rajashekar, P. Channakrishnaraju, V.K. Poornachary, SCEHO-IPSO: A nature-inspired metaheuristic optimization for task-scheduling policy in cloud computing, Appl. Sci. 13(19) (2023) 10850. https://doi.org/10.3390/app131910850.
[44] T. Hai, J. Zhou, D. Jawawi, Task scheduling in cloud environment: Optimization, security prioritization, and processor selection schemes, J. Cloud Comput. 12(15) (2023) 1–19. https://doi.org/10.1186/s13677-022-00374-7.
[45] H.S. Yahia, et al., Comprehensive survey for cloud computing-based nature-inspired algorithms optimization scheduling, Asian J. Res. Comput. Sci. 8(2) (2021) 1–16.
[46] M.I. Alghamdi, Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO), Sustainability 14 (2022) 11982. https://doi.org/10.3390/su141911982.
[47] M.E. Megahed, R.M. Ismail, N.L. Badr, M.F. Tolba, An enhanced cloud-based view materialization approach for peer-to-peer architecture, in: Multimedia Forensics and Security, Cham, 2017, pp. 77–95. https://doi.org/10.1007/978-3-319-44270-9_4.
[48] Q. Xu, Z. Xu, T. Wang, A data-placement strategy based on genetic algorithm in cloud computing, Int. J. Intell. Sci. 5(3) (2015) 145–157.
[49] A.W. Mohamed, A.A. Hadi, A.K. Mohamed, Gaining-sharing knowledge-based algorithm for solving optimization problems: A novel nature-inspired algorithm, Int. J. Mach. Learn. Cybern. 11 (2020) 1501–1529. https://doi.org/10.1007/s13042-019-01053-x.
[50] T. Mahalingam, S. Monikavasagom, A hybrid gray wolf and genetic whale optimization algorithm for efficient moving object analysis, Multimed. Tools Appl. 78 (2019). https://doi.org/10.1007/s11042-019-07768-x.

[51] A. Tummala, War Strategy Optimization Algorithm: A new effective metaheuristic algorithm for global optimization, IEEE Access 10 (2022). https://doi.org/10.1109/ACCESS.2022.3153493.
[52] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exp. 41(1) (2011) 23–50. https://doi.org/10.1002/spe.995.
[53] G.S. Panesar, R. Chadha, A hybrid optimization algorithm for efficient virtual machine migration and task scheduling using a cloud-based adaptive multi-agent deep deterministic policy gradient technique, Int. J. Intell. Syst. Appl. Eng. 12(6s) (2023) 30–45.
[54] Y. Zhang, J. Wang, Enhanced Whale Optimization Algorithm for task scheduling in cloud computing environments, J. Eng. Appl. Sci. 71 (2024) 121. https://doi.org/10.1186/s44147-024-00445-3.
[55] J. Ababneh, A Hybrid Approach Based on Grey Wolf and Whale Optimization Algorithms for Solving Cloud Task Scheduling Problem, Math. Probl. Eng. (2021).