Shahid Bahonar University of Kerman
Journal of Mahani Mathematical Research
Print ISSN: 2251-7952 Online ISSN: 2645-4505

APPLICATION OF PREDATOR-PREY OPTIMIZATION FOR TASK SCHEDULING IN CLOUD COMPUTING

Z. Jalali Khalil Abadi, B. Mohammad Hasani Zade, N. Mansouri ✉, and M.M. Javidi

Article type: Research Article
(Received: 23 January 2024, Received in revised form 02 October 2024)
(Accepted: 01 November 2024, Published Online: 25 November 2024)

Abstract. Cloud computing environments require scheduling to allocate resources efficiently and ensure optimal performance. It is possible to maximize resource utilization and minimize execution time by scheduling cloud systems effectively. Meta-heuristic algorithms aim to address this NP-hard problem by taking into account these QoS parameters. In order to deal with the task scheduling problem, we utilize a new meta-heuristic algorithm known as Predator-Prey Optimization (PPO). In PPO, predators and preys are modeled and their energy gains are determined by their body mass and interactions. Faster convergence rates enhance PPO's ability to find optimal solutions. The balance between exploration and exploitation makes it suitable for solving real-world problems in unknown spaces. The PPO-based Task Scheduling algorithm (PPOTS) has the goal of reducing execution time and makespan while increasing resource utilization. In this study, the PPOTS algorithm is compared to five well-known meta-heuristic algorithms: Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), Spotted Hyena Optimization Algorithm (SHO), Grasshopper Optimization Algorithm (GOA), and Sooty Tern Optimization Algorithm (STOA). Furthermore, the proposed PPOTS algorithm was compared with two new meta-heuristic based scheduling algorithms, and showed a better performance than the other two algorithms. Resource utilization and execution cost are enhanced by 8% and 15%, respectively, through the proposed method.

Keywords: Cloud Computing, Task Scheduling, Predator-Prey Optimization, Meta-heuristic.
2020 MSC: 68Qxx.

1. Introduction

Despite the many advantages of cloud computing, it is not always a good choice because it is slow to respond to existing requests. As a result, fog computing is more effective. In a fog environment, task scheduling is challenging. As Internet of Things (IoT) clients, it is important that tasks are executed on time and at lower costs; however, they also require that tasks are executed

✉ najme.mansouri@gmail.com, ORCID: 0000-0002-1928-5566
https://doi.org/10.22103/jmmr.2024.22855.1571
Publisher: Shahid Bahonar University of Kerman
How to cite: Z. Jalali Khalil Abadi, B. Mohammad Hasani Zade, N. Mansouri, M.M. Javidi, *Application of predator-prey optimization for task scheduling in cloud computing*, J. Mahani Math. Res. 2025; 14(1): 441-472.

securely. In addition to generating large amounts of data, rapid IoT deployment also requires transferring that data to the cloud. In order to solve this challenge, fog computing has been designed.

With fog and edge computing, users can be aware of their location and adjust their mobility. The data can also be stored and processed close to the point of use, which can overcome the limitations of cloud applications. In addition to reducing cloud data transfers, fog technology improves IoT performance by reducing the amount of data needed for processing, storing, and analyzing. As a result, data from the edge is temporarily stored and processed rather than transferred to the cloud, which reduces network traffic and delay. There are fog nodes that manage storage, computation, and networking [1]. As a distributed computing approach, fog and cloud are not integrated [2, 3]. Fog technology allows cloud latency to be reduced by using unused resources from nearby devices to overcome the latency issue. The cloud handles most tasks, however. Fog computing consists of using devices near the clients that have limited features, but high computing power, as opposed to cloud computing.

It is fundamental to distributed systems to utilize Load Balancing (LB), which is a technique that optimizes the resources available to VMs (Virtual Machines). Whenever workloads are dynamically distributed and resources are optimally utilized, load balancing plays an integral role. An efficient workload balance results in happier users and more efficient resource allocation. Load balancing reduces data transmission times and prevents unstable Quality of Service (QoS) situations in data centers caused by overloaded systems.

With the expansion of IoT in the digital world and the growth of real-time applications, equal distribution of workload has become increasingly important in the fog environment. By using load balancing, users are able to utilize resources more efficiently and experience greater levels of satisfaction. It will result in improved performance and resource utilization for the system. The result is a reduction in overuse and underutilization of resources. It is possible to reduce a system's overall operating costs by using a distributed workload. The user requests are continually passed to cloud-based fog architectures in bulk amounts to take full advantage of fog nodes [4]. The load balancer distributes tasks among all the processing nodes at the fog layer for execution by IoT devices. With load balancers, workload is distributed among multiple servers based on client requests, requests are routed only to available servers, and the server capacity can be adjusted according to demand.

The use of metaheuristic algorithms has received increasing attention in recent years for solving complex optimization problems. This is due to their high effectiveness and ability to find approximate optimal solutions in polynomial rather than exponential time, unlike conventional methods. Numerous metaheuristics have been employed to solve load balancing problems across a wide variety of fields, including cloud computing and fog computing. By utilizing natural methods, meta heuristic algorithms are striving to find the optimal location across the search space based on the global optimal solution. Cloud and

fog environments are dynamic, and metaheuristic algorithms are highly efficient in handling them. Nature- and population-based stochastic optimization techniques are the most popular, focusing on nonlinear solutions to complex problems.

Most algorithms are based on biological or physical phenomena, and often on strategies used by creatures to solve problems. In terms of nature-inspired algorithms, PPO (Predator-Prey Optimization) is an efficient algorithm that models the interaction between the predator and the prey by starting with two populations. After eating their prey, predators regain their lost energy and gain positive energy by searching for, catching, and handling their prey that requires little energy [5].

Improved Wild Horse Optimization (IWHO) algorithm was developed by Saravanan et al. [6] to address long scheduling time, high-cost consumption, and high virtual machine load in cloud computing. Firstly, IWHOLF-TSC constructs a cloud computing task scheduling and distribution model. Secondly, the best feasible plan for cloud computing task scheduling includes finding the best whale individual. Improved whale optimization algorithm employs an inertial weight strategy in order to improve local search capability. In order to increase the exploration ability of wild horse optimization and avoid premature convergence of the algorithm, the Levy Flight method is used.

Behera and Sobhanayak [7] proposed a hybrid algorithm combining Grey Wolf Optimization Algorithm (GWO) and Genetic Algorithm (GA). They designed a multi-objective cloud computing task scheduling algorithm with the goal of minimizing time, energy, and cost. Enhanced crossover operators are embedded in standard GA to improve its exploitation ability. In addition, the GA-based GWO algorithm can solve large scheduling problems more quickly. Cloudsim toolkit evaluations demonstrate the proposed algorithm's efficiency compared to existing approaches. In addition, it produces energy savings of 17%, 19%, and 23% over GWO, GA, and PSO.

Several researchers consider the task scheduling problem as a multiobjective problem, meaning that more than one objective affects the task scheduling. Many of them combine objectives into a linear weighted function, however some solutions have been ignored due to the non-convex nature of the problem.

According to Saif et al. [8], a multi-objective algorithm is used to find the best solution to the problem of task scheduling. Delay and energy consumption are two main objectives. According to the fitness function, the solution is chosen according to the minimum delay and energy consumption in the Multi-Objective Grey Wolf Optimizer (MGWO). Thus, the value of the fitness function is important for the proposed algorithm. MOP must improve multiple conflicting objectives simultaneously. A fitness function cannot be used to compare solutions in MOPs. In each iteration, the ideal solution is to implement Pareto dominance in order to choose the optimal solution.

This paper continues as follows. Section 2 presents the background of cloud computing, task scheduling, and PPO. Section 3 discusses existing methods

for task scheduling in cloud environments. Section 4 describes the PPOTS algorithm. Section 5 evaluates the performance of the PPOTS algorithm. In section 6, the conclusion and possible future works are discussed. Table 1 provides the full names of all the abbreviations used in the article.

—l—l—

Abbreviation	Full Name
IoT	Internet of Things
LB	Load Balancing
VMs	Virtual Machines
QoS	Quality of Service
PPO	Predator-Prey Optimization
PPOTS	PPO-based Task Scheduling algorithm
SaaS	Software-as-a-Services
IaaS	Infrastructure-as-a-Services
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
ETC	Expected Time to Compute
GWO	Grey Wolf Optimization
PPBACO	Performance and Budget-based Ant Colony Optimization
WOA	Whale Optimization Algorithm
COBL	Comprehensive Opposition-Based Learning
HGSO	Henry Gas Solubility Optimization
EG	Energy Gain
ABC	Artificial Bee Colony
DA	Dragonfly Algorithm
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
ACO	Ant Colony Optimization

FCFS	First Come First Serve
IWHO	Improved Wild Horse Optimization
MGWO	Multi-Objective Grey Wolf Optimizer
RU	Resource Utilization
SSA	Salp Swarm Algorithm
SHO	Spotted Hyena Optimization
GOA	Grasshopper Optimization Algorithm
STOA	Sooty Tern Optimization Algorithm
Q-ACOA	Q-based Ant Colony Optimization Algorithm
ACOA	Ant Colony Optimization Algorithm
OACT	Optimal Average Completion Time
HAGA	Hybrid Ant Genetic Algorithm
EC	Execution Cost
SO	Snake Optimization Algorithm
EWOA	Enhanced Whale Optimization Algorithm

2. Background

2.1. **Cloud computing.** Cloud Computing is a new idea that provides services and resources over the Internet to a large number of users on an as-needed basis as a result of technological advances [9]. Furthermore, cloud computing provides unlimited computing resources, including servers, storage, networks, and applications, that are geographically distributed [10, 11]. With a pay as you go model, cloud consumers can request resources at any time and from any location based on their needs. As a result of cloud computing, a variety of applications can be developed and maintained through access-based computing infrastructure [12]. It also utilizes internet computer resources instead of local computers for storing and processing data. Multiple computers are used at several locations for a simultaneous project being run by a computer group. A distributed analytics system that performs time-consuming data analysis is made more efficient by this distributed work [13]. The services offered include Software-as-a-Services (SaaS), Infrastructure-as-a-Services (IaaS), and Platform-as-a-Services (PaaS).

2.2. **Task scheduling.** An internet of everything (IoE) application generates a vast number of tasks with variable lengths, which need to be prioritized for

execution. The heterogeneous and resource-limited end devices in the network are difficult to manage. Resources must be shared among heterogeneous devices within heterogeneous environments. For this reason, agents should be assigned in accordance with their resource demands so that jobs will be executed appropriately [14]. The assignment of tasks and computing resources within the cloud environment is fundamentally determined by task scheduling in cloud computing. Task scheduling ensures optimal resource utilization, minimizes the makespan, and improves user service quality. Through efficient resource allocation and minimizing idle time, scheduling helps reduce energy consumption. An optimal task schedule improves system performance, reduces response times, and increases throughput. Task scheduling is directly related to cloud performance, resource utilization, and user satisfaction [15]. As a result of fog computing, which comes with the help of cloud computing, communication between IoT devices can be sped up and response times can be reduced [10].

The following are several key challenges related to fog computing for task scheduling:

- Heterogeneity of Devices: There are a variety of end devices that are limited in resources in fog environments. It is challenging to manage these heterogeneous devices and ensure efficient resource sharing.
- Scalability: Load balancing and task scheduling become more complicated as the number of devices increases. In order to handle this growth effectively, scheduling algorithms must be scalable.
- Latency and Real-Time Processing: The goal of fog computing is to reduce latency by processing data closer to its source. The ability to execute tasks quickly and in real-time is a crucial challenge for IoT applications, which generate vast amounts of data.
- Security: IoT devices are rapidly being deployed, so ensuring that tasks are executed securely is a major challenge. The security and privacy of data must be maintained across distributed fog nodes.
- Resource Management: It is difficult to manage and allocate resources efficiently in fog environments due to their dynamic nature. In order to accomplish this, resource utilization must be optimized while energy consumption and costs should be minimized.
- Cost Efficiency: It is essential to balance execution costs with the performance requirements of IoT clients. Resource utilization and task execution are also optimized.

Figure 1 illustrates how task scheduling works in a cloud computing environment. The following summary summarizes different workload balancing metrics proposed in various literatures [16]:

- Throughput: This metric is used to calculate the process completion rate.
- Cost: Cost is used by cloud providers to provide the lowest rate to all customers. Customers can only use software or services.

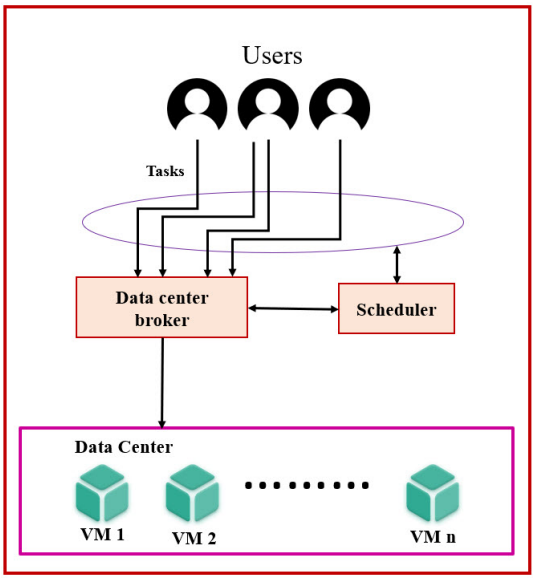


FIGURE 1. Cloud task scheduling model.

- Response time: A task's execution time is measured by this parameter.
- Execution time: In general, the execution time of a process depends on the input data, but it does not depend on when it was initiated.
- Makespan: Calculates how long it will take the user to complete a task or to get access to the resources.
- Energy consumption: Each node's energy consumption is calculated. Using load balancing, each node is equally loaded, which avoids overheating and reduces energy consumption.
- Scalability: When the number of nodes in the system increases, load balancing algorithms are capable of performing uniform load balancing as required among the nodes. Highly scalable algorithms are preferred.
- Migration time: Upon becoming overloaded, a node must be transferred to a node which is currently underloaded.

2.3. **Predator-Prey Optimization (PPO).** For estimating predator-prey interactions, PPO uses a mechanistic approach based on the obtained features at individual levels [5]. As a result of this method, predation is implied by motion by essence. A predator moves to snag a prey, and the prey must escape after being snatched.

Hunting involves three steps:

(1) It uses cyclic methods to locate prey. Predators consume energy to hover against their own weight. Archimedes' force and inertia play a crucial role in shaping movement. Due to their detection distance and

Algorithm 1 The pseudo-code of PPO algorithm

Input: Population of predators and prey, number of iterations, algorithm parameters (predation rate, reproduction rate, etc.), fitness function **Output:** The best solution (best-solution and best-fitness)

```
Begin
1.  best-solution = None;
2.  best-fitness = Infinity;
3.  For each iteration:
4.      Evaluate fitness of each predator and prey
5.      Sort predators and prey based on fitness values
6.      For each predator: // Predator Phase
7.          Select a prey based on fitness proximity
8.          Calculate new position of predator using prey's position
9.      End for
10.     Update predator's position if new position is better
11.     For each prey: // Prey Phase
12.         Calculate new position using random walk and predator avoidance
13.         Update prey's position if new position is better
14.     End for
15.     Select top-performing predators and prey to reproduce // Reproduction Phase
16.     Generate offspring to replace the worst-performing individuals
17.     current-best-solution = get-best-solution (population) // Update best solution found
18.     current-best-fitness = fitness-function (current-best-solution);
19.     If current-best-fitness < best-fitness:
20.         best-solution = current-best-solution;
21.         best-fitness = current-best-fitness;
22.     End If
23.     Adjust predation and reproduction rates dynamically // Update algorithm parameters if needed
24. End for
25.  Return best-solution, best-fitness // Output the best solution found
End
```

FIGURE 2. The pseudo-code for PPO algorithm.

abundance of prey, predators detect prey at a distance and encounter them.

(2) Successful encounters can lead to sequences. A prey moves toward a trap and then attempts to escape. Predators must keep buoyancy while handling prey by lifting themselves and the prey during handling.

(3) During eating and mechanical movements, a predator maintains its position in the water. In this case, the procedure is called handling.

According to Figure 2 and Figure 3, PPO's algorithm can be categorized into seven components based on its functions. So, we have: (1) Population definition that generates solutions, (2) Calculating prey and predator mass based on their fitness, (3) Calculation of search costs, (4) Catching, (5) Handling, (6) Calculating energy, and (7) Attempts to replace the worst solution with the best. By repeating heuristic searches, the desired solution will emerge.

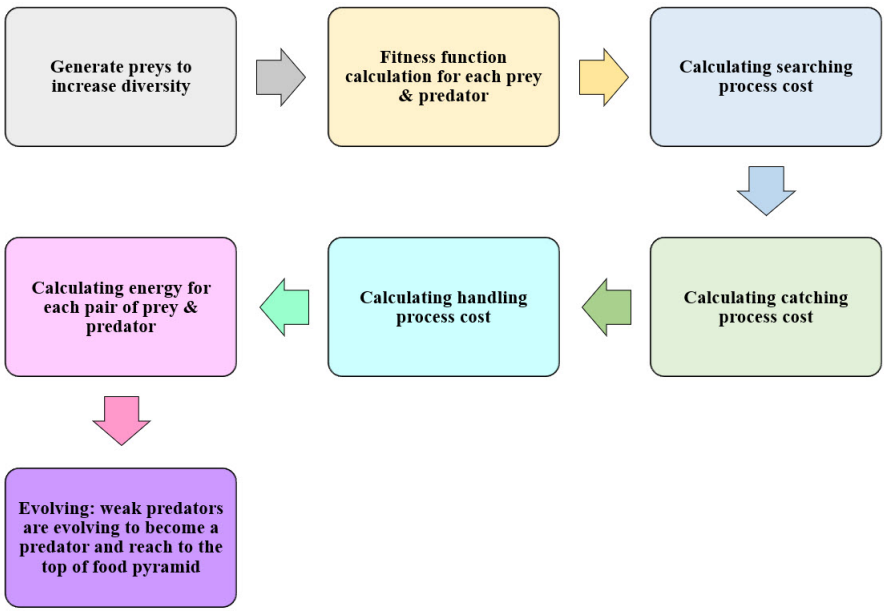


FIGURE 3. PPO main steps.

Due to the new parameter (e.g., energy gain), the PPO algorithm can find the optimal solution faster, helping to perform global and local searches. The PPO algorithm has also been applied to solving the problem of identifying features. Furthermore, PPO's performance for feature selection issues shows that it can solve real problems with unknown search spaces.

3. Related Work

Scheduling tasks involves assigning the tasks submitted by users to the available virtual machines (VMs) to ensure the least amount of response time. Scheduling tasks in cloud computing environments is an NP-Hard problem [17]. Many serious efforts have been made to find a solution to this problem, and as cloud computing is a relatively new field, there is still room for further research. The following is a brief overview of what has been accomplished.

The Particle Swarm Optimization (PSO) algorithm is used by Pradhan and Satapathy [18]. Therefore, a solution with efficient QoS parameters such as makespan, cloud utilization, and energy consumption was selected. Application models represent groups of tasks. A schedule is then compiled and a queue is created. Tasks have different properties, such as execution time and completion time. This model estimates the execution time of the application. Tasks are mapped to machines/clouds using a matrix called Expected Time to Compute (ETC). PSO scheduling method was used to develop a new energy model that

estimated total energy consumption. At each stage of the process, a fitness function is evaluated in order to identify the best solution. As a result of the proposed algorithm, the overall time and energy consumption will be optimized.

According to Indhumathi et al. [19], task scheduling is used to address the issue of workload. In this work, task execution is achieved with the help of grey wolf optimization (GWO). Additionally, fault detection and rescheduling of failed tasks are done simultaneously. For the purpose of analyzing the proposed framework, it is executed on JAVA. In order to perform tasks effectively and without lag, they must be scheduled. A two-stage process is used to implement the proposed system. The first step is task scheduling, followed by fault tolerance. Task scheduling begins with the user providing the task. Tasks are sent to the VM, which is referred to as a processing unit. After that, VM bandwidth is calculated. Estimating bandwidth is based on task priority. Performance optimization is used in this algorithm to select VMs. Using an optimization algorithm called grey wolf, the proposed work will select the virtual machine. This approach will be used to complete all tasks requested by the user. It is determined when the task will start and how long it will take to finish. The proposed approach allows resource allocation to be done effectively. The second stage of the process is the approach of fault tolerance after task scheduling is completed. The fault tolerance mechanism identifies the failed task and reschedules it. This proposed work aims to schedule tasks in a way that overcomes the workload issue. In this proposal, load balancing is also accomplished along with task scheduling.

According to Prem Jacob and Pradeep [20], two optimization algorithms have been combined into one, known as CPSO. In addition to reducing deadline violations and their costs, the proposed algorithm also reduces their duration. A local resource manager manages and monitors cloud computing environments. Resource costs are calculated by collecting CPU, memory, and individual task execution time data from nodes and sending it periodically to the global resource manager. To ensure efficient scheduling, task managers and resource managers provide schedulers with task and resource information. As a result of the algorithms matching the tasks with the appropriate resources according to deadlines and costs, the tasks are completed on time. Utilizing CloudSim 3.0 toolkit, the CPSO algorithm is evaluated. In comparison to PBACO (Performance and Budget-based Ant Colony Optimization), ACO (Ant Colony Optimization), MIN-MIN, and FCFS (First Come First Serve), the proposed work minimizes the makespan, cost, and deadline violation rate.

In order to meet end-users' dynamic requirements, advanced scheduling techniques must be in place to map tasks to cloud resources optimally. The whale optimization algorithm (WOA) is modified by Abd Elaziz and Attiya [21] to a comprehensive opposition-based learning algorithm for optimizing task scheduling. The method is called HGSWC. HGSWC improves solutions through WOA, while COBL (Comprehensive Opposition-Based Learning) improves worst solutions through computing their opposites. In contrast to conventional HGSO

and WOA, HGSWC is validated on 36 optimization benchmark functions. HGSWC can produce optimal task schedules with better makespan for all test instances, according to the results.

Huang et al. [22] presented a task scheduler for cloud computing using several discrete variants of particle swarm optimization. Population-based swarm intelligence algorithms involve two phases of optimization: global and local. When the two phases are balanced correctly, the algorithm can converge to the global best position. A global search operator is better than a local search operator for any algorithm. A global optimum can only be achieved by exploring more search areas. This balance is maintained by at least one coefficient in each algorithm. Inertia weight (w) is the only coefficient in PSO algorithms that plays this role. The inertia weight should be gradually reduced over iteration rather than a constant value being used. Providers of cloud computing services will receive requests for services. This means the cloud provider must establish a schedule that is optimal for the workflow application request made by the user. Based on the optimized schedule, the system's makespan can be minimized by choosing the optimal execution nodes (VMs). Comparing PSO-based schedulers to GSA (Gravitational Search Algorithm), ABC (Artificial Bee Colony), and DA (Dragonfly Algorithm), the use of logarithm decreasing strategies reduces average makespan by 19.12%, 2.1.42%, and 15.14%, respectively.

In Vijarania et al. [23], a load balance-aware task scheduling policy was proposed. Chromosomes are encoded using direct encoding methods. A chromosome's length is determined by its number of tasks, and its genes represent the resources used by the tasks. Chromosome fitness is measured by the fitness function. An individual's fitness value reflects its performance within a population. In determining whether a chromosome will move to the next generation, the selection operator evaluates the performance of each individual chromosome. In proportion selection method, chromosomes are selected based on their fitness values. Crossover operators select two chromosomes, then select their intersection point. There is an exchange of two chromosomes at the intersection point, which results in two new offspring. An intersection point is chosen using a single-point crossover operator that generates new offspring at random. By using the presented mechanism, costs and completion times are significantly reduced, and better results are obtained.

The Q-ACOA (Ant Colony Optimization Algorithm) algorithm is proposed by Su et al. [24] for resource allocation and task scheduling in cloud computing in order to analyze the current problems, expected time, and expected cost. The time taken to complete tasks in cloud computing, the data migration time, the cost of completing tasks, and user satisfaction are the evaluation indicators. Task scheduling systems in cloud computing should guarantee that the completion time for each task is no longer than the expected deadline of users, thus meeting QoS of time. Developed on the basis of improvements to the law of probability transition, the law of pheromone initialization, and the

law of pheromone update, the ACOA (Ant Colony Optimization Algorithm) is optimized. During optimization, pheromone size corresponds to the degree of mapping between the corresponding task and the data migration. When selecting the next node or direction, the probability transition rule is reflected. After optimization, the algorithm selects a global update for pheromone updates, and finally returns the optimal solution. Accordingly, Q-ACOA adjusts pheromone increments under comprehensive consideration of data migration time by normalizing task completion time and cost consumption. Cloud computing can benefit from the improved ACOA (Q-ACOA) when it comes to resource allocation and task scheduling.

As a part of their heuristic-based task scheduling approach, Tripathi and Kumar [25] designed an efficient load balancing mechanism for heterogeneous environments, which is intended to enhance both user and provider quality of service. As the threshold value, the method uses the Optimal Average Completion Time (OACT). The first VM is mapped using a circular FCFS manner as long as its completion time does not exceed the OACT. In order to minimize load imbalances, task loads are evenly distributed across VMs. This optimizes resource utilization and reduces the makespan (maximum completion time) for each VM. Additionally, this method reduces user processing costs and average waiting times. Experiments were conducted with a variety of task streams and VMs of varying lengths and speeds to ensure the proposed strategy was effective. Compared with existing scheduling policies like Round Robin, Conductance algorithm, and shortest job first, the proposed strategy improves QoS parameters and fulfills both users' and providers' requirements.

Krishnan and Rajalakshmi [26] presented a cost-optimized scheduling algorithm for multi-core data parallel tasks. Parallel executions can be performed on a multicore resource, which increases the number of parallel executions, thereby meeting the deadline. This paper develops a model to optimize data parallel task operational costs by assigning load fractions to multi-core resources. Data parallel tasks were explored in this work. In terms of processing tasks by deadline at an optimized cost, the work results in better solutions.

Task scheduling is an essential component of cloud computing to improve throughput, response time, energy consumption, and resource utilization. Task scheduling difficulties can be effectively solved by bio-inspired algorithms, but they need a lot of computational power and time because of cloud workload and complexity. Ajmal et al. [27] proposed the Hybrid Ant Genetic Algorithm (HAGA). This algorithm combines features of genetic algorithms and ant colony algorithms. The virtual machines are pheromoned after tasks are assigned. Detecting loaded virtual machines and dividing tasks into groups reduces solution space effectively. This algorithm has a small solution space, resulting in reduced convergence and response times. This solution minimizes the running time of workflows and tasks.

Table 2 compares the discussed scheduling algorithms. Most algorithms presented in Table 2 are based on reducing makespan or cost, which have a significant impact on cloud system efficiency. In this paper, we present a task scheduling algorithm that takes into account makespan, resource utilization, and execution cost parameters. —p2cm—p0.6cm—p1.8cm—p1.3cm—p1.6cm—p1.8cm—p2.2cm—

Reference	Year	Performance metrics	Simulator	Technique	Advantages	Disadvantages
Pradhan and Satapathy [18]	2023	Makespan, Energy consumption, Cloud utilization	Not reported	PSO-based task scheduling algorithm	Achieves an overall improvement in the makespan and generates cloud utilization and average energy consumption on cloud systems.	Advanced evolutionary scheduling strategies have not been analyzed with the proposed algorithm.
Indhumathi et al. [19]	2023	-Makespan, Execution time, Communication delay, Computational delay, Failure, Utilization rate	JAVA on CloudSim, JDK7.0, and Eclipse	Task scheduling and fault tolerance mechanism based on GWO	Task execution is more efficient with a lower failure rate and a higher throughput.	There is no load balancing.
Prem Jacob and Pradeep [20]	2019	Makespan, Cost, Deadline violation rate	CloudSim	Combine Cuckoo Search (CS) and Particle Swarm Optimization (PSO)	Performance and cost optimization of scheduling.	In addition, other QoS parameters weren't considered.
Abd Elaziz and Attiya [21]	2021	Makespan	CloudSim	HGSO based on WOA and COBL	Optimizes task scheduling for cloud computing resources.	Due to its higher number of control parameters, HGSWC may have less flexibility when implementing.
Huang et al. [22]	2020	Makespan	MATLAB 2014a	PSO-based scheduler	It reduces the makespan significantly compared to other algorithms.	Load balancing and energy consumption were not considered in this method.
Vijarania et al. [23]	2021	Cost, Completion time, Load of individual VM	MATLAB	GA-based task scheduling algorithm	Significant savings are made in terms of cost and time.	Other factors, such as time span, task completion time, and load balancing, were not taken into account.
Su et al. [24]	2021	Migration, Cost, Resource allocation, Completion time	CloudSim	Q-ACOA	In terms of task completion time, total data migration time, cost consumption, and user satisfaction, Q-ACOA performs the best.	For various reasons, correlation analysis is not performed. Task scheduling and resource allocation in cloud computing will be made easier by considering the correlation between tasks.
Tripathi and Kumar [25]	2022	Makespan, Resource utilization, Cost, Waiting time	CloudSim	Heuristic-based Task Scheduling Policy	Task loads are distributed across VMs effectively, reducing imbalance.	Complex to understand and implement. Calculating optimal completion times may introduce computational overhead. Its scalability with increasing tasks and heterogeneous resources.
Krishnan and Rajalakshmi [26]	2022	Cost	Alibaba Cluster Data V2017	Cost-optimized data parallel task scheduling in multi-core resources	Optimum load fractions are assigned to multicore resources to reduce operational costs.	This approach may be limited to certain types of parallel data processing.
Ajmal et al. [27]	2021	Execution time, Cost	CloudSim	Hybrid Ant Genetic Algorithm (HAGA)	Optimizes the use of resources, reducing waste and improving overall performance.	

Suitable for heterogeneous environments, this approach is versatile for cloud computing. Additional computation is involved in calculating the optimal average completion time (OACT) and managing the task distribution. A growing number of tasks and VMs may cause scalability issues.

The proposed method 2024 Execution cost, Resource utilization, Makespan MATLAB PPOTS Modeling predators and preys enables it to mimic their energy gain based on their body mass and interactions, resulting in an effective optimization process. In comparison to other meta-heuristic algorithms, PPOTS achieves faster convergence rates. It is important to consider security, scalability, load balancing, and availability.

4. **PPO-based Task Scheduling (PPOTS) Algorithm**

A simple definition of the task scheduling problem is provided, and then we explain how the problem is formulated. The task scheduling problem concepts are explained in Section 4.1, the objective functions are stated in Section 4.1.2, the PPO parameters explained in section 4.1.3, and finally, the upates of process is explained in Section 4.1.4.

Task scheduling in cloud and fog computing environments are comprehensively analyzed and innovatively discussed in this article. The following are some of the key contributions:

- Innovative Use of Metaheuristic Algorithms: Predator-Prey Optimization (PPO), in particular, is shown to be effective when solving complex optimization problems in cloud and fog computing. In dynamic environments, this algorithm provides approximate optimal solutions efficiently.
- Multi-objective Function: The proposed approach optimized makespan, execution cost, and resource utilization using a multi-objective objective function.
- Detailed Performance Evaluation: The article evaluates the PPOTS algorithm thoroughly, comparing it with existing methods, and demonstrating its effectiveness in terms of makespan, resource utilization, and execution cost. Analysis of the proposed solutions in detail helps understand their practical benefits.

4.1. **The model of task scheduling.** All tasks assigned among available VMs based on user needs and service quality. The main aim of this work is to increase the QoS metrics performance in cloud computing. Suppose that a cloud datacenter contains n task and m VM: $T = \{T_1, T_2, \ldots, T_n\}$, where T_i means the $i-th$ task and $VM = \{VM_1, VM_2, \ldots, VM_m\}$, where VM_j means the $j-th$ VM in the cloud environment, but the condition for execution of such tasks is: $n > m$.

In order to maximize virtual machine group utilization, cloud service providers strive to reduce the wait time and the makespan between virtual machine groups. Makespan can be minimized by assigning the set of jobs or tasks to a set of virtual machines, where the order of execution of the tasks or jobs

is irrelevant. Therefore, a lower value of makespan indicates a more efficient scheduling algorithm. Service and instance costs are determined by the type of service or instance and charged at a time-based rate. Costs are determined by the amount that the user must pay to the provider of the service. It is determined by the scheduling algorithm which VM will have the lowest execution cost when executing tasks. The algorithm provides QoS while reducing cloud computing costs by placing as many services as possible on virtual machines. In resource utilization, how much of the available resources is being consumed at any one time. To ensure the method is as productive as possible, we use it to plan how to utilize the resources more effectively. Multiple resource categories are able to be predicted by effective resource utilization. This will help you avoid reworking schedules and assigning tasks at the beginning of planning. Assigning employees to the project will also be possible.

According to the above description, Figure 4 shows a pseudocode of task scheduling based on the PPO method. The Figure 5 illustrates the obj (objective function) used by Algorithm 2 in Figure 4. Figure 6 shows the flowchart for the PPOTS algorithm. The algorithm focuses on three important criteria in the objective function: makespan, resource utilization, and execution cost.

Algorithm 2 Pseudo-code for PPOTS algorithm

Input: Tasks set, VMs set, the parameters of PPOTS algorithm, N (population size)
Output: Return best search
Begin
1. Initialize set of tasks, $T = \{T_1, T_2, \ldots, T_n\}$.
2. Initialize set VMs, $VM = \{VM_1, VM_2, \ldots, VM_m\}$.
3. Set the population size and maximum iteration.
4. Set the parameters of PPO.
5. [Prey, Predator] = initialization (N, T, VM). // Prey and Predator production process in 4.2.1
6. $t = 1$
7. **While** ($tmaximumiteration$)
8. **For** $i = 1$ to N
9. $[O_i]$= fobj ($Prey_i$, $Predator_i$); // using Algorithm 3.
10. Energy (i) = E_C ($Prey$, $Predator$, O); // Calculate energy gain for hunting process (search, hunting, and handling)
11. Find the best search agent;
12. **End** for
13. $t++$;
14. **End** while
15. **Return** best solution // The allocation matrix which minimizes the objective function.
End

FIGURE 4. Pseudo-code for PPOTS algorithm.

4.1.1. *Initialization.* The aim of PPOTS is to schedule all tasks to the available VMs which minimize makespan, resource utilization, and execution cost so that

Algorithm 3 Objective Function (fobj) Pseudocode

Input: Task, VM, Joblen (task information: task length and task ID), St (waiting time).
Output: Optimal solution (O), Makespan, Resource utilization (RU), Execution cost (EC).
Begin
1. For each Task
2. Execution time = (Joblen/VM size);
3. End for
4. For each VM
5. EC = ((Execution time/3600) * VM price);
6. Task processing time = (Joblen/(VM size * CPU utilization)) ;
7. End for
8. EC_mean = sum (sum (EC)); // Sum all of VMs excution cost.
9. Makespan = min (sum(execution time + St));
10. For $m = 1$ to size(VM)
11. Utilization (m) = (Joblen / Makespan);
12. End for
13. RU_mean = (sum (Utilization) / size (VM));
14. O = Makespan + RU_mean + EC_mean;
15. Return(O) //The best solution for mapping tasks to VMs is the one that minimizes objective function.
End

FIGURE 5. Objective Function (fobj) Pseudocode.

the user satisfied and the efficiency increases. Finally, the output of algorithm is a matrix with m column and n row that specifies by which VM each task should be executed. Objective function computed as follows:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix} \tag{1}$$

where x_{ij} is a decision variable and calculated by:

$$x_{ij} = \begin{cases} 1 & \text{if } \mathrm{T}_i \text{ is assigned to } \mathrm{VM}_j \\ 0 & \text{if } \mathrm{T}_i \text{ is not assigned to } \mathrm{VM}_j \end{cases} \tag{2}$$

With this condition:

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } 1 \leq \mathrm{i} \leq \mathrm{n} \tag{3}$$

Figure 7 shows how each individual in each population (prey and predator) is initially set up. After that, the task positions are randomly assigned based on the task. Numbers are then rounded up. Round numbers indicate tasks assigned to individual VMs. This means that the second VM is assigned to the fourth task. Figure 7 shows how an agent adapts to the task length based on the task information received.

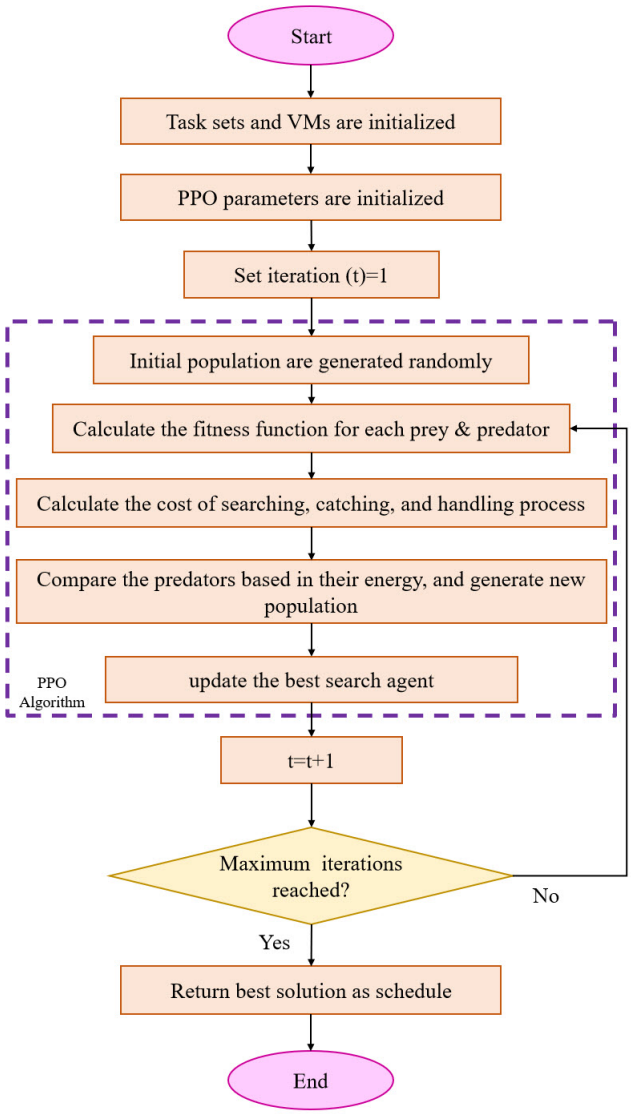


FIGURE 6. Flowchart of the PPO-based task scheduling algorithm.

4.1.2. *Fitness evaluation.* **Makespan:** The makespan is the total time it takes the resources to complete all tasks. In the cloud, VM utilization is determined by how effectively resources are utilized. It is an important issue, because researchers believe that the performance of the scheduling algorithm is depends

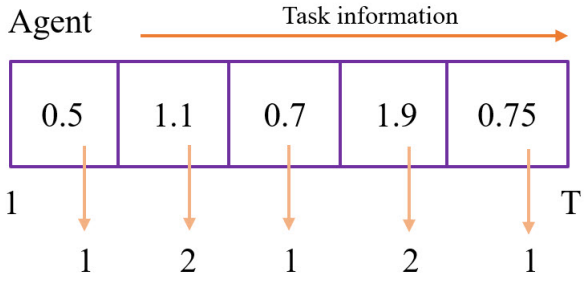


FIGURE 7. An overview of how the agent works.

on makespan. In addition, reducing the makespan rate satisfy the user and make the executions faster. The makespan formula define as follows:

$$Makespan = \max\left(Ext_j\right),\ 1 \leq j \leq m \tag{4}$$

Where Ext_j is the VM_j execution time which calculated based on Eq. 5.

$$Ext_j = \sum_{i=1}^{n} x_{ij} \times CT_{ij} \tag{5}$$

Where X_{ij} is the decision variable and CT_{ij} is the completion time of executing task i on VM_j which calculated by Eq. 6.

$$CT_{ij} = \frac{lengthoftheTask_i}{processingtimeoftheVM_j} \tag{6}$$

Resource Utilization (RU): System efficiency is the number of resources it utilizes effectively. By minimizing idle time of cloud service provider resources and keeping them busy executing customer tasks, it maximizes cloud service provider profits. Therefore, the RU can be defined as follows:

$$RU = \frac{TEXEC}{Makespan \times M} \tag{7}$$

Where M shows the number of VMs and $TEXEC$ is the total execution time which calculated by Eq. 8.

$$TEXEC_{ij} = \sum_{i=1}^{m} VMET_j,\ 1 \leq j \leq M \tag{8}$$

Where $VMET_j$ is the execution time of $j-th$ VM which calculated based on Eq. 9.

$$VMET_j = \sum_{i=0}^{M} x_{ij} \times EXEC_{ij},\ 1 \leq j \leq M \tag{9}$$

Which N shows the number of Tasks, and $EXEC_{ij}$ is the execution time of Task i on virtual machine VM_j that obtained based on Eq. 10.

$$EXEC_{ij} = \frac{L_{Task_i}}{Cap_{VM_j}} \tag{10}$$

Where L_{Task_i} is the computing time for executing Task i and Cap_{VM_j} is the capacity of virtual machine j.

Execution Cost: Execution cost refers to the amount the user pays to the cloud provider for renting a VM. The cost per unit time of a virtual machine is dependent on the time it takes to perform a task. To reduce execution costs, a suitable task scheduling algorithm allocates tasks to VMs optimally. The execution cost of Task i can be calculated as follows:

$$EC_{ij} = Price_j \times \frac{CT_{ij}}{3600} \tag{11}$$

Where $Price_j$ is the price of VM_j and CT_{ij} is the completion time of executing Task i on VM_j.

Finally, the optimization objective function is computed as follows:

$$F_{optimal} = Makespan + (\frac{RU}{M}) + (\frac{TEXEC}{N}) \tag{12}$$

4.1.3. *PPO parameters.* PPO involves individuals in prey populations exploring a large area of search space. Meanwhile, predators are looking for suitable prey to hunt (i.e., the prey which can provide the greatest amount of energy). To spread preys, three distributions are used, which are selected for initialization based on the ability to spread preys better.

There are two types of prey that each predator pursues in the PPO. The best prey (PR_{best}) and its corresponding prey (PR_i). Predator i (PRD_i) moves toward these preys. The movement of the $i-th$ predator can be calculated using Eq. 13 by choosing the vertical movement.

$$PDR_i^{ver}(t+1) = PRD_i(t) + V_i^{ver}(t+1) \tag{13}$$

Where $V_i^{ver}(t+1)$ calculated by Eq. 14.

(14)

$$V_i^{ver}(t+1) = (D \times W \times A \times V_i^{ver}(t)) + ((\frac{T}{t}) \times TP_{best} \times sin(\theta_1) \times T_{corr} \times sin(\theta_2))$$

Where D, W, and A show drag, weight, and Archimedes forces, respectively. The TP_{best} indicates the vector from the predator to the best prey. Predator-prey vectors are shown by the T_{corr}. θ_2 indicates the angle between the predator and best prey, and θ_1 the angle between predator and best prey.

The value set for D, W, and A affects exploration and exploitation in PPO. To demonstrate which setting is best for D, W, and A, the following settings are tested.

Setting 1: D= [1-0], W= [1-0], A= [1-0] //lower value for W, and A
Setting 2: D= [1-0], W= [2-0], A= [2-0] //standard value for W, and A
Setting 3: D= [2-0], W= [3-1], A= [3-1] //higher value for D, W, and A

Setting 4: D= [0.25-0], W= [0.5-0], A= [0.5-0] //very low value for D, W, and A

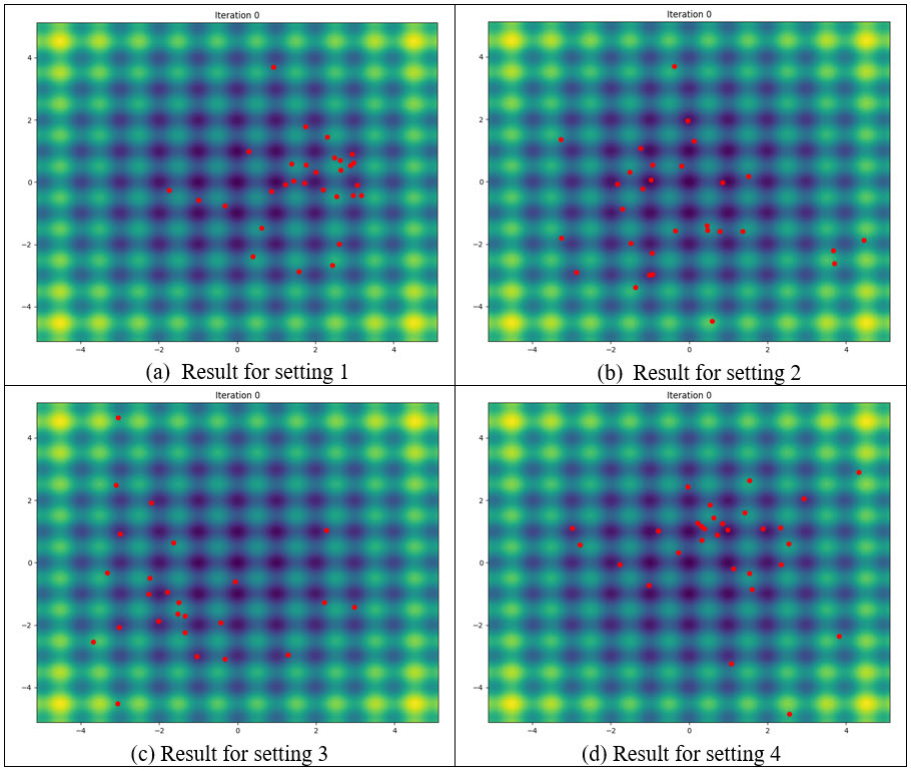


FIGURE 8. The distribution of individuals after an iteration for different settings of D, W, and A.

Figure 8 indicates that setting 2 is appropriate for algorithm because in early iterations individuals need to explore the search space instead of converge on a specific location.

4.1.4. *Update process.* Based on the objective function, the algorithm calculates the energy of predators and generates a new population before updating the search agent. Whenever the strongest predator (apex) has the highest energy gain, it is directly passed to the next iteration. All predators (other than apex predators) will survive for the next generation if they exert a positive influence on exploitation activities during hunting. Mutation and crossover operate to evolve predators that fail to hunt and have little effect on exploitation activities. In addition, survivors of preys produce new generations of children

for the next generation. In addition to avoiding predators, they explored unknown territories. In each iteration, predators and preys will be reassigned. A predator that consumes one prey can calculate its total energy gain (EG) [5].

There is a herd (i.e., team) of predators in the population (Predator) against a prey population (Prey). As a result, we try to improve them with regard to swarm behavior. According to paper [5] two populations are needed to generate new populations for the next iteration $(t+1)$ from current populations at the current iteration (t). There are three main steps in the evolving process:

(1) Identify weak predators and strong preys.
(2) Create a mixed population of predators, and predators with successful escape capabilities, contributing to the exploitation of prey.
(3) Determine which predators to create based on the solutions in the mixed population.

5. Simulation Experiment

The performance of the PPOTS algorithm evaluated in MATLAB R2018a software on a PC with Intel(R) Core (TM) i5-8250U CPU with 1.60, 1.80 GHz, and RAM of 12 GB running on 64-bit Windows 10 Pro operating system platform. This section reviews the experimental results and compares them on two levels. The first level includes the comparison of the proposed method with five well-known meta-heuristic algorithms, fully explained in section 5.1. The second level is given in section 5.2, in which the results of the proposed method are compared with two existing metaheuristic-based scheduling algorithms.

5.1. **Comparison with various meta-heuristic algorithms.** The PPOTS algorithm is compared with five well-known meta-heuristic algorithms, WOA (Whale Optimization Algorithm) [28], SSA (Salp Swarm Algorithm) [29], SHO (Spotted Hyena Optimization) [30], GOA (Grasshopper Optimization Algorithm) [31], and STOA (Sooty Tern Optimization Algorithm) [32].

All the mentioned algorithms have been converted into task scheduling algorithm using the objective function (fobj) used in this article. Also, the proposed algorithm is evaluated under equal conditions and on the same objective function (the proposed objective function in this paper) to show that the proposed method is suitable for the scheduling problem and can optimally allocate tasks to resources fairly. In order to evaluate the performance of the proposed method, 100-500 tasks are generated, each with its own id and size. The dataset consists of three types (low, medium, high) of MI. MATLAB generates tasks using the uniform distributed pseudorandom integer function randi(). The function returns a three-dimensional array [ID, size, type]. It consists of integers on the interval [min, max]. The proposed method has weakness in workflow datasets.

The worst complexity based on population size (N), problem dimension (D), maximum iteration (Max_iter) and cost function evaluation (CF) for PPOTS are given in Table 1 as follows: $Scenario1$: Tasks are fixed, while VMs

TABLE 1. The complexity of the algorithms in the worst case.

Methods	Complexity
PPOTS	$O(Max\_iter((N \times CF) + (N \times D)))$
WOA	$O(Max\_iter((N \times D) \times (N \times CF)))$
SSA	$O(Max\_iter((N \times D) \times (N \times CF)))$
GOA	$O(Max\_iter((N \times D) \times (N \times CF)))$
SHO	$O(N(D(1 + 2 \times Max\_iter)) \times CF)$
STOA	$O(Max\_iter((N \times D) \times (N \times CF)))$

range from 20 to 50. Table 2 shows details of the environmental simulation for scenario 1.

TABLE 2. Experiment setup details for scenario 1.

Parameters	Value
Number of tasks	100
Population size	30
Number of VMs	20-50
Maximum iteration	100

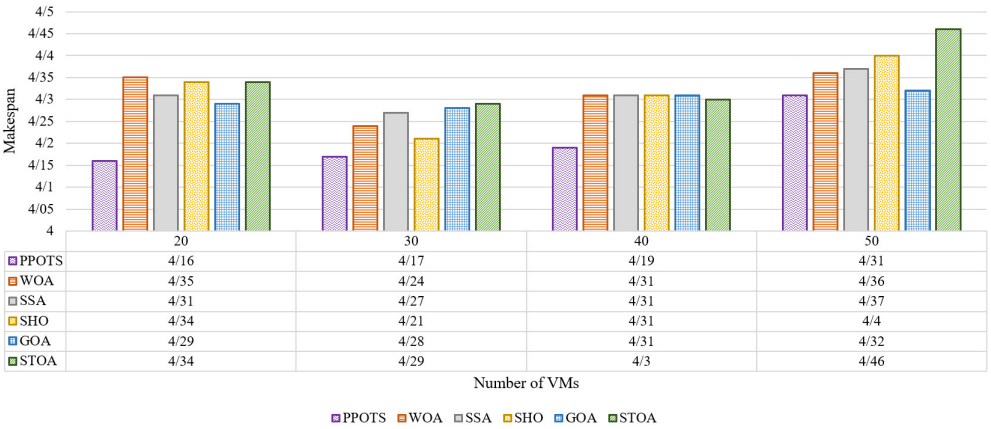
Figures 9, 10, and 11 show the comparisons of the makespan, resource utilization, and execution cost based on different numbers of VMs and fixed tasks. According to Fig. 9, PPOTS has the lowest makespan (about 15%) when the number of VMs is increased. Figure 10 illustrates the resource utilization of the algorithms. In comparison to WOATS, SSATS, SHOTS, GOATS, and STOATS, the PPOTS algorithm is more efficient. According to Figure 11, different algorithms have different degrees of execution cost. The proposed algorithm allocates tasks to resources optimally and distributes VMs efficiently.

*Scenario*2 : This experiment involves a fixed number of VMs with a variable number of tasks. Tasks range from 200 to 500. Scenario 2 parameters are shown in Table 3. By reducing the makespan value, scheduling has demonstrated that

TABLE 3. Experiment setup details for scenario 2.

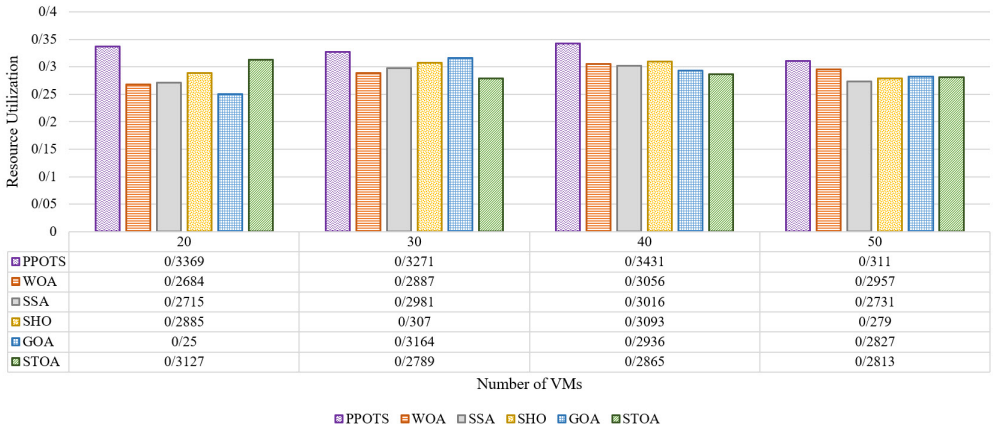
Parameters	Value
Number of tasks	200-500
Population size	30
Number of VMs	50
Maximum iteration	100

it can effectively allocate resources to tasks in an appropriate manner. Figure 12 shows the results of the comparison of the makespan metric between WOATS, SSATS, SHOTS, GOATS, and STOATS based on various numbers of tasks. In comparison with other methods, the PPOTS has a better makespan. Based on the resource utilization parameter, Figure 13 compares the PPOTS algorithm



	20	30	40	50
PPOTS	4/16	4/17	4/19	4/31
WOA	4/35	4/24	4/31	4/36
SSA	4/31	4/27	4/31	4/37
SHO	4/34	4/21	4/31	4/4
GOA	4/29	4/28	4/31	4/32
STOA	4/34	4/29	4/3	4/46

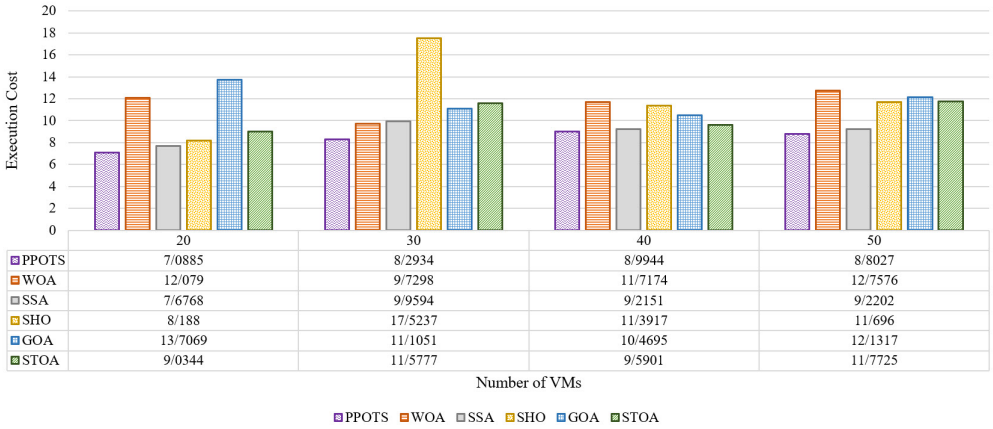
FIGURE 9. Comparison of Makespan (various numbers of VMs).



	20	30	40	50
PPOTS	0/3369	0/3271	0/3431	0/311
WOA	0/2684	0/2887	0/3056	0/2957
SSA	0/2715	0/2981	0/3016	0/2731
SHO	0/2885	0/307	0/3093	0/279
GOA	0/25	0/3164	0/2936	0/2827
STOA	0/3127	0/2789	0/2865	0/2813

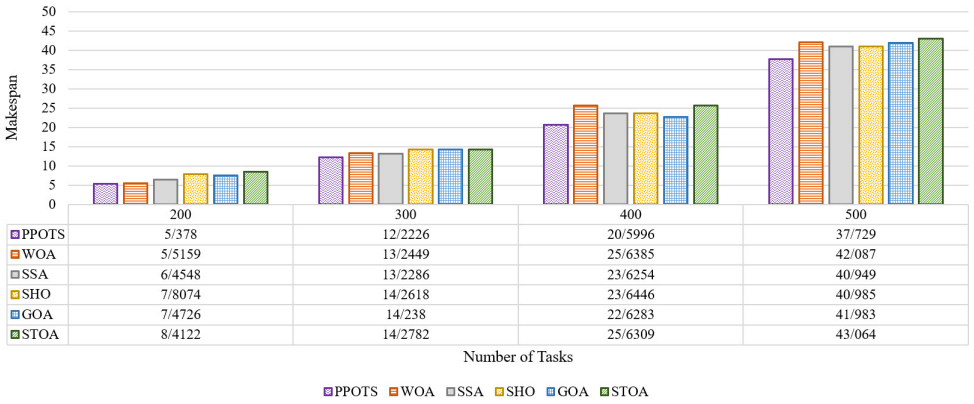
FIGURE 10. Comparison of resource utilization (various numbers of VMs).

with other meta-heuristic algorithms (about 8.25%). The resource utilization decreases as the number of tasks increases. Based on Figure 14, the proposed algorithm has a lower execution cost. The execution cost increases as the number of tasks increases. In convergence analysis, the PPO algorithm finds an optimal solution more rapidly, largely thanks to a new parameter (energy gain) that enables local and global search. With WOA, fewer analyses were considered (function evaluation) in order to minimize the total cost. When the scalability parameter is taken into account, SSA is capable of solving highly



	20	30	40	50
PPOTS	7/0885	8/2934	8/9944	8/8027
WOA	12/079	9/7298	11/7174	12/7576
SSA	7/6768	9/9594	9/2151	9/2202
SHO	8/188	17/5237	11/3917	11/696
GOA	13/7069	11/1051	10/4695	12/1317
STOA	9/0344	11/5777	9/5901	11/7725

FIGURE 11. Comparison of execution cost (various numbers of VMs).

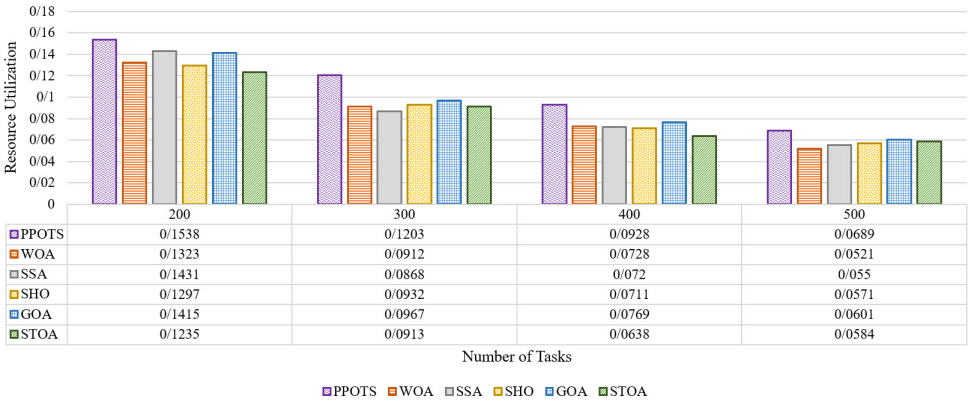


	200	300	400	500
PPOTS	5/378	12/2226	20/5996	37/729
WOA	5/5159	13/2449	25/6385	42/087
SSA	6/4548	13/2286	23/6254	40/949
SHO	7/8074	14/2618	23/6446	40/985
GOA	7/4726	14/238	22/6283	41/983
STOA	8/4122	14/2782	25/6309	43/064

FIGURE 12. Comparison of makespan (various numbers of tasks).

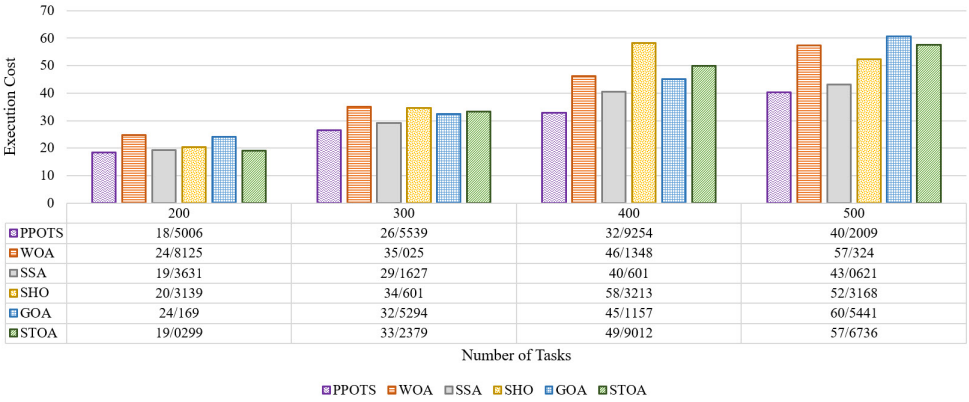
challenging test functions with and without noise. GOA makes a significant contribution to the solution of real problems with unknown search spaces. In comparison with other algorithms, STOA has an average performance in terms of makespan and cost, but it has a good performance when it comes to resource utilization. Moreover, STOA can solve challenging and high-dimensionality constrained real-world problems. SHO performs well in terms of utilization, but not in terms of cost and timeliness.

*Scenario*3 : The experiment is performed based on the various numbers of iteration in this scenario. The parameters details of this scenario illustrate in



	200	300	400	500
PPOTS	0/1538	0/1203	0/0928	0/0689
WOA	0/1323	0/0912	0/0728	0/0521
SSA	0/1431	0/0868	0/072	0/055
SHO	0/1297	0/0932	0/0711	0/0571
GOA	0/1415	0/0967	0/0769	0/0601
STOA	0/1235	0/0913	0/0638	0/0584

FIGURE 13. Comparison of resource utilization (various numbers of tasks).



	200	300	400	500
PPOTS	18/5006	26/5539	32/9254	40/2009
WOA	24/8125	35/025	46/1348	57/324
SSA	19/3631	29/1627	40/601	43/0621
SHO	20/3139	34/601	58/3213	52/3168
GOA	24/169	32/5294	45/1157	60/5441
STOA	19/0299	33/2379	49/9012	57/6736

FIGURE 14. Comparison of execution cost (various numbers of tasks).

Table 4. In order to increase the clarity, the objective function modified as

TABLE 4. Experiment setup details for scenario 3.

Parameters	**Value**
Number of tasks	300
Population size	50
Number of VMs	50
Maximum iteration	1-300

follow:

$$F_{optimal} = Makespan + (1 - RU) + TEXEC \tag{15}$$

Due to the fact that all objectives in Equation11 must become minimal, the method which found a solution with the lowest $F_{optimal}$ is the most efficient. Figure 15 illustrates the convergence of each method over time. As shown in Fig. 15, the proposed algorithm has a faster convergence speed than the others. It illustrates the convergence performance of various optimization algorithms over 300 iterations. In comparison to other algorithms, the proposed algorithm is more efficient. Moreover, because the outputs were not close to each other, the normalization function 'z_score' was used to normalize the outputs by centering the data on 0 and scaling it to 1. The WOA and SHO algorithms have poor exploitation abilities. In contrast, other algorithms (e.g., SSA, STOA, GOA) focus on exploring the search space during the first iteration but fall into the trap of local optimality and cannot reach the global optimality. During the middle iterations, most algorithms, including PPOTS, continue to decrease their fitness values but at a slower rate. As iterations increase, the fitness value of the PPOTS algorithm decreases. Towards the final iterations, PPOTS achieves a fitness value, which is lower than that of the other algorithms, indicating it has found a better solution. Because PPOTS compromises search, catching, and handling well, it is a good choice. In the last iteration, PPOTS searches the entire search space and avoids the local optimal trap to find the best global optimal solution. In Figure 15, it can be seen that PPOTS converges more effectively than the other algorithms. Additionally, it reaches and maintains the lowest fitness value towards the end of the iterations as well as rapidly decreases at the start. PPOTS finds a better optimal solution faster and maintains superior performance throughout the process of optimizing the given problem.

As the agents attempted to traverse the entire search space, there is a higher time complexity than in other metaheuristic algorithms. A greater number of variables that are affected by searching the problem space and assigned randomly. Improving the efficiency of the PPO algorithm itself remains an important area of research. The study needs to consider security, energy consumption, scalability, load balancing, and availability in future work.

It is necessary to conduct a more comprehensive statistical analysis of the results section. Table 5 presents the Statistical Friedman test result to better demonstrate superiority.

$SumofRank(EC) = 1 + 2 + 3 + 4 + 5 + 6 = 21$
$SumofRank(RU) = 1 + 2 \times 2.5 + 3 + 2 \times 4.5 = 18$
$SumofRank(Makespan) = 1 + 2 \times 2.5 + 3 + 2 \times 4.5 = 18$

The Friedman test is computed based on Eq. 16.

$$Q = \frac{12}{nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1) \tag{16}$$

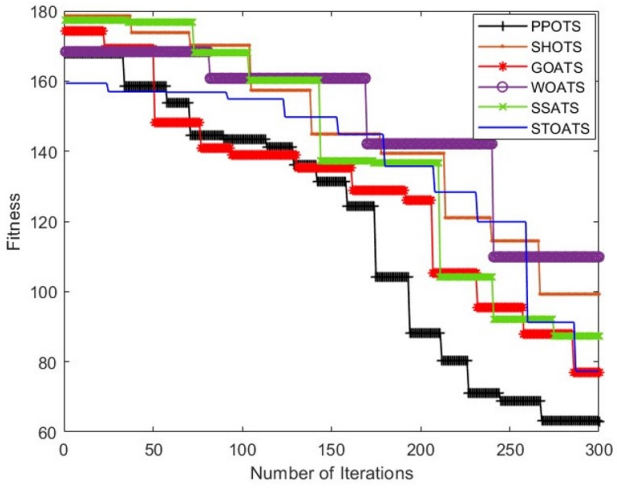


FIGURE 15. The algorithm's convergency.

TABLE 5. Statistical Friedman test on the proposed method and the other compared methods.

Methods	**EC**	**RU**	Makespan	Rank (EC)	Rank (RU)	Rank (Makespan)
PPOTS	46.20	0.064	12.97	1	1	1
WOA	56.87	0.055	12.99	4	2.5	2.5
SSA	48.06	0.058	12.99	2	3	2.5
SHO	54.38	0.055	13.05	3	2.5	4.5
GOA	60.54	0.059	13.02	6	4.5	3
STOA	57.67	0.059	13.05	5	4.5	4.5

Where the number of methods is n, which is equal to 6, the number of compared criteria is k, and R_j is the sum of the Rank for j.

$Q = \frac{12}{6\times3\times4} \times (21^2 + 18^2 + 18^2) - 3 \times 6 \times 4 = 109.5$

This critical value can be found by looking at the chi-square distribution with $k-1$ degrees of freedom. A degree of freedom of 2 is obtained for $k = 3$. At a significance level of $\alpha = 0.05$, the critical value of ^{2} with 2 degrees of freedom is approximately 5.99. The null hypothesis can be rejected since $Q = 109.5$ is greater than the critical value of 5.99. There is a significant difference between the results of the three compared methods.

5.2. **Comparison with metaheuristic-based task scheduling algorithms.** Cloud computing has gained a great deal of attention in recent years due to its expanding platform and features, such as the ability to multiplex users on shared infrastructure and provide on-demand resources. Cloud computing relies heavily on efficient utilization of computer resources. The scheduling of

tasks is crucial to optimizing the performance of cloud systems. It is challenging to schedule virtual machines in dynamic cloud environments characterized by uncertainty and constant change. Cloud task scheduling remains unresolved despite numerous efforts. In addition to varying scheduling approaches, researchers continue to improve overall cloud performance with the incorporation of diverse quality-of-service characteristics.

Zhang and Wang [33] proposed an Enhanced Whale Optimization Algorithm (EWOA). The EWOA combines the WOA with the Lévy fight. As a result of the incorporation of Lévy fight, WOA's search space will be widened, which will expedite convergence with adaptive crossover. The Cloudsim tool is used to simulate and evaluate EWOA under various test conditions. The effectiveness of EWOA is evaluated by comparing it with existing algorithms using various parameters. According to the results, EWOA outperforms other algorithms in terms of resource utilization and execution costs, which demonstrates its superiority in addressing the complexity of multi-objective cloud task scheduling.

Damera et al. [34] presented an innovative task-scheduling algorithm that improved time and energy efficiency as well as overall quality-of-service factors. Using sine chaos mapping, spiral search strategies, and dynamic adaptive weights, the proposed technique enhances Snake Optimization Algorithm (SO). In addition to escaping local optima, these enhancements improve global search. In comparison to other models, the proposed method performs better in cloud scheduling.

Based on the descriptions of the two methods above, we compare the proposed method PPOTS based on the settings of 500 tasks and 50 virtual machines. Figure 16 shows that the proposed method, PPOTS, is the most efficient when compared to others, despite the high number of tasks. PPOTS and EWOA cost close to each other, but the proposed method is less expensive. When discussing resource utilization in the scheduling algorithm, SO performs poorly, but PPOTS and EWOA do quite well. PPOTS has a better and more optimal resource utilization ratio among these two methods.

Because SO and EWOA algorithms ignore this parameter in their objective function formulation, their makespan rate is high. In contrast, in the proposed objective function, waiting time and task processing time are also considered, whereas in the objective functions of the other two methods, these two factors are not taken into account. Additionally, PPOTS explores the search space more thoroughly to find the optimal solution.

6. **Conclusion and Future Works**

Cloud computing can be greatly improved by optimizing scientific task scheduling. Although finding a suitable task scheduling algorithm is very important for cloud users and providers, most papers fail to offer an effective trade-off between makespan, resource utilization, and execution cost. In this paper, we introduce a PPO-based task scheduling algorithm named PPOTS that takes

FIGURE 16. Comparison of proposed method with the other scheduling algorithms.

into account makespan, resource utilization, and execution cost. When compared to WOA, SSA, SHO, GOA, and STOA, the proposed task scheduling method can improve system makespan, execution cost, and resource utilization by 15, 28, and 8.25%. Additionally, the PPOTS algorithm has a faster convergence time to find the optimal solution than other meta-heuristic algorithms. Also, the proposed method compared with two metaheuristic-based scheduling algorithms in terms of makespan, resource utilization and execution cost. In future work, we will consider security, scalability, load balancing, and availability. Additionally, we intend to improve the PPO algorithm's efficiency. Two perspectives can be used to examine the improvements:

(1) From the perspective of task scheduling, including:
   - Modify the proposed method so that accept workflow tasks
   - Add migration task (or VM) mechanism for load balancing
   - Enhance security mechanisms in the environment to ensure reliable connectivity for users

(2) From the perspective of the PPO algorithm, including:
   - Use reinforcement learning to learn policy of selecting search space
   - Set number of individuals in both populations dynamically based situation of exploring or exploiting the search space

References

[1] Gasmi, K., Dilek, S., Tosun, S., & Ozdemir, S. (2021). A survey on computation offloading and service placement in fog computing-based IoT. The Journal of Supercomputing, 78, 1983-2014. https://doi.org/10.1007/s11227-021-03941-y

[2] Maciel, P., Dantas, J., Melo, C., Pereira, P., Oliveira, F., Araujo, J., & Matos, R. (2022). A survey on reliability and availability modeling of edge, fog, and cloud computing. Journal of Reliable Intelligent Environments, 8, 227–245. https://doi.org/10.1007/s40860-021-00154-1

[3] Kant, U., & Kumar, V. (2022). IoT network used in fog and cloud computing. Internet of Things: Security and Privacy in Cyberspace, 165-187.

[4] Buyya, R., & Venugopal, S. (2005). A gentle introduction to grid computing and technologies. CSI Communications, 9-19.

[5] Mohammad Hasani Zade, B., & Mansouri, N. (2022). PPO: A new nature-inspired metaheuristic algorithm based on predation for optimization. Soft Computing, 26, 1331–1402. https://doi.org/10.1007/s00500-021-06404-x
[6] Saravanan, G., Neelakandan, S., Ezhumalai, & P. Maurya, S. (2023). Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. Journal of Cloud Computing, 12(24). https://doi.org/10.1186/s13677-023-00401-1
[7] Behera, I., & Sobhanayak, S. (2024). Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. Journal of Parallel and Distributed Computing, 183, 104766. https://doi.org/10.1016/j.jpdc.2023.104766
[8] Saif, F.A., Latip, R. Hanapi, Z.M., & Shafinah, K. (2023). Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. IEEE Access, 11, 20635-20646. https://doi.org/10.1109/ACCESS.2023.3241240
[9] Peter, M., & Grance, T. (2011). The NIST definition of cloud computing.
[10] Jalali Khalil Abadi, Z., Mansouri, N., & Khalouie, M. (2023). Task scheduling in fog environment — Challenges, tools and methodologies: A review. Computer Science Review, 48, 100550. https://doi.org/10.1016/j.cosrev.2023.100550
[11] Singh, H., Tyagi, S., & Kumar, P. (2021). Cloud resource mapping through crow search inspired metaheuristic load balancing technique. Computers and Electrical Engineering, 93, 107221. https://doi.org/10.1016/j.compeleceng.2021.107221
[12] Bhandari, G.P., & Gupta, R. (2019). An overview of edge/cloud computing architecture with its issues and challenges. Advancing Consumer-Centric Fog Computing Architectures, 1-37.
[13] Sriram, G.K. (2022). Edge computing vs. cloud computing: An overview of big data challenges and opportunities for large enterprises. International Research Journal of Modernization in Engineering Technology and Science, 4, 1331-1337. http://dx.doi.org/10.0202/Computin.2022197786
[14] Ghafari, R., Hassani Kabutarkhani, F., & Mansouri, N. (2022). Task scheduling algorithms for energy optimization in cloud environment: A comprehensive review. Cluster Computing, 25, 1035-1093. https://doi.org/10.1007/s10586-021-03512-z
[15] Mansouri, N., & Javidi, M.M. (2020). A review of data replication based on metaheuristics approach in cloud computing and data grid. Soft Computing, 24(19). https://doi.org/10.1007/s00500-020-04802-1
[16] Pradeep, K., Gobalakrishnan, N., Manikandan, N., Javid Ali, L., Parkavi, K., & Vijayakumar, K.P. (2021). A review on task scheduling using optimization algorithm in clouds. 5th International Conference on Trends in Electronics and Informatics (ICOEI). https://doi.org/10.1109/ICOEI51242.2021.9452837
[17] Gray, M.R., and Johnson, D.S. (1979). Computers and intractability: A guide to the theory of NP-completeness. The Journal of Symbolic Logic, 48(2), 90-91.
[18] Pradhan, R., & Satapathy, S.C. (2023). Particle Swarm Optimization-based energy-aware task scheduling algorithm in heterogeneous cloud. Communication, Software and Networks, 439-450. https://doi.org/10.1007/978-981-19-4990-6_40
[19] Indhumathi, R., Amuthabala, K., Kiruthiga, G., Yuvaraj, N., & Pandey, A. (2023). Design of task scheduling and fault tolerance mechanism based on GWO algorithm for attaining better QoS in cloud system. Wireless Personal Communications, 128, 2811-2829. https://doi.org/10.1007/s11277-022-10072-x
[20] Prem Jacob, T., & Pradeep, K. (2019). A multi objective optimal task scheduling in cloud environment using Cuckoo Particle Swarm Optimization. Wireless Personal Communications, 109, 315-331. https://doi.org/10.1007/s11277-019-06566-w
[21] Abd Elaziz, M., & Attiya, I. (2021). An improved Henry Gas Solubility Optimization algorithm for task scheduling in cloud computing. Artificial Intelligence Review, 54, 3599-3637. https://doi.org/10.1007/s10462-020-09933-3

[22] Huang, X., Li, C., Chen, H., & An, D. (2020). Task scheduling in cloud computing using Particle Swarm Optimization with time varying inertia weight strategies. Cluster Computing, 23, 1137-1147. https://doi.org/10.1007/s10586-019-02983-5

[23] Vijarania, M., Agrawal, A., & Sharma, M.M. (2021). Task scheduling and load balancing techniques using Genetic Algorithm in cloud computing. Soft Computing: Theories and Applications, 97-105. https://doi.org/10.1007/978-981-16-1696-9_9

[24] Su, Y., Bai, Z., & Xie, D. (2021). The optimizing resource allocation and task scheduling based on cloud computing and Ant Colony Optimization Algorithm. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-021-03445-w

[25] Tripathi, G., & Kumar, R. (2022). A heuristic-based task scheduling policy for QoS improvement in cloud. International Journal of Cloud Applications and Computing, 12(2), 1-22. https://doi.org/10.4018/IJCAC.295238

[26] Krishnan, S., & Rajalakshmi, N.R. (2022). A cost-optimized data parallel task Sscheduling in multi-core resources under deadline and budget constraints. International Journal of Cloud Applications and Computing, 12(2), 1-16. https://doi.org/10.4018/IJCAC.305857

[27] Ajmal, M.S., Iqbal, Z., Khan, F.Z., Ahmad, M., Ahmad, I., & Gupta, B.B. (2021). Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers. Computers and Electrical Engineering, 95, 107419. https://doi.org/10.1016/j.compeleceng.2021.107419

[28] Mirjalili, S.A., & Lewis, A. (2016). The whale optimization algorithm. Advances in Engineering Software, 95, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[29] Mirjalili, S.A., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., & Mirjalili, S.M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163-191. https://doi.org/10.1016/j.advengsoft.2017.07.002

[30] Dhiman, G., & Kumar, V. (2017). Spotted Hyena Optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Advances in Engineering Software, 114, 48-70. https://doi.org/10.1016/j.advengsoft.2017.05.014

[31] Saremi, S., Mirjalili, S.A., & Lewis, A. (2017). Grasshopper Optimization Algorithm: Theory and application. Advances in Engineering Software, 105, 30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004

[32] Dhiman, G., & Kumar, V. (2019). STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Engineering Applications of Artificial Intelligence, 82, 148-174. https://doi.org/10.1016/j.engappai.2019.03.021

[33] Zhang, Y., & Wang, J. (2024). Enhanced Whale Optimization Algorithm for task scheduling in cloud computing environments. Journal of Engineering and Applied Science, 71(121). https://doi.org/10.1186/s44147-024-00445-3

[34] Damera, V.K., Vanitha, G., Indira, B., Sirisha, G., & Vatambeti, R. (2023). Improved snake optimization-based task scheduling in cloud computing. Computing, 106, 3353-3385. https://doi.org/10.1007/s00607-024-01323-9

Zahra Jalali Khalil Abadi
Orcid number: 0000-0003-3778-6548
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: zj190179@gmail.com

Behnam Mohammad Hasani Zade
Orcid number: 0000-0003-3463-7520
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: behnamhasani707@gmail.com

Najme Mansouri
Orcid number: 0000-0002-1928-5566
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: najme.mansouri@gmail.com

Mohammad Masoud Javidi
Orcid number: 0009-0002-7955-8220
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: javidi@uk.ac.ir