Shahid Bahonar University of Kerman

Journal of Mahani Mathematical Research

Print ISSN: 2251-7952 Online ISSN: 2645-4505

COST-AWARE AND ENERGY-EFFICIENT TASK SCHEDULING BASED ON GREY WOLF OPTIMIZER

R. Ghafari and N. Mansouri

Article type: Research Article

(Received: 05 April 2022, Received in revised form: 06 June 2022)

(Accepted: 14 July 2022, Published Online: 24 July 2022)

Abstract. One of the principal challenges in the cloud is the task scheduling problem. Appropriate task scheduling algorithms are needed to achieve goals such as load balancing, minimum cost, minimum energy consumption, etc. Using meta-heuristic algorithms is a good way to solve scheduling problems in the cloud because scheduling is an NP-hard problem. In recent years, various meta-heuristic algorithms have been introduced, one of the most popular meta-heuristic algorithms to deal with optimization problems is the Grey Wolf Optimizer (GWO) algorithm. This paper introduces a novel GWO-based task scheduling (GWOTS) algorithm to map tasks over the available resources. The principal goal of this paper is to decrease execution cost, energy consumption, and makespan. The efficiency of the GWOTS algorithm is compared with the well-known meta-heuristic algorithms, namely Genetic Algorithm (GA), Dragonfly Algorithm (DA), Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Ant Colony Optimization (ACO), Gravitational Search Algorithm (GSA), Sooty Tern Optimization Algorithm (STOA), Artificial Hummingbird Algorithm (AHA), Multi-Verse Optimizer (MVO), and Sine Cosine Algorithm (SCA). In addition, the performance of GWOTS is compared with three recently scheduling algorithms, namely SOATS, IWC, and CETSA. Experimental results show that the GWOTS algorithm improves performance in terms of makespan, cost, energy consumption, total execution time, resource utilization, throughput, and degree of resource load balance compared to other algorithms.

Keywords: Cloud Computing, Task scheduling, GWO, Meta-heuristic.
2020 MSC: 68T20.

1. Introduction

Cloud computing has become a famous term in computer science in recent decades due to its advantages and advances such as hiding and abstracting complexity, visualized resources, and efficient use of distributed resources [6]. Cloud computing manages a variety of virtual resources, so scheduling is a significant component of cloud computing. The main idea of task scheduling is assigning tasks in such a way that one or more objectives are optimized [37].

✉ Najme.mansouri@gmail.com, ORCID: 0000-0002-1928-5566
DOI: 10.22103/jmmr.2022.19250.1230
Publisher: Shahid Bahonar University of Kerman
How to cite: R. Ghafari, N. Mansouri, *Cost-Aware and Energy-Efficient Task Scheduling Based on Grey Wolf Optimizer*, J. Mahani Math. Res. 2023; 12(1): 257-288.

The problem of scheduling is an NP-hard problem because resources are heterogeneous and tasks have various properties. There is no specific way to get a polynomial-time solution for NP-hard problems. One of the solutions that have been considered by researchers to obtain a near-optimal solution to these complex problems is the use of meta-heuristic algorithms [25]. One of the interesting meta-heuristic algorithms is the Grey Wolf Optimizer (GWO) algorithm, which is to deals with continuous optimization problems. This paper presents a task scheduling algorithm using a grey wolf optimizer for the cloud system. The goal of the GWOTS algorithm is to reduce makespan, energy consumption, and cost.

The principal contributions of this paper are as follows:

(1) A task scheduling algorithm for the cloud environment is suggested that simultaneously considers execution cost, energy consumption, and makespan.
(2) The task scheduling problem is formulated and objective functions are introduced to optimally map tasks to virtual machines.
(3) Because GWO can make a suitable trade-off between exploration and exploitation, it has been used to solve scheduling problems.
(4) Comprehensive experimental analysis is performed to compare the efficiency of the presented algorithm with GA, PSO, ACO, DA, WOA, GSA, STOA, AHA, MVO, and SCA. Also, the proposed algorithm is compared with three scheduling algorithms (i.e., SOATS, IWC, and CETSA).

The rest of the paper's content is organized as follows. In Section 2, the main concepts and preliminaries are described. In Section 3, the relevant paper is discussed. In section 4, the GWOTS algorithm is described. In section 5, the performance of the GWOTS algorithm is evaluated. Finally, in section 6, the conclusion and future works are discussed.

2. Preliminaries

This section describes the preliminaries and important definitions of cloud computing, task scheduling, meta-heuristic algorithms, and GWO algorithm.

2.1. **Cloud Computing.** Cloud computing is a distributed computing environment for data storage, data processing, and data networking. It is a huge revolution in the provision of virtualization-based Information Technology (IT) services [7]. Cloud computing provides on-demand services for cloud users. Users can remotely save their data and access them anytime and anywhere through the Internet [40] Clouds are applied in a variety of areas, including Wireless Sensor Networks (WSNs) and big data. As shown in Fig. 1, there are four kinds of deployment models in cloud system: 1) Public: Enables cloud services for public utilization, 2) Private: Enables cloud services just for the organization itself, 3) Hybrid: It is a combination of public and private cloud features and 4) Community: Shares cloud infrastructure between multiple organizations or individuals. Each type of cloud has different IT management and security risks. As shown in Fig. 1, there are three models for cloud services: 1) SAAS (Software as a Service), 2) IAAS (Infrastructure as A Service), and 3) PAAS (Platform as a Service). In addition, cloud computing has five main features that are shown in Fig. 1. In comparison with other distributed systems (e.g., supercomputers, data grids), the cloud can provide a computing environment with higher access, more reliability, cheaper, and more scalable [27].

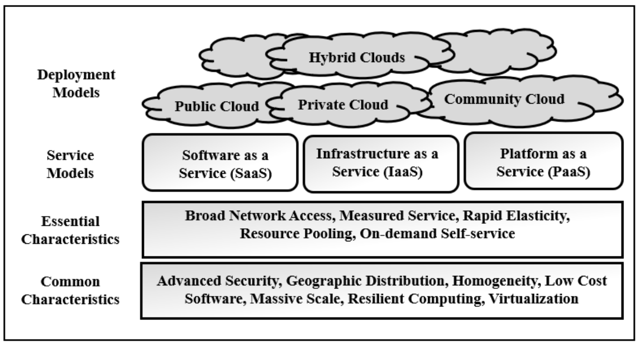
Deployment Models
Hybrid Clouds
Public Cloud
Private Cloud
Community Cloud
Service Models
Software as a Service (SaaS)
Infrastructure as a Service (IaaS)
Platform as a Service (PaaS)
Essential Characteristics
Broad Network Access, Measured Service, Rapid Elasticity, Resource Pooling, On-demand Self-service
Common Characteristics
Advanced Security, Geographic Distribution, Homogeneity, Low Cost Software, Massive Scale, Resilient Computing, Virtualization

FIGURE 1. Cloud computing definition [4].

Due to the exponential growth in demand for services, there is a need for continuous improvement in cloud data centers. An important part of a cloud environment is efficient task scheduling to achieve maximum efficiency, minimum energy consumption, minimum makespan, and optimal utilization of resources [39].

2.2. **Task Scheduling.**

Cloud service providers must provide services to customers. In the cloud system, with the increase in the number of requests, cloud service providers face different issues such as task scheduling, privacy, and security. One of the principal challenges in the field of cloud computing is the problem of scheduling [12]. Task scheduling is the process of allocating submitted tasks to available resources so that resource utilization is increased and the Quality of Service (QoS) is maximized. Thus, tasks are assigned to various resources based on restrictions imposed by users and cloud providers [18]. Figure 2 shows the scheduling model in the cloud system. Input tasks are sent to the cloud system task queue. Then, input tasks are received by the Virtual Machine (VM) manager from the task queue. Then, the VM manager analyses the status of the resources. If it is possible to assign tasks to existing VMs, tasks will be allocated to VMs, otherwise, new VMs will be created.

The problem of task scheduling is an NP-hard problem in the cloud, meaning that no optimal solution can be found in a polynomial time. This is due

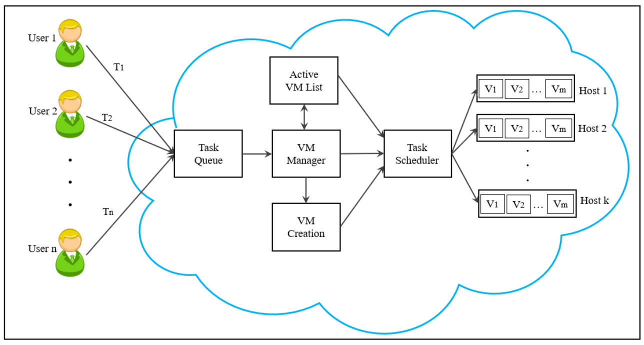

FIGURE 2. Model of assigning tasks to VMs in the cloud system [36].

to it taking a lot of time to find the optimal solution because the solution space is large and therefore no strategy can be used to get the optimal solution in a polynomial time for the scheduling problem [45]. Meta-heuristic algorithms are a suitable selection for solving NP-hard problems. Meta-heuristic algorithms formulate the task scheduling problem as an optimization problem. Meta-heuristic algorithms can provide optimal or near-optimal solutions with minimal complexity [21].

2.3. **Meta-heuristic Algorithm.** Meta-heuristic algorithms are a significant area of research with significant advances in solving complex optimization problems. The term meta-heuristic describes higher-level heuristics that have been presented to solve a wide range of optimization problems [10]. In recent years, a lot of meta-heuristic algorithms have been applied successfully to solve large computational problems. The attractiveness of using meta-heuristic algorithms is that they offer near-optimal solutions to complex problems in a suitable time. Meta-heuristic algorithms are divided into two categories: 1) single-solution meta-heuristics (Consider a single solution at a time), 2) population meta-heuristics (Evolve several solutions simultaneously). Meta-heuristic algorithms are also named approximation algorithms that have better results than deterministic algorithms [41]. Figure 3 shows the classification of meta-heuristic algorithms into four main categories. As shown in Fig. 3, optimization algorithms are divided into four main categories.

One of the interesting categories of population-based meta-heuristic algorithms is swarmed intelligence. The main idea of the swarm intelligence algorithms is the collective social behavior of natural swarms, systems, or communities such as fish schools, insect colonies, animal herds, and bird flocks [24]. The Grey Wolf Optimizer (GWO) is one of the well-known swarm intelligence algorithms, which is proposed by Mirjalili et al [30]. The leadership hierarchy as well as the mechanism of hunting grey wolves in nature, which are looking

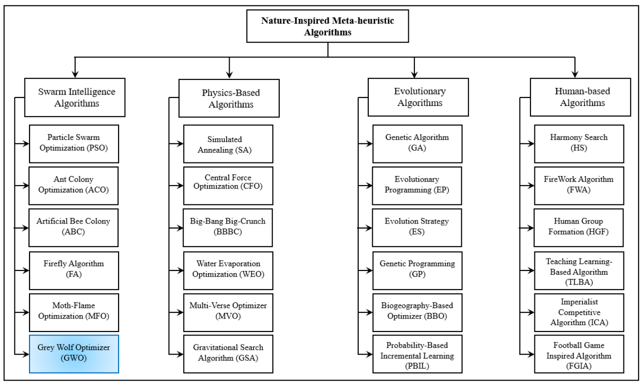
Nature-Inspired Meta-heuristic Algorithms
Swarm Intelligence Algorithms
Particle Swarm Optimization (PSO)
Ant Colony Optimization (ACO)
Artificial Bee Colony (ABC)
Firefly Algorithm (FA)
Moth-Flame Optimization (MFO)
Grey Wolf Optimizer (GWO)
Physics-Based Algorithms
Simulated Annealing (SA)
Central Force Optimization (CFO)
Big-Bang Big-Crunch (BBBC)
Water Evaporation Optimization (WEO)
Multi-Verse Optimizer (MVO)
Gravitational Search Algorithm (GSA)
Evolutionary Algorithms
Genetic Algorithm (GA)
Evolutionary Programming (EP)
Evolution Strategy (ES)
Genetic Programming (GP)
Biogeography-Based Optimizer (BBO)
Probability-Based Incremental Learning (PBIL)
Human-based Algorithms
Harmony Search (HS)
FireWork Algorithm (FWA)
Human Group Formation (HGF)
Teaching Learning-Based Algorithm (TLBA)
Imperialist Competitive Algorithm (ICA)
Football Game Inspired Algorithm (FGIA)

Figure 3. Classification of the different meta-heuristic algorithms [29].

for the optimal way to hunt prey, have been the main ideas on which the GWO algorithm is based. Experimental results in [30] showed that GWO can have a better performance compared to Evolutionary Programming (EP), Differential Evolution (DE), Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), and Evolution Strategy (ES). The outcomes also represent that GWO can provide a good trade-off between exploration and exploitation, which leads to the avoidance of local optimal. In addition, GWO can be used to solve a large number of optimization problems because GWO has significant features such as very low parameters, flexibility, no need for derivative information in the initial search, simplicity, ease to use, and scalability. GWO algorithm can also create a good balance between exploration and exploitation, which leads to desired convergence [15]. Thus, GWO was able to gain the research interest of audiences in various fields in a short time.

2.4. **Grey Wolf Optimizer (GWO).** GWO algorithm is a swarm intelligence algorithm that is one of the most famous meta-heuristic algorithms inspired by nature. Grey wolves live in a pack. The primary inspiration for the GWO algorithm is the grey wolf swarm intelligence in leadership and hunting. There is a social dominance hierarchy in each group of grey wolves. Wolves' domination has four classes: Alpha, Beta, Delta, and Omega. The strongest wolf is called Alpha, who is the leader and leads the whole pack in hunting, migrating, and feeding. Beta is the alpha advisor and helps alpha make decisions, and Beta is the group leader in the absence of Alpha. Delta wolves are introduced as subordinates and must follow the alpha and beta but dominate the lower level i.e., omega. This hierarchy is one of the main inspirations of the GWO algorithm.

Another inspiration for the GWO algorithm is the hunting approach of grey wolves. Grey wolves follow the following steps when hunting prey:

• Tracking, chasing, and approaching prey,
• Pursuing, encircling, and harassing the prey until it stops moving,
• Attack towards the prey.
In the following, the mathematical model of the GWO algorithm is presented.

Encircling prey: Grey wolves encircle their prey during the hunting process, which can be modeled by Eq. (1) and Eq. (2) [15].

$$D = |C.X_p(t) - X(t)| \tag{1}$$

$$X(t+1) = X_p(t) - A.D \tag{2}$$

Where t represents the current iteration, X_p indicates the position of prey and X is the position of the wolf. In addition, A and C are the coefficient vectors and calculated using the following equation [15]:

$$A = 2a.d_1 - a \tag{3}$$

$$C = 2.d_2 \tag{4}$$

Where d_1 and d_2 indicate random vectors in $[0, 1]$, and a is linearly decreased from 2 to 0 and computed using Eq. (5) [15].

$$a = 2 - t(\frac{2}{T}) \tag{5}$$

Where t is the current iteration and T indicates the maximum number of iterations.

Hunting: This stage is led by the alpha wolf (best solution) and beta and delta wolves because they have enough information about the position of prey. Therefore, other wolves must use the position of the best agents to update their position. The following are mathematical modeling equations for position updating [15]:

$$D_\alpha = |C_1.X_\alpha - X|\,, D_\beta = |C_2.X_\beta - X|\,, D_\delta = |C_3.X_\delta - X| \tag{6}$$

$$X_1 = X_\alpha - A_1.(D_\alpha), X_2 = X_\beta - A_2.(D_\beta), X_3 = X_\delta - A_3.(D_\delta), \tag{7}$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{8}$$

Where X_1, X_2, and X_3 indicate position vectors of alpha, beta, and delta wolves, respectively.

Attacking Prey: In this stage, if the prey stops, the grey wolves will attack it. For modeling, the authors reduced the value of a from 2 to 0 (according to Eq. (5)). The amount of A also decreases depending on the a. Since the value of A has an important role, grey wolves attack prey if $|A| < 1$. The algorithm also provides a search or exploration step to avoid falling into the local optimal. Therefore, if $|A| > 1$, the grey wolves will go away from the prey and look for another prey. This problem is shown in Fig. 4.

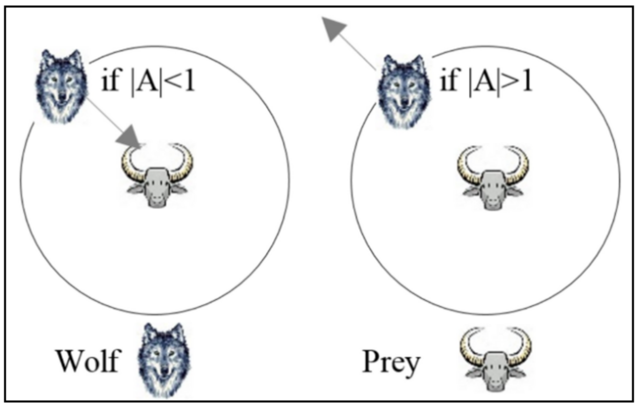

FIGURE 4. Exploration phase vs. Exploration phase [38].

3. Related Work

The problem of task scheduling in the cloud is an NP-hard problem. Recently, different algorithms have been presented to solve the task schedule problem. This section provides an overview of recent scheduling algorithms.

Ajmal et al. [2] offered a novel task scheduling algorithm using a hybrid of ant and genetic algorithms and named it HAGA. The HAGA algorithm divides tasks and VMs into smaller groups using the features of the GA and the ACO algorithm. The proposed algorithm adds a pheromone to VMs after distributing tasks among VMs. According to the HAGA algorithm, if the VM is loaded, it will not be included in the solution space. Therefore, due to the reduction of solution space, convergence and response time are considerably reduced. This study represented that the suggested method improves performance in terms of minimizing response time, makespan, cost of running workflow, convergence time, and Service Level Agreement (SLA) violation compared to other algorithms.

Elaziz et al. [14] offered a new scheduling algorithm. In the presented algorithm, the Moth Search Algorithm (MSA) has been improved using DE to be able to optimally assign tasks to VMs. Since one of the important features of the meta-heuristic algorithm is having the capability to explore and exploit properly, but the MSA does not have a good exploitation ability, the DE algorithm is applied to enhance the exploitation capability of the MSA. The experimental outcomes represented that the proposed algorithm achieves the optimal solution in a shorter time than other algorithms.

Fu et al. [16] suggested a particle swarm optimization genetic hybrid algorithm based on phagocytosis for scheduling tasks in the cloud system and named it PSO-PGA. The proposed algorithm changes the strategy of updating position and velocity in standard PSO. In the proposed algorithm, each generation of particles is split and the particle position is updated based on the mechanism of phagocytosis and crossover mutation of the GA. This is because it makes the search space of the solution space larger. Sub-population merging operations are then performed, which leads to particle diversity in the population and decreases the possibility of falling into the local optimal. The experimental outcomes represented that the PSO-PGA algorithm improves efficiency in terms of overall completion time and convergence accuracy than other algorithms.

Jacob and Pradeep [42] introduced the CPSO algorithm, which combines Cuckoo Search (CS) and PSO to schedule tasks. The proposed algorithm goals to decrease the makespan, deadline violation rate, and cost. To evaluate the performance of the CPSO algorithm, the authors used a CloudSim simulator. The experimental outcomes demonstrated that the CPSO method has better efficiency in comparison to other algorithms.

Guo [19] offered a task scheduling algorithm using the fuzzy self-defense algorithm that aimed to make a balance between the shortest time, the cost, and the degree of load balancing. The global optimal solution is calculated based on solving the objective function with a fuzzy self-defense algorithm by searching. The experimental outcomes represented that the proposed algorithm can improve efficiency compared to other algorithms.

Kumar and Venkatesan [44] introduced a task scheduling algorithm that stores tasks in a queue and calculates priorities. The suggested algorithm assigns tasks to the resource if it is a duplicate task, otherwise, it analyzes new tasks that are in the on-demand queue. Tasks in the on-demand queue are mapped to resources using the Hybrid Genetic-Particle Swarm Optimization (HGPSO) algorithm (which is a combination of PSO and GA). This study represented that the proposed method can improve performance in terms of execution time, scalability, and availability than other algorithms.

Emami [13] presented the Enhanced Sunflower Optimization (ESFO) algorithm to optimally assign tasks to resources with minimal search complexity. In order to be able to search the solution space well, in the ESFO algorithm, the pollination operator of the standard SFO algorithm has been improved and the ESFO algorithm can achieve the optimal solution in a polynomial time. The experimental outcomes represented the ESFO algorithm The experimental outcomes represented that the ESFO algorithm improves the efficiency in comparison to other algorithms.

Velliangiri et al. [46] offered a Modified Electro Search (HESGA) algorithm to achieve optimal scheduling in the cloud system. The presented algorithm aims to enhance the efficiency of the scheduling algorithm by considering parameters such as cost, load balancing, makespan, and resource utilization. HESGA algorithm is based on modifying the electronic search algorithm with the principles of genetic algorithm. The GA can find the best local optimal solutions, while the electro search algorithm can find the best global optimal solutions. The experimental outcomes demonstrated that the HESGA algorithm improves performance than other algorithms.

Imene et al. [22] suggested a new task scheduling algorithm using a third-generation Multi-objective optimization method called Nondominated Sorting Genetic Algorithm (NSGA-III). The proposed algorithm aims to reduce runtime, power consumption, and cost. The evaluations indicated that the proposed method outperforms in terms of runtime, cost, and energy usage in comparison with the Non-dominated Sorting Genetic Algorithm (NSGAII).

Abed-alguni and Alawad [1] offered a scheduling strategy for workflow applications using the Distributed Grey Wolf Optimizer (DGWO) technique. The purpose of the proposed algorithm is to reduce the cost of computation and data transmission by optimally allocating the tasks of a workflow application to machines. Continuous candidate solutions produced by DGWO are converted into discrete candidate solutions using the Largest Order Value (LOV) procedure. Simulation results indicated that DGWO assigns tasks faster between resources than other methods. In addition, the experimental results showed that DGWO obtains a reasonable makespan in comparison with other algorithms.

In Table 1, the discussed scheduling algorithms are compared. As shown in Table 1, it can be concluded that most of the presented algorithms are based on reduction of makespan, or energy consumption or cost, while all these parameters have a significant impact on the efficiency of the cloud system. In this paper, the introduced algorithm considers three parameters of makespan, cost, and energy consumption simultaneously. This paper presents an optimal solution to the task scheduling problem. In other words, the main advantage of the proposed algorithm is that it considers the three parameters of cost, makespan, and energy that conflict with each other and uses the GWO algorithm to find the global optimal solution.

4. Grey Wolf Optimizer-based Task Scheduling Algorithm (GWOTS)

In this Section, the task scheduling problem is formulated and the objective functions are introduced. Subsection 4.1 introduces the task scheduling problem concepts in the cloud environment, Subsection 4.2 states the objective functions, and finally, in Subsection 4.3 GWOTS algorithm is described.

Table 1. Comparison of different scheduling algorithms.

Reference	**Year**	**Makespan**	**Cost**	**Resource Utilization**	**Security**	**Energy consumption**	**Priority constraint**	**Tool**	**Technique**	**Advantage**	**Disadvantage**
Ajmal et al. [2]	2021	+	+	-	-	-	-	CloudSim	ACO and GA	- Decreases running time, - Divides tasks and VMs into smaller groups, - Reduces convergence and response time.	- Does not manage servers, - Does not pay attention to dependencies between tasks.
Elaziz et al. [14]	2019	+	-	-	-	-	-	CloudSim	MSA and DE	- Provides better performance than the traditional MSA, - Performs task scheduling with minimum makespan.	- The presented algorithm is a single-objective and does not take into account other parameters such as overloads, usage of memory, and the peak of the demand, - Time complexity is high.
Fu. [16]	2020	+	-	-	-	-	-	CloudSim	PSO and GA based on phagocytosis	- Has high convergence accuracy, - Improves completion time.	- The energy efficiency of the proposed algorithm is very low.
Jacob and Pradeep [42]	2019	+	+	-	-	-	-	CloudSim	CS and PSO	- Obtains minimal deadline violation rate, - Decreases the makespan and cost.	- Does not consider energy consumption and other effective QoS parameters, - Does not consider load balancing and does not provide elasticity.
Guo [19]	2021	-	+	+	-	-	-	—	Fuzzy self-defense algorithm	- Shortest maximum scheduling completion time, - A low deadline violation rate.	- Does not consider constraint like priority, - Energy efficiency of the proposed algorithm is very low.
Kumar and Venkatesan [44]	2018	+	-	-	-	-	+	—	PSO and GA	- Computes priority, - Improves availability and scalability rate.	- Does not compare with any state-of-art algorithm, - Does not consider effective QoS parameters like cost, energy, etc.
Emami [13]	2021	+	-	-	-	+	-	CloudSim	Enhanced sunflower optimization	- Improves makespan and energy consumption, - Searchs the solution space rightly by improving pollination operator of the SFO.	- Does not consider SLA violation and priority constraint, - Does not consider QoS factors like reliability, availability etc.
Velliangiri et al. [13]	2020	+	+	+	-	-	-	CloudSim	Electro Search algorithm and GA	- Decreases makespan, cost, and response time, - combined the advantage of GS and electro search algorithm.	- Does not consider degree of imbalance and energy efficiency for comparison.
Imene et al. [22]	2022	-	+	-	-	+	-	CloudSim	NSGA-III	- Decreases cost and enrgy consumption, - Improves runtime.	- The proposed algorithm has a high runtime compared to NSGA-II, - Low convergence speed.
Abed-alguni and Alawad [1]	2021	+	+	-	-	-	-	Java programming language	Distributed grey wolf optimizer	- Distributes tasks to VMs fast, - Improves makespan.	- The performance of the proposed algorithm has not been evaluated using complex scientific workflow applications, - Scheduling is based on cost only.

4.1. **System Model.** The datacenter broker and cloud information service assign user tasks to VMs based on user needs and service quality. In cloud computing, task scheduling aims to increase the performance of various QoS metrics. Suppose a cloud data center consists of n number of the task such as: $Task = \{Task_1, Task_2, \ldots, Task_n\}$, where $Task_i$ indicated the $i-th$ task, and m number of VMs such as: $VM = \{VM_1, VM_2, \ldots, VM_m\}$, where VM_j indicated the $j-th$ VM. The condition for executing such tasks is that $n > m$.

4.2. **Objective Function.** The principal aim of this paper is to minimize makespan, energy consumption, and cost to increase user satisfaction and increase the profit of the service provider. Therefore, the objective functions are computed as follows:

Makespan: The total time from sending the first task to completing the last task is called makespan. Makespan is the popular optimization parameter when assigning tasks to resources. This is because most users want their tasks executed faster. Makespan can be computed as follows [35]:

$$MS = \max\left(ExeT_j\right) \tag{9}$$

Where $ExeT_j$ indicates the execution time of the VM_j and it can be calculated based on Eq. (10) [35].

$$ExeT_j = \sum_{i=1}^{n} X_{ij} \times CT_{ij} \tag{10}$$

Where X_{ij} represents the decision variable and CT_{ij} indicates the completion time of the $Task_i$ in the VM_j and it can be computed by Eq. (12) [35]:

$$X_{ij} = \begin{cases} 1 & \text{if } Task_i \textit{ is assigned to } VM_j \\ 0 & \text{if } Task_i \textit{ is not assigned to } VM_j \end{cases} \tag{11}$$

$$CT_{ij} = \frac{Len_i}{PT_j} \tag{12}$$

Where Len_i indicates the length of the $Task_i$ and PT_j indicates the processing time of the VM_j

Energy Consumption: In recent years, the amount of energy consumed by the cloud data center has increased considerably. A lot of energy is consumed by computing resources and cooling equipment, which leads to increased energy costs and carbon emissions. Today, one of the main issues in the field of cloud computing is decreasing energy consumption. Decreasing energy consumption reduces energy costs, increases the lifespan of high-performance computing resources, and reduces carbon emissions. During the execution of tasks, a VM can be idle or active. Suppose VM_j consumes energy equal to γ_j (joule / MI) in active mode and ω_j (joule / MI) in idle mode. A VM_j stays (ET_j) seconds in active mode and $(MS–ET_j)$ seconds in idle mode. Therefore, the total energy consumption can be considered as follows [5]:

$$(13)\quad Energy - Consumption = \sum_{j=1}^{m} [[ET_j \times \gamma_j + (MS - ET_j) \times \omega_j] \times PT_j]$$

Execution Cost: Execution cost means the amount that the user has to pay to the cloud provider to rent a VM. A suitable task scheduling algorithm should reduce the execution cost by optimally allocating tasks to VMs. The execution cost for a task is besed on the cost of the VM per unit time and the execution time of that task. So, the execution cost of $Task_i$ can be calculated as follows [3]:

$$(14)\qquad EC_{ij} = Price_j \times \frac{CT_{ij}}{3600}$$

where $Price_j$ indicates the price of VM_j and CT_{ij} indicates the completion time of executing task $Task_i$ on VM_j.

In this paper, the final optimization objective function is described as follows:

$$(15)\qquad F_optimal = \min\{\alpha_1 \times \tfrac{MS}{MaxMS} + \alpha_2 \times \tfrac{Energy-Consumption}{MaxEnergy-Consumption} + \alpha_3 \times \tfrac{EC}{MaxEC}\}$$

Where α_1, α_2, α_3 indicates weight values in the range [0, 1], and $MaxMS$, $MaxEnergy\_Consumption$, and $MaxEC$ represent the maximum makespan, maximum energy consumption, and max execution cost, respectively.

4.3. **GWOTS Algorithm.** The proposed algorithm considers three important criteria, namely makespan, cost, and energy consumption, simultaneously in the objective function. Makespan, which is one of the most common metrics in scheduling, refers to the time of completing the last task and exiting the cloud system. Therefore, the lower value of makespan shows that the scheduling algorithm is more efficient. Another objective of the optimization is cost. The user's profit increases as the cost are minimized. The cost represents the amount that the user has to pay to the service provider depending on the use of resources. To minimize the execution cost, the scheduling algorithm decides which VM has the lowest execution cost to execute the task. Energy consumption is another criterion in the objective function. Minimizing energy consumption is one of the main and vital issues in the field of cloud computing and is one of the provider-desired criteria. One effective way to reduce energy consumption is to turn off idle systems using an energy-efficient scheduling algorithm. Reducing energy consumption decreases energy-related costs, helps protect the environment by reducing carbon emissions, and improves the overall performance, reliability, and availability of the system. Since energy costs are one of the highest prices in the cloud system, cloud providers are looking for a way to decrease energy consumption. Energy consumption can be reduced by reducing the number of VMs used to execute users' tasks. But this increases the makespan and makes users dissatisfied because users tend to get their tasks done faster. Therefore, this paper presents efficient scheduling using the GWO algorithm to minimize energy consumption, makespan, and cost. In this subsection, the pseudo-code of the proposed algorithm is introduced (Algorithm 1). Also, the flowchart of the GWOTS algorithm is shown in Fig. 5.

Algorithm 1: Pseudo-code for assigning tasks to VMs

Input: Set of tasks, set of VMs, GWO algorithm parameters

Output: Mapping of the task to the appropriate VMs

```
1. $Num$ indicates population size, Factor $a$, $A$, and $C$ are coefficient vectors, $Max_{iter}$ indicates
   the maximum number of iterations
2. Initialize set of tasks, $T = \{T_1, T_2, \ldots, T_n\}$
3. Initialize set of VMs, $VM = \{VM_1, VM_2, \ldots, VM_m\}$
4. Initialize the position of Alpha, Beta, and Delta $(X_\alpha, X_\beta, X_\delta)$
5. For $i = 1 : Num$
6.   Initialize the positions of search agents randomly
7. End for
8. Set $t = 1$
9. While $(t <= Max_{iter})$
10.    For $i = 1 : Num$
11.        Calculate the objective function for each search agent
12.        Update the position of Alpha, Beta, and Delta $(X_\alpha, X_\beta, X_\delta)$
               // The grey wolf with the most fitness is denoted as Alpha
               // The grey wolf with the second most fitness is denoted as Beta
               // The grey wolf with the third most fitness is denoted as Delta
13.    End for
14.    The value of a decrease from 2 to 0
15.    For each search agent
16.        Using Eq. (3) and Eq. (4) the coefficient of $A$ and $C$ are updated, respectively
17.        Using Eq. (8) each grey wolf position is updated
18.    End for
19.    Set $t = t + 1$
20. End while
21. Return Alpha $(X_\alpha)$ as the nearest optimal from the search space
```

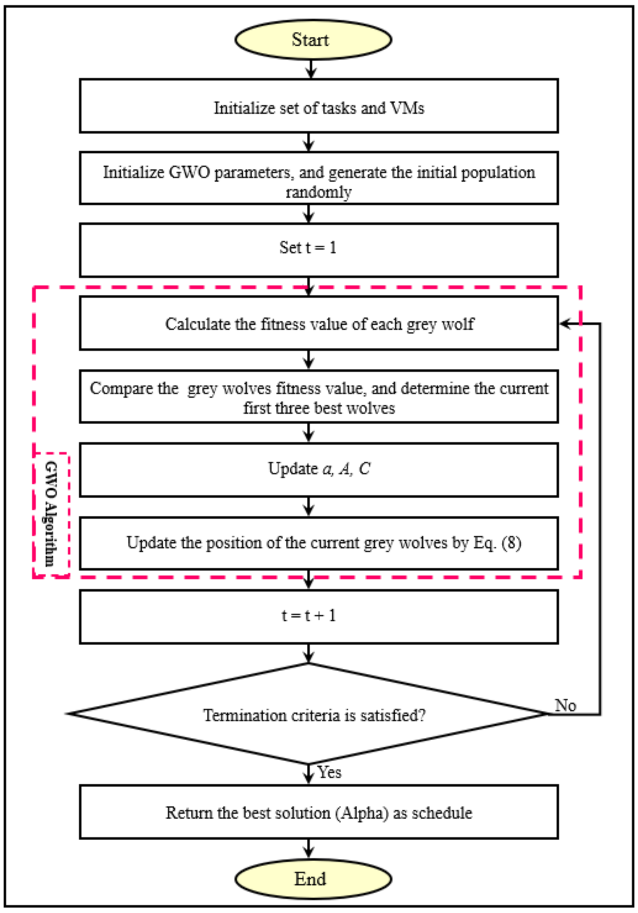
Start
Initialize set of tasks and VMs
Initialize GWO parameters, and generate the initial population randomly
Set t = 1
Calculate the fitness value of each grey wolf
Compare the grey wolves fitness value, and determine the current first three best wolves
Update *a*, *A*, *C*
Update the position of the current grey wolves by Eq. (8)
GWO Algorithm
t = t + 1
Termination criteria is satisfied?
No
Yes
Return the best solution (Alpha) as schedule
End

Figure 5. Flowchart of GWOTS algorithm.

5. Performance Evaluation

In this section, the efficiency of the GWOTS algorithm is evaluated based on various scenarios. In scenario 1, scenario 2, and scenario 3, the performance of the GWOTS algorithm is compared with ten popular meta-heuristic algorithms, Genetic Algorithm (GA) [20], Dragonfly Algorithm (DA) [32], Particle Swarm Optimization (PSO) [23], Whale Optimization Algorithm (WOA) [31], Ant Colony Optimization (ACO) [11], Gravitational Search Algorithm (GSA) [43], Sooty Tern Optimization Algorithm (STOA) [9], Artificial Hummingbird Algorithm (AHA) [48], Multi-Verse Optimizer (MVO) [33], and Sine Cosine Algorithm (SCA) [34]. In order to fairly show that the GWO algorithm is suitable for the scheduling problem and can optimally assign tasks to resources, all proposed meta-heuristic algorithms under equal conditions and in the same objective function (objective function proposed in this paper) have been evaluated. For this reason, GATS (GA-based Task Scheduling), PSOTS (PSO-based Task Scheduling), ACOTS (ACO-based Task Scheduling), DATS (DA-based Task Scheduling), WOATS (WOA-based Task Scheduling), GSATS (GSA-based Task Scheduling), STOATS (STOA-based Task Scheduling), AHATS (AHA-based Task Scheduling), MVOTS (MVO-based Task Scheduling), and SCATS (SCA-based Task Scheduling) are written in the charts and tables. In scenario 4, the performance of the GWOTS is evaluated with the various number of iterations and population size. In scenario 5, the performance of the GWOTS algorithm is compared with the scheduling algorithm, SOATS [17], IWC [8], and CETSA [28]. MATLAB software is used to simulate algorithms.

Scenario 1: In this scenario, the number of tasks varies between 100 and 500, and the number of VMs is fixed. Table 2 shows the parameters set for the cloud environment and the GWO algorithm.

Table 2. Parameter setting for scenario 1.

Parameters	**Values**
Tasks range	100-500
Size of tasks	1000-4000
VMs number	50
VMs processing speed	1000-5000
Maximum iteration	100
Population size	60
α	[2,0]

Makespan Defines the time required to complete the last task and exit the cloud system, which is one of the most important parameters in scheduling. The comparison results between the GWOTS algorithm and other algorithms are represented in Table 3. As the number of tasks in VMs increases, also the makespan is increased. As shown in the table, the GWOTS algorithm decreases makespan by up to 21% compared to GATS, up to 5% compared to PSOTS, up to 32% compared to ACOTS, up to 16%compared to DATS, up to 13% compared to WOATS, up to 16% compared to GSATS, up to 20% compared to STOATS, up to 1% compared AHATS, up to 2% compared to MVOTS, and up to 9% compared to SCATS for 500 number of tasks. This is because the GWO algorithm uses the social dominance hierarchy that exists in grey wolf groups. In other words, this is one of the strengths of the GWO algorithm because it saves the best solutions achieved so far during iteration.

Cloud data centers use significant energy. Increasing energy consumption increases cost and CO_2 emissions. Decreasing energy consumption is a significant issue in the cloud system. Table 4 shows the comparison of energy consumption between various algorithms. As can be seen, the GWOTS algorithm improves efficiency in terms of reducing energy consumption than other algorithms. The energy consumption minimization by GWOTS was 32%–16% less than that of GATS for 100 through 500 tasks, respectively. Moreover, the energy consumption minimization by GWOTS was 53%–23% less than that of ACOTS for 100 through 500 tasks, respectively. Also, the energy consumption minimization by GWOTS is 31%–10% less than that of GSATS for 100 through 500 tasks, respectively. In addition, energy consumption minimization by GWOTS is 24%–14% less than that of STOATS for 100 through 500 tasks, respectively. This is due to the proposed algorithm making a good balance between exploration and exploitation. In other words, the GWO algorithm guarantees the exploration and exploitation ability by the adaptive values of a and A. Parameter a is reduced from 2 to 0 to emphasize exploration and exploitation. Also, the adaptive value of A strikes a balance between exploration and exploitation, so that in the GWO algorithm, half of the iterations are devoted to exploration ($|A| \geq 1$) and the rest to exploitation ($|A| < 1$). This mechanism causes the proposed algorithm to have a good exploration and exploitation ability and also does not fall into the local optimal.

Execution cost is the total amount that the user pays to the service provider based on the number of resources used. The execution cost results between the different algorithms are shown in Table 5. It can be seen that the GWOTS algorithm can improve performance more than all other algorithms in reducing the cost for the different number of tasks. The execution cost in the GWOTS algorithm is 5%, 4%, 8%, 4%, 4%, 6%, 3%, 5%, 1%, and 6% less than GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS for 300 instances of tasks, respectively.

The comparison between the total execution time of the proposed algorithm and other meta-heuristic algorithms for the different number of tasks is performed and the results of this comparison are shown in Table 6. The execution time indicates the amount of time required to complete the execution of the task. The total execution time can be calculated based on the following equation [26]:

$$TExeT = \sum_{j=1}^{m} ExeT_j \tag{16}$$

Where m indicates the number of VMs and $ExeT_j$ is the execution time of the VM_j and it can be computed using Eq. (10).

As can be seen from the results, the proposed algorithm has a better total execution time than other algorithms for a different number of tasks. This is due to the GWO algorithm's ability to detect local optimal and exit local optimal. This feature gives the algorithm the ability to execute tasks faster and reduce total execution time.

Resource utilization refers to the resources utilized for the task scheduling process. If the value of resource utilization is high, the profit of the service provider will be higher. In other words, for more profit, the service provider leases limited resources to users so that the resources are fully utilized. The average resource utilization can be computed using the following equation [35]:

$$Ave\_ReUt(VM_{ij}) = \frac{\sum_{j=1}^{m} ReUt\,(V_{ij})}{m} \tag{17}$$

Where m is the number of VMs, $ReUt(VM_{ij})$ indicates the resource utilization and it can be computed using Eq. (18) [35].

$$ReUt(VM_{ij}) = \frac{CT_{ij}}{MS} \tag{18}$$

Where CT_{ij} is the completion time and MS represent makespan and they can be computed by Eq. (12) and Eq. (9), respectively.

Table 7 shows the comparison results between the GWOTS algorithm and other algorithms in terms of resource utilization. As the results show, the GWOTS algorithm performs better than the other meta-heuristic algorithms by increasing the resource utilization by 30%, 21%, 34%, 23%, 20%, 36%, 50%, 46%, 47%, and 49% while comparing with GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS for 400 instances of task, respectively.

Throughput represents the total number of tasks completed per unit time. High throughput values show that the scheduling algorithm is more efficient and optimally assigns tasks to resources. The average throughput is computed using Eq.(19) [35].

$$Average\_Throughput = \sum_{j=1}^{m} Throughpuut \tag{19}$$

Where m represents the number of VMs and $Throughput$ can be calculated as follows [35]:

$$Throughput = \frac{ReUt(VM_{ij})}{MS} \tag{20}$$

Where $ReUt(VM_{ij})$ indicates resource utilization and MS represents makespan and can be calculated by Eq. (18) and Eq.(9), respectively.

Table 8 shows the results of comparing throughput between different algorithms. As the results clearly show, GWOTS performs better than other algorithms in most cases. The throughput in the GWOTS algorithm is 49%, 32%, 59%, 23%, 43%, 72%, 72%, 76%, 68%, and 76% more than GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS, respectively. This is because the GWO algorithm uses its parameters to transition smoothly between exploration and exploitation.

Load balance is one of the most important factors in scheduling in the field of cloud computing. In scheduling, there can be the condition when VMs may execute more than one task. An efficient scheduling algorithm must distribute the load evenly between VMs and prevent overload and underload of VMs. The degree of load balance can be calculated using the following equation [26]:

$$\varpi = \frac{\sqrt{\frac{\sum_{j=1}^{m}(ExeT_j - mExeT_j)^2}{m}}}{n} \tag{21}$$

Where n indicates the number of tasks, m represents the number of VMs, $ExeT_j$ and $mExeT_j$ indicate execution time VM_j and mean execution time VM_j, respectively.

The results of comparisons between various algorithms are shown in Table 9. The results clearly show that the GWOTS algorithm has a better degree of resource load balance in most cases compared to other algorithms.

To evaluate the results, the Wilcoxon statistical test, which is one of the non-parametric tests and is used in statistical analysis, has been used [47]. Table 10 shows the results obtained from the Wilcoxon rank-sum. In other words, Table 10 shows the Wilcoxon rank-sum test results between the GWOTS algorithm and other algorithms that take into account the number of wins. In Table 10, "+" indicates that the comparison algorithm performed worse than GWOTS, and "-" indicates that the comparison algorithm performed better than GWOTS. According to the obtained results, it can be seen that the proposed algorithm has the best performance and in comparison with GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS algorithms, it has won 5, 5, 5, 5, 5, 5, 5, 4, 5, and 5, respectively. This is due to the balance between exploration and exploitation in the GWO algorithm and that this algorithm does not fall in local optimal.

Table 3. Makespan comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**3.18**	4.72	4.12	7.47	3.62	3.83	4.5	4.04	3.59	3.2	4.2
200	**5.1**	8.63	6.75	11.58	6.33	6.14	6.75	6.55	5.58	5.42	6.75
300	**8.53**	12.51	10.77	14.23	10.06	11.77	12.29	9.91	9.58	8.62	12.22
400	**11.41**	17.52	15.11	18.88	15.95	15.07	15.87	15.97	13.38	12.88	15.32
500	**16.63**	20.94	17.54	24.49	19.7	19.09	19.79	20.81	16.72	16.92	18.34
Number of wins(Grade)	5(1)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)

Table 4. Energy consumption comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**111676**	163185	151058	236977	132482	128918	161687	149087	129720	120585	150001
200	210803	297434	250592	394193	225871	**207072**	250592	241472	212439	212670	250592
300	305270	421351	374679	445211	344571	384715	404545	336535	334280	**302195**	402229
400	**421665**	575545	503534	586661	531660	497080	539099	505571	440000	470411	528768
500	597661	709245	634172	778869	686082	650940	660562	695216	**584688**	598555	668618
Number of wins(Grade)	2(1)	0(3)	0(3)	0(3)	0(3)	1(2)	0(3)	0(3)	1(2)	1(2)	0(3)

Table 5. Execution cost comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**0.162**	0.172	0.169	0.179	0.165	0.164	0.168	0.170	0.165	0.168	0.171
200	**0.164**	0.170	0.167	0.174	0.169	0.165	0.167	0.169	0.166	0.165	0.167
300	**0.169**	0.179	0.177	0.184	0.175	0.177	0.180	0.173	0.178	0.171	0.180
400	**0.175**	0.183	0.180	0.184	0.182	0.180	0.182	0.184	0.180	0.176	0.180
500	0.178	0.181	0.179	0.185	0.177	0.180	0.179	0.179	**0.174**	0.176	0.178
Number of wins(Grade)	4(1)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	1(2)	0(3)	0(3)

Table 6. Total execution time comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**80.28**	89.40	86.00	93.46	82.76	81.81	107.68	114.37	104.86	109.99	104.55
200	**164.26**	175.25	170.96	182.07	174.21	167.15	217.35	208.48	211.51	203.39	217.35
300	**261.77**	288.28	283.19	300.86	279.16	283.01	320.27	321.03	324.41	302.71	331.63
400	**373.08**	402.52	391.07	407.27	400.79	391.76	427.57	421.93	424.15	412.15	422.18
500	**487.01**	500.04	491.87	519.14	492.04	497.20	492.32	491.50	490.54	495.97	497.16
Number of wins(Grade)	5(1)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)	0(2)

Scenario 2: In this scenario, the simulation is performed based on the various number of VMs with a fixed number of tasks. The simulation parameters used in this scenario are represented in Table 11.

Table 7. Resource utilization comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**0.504**	0.379	0.418	0.250	0.458	0.427	0.290	0.254	0.245	0.295	0.253
200	**0.644**	0.406	0.507	0.314	0.551	0.544	0.255	0.372	0.386	0.344	0.255
300	**0.614**	0.461	0.526	0.423	0.555	0.481	0.362	0.359	0.333	0.371	0.337
400	**0.654**	0.460	0.518	0.431	0.503	0.520	0.417	0.326	0.350	0.346	0.333
500	0.680	0.478	0.561	0.424	0.604	0.521	0.497	0.472	**0.688**	0.591	0.531
Number of wins(Grade)	4(1)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	1(2)	0(3)	0(3)

Table 8. Throughput comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**7.92**	4.02	5.07	1.68	6.33	5.57	1.96	1.41	1.43	1.98	1.54
200	**6.31**	2.35	3.75	1.36	4.35	4.43	0.75	1.66	1.76	1.45	0.75
300	**3.59**	1.84	2.44	1.49	2.76	2.04	1.02	1.00	0.85	1.13	0.86
400	**2.86**	1.31	1.71	1.14	1.58	1.73	1.02	0.63	0.72	0.73	0.66
500	1.96	1.14	1.60	0.87	1.94	1.36	1.26	1.13	**2.13**	1.82	1.45
Number of wins(Grade)	4(1)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	1(2)	0(3)	0(3)

Table 9. Degree of resource load balance comparison (various number of tasks).

Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	**0.011**	0.013	0.014	0.016	0.012	0.013	0.019	0.024	0.019	0.017	0.018
200	**0.013**	0.016	0.015	0.018	0.015	0.014	0.022	0.019	0.020	0.018	0.022
300	**0.015**	0.018	0.018	0.020	0.017	0.017	0.022	0.021	0.022	0.019	0.023
400	**0.017**	0.020	0.019	0.020	0.019	0.019	0.022	0.022	0.023	0.021	0.022
500	0.019	0.020	0.019	0.022	0.018	0.020	0.019	0.020	**0.017**	0.018	0.019
Number of wins(Grade)	4(1)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	0(3)	1(2)	0(3)	0(3)

Figures 6-12 show the performance of the GWOTS algorithm in comparison to other algorithms based on various criteria for the various number VMs.

It is very important to calculate the makespan parameter because it will help to meet the task deadline. As shown in Fig. 6, makespan decreases with the increasing number of VMs. The X-axis represents the number of VMs and the Y-axis represents the makespan. The GWOTS improves efficiency in terms of makespan minimization than other algorithms. This is because the

Table 10. Sum win and Wilcoxon rank-sum for results of algorithms.

Sum win											
Number of tasks	**Algorithm**										
	GWOTS	**GATS**	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	7	0	0	0	0	0	0	0	0	0	0
200	6	0	0	0	0	1	0	0	0	0	0
300	6	0	0	0	0	0	0	0	0	1	0
400	7	0	0	0	0	0	0	0	0	0	0
500	2	0	0	0	0	0	0	0	5	0	0
Sum	28	0	0	0	0	1	0	0	5	1	0
Grade	(1)	(4)	(4)	(4)	(4)	(3)	(4)	(4)	(2)	(3)	(4)

Wilcoxon rank-sum										
Number of tasks	**Algorithm**									
	GATS	**PSOTS**	**ACOTS**	**DATS**	**WOATS**	**GSATS**	**STOATS**	**AHATS**	**MVOTS**	**SCATS**
100	+	+	+	+	+	+	+	+	+	+
200	+	+	+	+	+	+	+	+	+	+
300	+	+	+	+	+	+	+	+	+	+
400	+	+	+	+	+	+	+	+	+	+
500	+	+	+	+	+	+	+	-	+	+
Sum	5	5	5	5	5	5	5	4	5	5

Table 11. Parameter setting for scenario 2.

Parameters	**Values**
Tasks range	300
Size of tasks	1000-4000
VMs number	20-50
VMs processing speed	1000-5000
Maximum iteration	100
Population size	60
α	[2,0]

GWO algorithm defines a circular neighborhood around solutions based on its encirclement mechanism. The A and C parameters in the GWO algorithm help the candidate solutions to have hyper-spheres with various random radii. This causes the GWO algorithm to have better performance than other algorithms and achieves better solutions.

Energy consumption is one of the important parameters that have an important role in system efficiency. Since suitable resource utilization will reduce energy consumption, so there is a close relationship between energy consumption and resource utilization. As shown in Fig. 7, the presented algorithm performs better in reducing energy consumption than other algorithms. The GWOTS algorithm improvement rates are 35%, 21%, 28%, 16%, 21%, 29%, 18%, 9%, 24%, and 28% over GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS for 40 VMs, respectively. The results represent that the performance of the GWOTS algorithm using the various number of VMs is not negatively affected.

One of the goals of the GWOTS is to allocate tasks to VMs in a way that reduces execution costs. Figure 8 shows the execution costs for different algorithms with variable numbers of VMs and a fixed number of tasks. The proposed algorithm has a lower execution cost than other algorithms. The execution cost minimization by the GWOTS algorithm was 4%–5% less than that of GATS for 20 through 50 VMs, respectively. In addition, the execution cost minimization by GWOTS was 6%–8% less than that of ACOTS for 20 through 50 VMs, respectively. Also, the execution cost minimization by GWOTS was 1%–6% less than that of GSATS for 20 through 50 VMs, respectively. Moreover, the execution cost minimization by GWOTS was 1%–5% less than that of AHATS for 20 through 50 VMs, respectively. This shows that the GWO algorithm can allocate tasks to resources more efficiently and reduce execution costs compared to other algorithms.

Figure 9 shows the total execution time for different algorithms with various numbers of VMs. As clearly shown in Fig. 9, the proposed scheduling algorithm has obvious advantages in achieving a minimum total execution time compared to the algorithms. The total execution time in the GWOTS algorithm is 9% less than that of GATS, 8% that of PSOTS, 13% that of ACOTS, 6% that of DATS, 8% that of WOATS, 10% that of GSATS, 4% that of STOATS, 8% that of AHATS, 2% that of MVOTS, and 11% that of SCATS for 50 number of VMs.

Figure 10 shows the results of comparing resource utilization between the GWOTS algorithm and other algorithms with the various number of virtual VMs. The resource utilization is influenced by the makespan. Because according to Eq. (18), resource utilization is inversely related to makespan. It can be clearly seen that when there are various numbers of VMs with a fixed number of tasks, the GWOTS algorithm performs better in terms of resource utilization.

Figure 11 shows the performance of various algorithms in terms of throughput. The X-axis indicates the number of VMs, while the Y-axis shows the throughput. It can be seen, that the GWOTS algorithm has better performance than the other algorithms by increasing the throughput by 64%, 49%, 76%, 12%, 46%, 22%, 39%, 17%, 21%, and 43% while comparing with GATS, PSOTS, ACOTS, DATS, WOATS, GSATS, STOATS, AHATS, MVOTS, and SCATS for 20 VMS, respectively. This is because GWO does not fall into the local optimal trap and uses adaptive values of parameters$a$ and A to slowly switch between exploration and exploitation.

Load balancing, which is one of the most important parameters in scheduling, indicates the load distribution between VMs. The results of comparing the degree of load balance between different algorithms are shown in Fig. 12. The proposed algorithm optimally assigns tasks to resources and distributes the load efficiently among VMs. The results represent that the GWOTS can be efficiently used for task scheduling with the various number of VMs and a fixed number of tasks.

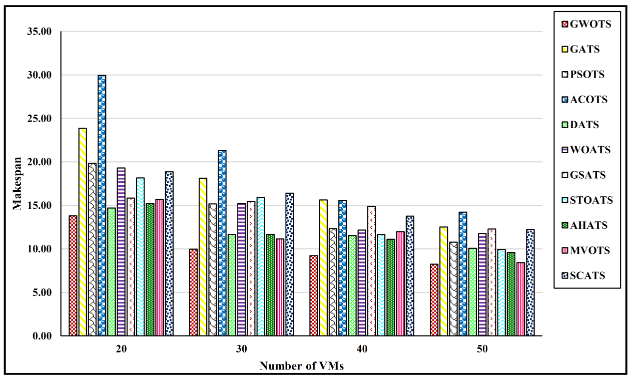

FIGURE 6. Makespan comparison (various numbers of VMs).

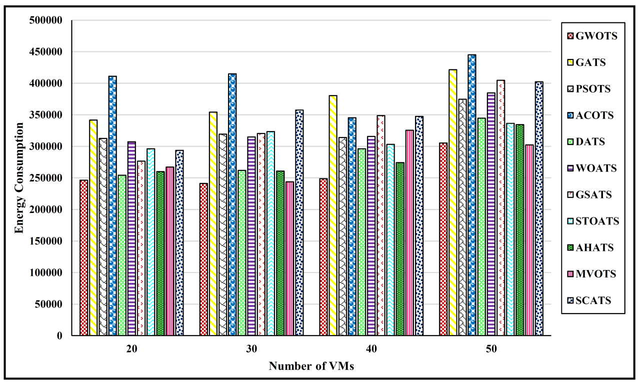

FIGURE 7. Energy consumption comparison (various numbers of VMs).

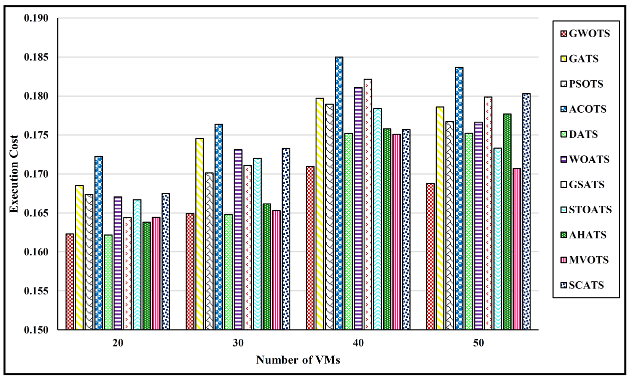

FIGURE 8. Execution cost comparison (various numbers of VMs).

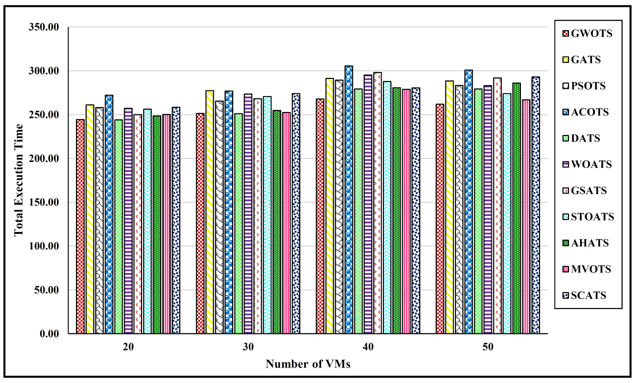

FIGURE 9. Total execution time comparison (various numbers of VMs).

Scenario 3: In this scenario, the performance of the GWOTS algorithm is evaluated with other algorithms based on increasing the number of iterations and the fitness value. Table 12 represents the parameter setting for this scenario.

The convergence speed of the presented algorithm and the other algorithms are shown in Fig. 13. As can be seen, the presented algorithm performs better compared to other algorithms. As shown in the figure, the disadvantage of the GA and ACO algorithms is that these algorithms have poor exploitation ability. Also, GA and ACO are exploring solution space during different iterations and do not converge to the optimal solution. In these algorithms, there is no balance between exploration and exploitation. Other algorithms (i.e., PSO,

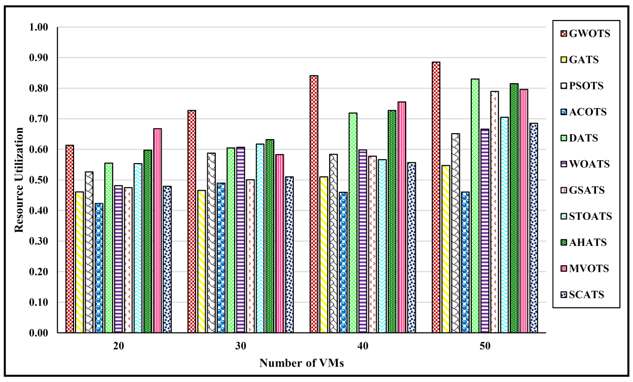

Figure 10. Resource utilization comparison (various numbers of VMs).

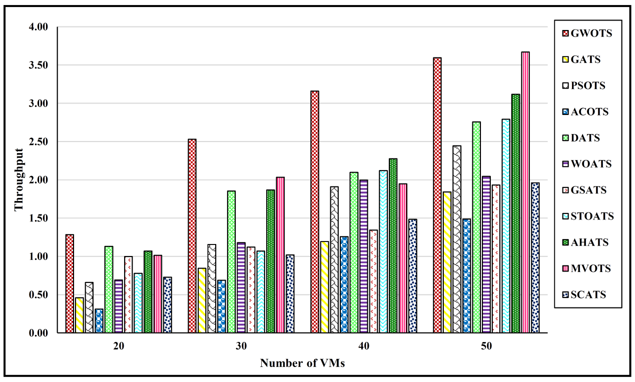

Figure 11. Throughput comparison (various numbers of VMs).

Table 12. Parameter setting for scenario 3.

Parameters	**Values**
Tasks range	100
Size of tasks	1000-4000
VMs number	20
VMs processing speed	1000-5000
Maximum iteration	100
Population size	50
α	[2,0]

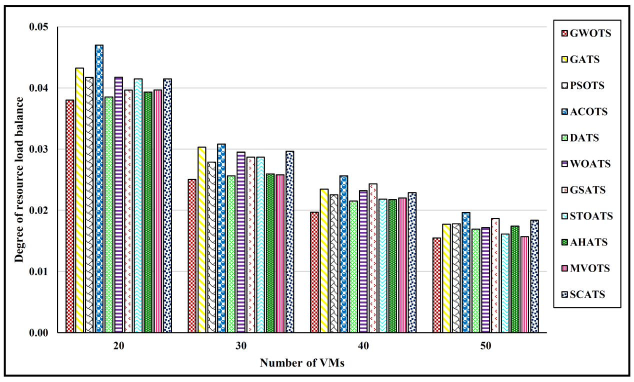

Figure 12. Degree of resource load balance comparison (various numbers of VMs).

DA, WOA, SCA, MVO, AHA, GSA, STOA) explore the search space in the first iterations, but in later iterations, they fall into the trap of local optimal and cannot converge to the global optimal. While the fitness value of the GWO algorithm reduces with the increasing number of iterations. This is because GWO represents a good trade-off between exploration and exploitation. In other words, in the first iterations, GWO tries to search the entire search space and avoids falling into the trap of the local optimal, and in the last iterations, GWO tries to converge to the best global optimal solution. This superior ability is because of the adaptive value of A.

Figure 14 compares the time consumed between different algorithms. GWO takes less time than most algorithms. Also, results indicate that DA takes longer than other algorithms.

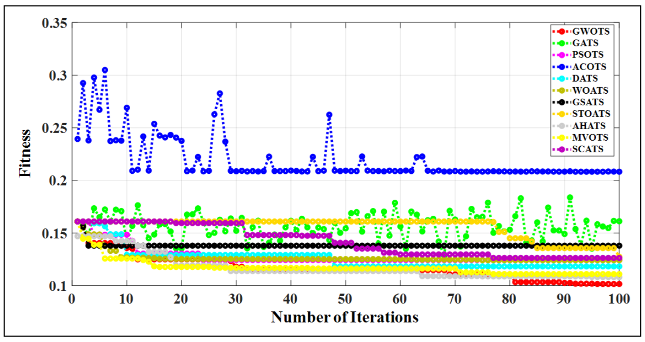

Figure 13. Convergence of the different algorithms.

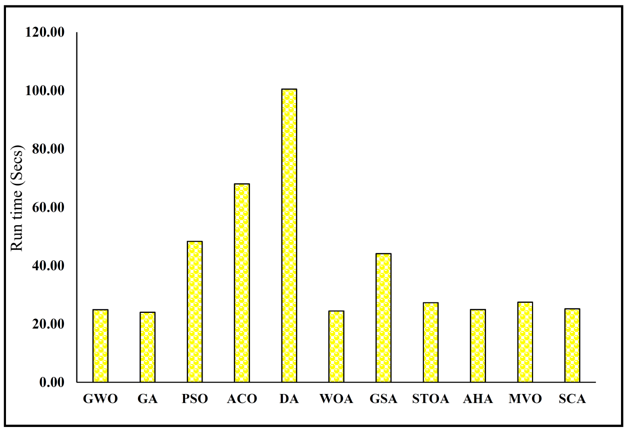

Figure 14. Time comparison between different algorithms.

Scenario 4: In this scenario, the fitness value is obtained in which the tasks number, as well as VMs number, are constant while the population size, as well as the number of iterations, are variable. The simulation parameters used for this scenario are represented in Table 13.

Table 13. Parameter setting for scenario 4.

Parameters	**Values**
Tasks range	200
Size of tasks	1000-4000
VMs number	30
VMs processing speed	1000-5000
Maximum iteration	60-100
Population size	40-80
α	[2,0]

As shown in Fig. 15, the number of iterations starts at 60 and increases to 100 iterations. Also, the number of agents started from 40 and has increased to 80 agents. The GWOTS algorithm can achieve the best fitness value with 100 iterations and 60 agents. As the number of agents increases, more parts of the search space are searched. But it is significant to note that increasing the number of agents increases computational complexity and is time-consuming. Also, with fewer agents, the solution may not be appropriate because the fitness value is low.

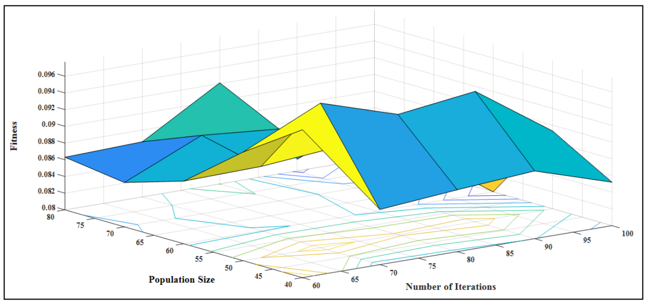

Figure 15. Fitness value for the various number of agents and iterations.

Scenario 5: In the last scenario, a comparison is made between the performance of the GWOTS algorithm and the SOATS [17], IWC [8], and CETSA [28] algorithms. The parameter settings are shown in Table 14.

Table 14. Parameter setting for scenario 5.

Parameters	**Values**
Tasks range	500
Size of tasks	1000-4000
VMs number	50
VMs processing speed	1000-5000
Maximum iteration	100
Population size	60
α	[2,0]

Table 15 shows the results of comparisons between the GWOTS algorithm and other scheduling algorithms in terms of makespan, cost, energy consumption, total execution time, resource utilization, throughput, and degree of resource load balance.

- SOATS [17]: They proposed an SOA-based Task Scheduling algorithm and called it SOATS. The purpose of the SOATS algorithm is to create a trade-off between cost, energy consumption, waiting time, makespan, and load balance.
- IWC [8]: They presented an Improved WOA (IWC) algorithm to improve the search ability of the WOA. Then, a scheduling algorithm is offered to improve the performance of the cloud system based on the IWC.
- CETSA [28]: They introduced a Cost and Energy-aware Task Scheduling Algorithm and called it CETSA. The purpose of the CETSA is

to reduce makespan, cost, and energy consumption, while the load on the resources is considered to prevent overloading of resources.

Experimental results clearly show that GWOTS performed better than other scheduling algorithms. This is because the GWOTS algorithm considers three parameters, makespan, energy, and execution cost simultaneously and finds the optimal solution using the GWO algorithm, which is a powerful optimizer, and thus GOWTS has better efficiency.

Table 15. Comparison between different scheduling algorithms.

Objectives/ Methods	**Makespan**	**Energy Consumption**	**Execution Cost**	**Total Execution Time**	**Resource Utilization**	**Throughput**	**Degree of resource load balance**
GWOTS	16.63	597661	0.178	487.01	0.680	1.96	0.019
SOATS	30.9818	918800	0.184	514.46	0.332	0.536	0.022
IWC	31.55	919562	0.188	533.65	0.338	0.536	0.023
CETSA	28.3	631162	0.18	496.99	0.41	1	0.02

6. Conclusion and Future work

Cloud computing allows companies and end-users to consume virtual resources based on a pay-per-use model. In cloud computing, adopting a proper task scheduling algorithm that optimally maps tasks to VMs is critical for both users and service providers. Therefore, this paper presents a task scheduling algorithm based on the popular nature-inspired GWO algorithm. The proposed algorithm has been offered to simultaneously optimize three important objectives, namely: minimizing makespan, minimizing execution cost, and minimizing energy consumption. The simulation results represent the effective performance of the presented algorithm than other algorithms. Experimental outcomes show that the GWOTS algorithm for the various number of tasks and VMs improves makespan, energy consumption, cost, total execution time, resource utilization, throughput, and degree of resource load balance compared to GA, PSO, ACO, DA, WOA, GSA, STOA, AHA, MVO, SCA. The proposed algorithm can also avoid falling into the local optimal trap and has better convergence compared to other algorithms. In addition, the results of the comparison between GWOTS and other scheduling algorithms (i.e., SOATS, IWC, CETSA) show that GWOTS has better performance. For future work, we intend to consider other important parameters such as security, scalability, and availability. We also intend to enhance the efficiency of the GWO algorithm

References

[1] B.H. Abed-Alguni, N.A. Alawad, *Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments*, Applied Soft Computing. 102 (2021).

[2] M.S. Ajmal, Z. Iqbal, F.Z. Khan, M. Ahmad, I. Ahmad, B.B. Gupta, *Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers*, Computers and Electrical Engineering. 95 (2021).

[3] D. Alboaneen, H. Tianfield, Y. Zhang, B. Pranggono, *A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers*, Future Generation Computer Systems. 115 (2021) 201–212.

[4] K. Alzhrani, F. Alotaibi,*Ensuring Security and Privacy for Cloud-based E-Services*, International Journal of Computer Applications. 149 (2016) 8–13.

[5] S.A. Alsaidy, A.D. Abbood, M.A. Sahib, *Heuristic initialization of PSO task scheduling algorithm in cloud computing*, Journal of King Saud University - Computer and Information Sciences. (2020).

[6] N. Arora, R.K. Banyal, *A Particle Grey Wolf Hybrid Algorithm for Workflow Scheduling in Cloud Computing*, Wireless Personal Communications. (2021) 1–33.

[7] S.A. Bello, L.O. Oyedele, O.O. Akinade, M. Bilal, J.M. Davila Delgado, L.A. Akanbi, A.O. Ajayi, H.A. Owolabi, *Cloud computing in construction industry: Use cases, benefits and challenges*, Automation in Construction. 122 (2021).

[8] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, J. Murphy, *A woa-based optimization approach for task scheduling in cloud computing systems*, IEEE Systems Journal. 14 (2020) 3117–3128.

[9] G. Dhiman, A. Kaur, *STOA: a bio-inspired based optimization algorithm for industrial engineering problems*, Engineering Applications of Artificial Intelligence. 82 (2019) 148–174.

[10] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, *A survey on new generation metaheuristic algorithms*, Computers and Industrial Engineering. 137 (2019).

[11] M. Dorigo, V. Maniezzo, A. Colorni, *Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems*, Man, and Cybernetics, Part B (Cybernetics). 26 (1996) 29–41.

[12] K. Dubey, S.C. Sharma, *A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing*, Sustainable Computing: Informatics and Systems. 32 (2021).

[13] H. Emami, *Cloud task scheduling using enhanced sunflower optimization algorithm*, ICT Express. (2021).

[14] M.A. Elaziz, S. Xiong, K.P.N. Jayasena, L. Li, *Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution*, Knowledge-Based Systems. 169 (2019) 39–52.

[15] H. Faris, I. Aljarah, M.A. Al-Betar, S. Mirjalili, *Grey wolf optimizer: a review of recent variants and applications*, Neural Computing and Applications. 30 (2018) 413–435.

[16] X. Fu, Y. Sun, H. Wang, H. Li, *Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm*, Cluster Computing. (2021).

[17] R. Ghafari, N. Mansouri, *An Efficient Task Scheduling Based on Seagull Optimization Algorithm for Heterogeneous Cloud Computing Platforms*, International Journal of Engineering. 35 (2022) 433–450.

[18] R. Ghafari, F.H. Kabutarkhani, N. Mansouri, *Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review*, Cluster Computing. (2022).

[19] X. Guo, *Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm*, Alexandria Engineering Journal. 60 (2021) 5603–5609.

[20] J.H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

[21] E.H. Houssein, A.G. Gad, Y.M. Wazery, P.N. Suganthan, *Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges*, and Future Trends, Swarm and Evolutionary Computation. 62 (2021).

[22] L. Imene, S. Sihem, K. Okba, B. Mohamed, *A third generation genetic algorithm NSGAIII for task scheduling in cloud computing*, Journal of King Saud University-Computer and Information Sciences. (2022).

[23] J. Kennedy, R. Eberhart, *Particle swarm optimization*, in: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE, 1995: pp. 1942–1948.

[24] J.K. Konjaang, L. Xu, *Meta-heuristic Approaches for Effective Scheduling in Infrastructure as a Service Cloud: A Systematic Review*, Journal of Network and Systems Management. 29 (2021).

[25] N. Manikandan, N. Gobalakrishnan, K. Pradeep, *Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment*, Computer Communications. 187 (2022) 35–44.

[26] N. Mansouri, B.M.H. Zade, M.M. Javidi, *Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory*, Computers and Industrial Engineering. 130 (2019) 597–633.

[27] N. Mansouri, R. Ghafari, B.M.H. Zade, *Cloud computing simulators: A comprehensive review*, Simulation Modelling Practice and Theory. 104 (2020) 102144.

[28] N. Mansouri, R. Ghafari, *Cost-efficient task scheduling algorithm to reduce energy consumption and makespan of cloud computing*, Computer and Knowledge Engineering. (2022).

[29] Y. Meraihi, A.B. Gabis, A. Ramdane-Cherif, D. Acheli, *A comprehensive survey of Crow Search Algorithm and its applications*, Artificial Intelligence Review. 54 (2021) 2669–2716.

[30] S. Mirjalili, S.M. Mirjalili, A. Lewis, *Grey wolf optimizer*, Advances in Engineering Software. 69 (2014) 46–61.

[31] S. Mirjalili, A. Lewis, *The whale optimization algorithm*, Advances in Engineering Software. 95 (2016) 51–67.

[32] S. Mirjalili, *Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems*, Neural Computing and Applications. 27 (2016) 1053–1073.

[33] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, *Multi-verse optimizer: a nature-inspired algorithm for global optimization*, Neural Computing and Applications. 27 (2016) 495–513.

[34] S. Mirjalili, *SCA: a sine cosine algorithm for solving optimization problems*, Knowledge-Based Systems. 96 (2016) 120–133.

[35] K. Mishra, J. Pati, S.K. Majhi, *A dynamic load scheduling in IaaS cloud using binary JAYA algorithm*, Journal of King Saud University-Computer and Information Sciences. (2020).

[36] S.K. Mishra, B. Sahoo, P.P. Parida, *Load balancing in cloud computing: a big picture*, Journal of King Saud University-Computer and Information Sciences. 32 (2020) 149–158.

[37] B. Mohammad Hasani Zade, N. Mansouri, M.M. Javidi, *SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment*, Expert Systems with Applications. 176 (2021).

[38] A. Mohammadzadeh, M. Masdari, F.S. Gharehchopogh, A. Jafarian, *Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing*, Evolutionary Intelligence. 14 (2021) 1997–2025.

[39] R. NoorianTalouki, M. Hosseini Shirvani, H. Motameni, *A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms*, Journal of King Saud University - Computer and Information Sciences. (2021).

[40] S.K. Panda, P.K. Jana, *An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems*, Cluster Computing. 22 (2019) 509–527.

[41] A. Pradhan, S.K. Bisoy, A. Das, A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment, Journal of King Saud University - Computer and Information Sciences. (2021).

[42] T. Prem Jacob, K. Pradeep, A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization, Wireless Personal Communications. 109 (2019) 315–331.

[43] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Information Sciences. 179 (2009) 2232–2248.

[44] A.M. Senthil Kumar, M. Venkatesan, Task scheduling in a cloud computing environment using HGPSO algorithm, Cluster Computing. 22 (2019) 2179–2185.

[45] H. Singh, S. Tyagi, P. Kumar, S.S. Gill, R. Buyya, Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions, Simulation Modelling Practice and Theory. 111 (2021).

[46] S. Velliangiri, P. Karthikeyan, V.M. Arul Xavier, D. Baswaraj, Hybrid electro search with genetic algorithm for task scheduling in cloud computing, Ain Shams Engineering Journal. 12 (2021) 631–639.

[47] T. Wang, P. Zhang, J. Liu, M. Zhang, Many-objective cloud manufacturing service selection and scheduling with an evolutionary algorithm based on adaptive environment selection strategy, Applied Soft Computing. 112 (2021).

[48] W. Zhao, L. Wang, S. Mirjalili, Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications, Computer Methods in Applied Mechanics and Engineering. 388 (2022) 114194.

Reyhane Ghafari
Orcid number:0000-0002-3551-7523
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: Reihaneh.ghafary@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