International Journal of Engineering

Journal Homepage: www.ije.ir

An Efficient Task Scheduling Based on Seagull Optimization Algorithm for Heterogeneous Cloud Computing Platforms

R. Ghafari, N. Mansouri*

Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran

P A P E R I N F O

Paper history:
Received 18 August 2021
Received in revised form 19 November 2021
Accepted 29 November 2021

Keywords:
Cloud Computing
Task Scheduling
Meta-heuristic
Seagull Optimization

A B S T R A C T

Cloud computing provides computing resources like software and hardware as a service by the network for several users. Task scheduling is one of the main problems to attain cost-effective execution. The main purpose of task scheduling is to allocate tasks to resources so that it can optimize one or more criteria. Since the problem of task scheduling is one of the Nondeterministic Polynomial-time (NP)-hard problems, meta-heuristic algorithms have been widely employed for solving task scheduling problems. One of the new bio-inspired meta-algorithms is Seagull Optimization Algorithm (SOA). In this paper, we proposed an energy-aware and cost-efficient SOA-based Task Scheduling (SOATS) algorithm. The aims of proposed algorithm to make a trade-off between five objectives (i.e., energy consumption, makespan, cost, waiting time, and load balancing) using a fewer number of iterations. The experiment results by comparing with several meta-heuristic algorithms (i.e., Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Whale Optimization Algorithm (WOA)) prove that the proposed technique performs better in solving task scheduling problem. Moreover, we compared the proposed algorithm with well-known scheduling methods: Cost-based Job Scheduling (CJS), Moth Search Algorithm based Differential Evolution (MSDE), and Fuzzy-GA (FUGE). In the heavily loaded environment, the SOATS algorithm improved energy consumption and cost saving by 10 and 25%, respectively.

***doi**: 10.5829/ije.2022.35.02b.20*

1. INTRODUCTION

In the era of technology, cloud computing is developing as a technology that dynamically provides the infrastructure to end users [1]. One of the most extensive areas of research in cloud computing is task scheduling. The main challenge in task scheduling is finding the optimal resource for input tasks. In single task scheduling, solely one parameter is taken into account, while in multi-objective task scheduling, two or more criteria are taken into account as one objective [2]. Researchers have used various kinds of task scheduling strategies. However, meta-heuristic scheduling has better results than traditional heuristic scheduling. Most existing task scheduling algorithms are more concerned with achieving better task execution time. In the cloud environment, not only we should consider the completion time, but also pay attention to the other Quality of Service (QoS) factors (e.g., costs and energy consumption). Among the existing meta-heuristic algorithms, Seagull Optimization Algorithm (SOA) [3] is one of the meta-heuristic algorithms used to solve optimization problems. In this paper, we present a task scheduling algorithm based on SOA, which takes into account several important parameters, namely energy consumption, cost, waiting time, load balancing, and makespan at the same time.

1. 1. Cloud Computing Cloud computing is known as a popular paradigm of business computing. Cloud computing can suggest to users the different computing services such as applications, servers, storage, and networks using the Virtual Machine (VM) over the internet [4]. Cloud computing can speed up the prediction process by utilization of high-speed computing. In the case of the COVID-19 epidemic, a prediction scheme

*Corresponding Author Insitiutional Email: n.mansouri@uk.ac.ir (N. Mansouri) based on the machine learning model could be used in remote cloud nodes for real-time prediction permitting governments and citizens to reply proactively [5].

As shown in Figure 1, cloud computing has five basic characteristics: on-demand self-service, resource pooling, rapid elasticity, broad network access, and measured services. The cloud has three service models: Software as a Service (SaaS) is cloud-based construction software that can be purchased for use on a pay-as-you-go basis; therefore, decreasing the cost of ownership, Infrastructure as a Service (IaaS) which presents infrastructure services such as storage systems, and computing resources, and Platform as a Service (PaaS) which can be procured to integrate databases from various project data generated by the various professionals on-site and those in the back office. Cloud services can be deployed as a private service, a public service, a community service, or a hybrid service depending on the access method as well as the classification of eligible users to access the service.

Cloud service providers sell resources to users as virtual resources. Users use these resources and execute tasks. Task scheduling is one of the most important applications used by end-users and cloud service providers [7]. One of the most challenging problems in task scheduling is finding the optimal resource for input tasks [8].

1. 2. Task Scheduling The problem of task planning is to schedule a set of specific tasks in a specific set of resources in the form of VMs that have limitations for optimizing some objective functions [9]. Figure 2 shows a model of task scheduling in the cloud environment. The Datacenter Broker (DB) is responsible for identifying and collecting all information about available resources (VMs) and any residual resource that may be available in the future, which collects this information with the aid of the Cloud Information Services (CIS). The interface between the host operating system and the VMs is a hypervisor. Tasks are sent to the task queue to be scheduled for VMs according to the scheduling algorithm defined in the DB.

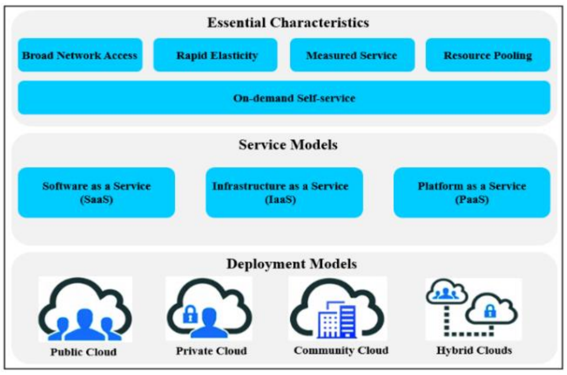

Figure 1. Cloud computing definition [6]

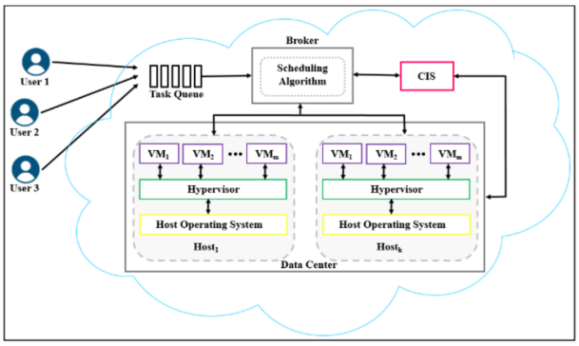

Figure 2. Task scheduling model in the cloud [10]

Task scheduling is an NP-hard optimization problem because the number of tasks increases and the length of the task varies rapidly [11]. In cloud computing, task scheduling efficiency is measured by different system performance criteria. In general, these cloud-based optimization metrics can be categorized into the goals of cloud users and the goals of cloud service providers. On the one hand, some metrics such as makespan and waiting time are user metrics. On the other hand, metrics such as the cost of the provider and energy consumption are the metrics of the provider [12, 13]. The popularity of cloud computing is growing day by day, so with an increasing demand for high-performance computing resources, energy consumption in the cloud data center is greatly increased [14]. Energy consumed by computing resources and connected cooling facilities is the main component of energy costs and high carbon emissions. According to research conducted by Uchechukwu and Shen [15], it is estimated that energy consumption by data centers around the world is about 1.4% of electricity consumption worldwide and is growing at a rate of 12% annually. The energy consumption of processing units is approximately 42%, cooling facilities about 15.4%, and storage facilities nearly 14.3% [16]. As a result, one of the main concerns in cloud computing is how to decrease energy consumption and related costs while keeping execution performance. Minimizing energy consumption improves overall efficiency and also increases system reliability and availability [17]. In other words, minimizing energy consumption while ensuring the user's QoS preferences is critical to achieving maximum profit for service providers and ensuring the user's service level agreement (SLA). Moreover, minimizing energy consumption decreases energy costs as well as aiding to protect our natural environment because it decreases carbon emissions [18]. In addition to energy consumption, the cost of a cloud provider can be reduced by assigning the task to a suitable VM that executes the task at minimal cost and without violating QoS restrictions [19], which have not been addressed in most papers. Thus, efficient resource management is the key to balancing performance and cloud costs while keeping service availability. We need a suitable task scheduling algorithm to find a trade-off between user goals (such as reducing makespan) and service provider goals (such as reducing energy consumption and cost). To solve such a problem, a large number of researchers focused their research work on heuristic, meta-heuristic, and hybrid scheduling algorithms [20, 21]. Currently, swarm intelligence algorithms are widely used to solve these types of problems.

1. 3. Meta-heuristic Algorithm The task scheduling problem in cloud computing is known as an NP-hard problem because of the large space of solutions. Therefore, we need a long time to discover an optimal solution [22]. It is possible to reach a near-optimal solution in a short time for such problems by using meta-heuristic strategies [23]. One of the advantages of meta-heuristic algorithms is that they are problem-independent and have a good approach to solve problems in different domains [24]. There are a variety of meta-heuristic algorithms. As shown in Figure 3, the bio-inspired meta-heuristic algorithms can generally be divided into three main categories [25]: evolution-based methods (are inspired by the laws of natural evolution), swarm-based methods (imitate the social behavior of groups of animals), and bacterial foraging methods (inherit the characteristics of bacterial foraging patterns).

Swarm intelligence is one of the attractive branches of population-based meta-heuristic algorithms. Concepts of swarm intelligence were first introduced in 1993 [26]. Swarm intelligence strategies mimic the social behaviors of organisms living in colonies, flocks, or herds [27]. Among the most popular swarm intelligence strategies are Particle Swarm Optimization (PSO) [28] and Ant Colony Optimization (ACO) [29]. One of the meta-heuristic algorithms that have been introduced in recent years is the SOA [3] to solve expensive computational problems. The principal inspiration of the SOA is the migration and attacking behavior of seagulls in nature.

The SOA starts by generating a random initial population. Search agents update their positions according to the best search agent during different iterations. Seagulls explore various promising areas of the search space. At the beginning of the optimization process, the search agents vary quickly. The experimental results are obtained by comparing SOA with other popular meta-heuristic algorithms (e.g., Spotted Hyena Optimizer (SHO), Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Moth-Flame Optimization (MFO), Multi-Verse Optimizer (MVO), Sine Cosine Algorithm (SCA), Gravitational Search Algorithm (GSA), Genetic Algorithm (GA), and Differential Evolution (DE)) showed that SOA represents three various convergence behaviors while optimizing test functions [3]. In the early stages of iterations, SOA converges more quickly to the promising areas due to its adaptive mechanism. Also, SOA performs better in terms of average running time compared to other meta-algorithms. This is because SOA does not require crossover and mutation operators. As a result, SOA's computational efficiency is much better than other methods.

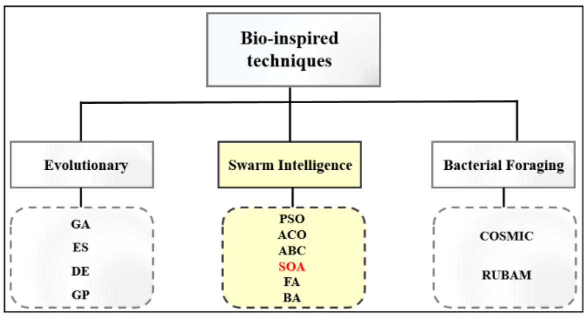

Figure 3. Taxonomy of bio-inspired techniques [25]

The main contributions are shown as follows:

1) The multi-objective optimized task scheduling algorithm is proposed considering multiple factors (i.e., energy consumption, makespan, cost, waiting time, and load balancing).
2) The Dynamic Voltage Frequency Scale (DVFS) model is included in the optimization method to reduce energy consumption.
3) The SOA is considered a global optimizer because it has good exploration and exploitation capability.
4) To show the applicability of the proposed algorithm in different scenarios, extensive experiments have been performed.

The rest of the paper is arranged as follows: Section 2 discusses the related papers which deal with existing strategies for scheduling in the cloud. Section 3 describes the SOA. Section 4 introduces the proposed algorithm. Section 5 deals with performance evaluation and experimental results. Section 6 contains the conclusion and future works.

2. RELATED WORKS

Task scheduling techniques that can effectively assign tasks to resources are still one of the challenges in the cloud environment. This is because requirements such as storage, response time, bandwidth, and resource cost may be different for each task, which greatly complicates the optimization problem, and also the heterogeneity and dynamics of the cloud environment make the issue more complex. Various techniques have been proposed to make good use of cloud resources.

Sreenu and Sreelatha [30] introduced a task scheduling algorithm for assigning tasks to suitable VMs in the cloud based on a multi-objective model and a Whale Optimization Algorithm (WOA) [31] and named it W-Scheduler. To calculate the fitness value, the authors first obtained the fitness value by calculating the cost function of CPU and memory, and then makespan, as well as the budget cost function, are added to calculate the fitness value. They used the WOA to optimally assign tasks to VMs. The WOA for finding the optimal solution supposes that the current solution is the best and tries to find the best optimal solution based on the best search agent. Experimental results showed that W-Scheduler can optimize task scheduling and perform better in terms of makespan and average cost compared to PBACO [32], SLPSO-SA [33], and SPSO-SA [33]. Nevertheless, energy consumption is not considered.

Sreenivasulu and Paramasivam [34] presented a hybrid algorithm to efficiently assign tasks to VMs. The proposed algorithm first uses a hierarchical process to prioritize tasks. Then, it applies the Bandwidth-aware divisible task (BAT) model [35] and BAR model [36] to consider task properties and VM attributes for task scheduling. The authors used the minimum overload and minimum lease policy to apply pre-emption in the data center and decrease the overload of the VMs. The experimental results showed that the proposed algorithm has better performance in terms of resource utilization, bandwidth utilization, and memory utilization compared to other algorithms. The main weakness of the presented algorithm is that it does not take into account key QoS parameters such as cost and energy consumption.

Mansouri and Javidi [37] suggested a new job scheduling based on the cost and called it CJS. The proposed algorithm, in addition to simultaneously using the data-intensive and computation-intensive of the job, also takes into account the similar factors of the available distributed environment. To assign jobs, CJS considers processing power, data, and network features. The proposed algorithm calculates three important costs, namely network cost, computation cost, and data transfer cost. The results of simulations using CloudSim [38] showed that CJS performs better in terms of makespan and resource utilization compared to FUGE [39], Berger [40], MQS [41], and HPSO [42] algorithms. However, the CJS algorithm does not consider energy consumption.

Kumar and Kalra [43] suggested a hybrid task scheduling algorithm that combines Genetic Algorithm (GA) [44] and Artificial Bee Colony (ABC) [45] algorithms. GA is a bio-inspired algorithm and consists of two distinct operations (i.e., crossover and mutation). The goal of the proposed algorithm is to decrease makespan and energy consumption. The authors used the DVFS [46] power model to compute the total energy consumed by resources. The experimental results demonstrated that the proposed hybrid algorithm performs better in terms of makespan and total energy consumption compared to the modified GA [47]. But, conflicting objectives such as time and cost have not been discussed.

Jacob and Pradeep [48] offered a multi-objective task scheduling algorithm based on a combination of Cuckoo Search (CS) [49] and PSO [28] algorithms and called it CPSO. The authors considered cost, makespan, and deadline violation rate as a multi-objective function, and based on the multi-objective function, they reached the near-optimal task scheduling. To evaluate CPSO's performance, the authors used the CloudSim [38] simulator. Experimental results showed that the CPSO algorithm has better performance in terms of cost, makespan, and deadline violation rate than PBACO, ACO, MIN-MIN, and FCFS. However, CPSO also has weaknesses. One of CPSO's principal weaknesses is that there is a high probability that resources will be overloaded.

Wu [50] proposed a novel task scheduling algorithm based on improved PSO. The author improved the PSO algorithm by adding iterative selection inhibition operators and used the improved PSO to assign tasks to VMs. The advantages of the improved PSO algorithm include high convergence speed that helps to reduce task scheduling time costs, keep away from falling into local optimum through effective search and proper distribution of computational resources, improved optimization capability, and consideration of usability and scalability in resource allocation. Simulation results demonstrated that the improved PSO has a better performance compared to PSO in terms of average execution time. However, the authors did not consider the cost and energy consumption during the scheduling process.

Elaziz et al. [51] suggested a task scheduling algorithm in the cloud environment based on a combination of Moth Search Algorithm (MSA) [52] and Differential Evolution (DE) [53] named it MSDE. The purpose of the MSDE algorithm is to assign tasks to VMs in a way that minimizes makespan. The authors considered the DE algorithm as a local search strategy to improve MSA exploitation capability. Experimental results demonstrated that the MSDE algorithm performs better in terms of makespan for both synthetical and real trace data than Shortest Job First (SJF), Round Robin (RR), PSO, WOA, and MSA. But, MSDE focuses only on makespan and does not consider other QoS parameters such as energy consumption or cost.

Shojafar et al. [39] introduced a hybrid job scheduling based on fuzzy theory and a GA and name it FUGE. FUGE's goal is to create the optimal load balance by considering run time and cost. The authors applied fuzzy theory to improve the standard GA to devise a fuzzy-based steady-state GA to improve standard GA performance in terms of makespan. The proposed algorithm for assigning jobs to resources takes into account VM processing speed, VM memory, VM bandwidth, and job length. The experimental results showed that the FUGE performs better in terms of execution time, execution cost, and average degree of imbalance compared to other algorithms. Nevertheless, the proposed algorithm does not include energy consumption.

Table 1 compares the discussed scheduling algorithms. As shown in Table 1, although most algorithms take into account makespan, cost, or energy, they did not simultaneously consider energy, cost, and makespan despite their important impact in the cloud environment. Considering all these objectives at the same time is a complex issue. To solve complex optimization problems in a reasonable time, using meta-heuristic techniques to find a near-optimal solution can be effective. Meta-heuristic algorithms are non-deterministic strategies that have been proposed to significantly solve the problem of task scheduling in a polynomial time. In this paper, we present an SOA-based task scheduling algorithm that simultaneously considers five objectives: waiting time, cost, energy consumption, makespan, and load balancing.

3. SEAGULL OPTIMIZATION ALGORITHM(SOA)

The Seagull Optimization Algorithm (SOA) [3] is a new meta-heuristic optimization algorithm inspired by the natural behavior of seagulls. Several types of seagulls vary in size and length. Seagulls are omnivorous and feed insects, fish, earthworms, reptiles, and amphibians. The Seagulls, that scientific name is Laridae, are smart birds. They use breadcrumbs to absorb fish and also absorb earthworms by making the rain-like sound with their feet. Seagulls generally live in colonies. They frequently migrate from one place to another place to find plenty of food. Seagulls attack prey when they reach a new place.

TABLE 1. Comparison of task scheduling algorithms

	Year	**Makespan**	**Monetary cost**	**Resource utilization**	**Reliability**	**Energy consumption**	**Load balancing**	**Technique**	**Disadvantage**
Sreenu and Sreelatha [30]	2017	✓	✓	×	×	×	×	Using multi-objective model and WOA	- Does not include objectives such as energy consumption and guarantee QoS.
Sreenivasulu and Paramasivam [34]	2020	×	×	✓	×	×	✓	Using BAT and Bar models	- The energy efficiency of the algorithm is very low, - Does not discuss a trade-off solution between conflict QoS parameters such as time and cost.
Mansouri and Javidi [37]	2019	✓	×	×	×	×	✓	Using data, processing power, and network characteristics to assign jobs to resources	- Does not take into account significant criteria such as energy consumption. - Resources may be overloaded or underutilization.
Kumar and Kalra [43]	2019	✓	×	×	×	✓	×	Using GA and ABC along with DVFS	- Does not consider the deadline and priority constraint as well as SLA violations, - Cost, reliability, and other QoS parameters do not consider.
Jacob and Pradeep [48]	2019	✓	✓	×	×	×	×	Using CS and PSO	- Cannot distribute the load uniformly, - Does not optimize QoS parameters such as energy consumption.
Wu [50]	2018	✓	×	×	×	×	×	Using Improved PSO	- The load balance on resources is not monitored during runtime, - The proposed algorithm is a single objective and does consider other QoS parameters such as cost, energy consumption, etc.
Elaziz et al. [51]	2019	✓	×	×	×	×	×	Using a combination of MSA and DE	- Does not take into account the usage of memory, the peak of the demand, and overloads, - High-time complexity.
Shojafar et al. [39]	2014	✓	✓	×	×	×	✓	Using GA and fuzzy theory	- Does not take into account VM energy consumption, - High monitoring overhead.

The most significant thing about seagulls is their migratory and attacking behavior. Therefore, SOA focuses on these two natural behaviors and provides a mathematical model. Figure 4 shows a conceptual model of these behaviors.

Initially, seagulls perform migratory behavior (indicating the exploration ability of SOA). When migrating, members of a group of seagulls should avoid colliding with each other. To obtain this, an additional variable A is used to compute the position of the new search agent.

$$\overrightarrow{C_s} = A \times \overrightarrow{P_s}(x) \tag{1}$$

where $\overrightarrow{C_s}$ indicates the position of the search agent which does not collide with other search agents, $\overrightarrow{P_s}$ indicates the current position of the search agent, x shows the current iteration, and A represents the movement behavior of the search agent.

$$A = f_c - \left(x \times \left(f_c / Max_{iteration}\right)\right) \tag{2}$$

where f_c is presented to manage the frequency of employing variable A which is linearly decreased from the initial value of f_c to 0. After avoiding collisions among neighbors, search agents move toward the best search agent.

$$\overrightarrow{M_s} = B \times (\overrightarrow{P_{bs}}(x) - \overrightarrow{P_s}(x)) \tag{3}$$

where $\overrightarrow{M_s}$ indicates the position of the search agent $\overrightarrow{P_s}$ towards the best search agent $\overrightarrow{P_{bs}}$ (i.e., the most suitable seagull). The coefficient B is a random value that can be used to make a trade-off between exploitation and exploration. B is computed as follows:

$$B = 2 \times A^2 \times rd \tag{4}$$

where rd indicates a random number in the range [0, 1]. Since search agents move toward the most appropriate search agent, they may stay close to each other.

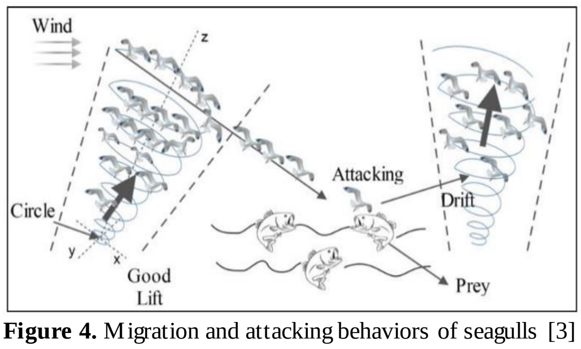

Figure 4. Migration and attacking behaviors of seagulls [3]

Therefore, search agents can update their position according to the best search agent based on the following equation:

$$\overrightarrow{D_s} = |\overrightarrow{C_s} + \overrightarrow{M_s}| \tag{5}$$

where $\overrightarrow{D_s}$ indicates the distance between the search agent and the best search agent.

Secondly, seagulls attack prey in a spiral movement after reaching a new place (indicating the exploitation ability of SOA). This behavior in x, y, and z planes is defined below:

$$x = r \times \cos(k) \tag{6}$$

$$y = r \times \sin(k) \tag{7}$$

$$z = r \times k \tag{8}$$

$$r = u \times e^{kv} \tag{9}$$

where r indicates the radius of each urn of the spiral, k represents a random number in the range $[0 \leq k \leq 2\pi]$. u and v are constants, and e is the base of the natural logarithm. The updated position of the search agent is computed as follows:

$$\overrightarrow{P_s}(x) = (\overrightarrow{D_s} \times x \times y \times z) + \overrightarrow{P_{bs}}(x) \tag{10}$$

Figure 5 represents the pseudocode of SOA.

4. SOA-BASED TASK SCHEDULING ALGORITHM

This section consists of three subsections. In subsection 4.1, the basic concepts related to the problem of task scheduling are explained. In subsection 4.2, the objective function and mathematical model are described. In subsection 4.3, the proposed algorithm is introduced.

4. 1. Task Scheduling Model

Assigning all tasks among available VMs and discovering the optimal solution in the cloud environment is not simple work. For this reason, we need an effective task scheduling algorithm to balance the VM load and assign all user's tasks to the appropriate resources.

Suppose a cloud datacenter contains n tasks such as: $T = \{T_1, T_2, ..., T_n\}$, where T_i represents the i-th task in the task queue, m VMs such as: $V = \{V_1, V_2, ..., V_m\}$, where V_j represents the j-th VM in the cloud environment, but the condition for execution of such tasks is: $n > m$.

4. 2. Objective Functions

The primary goal of the SOATS is to optimally schedule all the input tasks to the available VMs to minimize cost, makespan, load, energy

Algorithm : Seagull Optimization Algorithm

```
Input: Seagull population $\vec{P_s}$
Output: Optimal search agent $\vec{P_{bs}}$
procedure SOA
Initialize the parameters $A$, $B$, and $Max_{iteration}$
    Set $f_c$
    Set $u$
    Set $v$
    while ($x < Max_{iteration}$) do
        $\vec{P_{bs}} \leftarrow$ ComputeFitness($\vec{P_s}$)    /* Calculate the fitness values of each search agent using ComputeFitness function*/
/* Migration behavior */
        $rd \leftarrow Rand(0, 1)$        /* To generate the random number in range [0, 1] */
        $k \leftarrow Rand(0, 2\pi)$       /* To generate the random number in range [0, 2π] */
/* Attacking behavior */
        $r \leftarrow u \times e^{kv}$        /* To generate the spiral behavior during migration */
        Calculate the distance $\vec{D_s}$ using Eq. (5)
        $P \leftarrow x' \times y' \times z'$      /* Compute x, y, z planes using Eqs. (6) - (9) */
        $\vec{P_s}(x) \leftarrow (\vec{D_s} \times P) + \vec{P_{bs}}$
        $x \leftarrow x + 1$
    end while
return $\vec{P_{bs}}$
end procedure

procedure ComputeFitness($\vec{P_s}$)
    for i ← 1 to n do     /* Here, n represents the dimension of a given problem */
        $FIT_s[i] \leftarrow FitnessFunction(P_s(\vec{i}, :))$    /* Calculate the fitness of each individual */
    end for
    $FIT_{s_{best}} \leftarrow$ BEST($FIT_s[]$)    /* Calculate the best fitness value using BEST function */
return $FIT_{s_{best}}$
end procedure

procedure BEST($FIT_s[]$)
    $Best \leftarrow FIT_s[0]$
    for i ← 1 to n do
        if($FIT_s[i] < Best$) then
            $Best \leftarrow FIT_s[i]$
        end if
    end for
return $Best$        /* Return the best fitness value */
end procedure
```

Figure 5. The pseudocode of SOA [3]

consumption, and waiting time to keep both the user satisfied and the provider profit. The objective function is computed as follows. The final output of the scheduling algorithm is an $n \times m$ assignment matrix that specifies by which VM each task should be executed. We define the assignment matrix as follows:

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

where x_{ij} is a decision variable and calculated by Equation (12):

$$x_{ij} = \begin{cases} 1 & if\ T_i\ is\ assigned\ to\ V_j \\ 0 & if\ T_i\ is\ not\ assigned\ to\ V_j \end{cases} \tag{12}$$

With the condition:

$$\sum_{j=0}^{m} x_{ij} = 1 \quad for\ 1 \le i \le n \tag{13}$$

Cost: Task scheduling in a cloud system (as a business service) in addition to being an efficient scheduler, must also decrease costs. Scheduling that decreases costs without violating QoS leads to both user and service provider satisfaction. To estimate the assignment cost, each use of resources such as processing element, memory, etc. must be computed. The following equation is used to calculate the cost of task completion [54]:

$$C_{V_j} = \sum_{j=1}^{m} sum(VM_j) \times (V_{cpu_j} + V_{ram_j} + V_{bw_j}) \tag{14}$$

where *sum(VMj)* indicates the total number of tasks assigned to V_j. Equation (14) shows the processing cost in a V_j, which is closely related to CPU (V_{cpu_j}), memory (V_{ram_j}), and bandwidth performance (V_{bw_j}) of VMs.

Makespan: Makespan shows the completion time of the last task. One of the most popular scheduling criteria that researchers use to measure the performance of scheduling algorithms is makespan. This is because researchers believe that the performance of the scheduling algorithm is highly makespan-dependent. In addition, minimizing the makespan makes the user application execute faster; thus, reducing the makespan increases user satisfaction. Makespan can be described mathematically by Equation (15) [55]:

$$MS = Max\{VET_j\} \quad for\ 1 \le j \le m \tag{15}$$

where VET_j represents the *j-th* VM execution time and it is computed based on the decision variable x_{ij} by Equation (16):

$$VET_j = \sum_{i=1}^{n} x_{ij} \times ET_{ij} \quad for\ 1 \le j \le m \tag{16}$$

where ET_{ij} is the approximation calculation time for executing T_i on V_j and computed based on Equation (17):

$$ET_{ij} = \frac{TL_i}{PS_j} \tag{17}$$

where TL_i indicates the *i-th* task length in Million Instructions (MI) and PS_j indicates the *j-th* VM execution speed in Million Instructions Per Second (MIPS).

Load balancing: VMs are mostly processing elements in cloud environments. In scheduling, there is a situation where more than one task is assigned to each VM. Load balance distributes loads evenly between different cloud resources. The scheduler must be able to distribute the workload among available resources in a way that prevents resources from being overloaded or underloaded. Load balancing increases resource utilization and thus improves overall scheduling performance. The equation for calculating the degree of resource load balance in a VM is as follows [54]:

$$\varphi = \frac{\sqrt{\dfrac{\sum_{j=1}^{m}(VET_j - \overline{VET_j})^2}{m}}}{n} \tag{18}$$

where VET_j represents the total execution time of the V_j; $\overline{VET_j}$ represents the mean execution time of the V_j.

Energy consumption: One of the most important issues for individuals, organizations, and governments is energy consumption. There is a global concern about minimizing carbon emissions because it affects our environment in a way that endangers a healthy life and human health. CPU utilization and resource utilization directly affect the energy consumed by a task. Energy consumption will be high when CPUs are not properly utilized. This is because idle power is not effectively used. Sometimes energy consumption increases due to high requests for resources, and this may reduce efficiency. Proper scheduling algorithms are very significant to find the optimal assignment of tasks so that energy consumption is reduced. The total energy consumption of a DVFS-enable resource (DVFS lets resources operate at various voltage and frequency sets) contains static energy because of leakage current and dynamic energy because of switching activities. As shown in Equation (23), in this paper we consider only dynamic energy consumption [56]:

$$E = E_{sta} + E_{dyn} \tag{19}$$

where E_{sta} represents static energy and E_{dyn} represents dynamic energy consumption.

$$E_{dyn} = \alpha \times v_{j,s}^2 \times f_s \tag{20}$$

where α is a constant value, $v_{j,s}^2$ is V_j voltage, and f_s is the corresponding frequency of $v_{j,s}$.

$$E_{active} = \sum_{j=1}^{m} E_{dyn} \times ET_{ij} \tag{21}$$

where ET_{ij} is the execution time of the T_i executed on V_j.

$$E_{idle} = \sum_{j=1}^{m} \alpha \times v_0^2 \times f_0 \times t_{idle,j} \tag{22}$$

where α indicates a constant value, v_0 and f_0 is the resource minimum voltage and resource minimum frequency, respectively, and $t_{idle,j}$ represents the idle time of the V_j.

$$E_{total} = E_{active} + E_{idle} \tag{23}$$

Waiting time: It is the difference between the start time of execution and the submission time of the task. Reducing waiting time increases user satisfaction because the user has to wait less time. User waiting time can be defined mathematically as follows [54]:

$$WT_i = \underset{j=1}{\overset{m}{Max}} \sum_{i=1}^{sum(VM_j)} ET_{ij} \tag{24}$$

where ET_{ij} refers to the execution time T_i on V_j.

The main goal is to minimize the values of the above five functions, which is a multi-objective optimization problem; because each of the functions has various purposes that can conflict with each other. With a powerful CPU, we can increase the processing speed of a task, but the cost also increases. Also, for the situation that a VM with a large memory will be able to load a lot of tasks, but the makespan could be long if the computing power of the CPU is low. Because the task scheduling function is not determined by a single objective function, the presented algorithm creates a task scheduling satisfaction function based on a priori preferences. Therefore, we turn the multi-objective problem of task scheduling into a single-objective problem. Assume that the cost range of task completion is [C_{min}, C_{max}], the range of makespan is [MS_{min}, MS_{max}], the satisfaction range of VM load balancing degree is [$\varphi_{\min}, \varphi_{\max}$], the suitable range of energy consumption is [E_{min}, E_{max}], and the range of the user's shortest waiting time is [WT_{min}, WT_{max}]. By introducing the minimum amount of ε [57], the five objectives are computed as follows:

$$O(C_{v_j}) = \begin{cases} 1 & C_{v_j} \le C_{\min} \\ \dfrac{C_{\max} - C_{v_j}}{C_{\max} - C_{\min}} & C_{v_j} \in (C_{\min}, C_{\max}) \\ \dfrac{\varepsilon}{C_{v_j}} & C_{v_j} \ge C_{\max} \end{cases} \tag{25}$$

$$O(MS) = \begin{cases} 1 & MS \le MS_{\min} \\ \dfrac{MS_{\max} - MS}{MS_{\max} - MS_{\min}} & MS \in (MS_{\min}, MS_{\max}) \\ \dfrac{\varepsilon}{MS} & MS \ge MS_{\max} \end{cases} \tag{26}$$

$$O(\varphi) = \begin{cases} 1 & \varphi \le \varphi_{\min} \\ \dfrac{\varphi_{\max} - \varphi}{\varphi_{\max} - \varphi_{\min}} & \varphi \in (\varphi_{\min}, \varphi_{\max}) \\ \dfrac{\varepsilon}{\varphi} & \varphi \ge \varphi_{\max} \end{cases} \tag{27}$$

$$O(E) = \begin{cases} 1 & E_{total} \le E_{\min} \\ \dfrac{E_{\max} - E_{total}}{E_{\max} - E_{\min}} & E_{total} \in (E_{\min}, E_{\max}) \\ \dfrac{\varepsilon}{E_{total}} & E_{total} \ge E_{\max} \end{cases} \tag{28}$$

$$O(WT_i) = \begin{cases} 1 & WT_i \le WT_{\min} \\ \dfrac{WT_{\max} - WT_i}{WT_{\max} - WT_{\min}} & WT_i \in (WT_{\min}, WT_{\max}) \\ \dfrac{\varepsilon}{WT_i} & WT_i \ge WT_{\max} \end{cases} \tag{29}$$

We used the geometric average method to convert five objectives into one objective. Therefore, the final optimization function which will be minimized through the proposed algorithm is as follows:

$$F_{opt} = Min\left\{ \sqrt[5]{O(C_{v_j}) \times O(MS) \times O(\varphi) \times O(E) \times O(WT_i)} \right\} \tag{30}$$

4. 3. The SOATS Algorithm Based on all the above, Figure 6 represents the pseudocode of task scheduling based on SOA technique.

In addition, the flowchart of SOATS algorithm for task scheduling is shown in Figure 7. The principal steps of the SOATS algorithm can be described as follows:

Step 1) At first, initialization is performed, which usually contains mapping among cloud tasks and seagulls and initialization of seagulls positions. Also, some execution factors such as the number of search agents, the maximum number of iterations, and search space dimensions are initialized.

Step 2) The process of finding the optimal solution starts based on SOA. In this step, based on position information, the amount of cost, makespan, load, energy consumption, and waiting time are calculated according to Equations (14), (15), (18), (23), and (24), respectively. Then, according to Equation (30), the objective function value of each seagull is calculated. The position of the seagull with the smallest fitness value (i.e., fittest seagull)

Algorithm: Pseudocode for mapping of tasks onto VMs using SOA algorithm

```
Input: Tasks set, VMs set.
Output: Allocating tasks to VMs.
Begin
1  Initialize tasks set as: $T = \{T_1, T_2, \ldots, T_n\}$.
2  Initialize VMs set as: $V = \{V_1, V_2, \ldots, V_m\}$.
3  Initialize number of seagulls (i.e., population size), and maximum iteration.
4  Initialize SOA parameters (e.g., $f_c$, $u$, $v$).
5  Initialize $C_{min}$, $C_{max}$, $MS_{min}$, $MS_{max}$, $E_{min}$, $E_{max}$, $WT_{min}$, $WT_{max}$, $\phi_{min}$, $\phi_{max}$, and $\varepsilon$ that represent limits of cost, makespan, energy consumption, waiting time, load balancing, and minimum value respectively.
6  Initialize seagulls' position randomly.
7  t = 1
8  While (t ≤ maximum iteration)
9      For (i =1 to number of seagulls)
10         Calculate fitness value for each seagull;
11         Find best seagull so far and set its position as $P_{bs}$;
12     End for
13     Update variable $A$ according to Equation (2);
14     For (i =1 to number of seagulls)
15         Perform migration according to Equation (5);
16         Perform attacking according to Equation (10);
17     End for
18     t++;
19 End while
20 Set the allocation matrix for the best seagull.
```

Figure 6. The pseudocode of SOATS

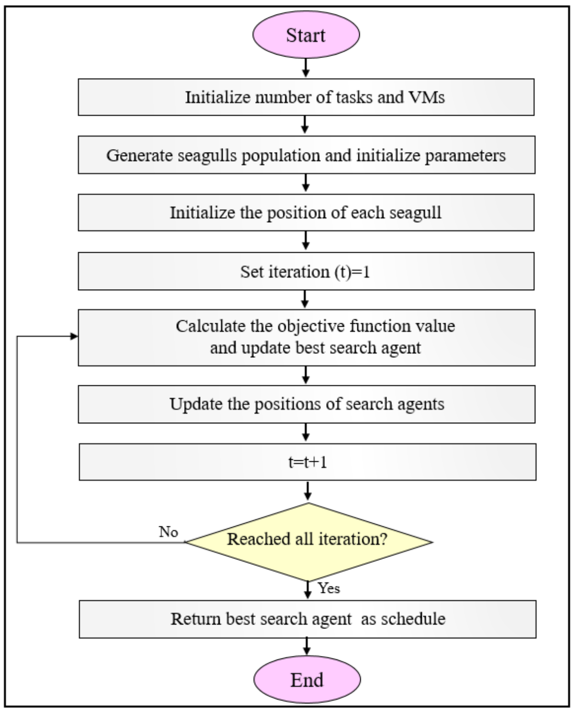
Start
Initialize number of tasks and VMs
Generate seagulls population and initialize parameters
Initialize the position of each seagull
Set iteration (t)=1
Calculate the objective function value and update best search agent
Update the positions of search agents
t=t+1
Reached all iteration?
No
Yes
Return best search agent as schedule
End

Figure 7. Flowchart of SOATS

will be recorded (which indicates the optimal solution so far).

Step 3) In this step, the positions of seagulls will be updated according to Equation (10).

Step 4) When the positions of all the seagulls are updated, an iteration is performed. If the maximum number of iterations is done, the search process is terminated and the position of the best search agent is transferred to the x_{ij} decision variables and finally returned as the best scheduling solution, otherwise, it goes to step 2 for the new search.

5. PERFORMANCE ANALYSIS

In this section, to evaluate the performance of the SOATS algorithm, we use MATLAB software that is installed on a PC with Intel(R) Core(TM) i5-7200U CPU with 2.50 GHz, and RAM of 8 GB running on 64-bit Windows 10 Pro operating system platform. The SOA-based task scheduling algorithm is compared with other well-known meta-heuristic algorithms, namely GA, PSO, ACO, and WOA for validation. In addition, we compare the performance of the SOATS algorithm in a heavily loaded environment with three scheduling algorithms, namely CJS, FUGE, and MSDE.

Table 2 shows the specific parameter settings for the comparative meta-heuristic algorithms [3, 58, 59].

Also, for each different scenario, the table of simulation parameters is presented. Most of the

TABLE 2. Parameters settings of caparisoned meta-heuristic algorithms

Algorithms	Parameters	Values
GA	Crossover	0.9
	Mutation	0.05
PSO	C1	1.8
	C2	2
	Inertia factor	0.75
ACO	ρ	0.7
	P	0.3
WOA	a	[2, 0]

simulation parameters have been selected to conform to existing studies for the real representation of a typical cloud environment [60]. In addition, the parameters related to the SOA algorithm are also set [3]. According to each different scenario that is proposed, one of the parameters in each scenario is variable and the results are analyzed based on this parameter.

5. 1. Number of Tasks

In this experiment, the number of tasks is changed among 100 and 500 tasks with a step of 100. The parameters of the cloud system and the SOA are described in Table 3.

In many works, makespan is used as one of the most popular performance criteria. Reducing the makespan value demonstrates the ability of scheduling to effectively choose resources for the appropriate allocation of tasks. Figure ٨ shows a graphical comparison of the makespan between SOATS and the task scheduling based on GA, PSO, ACO, and WOA using various numbers of tasks. Makespan is drawn on the vertical axis and the number of tasks on the horizontal

TABLE 3. Parameters setting (different number of tasks)

Parameters	Values
Number of tasks	100-500
Tasks size (MI)	100-2000
Number of VMs	40
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	100
Population size	50
F_c	1
Constant u and v	1

axis. According to the results, it is clear that SOATS has a better makespan compared to other algorithms by increasing the number of tasks. The makespan minimization by SOA is 5-10% less than that of PSO for 100 through 500 number of tasks, respectively. This is because the SOA has good exploration and exploitation ability because variable *B* in the SOA is responsible for the smooth transfer between exploration and exploitation.

As shown in Figure 9, the proposed SOA-based task scheduling algorithm has obvious benefits in obtaining load balancing compared to other meta-heuristic algorithms. Load balancing must be done in such a way that all VMs must be balanced to achieve optimal use of their capabilities and improve system performance. The SOATS obtains the best balance between VMs in all numbers of tasks. Conversely, ACO-based task scheduling has the worst workload for all cases.

A comparison of the costs of using the VM for the SOATS and other meta-heuristic algorithms is shown in Figure 10. The cost increases as the number of tasks increases. Proper estimation of VM cost in a cloud computing environment is very important because as the cost decreases, the service provider's profit increases. The cost minimization by SOA is 12-3% less than that of WOA for 100 through 500 instances of tasks, respectively. Also, the cost minimization by SOA is 10-2% less than that of ACO for 100 through 500 tasks, respectively. The main reason is that ACO does not search the search space well and falls into the trap of local optimum.

Energy consumption is also one of the main metrics in maximizing the overall performance of the cloud system. There is a direct linear relationship between energy consumption and VMs utilization because the optimal VMs utilization reduces the energy consumption of a server. The X-axis represents the number of tasks and the Y-axis indicates the energy consumption. In Figure 11, SOATS is more efficient and has a lower energy consumption in comparison to other algorithms. The energy consumption in the proposed algorithm is 31% better than that of GA, 22% that of PSO, 28% that of ACO, and 20% that of WOA in the case of 500 tasks assigned.

Waiting time is the total time a task spends in the task queue waiting for a VM to execute. Figure 12 shows the experimental results for the waiting time. As shown in Figure 12, SOATS waiting time is better than other algorithms for all cases. The GA provides the worst waiting time when the number of tasks is 500. The

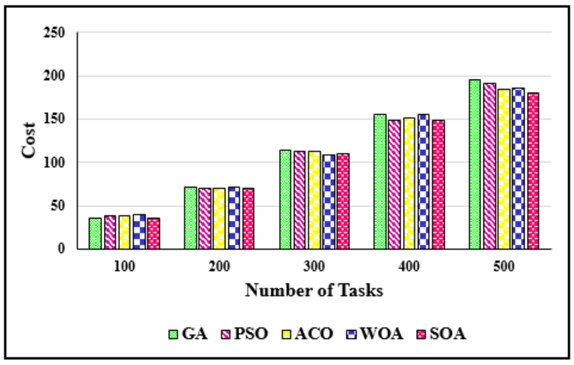

Figure 10. Cost with the different number of tasks

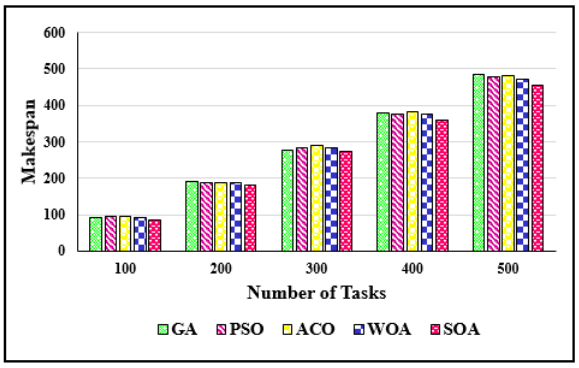

Figure 8. Makespan time with different numbers of tasks

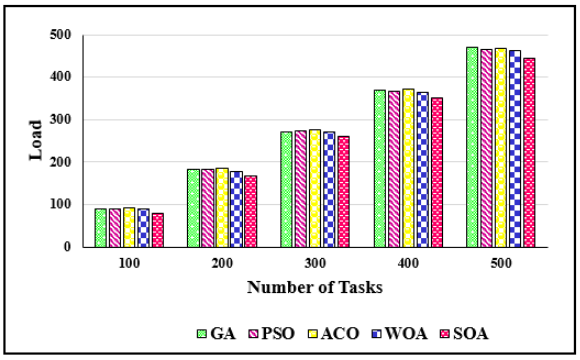

Figure 9. Degree of load balancing with different numbers of tasks

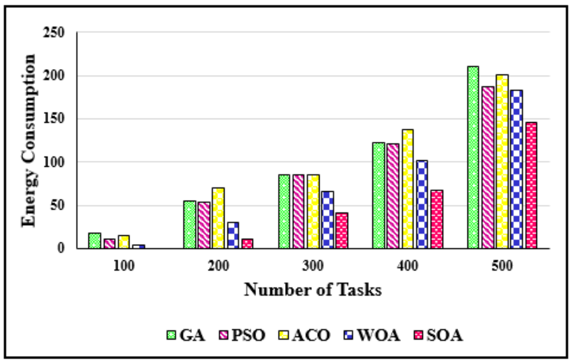

Figure 11. Energy consumption with the different number of tasks

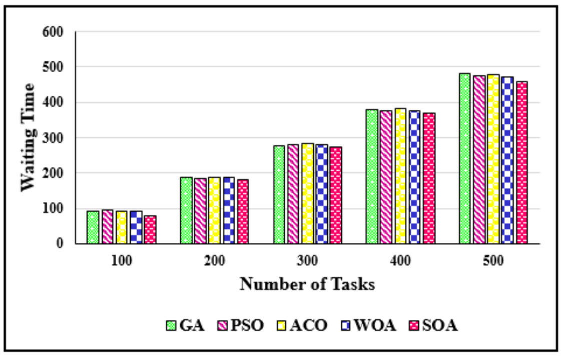

Figure 12. Waiting time with the different number of tasks

waiting time in the proposed approach is 15-3%, 13-4%, and 13-2% less than PSO, ACO, and WOA for 100 through 500 tasks, respectively. Since f_c decreases from the initial value of f_c to 0, this allows the SOA to search well at the beginning and converges to the optimal solution by increasing the number of iterations.

5. 2. Number of VMs This experiment is performed with a variable number of VMs (between 10 and 50) while the number of tasks is considered fixed (500 tasks). Table 4 represents the parameters of the cloud system and the SOA.

Comparison of performance in terms of makespan, load, cost, energy consumption, and waiting time is shown in Figures 13-17 for different numbers of VMs with bar charts between different algorithms. It is clear that as the number of VMs increases, scheduling algorithms can process tasks in a shorter time, so parameters such as makespan and waiting time decrease with the increasing number of VMs (Figures 13 and 17, respectively). However, an increase in VMs number is increasing energy consumption. In Figure 16, as

TABLE 4. Parameters setting (different number of VMs)

Parameters	Values
Number of tasks	500
Tasks size (MI)	100-2000
Number of VMs	10-50
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	1-100
Population size	60
F_c	1
Constant u and v	1

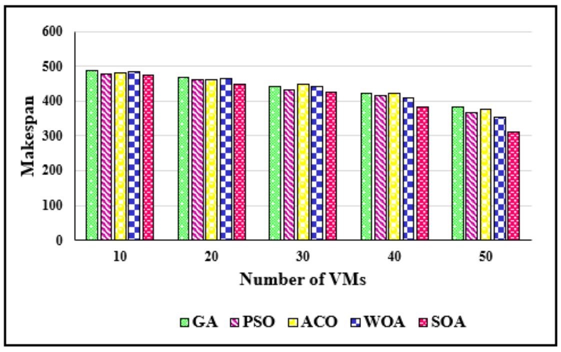

Figure 13. Makespan time with different numbers of VMs

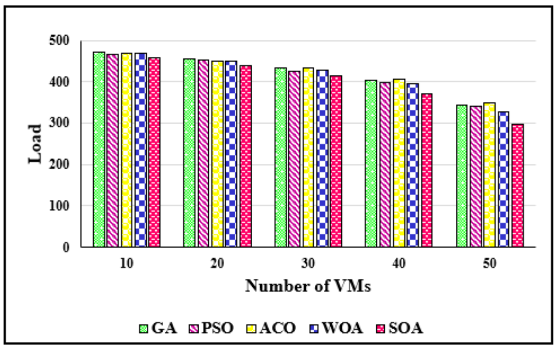

Figure 14. Degree of load balancing with different numbers of VMs

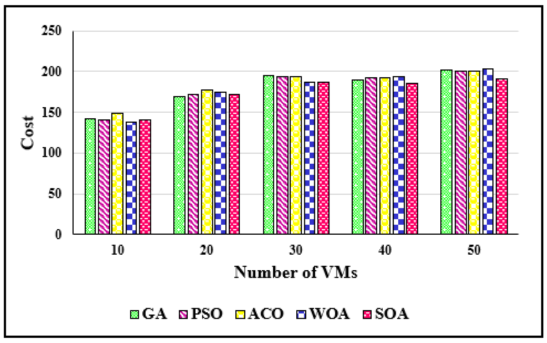

Figure 15. Cost with the different number of VMs

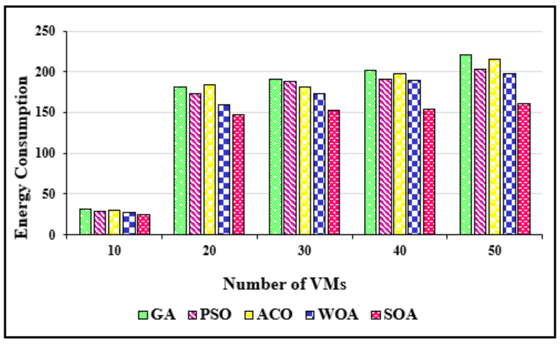

Figure 16. Energy consumption with the different number of VMs

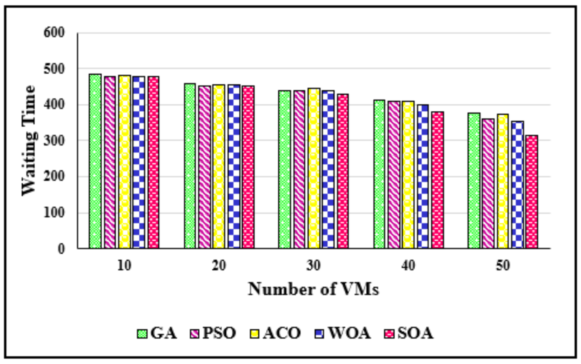

Figure 17. Waiting time with the different number of VMs

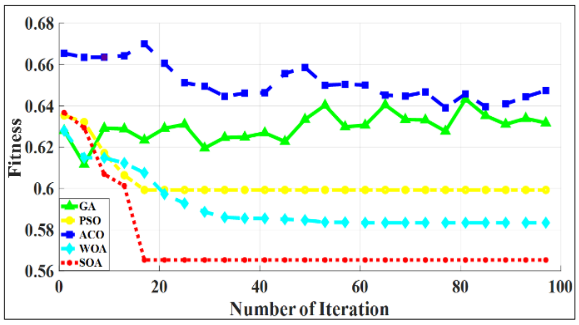

Figure 18. Convergence plot based on the number of iterations

expected, the energy consumption increases as the number of VMs increases. SOATS algorithm has the lowest energy consumption for different numbers of VMs and also GA in most cases has the maximum energy consumption due to poor exploitation capability. The difference between scheduling algorithms is evident in difficult situations such as when the number of VMs is low. As shown in Figures 13-17, the SOATS algorithm in most cases performs better than other algorithms for a different number of VMs with a certain number of tasks. This is because the SOA algorithm makes a balance between exploration and exploitation.

5. 3. Number of Iterations

In the second scenario, we examine the performance of the SOATS compared to other algorithms by increasing the number of iterations. Table 5 presents the simulation parameters used in this scenario.

Figure 18 shows the convergence speed comparison of five meta-heuristic algorithms to solve the scheduling problem. Figure 18 shows that the fitness of the PSO, WOA, and SOA algorithms decreases with an increase in number of iterations, which indicates the efficiency of these algorithms in scheduling in the cloud environment. As shown in Figure 18, WOA and PSO perform better than GA and ACO algorithms. This is because PSO and WOA have better search capability and exploitation capability than GA and ACO. However, it can be observed that SOA has the best performance and, an increase in the number of iterations, SOA can achieve its optimal solution faster than PSO and WOA. SOA has better performance than the other four algorithms in terms of convergence speed and accuracy. This is because there is a trade-off between the local optimal value and the global optimal value in the search process. In other words, SOA has good exploration and exploitation capabilities. Initially, it searches the search space well and does not fall into the local optimal and then converges to the global optimal solution. Therefore, SOA has a good ability to solve complex optimization problems.

The convergence analysis of meta-heuristic algorithms is used for a better understanding of exploration and exploitation capabilities. Figure 19 shows the average convergence time of SOA and other metaheuristic algorithms. It can be seen that SOA takes less convergence time than other methods. The SOA

TABLE 5. Parameters setting (different number of iterations)

Parameters	Values
Number of tasks	400
Tasks size (MI)	100-2000
Number of VMs	20
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	1-100
Population size	40
F_c	1
Constant u and v	1

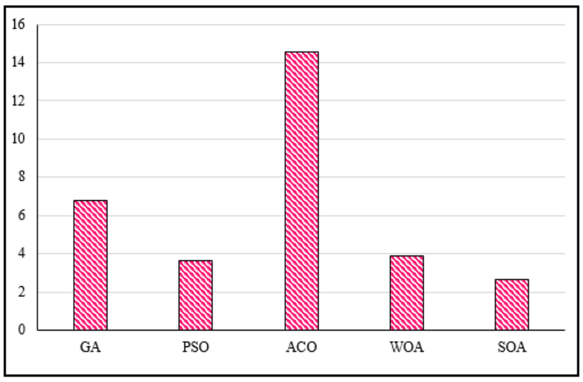

Figure 19. Average convergence time of meta-heuristic algorithms

converges after about 2.62 seconds, PSO 3.64 seconds, WOA about 3.91 seconds, GA 6.76 seconds, and ACO 14.54 seconds.

5. 4. Number of Search Agents In this subsection, we compare the performance of SOATS with other algorithms according to different population sizes. The cloud parameters and SOA parameters are presented in Table 6.

The number of seagulls in the SOA algorithm is known as the population size. Increasing the size of the population creates more parts of the search space that must be covered in each iteration. By increasing the population size, the number of iterations required to achieve the optimal solution can be reduced. However, increasing the population size increases the computational complexity in each iteration; therefore, it is time-consuming. In this experiment, we examine the performance of all five meta-heuristic algorithms in terms of task scheduling by considering the number of 100 iterations and different population sizes. We started the simulation with 40 search agents and increased it to 80 agents. The results in Figure 20, show that the ACO in most cases has the worst fitness and SOA has the best fitness value in all population sizes compared to other algorithms.

5. 5. F_c Parameter In this scenario, we run the SOA with different f_c values and compare the results. Table 7 proposed the simulation parameters used in this scenario.

f_c is one of the most important parameters in the SOA algorithm, which is introduced to control the frequency of variable *A* and reduces linearly from the initial value of f_c to 0. We implemented the SOA algorithm for various values of the f_c parameter by keeping the number of iterations and the number of search agents constant.

TABLE 6. Parameters setting (different number of agents)

Parameters	**Values**
Number of tasks	400
Tasks size (MI)	100-2000
Number of VMs	10
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	100
Population size	40-80
F_c	1
Constant u and v	1

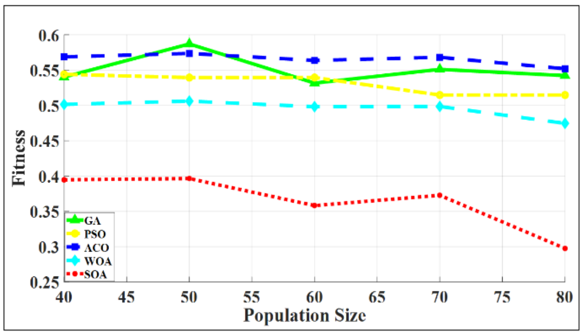

Figure 20. Convergence plot based on population size

TABLE 7. Parameters setting (different values of f_c)

Parameters	**Values**
Number of tasks	400
Tasks size (MI)	100-2000
Number of VMs	20
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	100
Population size	40
F_c	1-5
Constant u and v	1

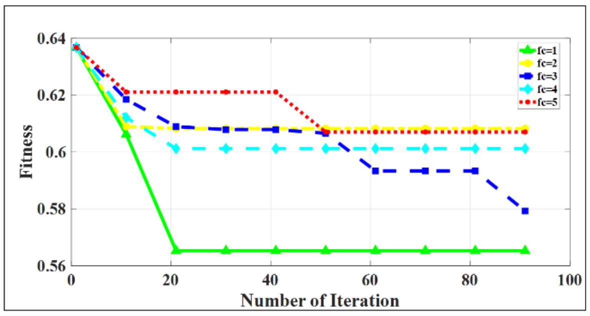

Figure 21. Effect of parameter f_c in our task scheduling algorithm

The f_c values used in experiments are 1, 2, 3, 4, and 5. As shown in Figure 21, the scheduling algorithm presented in this paper obtains the best optimal solution when the value of f_c is set to 1.

5. 6. Other Scheduling Algorithms In the last scenario, we compare SOATS performance with other scheduling algorithms, namely CJS [37], FUGE [39], and MSDE [51]. Table 8 shows the parameters settings. The results obtained by comparing SOATS with other scheduling algorithms in terms of makespan, cost, degree of load, energy consumption, and waiting time are shown in Table 9.

Reducing makespan and waiting time is one of the most important requests of users because it makes their tasks execute faster. It is clear from Table 9 that SOATS has a shorter makespan and waiting time than other scheduling algorithms. Because the SOATS algorithm tries to distribute the tasks optimally between resources by considering the makespan and waiting time in the objective function and evaluating the value of the objective function in each iteration, which helps to reduce the execution time.

As Table 9 shows, the costs in SOA are 7, 10, and 12% lower than CJS, FUGE, and MSDE, respectively. MSDE has the worst performance in terms of cost compared to the rest, because MSDE only focused on reducing makespan and did not consider the cost. Also, the proposed algorithm performs better in terms of load balance as well as reducing energy consumption. SOA decreases energy consumption up to 27% in comparison with CJS, up to 24% in comparison with FUGE, and up to 23% in comparison with MSDE. Since SOATS uses the DVFS model and so consumes less energy because the resources operate with the minimum voltage and frequency required.

TABLE 8. Parameters setting (different scheduling algorithms)

Parameters	Values
Number of tasks	500
Tasks size (MI)	100-2000
Number of VMs	40
VMs execution speed (MIPS)	500-4500
Storage cost	\$0.1 per GB
Processing cost	\$1 per 10^6 MI
Data transfer cost	\$0.05 per GB
Maximum iteration	100
Population size	50
F_c	1
Constant u and v	1

TABLE 9. The comparison between the different scheduling algorithms

Objectives/ Methods	Makespan	Cost	Degree of load	Energy	Waiting time
CJS	460	193	457	200	462
FUGE	463	201	453	191	460
MSDE	459	205	459	190	463
SOATS	455	180	445	146	457

6. CONCLUSION AND FUTURE WORKS

The problem of scheduling in cloud computing is an NP-hard problem due to the many parameters that exist (such as task priority, the dependency among tasks, and communication costs). One of the solutions to solve these problems is to use meta-heuristic algorithms. Although in the cloud system, finding a suitable task scheduling algorithm is very important for users and providers, most papers fail to offer an effective trade-off between makespan, energy consumption, and cost. In this paper, we present a new SOA-based task scheduling algorithm that simultaneously considers makespan, energy consumption, cost, load, and waiting time and named it SOATS. The experimental results show that the proposed tasks scheduling algorithm can improve the performance of the cloud computing system in terms of system load, makespan, cost, energy consumption, and waiting time compared to other well-known meta-heuristic algorithms such as GA, PSO, ACO, and WOA. In addition, SOATS has a better convergence speed and can find the optimal solution with more accuracy and speed compared to other meta-heuristic algorithms. This is because SOA has a good ability to explore and exploit. In the heavily loaded cloud environment. The proposed algorithm reduces energy consumption, cost saving and degree of load balancing by 10 and 25 and 3%, respectively. As part of our future work, we intend to combine the proposed algorithm with other meta-heuristic algorithms. In addition, we will consider other computational criteria such as security and availability. We also want to improve the proposed algorithm using fuzzy theory.

7. REFERENCES

1. Mohammad Hasani Zade. B, Mansouri. N, and Javidi. M. M, "Multi-objective scheduling technique based on hybrid hitchcock bird algorithm and fuzzy signature in cloud computing", ***Engineering Applications of Artificial Intelligence***, Vol. 104, (2021), DOI: 10.1016/j.engappai.2021.104372.
2. Kumar. M, Sharma. S. C, Goel. A, and Singh. S. P, "A comprehensive survey for scheduling techniques in cloud computing", ***Journal of Network and Computer Applications***, Vol. 143, (2019), 1-33, DOI: 10.1016/j.jnca.2019.06.006.
3. Dhiman. G, and Kumar. V, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems", ***Knowledge-Based Systems***, Vol. 165, (2019), 169-196, DOI: 10.1016/j.knosys.2018.11.024.
4. Mansouri. N, Mohammad Hasani Zade. B, Javidi. M. M, "SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment", ***Expert Systems with Applications***, Vol. 176, (2021), DOI: 10.1016/j.eswa.2021.114915.

5. Pradhan. A, Bisoy. S. K, and Das. A, "A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment", ***Journal of King Saud University - Computer and Information Sciences***, (2021), DOI: 10.1016/j.jksuci.2021.01.003.

6. Shafiq. D. A, Jhanjhi. N. Z, and Abdullah. A, "Load balancing techniques in cloud computing environment: A review", ***Journal of King Saud University - Computer and Information Sciences***, (2021), DOI: 10.1016/j.jksuci.2021.02.007.

7. Velliangiri. S, Karthikeyan. P, Arul Xavier. V. M, and Baswaraj. D, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing", ***Ain Shams Engineering Journal***, Vol. 12, No. 1, (2021), 631-639, DOI: 10.1016/j.asej.2020.07.003.

8. Wilczyński. A, and Kołodziej. J, "Modelling and simulation of security-aware task scheduling in cloud computing based on Blockchain technology", ***Simulation Modelling Practice and Theory***, Vol. 99, (2020), DOI: 10.1016/j.simpat.2019.102038.

9. NoorianTalouki. R, Hosseini Shirvani. M, and Motameni. H, "A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms", ***Journal of King Saud University - Computer and Information Sciences***, (2021), DOI: 10.1016/j.jksuci.2021.05.011.

10. Alsaidy. S. A, Abbood. A. D, and Sahib. M. A, "Heuristic initialization of PSO task scheduling algorithm in cloud computing", ***Journal of King Saud University - Computer and Information Sciences***, (2020), DOI: 10.1016/j.jksuci.2020.11.002.

11. Pradhan. A, and Bisoy. S. K, "A novel load balancing technique for cloud computing platform based on PSO", ***Journal of King Saud University-Computer and Information Sciences***, (2020), DOI: 10.1016/j.jksuci.2020.10.016.

12. Kaur. R, Laxmi. V, and Balkrishan, "Performance evaluation of task scheduling algorithms in virtual cloud environment to minimize makespan", ***International Journal of Information Technology***, (2021), DOI: 10.1007/s41870-021-00753-4.

13. Sreenivasulu. G, and Paramasivam. I, "Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing", ***Evolutionary Intelligence***, Vol. 14, No. 2, (2021), 1015-1022, DOI: 10.1007/s12065-020-00517-2.

14. Zandvakili. A, Mansouri. N, and Javidi. M. M, "Energy-aware task scheduling in cloud compting based on discrete pathfinder algorithm", ***International Journal of Engineering, Transactions C: Aspects***, Vol. 34, No. 9, (2021), 2124-2136, doi: 10.5829/ije.2021.34.09c.10.

15. Uchechukwu. A, Li. K, and Shen. Y, "Energy consumption in cloud computing data centers", ***International Journal of Cloud Computing and Services Science***, Vol. 3, No. 3, (2014), 31-48, doi: 10.11591/closer.v3i3.6346.

16. Barroso. L. A, Clidaras. J, and Hölzle. U, "The datacenter as a computer: An introduction to the design of warehouse-scale machines", ***Synthesis Lectures on Computer Architecture***, Vol. 8, No. 3, (2013), 1-154, DOI: 10.2200/S00193ED1V01Y200905CAC006.

17. Sharma. M, and Garg. R, "HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers", ***Engineering Science and Technology, an International Journal***, Vol. 23, No. 1, (2020), 211-224, DOI: 10.1016/j.jestch.2019.03.009.

18. Hussain. M, Wei. L.-F, Lakhan. A, Wali. S, Ali. S, and Hussain. A, "Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing", ***Sustainable Computing: Informatics and Systems***, Vol. 30, (2021), DOI: 10.1016/j.suscom.2021.100517.

19. Dong. M, Fan. L, and Jing. C, "ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems", ***Journal of Systems and Software***, Vol. 158, (2019), DOI: 10.1016/j.jss.2019.110405.

20. Singh. H, Tyagi. S, Kumar. P, Gill. S. S, and Buyya. R, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions", ***Simulation Modelling Practice and Theory***, Vol. 111, (2021), DOI: 10.1016/j.simpat.2021.102353.

21. Meshkati. J, and Safi-Esfahani. F, "Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing", ***The Journal of Supercomputing***, Vol. 75, No. 5, (2019), 2455-2496, DOI: 10.1007/s11227-018-2626-9.

22. Sanaj. M. S, and Joe Prathap. P. M, "An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment", ***Materials Today: Proceedings***, Vol. 37, (2021), 3199-3208, DOI: 10.1016/j.matpr.2020.09.064.

23. Alboaneen. D, Tianfield. H, Zhang. Y, and Pranggono. B, "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers", ***Future Generation Computer Systems***, Vol. 115, (2021), 201-212, DOI: 10.1016/j.future.2020.08.036.

24. Houssein. E. H, Gad. A. G, Wazery. Y. M, and Suganthan. P. N, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends", ***Swarm and Evolutionary Computation***, Vol. 62, (2021), DOI: 10.1016/j.swevo.2021.100841.

25. Rai. D, and Tyagi. K, "Bio-inspired optimization techniques: a critical comparative study", ***ACM SIGSOFT Software Engineering Notes***, Vol. 38, No. 4, (2013), 1-7, DOI: 10.1145/2492248.2492271.

26. Beni. G, and Wang. J, "Swarm intelligence in cellular robotic systems", ***Robots and Biological Systems: Towards a New Bionics?***, Springer, (1993), 703-712, DOI: 10.1007/978-3-642-58069-7_38.

27. Shaheen. A. M, Spea. S. R, Farrag. S. M, and Abido. M. A, "A review of meta-heuristic algorithms for reactive power planning problem", ***Ain Shams Engineering Journal***, Vol. 9, No. 2, (2018), 215-231, DOI: 10.1016/j.asej.2015.12.003.

28. Kennedy. J, and Eberhart. R, "Particle swarm optimization", IEEE Proceedings of ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, (1995), DOI: 10.1109/ICNN.1995.488968.

29. Dorigo. M, Maniezzo. V, and Colorni. A, "Ant system: optimization by a colony of cooperating agents", ***IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)***, (1996), 1996, DOI: 10.1109/3477.484436.

30. Sreenu. K, and Sreelatha. M, "W-Scheduler: whale optimization for task scheduling in cloud computing", ***Cluster Computing***, Vol. 22, (2019), 1087-1098, DOI: 10.1007/s10586-017-1055-5.

31. Mirjalili. S, and Lewis. A, "The whale optimization algorithm", ***Advances in Engineering Software***, Vol. 95, (2016), 51-67, DOI: 10.1016/j.advengsoft.2016.01.008.

32. Zuo. L, Shu. L, Dong. S, Zhu. C, and Hara. T, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing", ***IEEE Access***, Vol. 3, (2015), 2687-2699, DOI: 10.1109/ACCESS.2015.2508940.

33. Zuo. X, Zhang. G, and Tan. W, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud", ***IEEE Transactions on Automation Science and Engineering***, Vol. 11, No. 2, (2013), 564-573, DOI: 10.1109/TASE.2013.2272758.

34. Sreenivasulu. G, and Paramasivam. I, "Hybrid optimization algorithm for task scheduling and virtual machine allocation in

cloud computing", ***Evolutionary Intelligence***, Vol. 14, (2021), DOI: 10.1007/s12065-020-00517-2.

35. Lin. W, Liang. C, Wang. J. Z, and Buyya. R, "Bandwidth-aware divisible task scheduling for cloud computing", ***Software: Practice and Experience***, Vol. 44, No. 2, (2014), 163-174, DOI: 10.1002/spe.2163.

36. Del Acebo. E, and de-la Rosa. J. L, "Introducing bar systems: a class of swarm intelligence optimization algorithms", In AISB 2008 Convention Communication, Interaction and Social Intelligence, Vol. 1, (2008), 1-18.

37. Mansouri. N, and Javidi. M. M, "Cost-based job scheduling strategy in cloud computing environments", ***Distributed and Parallel Databases***, Vol. 38, No. 2, (2020), 365-400, DOI: 10.1007/s10619-019-07273-y.

38. Calheiros. R. N, Ranjan. R, Beloglazov. A, De Rose. C. A. F, and Buyya. R, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", ***Software: Practice and Experience***, Vol. 41, No. 1, (2011), 23-50, DOI: 10.1002/spe.995.

39. Shojafar. M, Javanmardi. S, Abolfazli. S, and Cordeschi. N, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method", ***Cluster Computing***, Vol. 18, No. 2, (2015), 829-844, DOI: 10.1007/s10586-014-0420-x.

40. Xu. B, Zhao. C, Hu. E, and Hu. B, "Job scheduling algorithm based on Berger model in cloud environment", ***Advances in Engineering Software***, Vol. 42, (2011), No. 7, 419-425, DOI: 10.1016/j.advengsoft.2011.03.007.

41. Karthick. A. V, Ramaraj. E, and Subramanian. R. G, "An efficient multi queue job scheduling for cloud computing", 2014 World Congress on Computing and Communication Technologies, Trichirappalli, India, (2014), DOI: 10.1109/WCCCT.2014.8.

42. Babu. G, and Krishnasamy. K, "Task scheduling algorithm based on Hybrid Particle Swarm Optimization in cloud computing environment", ***Journal of Theoretical and Applied Information Technology***, Vol. 55, (2013), 33-38.

43. Kumar. S, and Kalra. M. A, " Hybrid Approach for Energy-Efficient Task Scheduling in Cloud ", Proceedings of 2nd International Conference on Communication, Computing and Networking, Singapore, (2018), DOI: 10.1007/978-981-13-1217-5_99.

44. Holland. J. H, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", MIT press, (1992).

45. Karaboga. D, " An idea based on honey bee swarm for numerical optimization " Technical report-tr06, Erciyes university, Engineering faculty, Computer engineering department, (2005).

46. Cotes-Ruiz. I. T, Prado. R. P, García-Galán. S, Muñoz-Expósito. J. E, and Ruiz-Reyes. N, "Dynamic voltage frequency scaling simulator for real workflows energy-aware management in green cloud computing", ***PloS One***, Vol. 12, No. 1, (2017), DOI: 10.1371/journal.pone.0169803.

47. Singh. S, and Kalra. M, "Scheduling of independent tasks in cloud computing using modified genetic algorithm", 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, India, (2014), DOI: 10.1109/CICN.2014.128.

48. Prem Jacob. T, and Pradeep. K, "A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization", ***Wireless Personal Communications***, Vol. 109, No. 1, (2019), 315-331, DOI: 10.1007/s11277-019-06566-w.

49. Yang. X.-S, and Deb. S, "Cuckoo search via Lévy flights", World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, (2009), doi: 10.1109/NABIC.2009.5393690.

50. Wu. D, "Cloud computing task scheduling policy based on improved particle swarm optimization", Proceedings - 2018 International Conference on Virtual Reality and Intelligent Systems, ICVRIS 2018, Hunan, China, (2018), DOI: 10.1109/ICVRIS.2018.00032.

51. Elaziz. M. A, Xiong. S, Jayasena. K. P. N, and Li. L, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution", ***Knowledge-Based Systems***, Vol. 169, (2019), 39-52, DOI: 10.1016/j.knosys.2019.01.023.

52. Wang. G.-G, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems", ***Memetic Computing***, Vol. 10, (2018), No. 2, 151-164, DOI: 10.1007/s12293-016-0212-3.

53. Storn. R, and Price. K, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces", ***Journal of Global Optimization***, Vol. 11, No. 4, (1997), 341-359, DOI: 10.1023/A:1008202821328.

54. Guo. X, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm", ***Alexandria Engineering Journal***, Vol. 60, No. 6, (2021), 5603-5609, DOI: 10.1016/j.aej.2021.04.051.

55. Sharma. M, and Garg. R, "An artificial neural network based approach for energy efficient task scheduling in cloud data centers", ***Sustainable Computing: Informatics and Systems***, Vol. 26, (2020), DOI: 10.1016/j.suscom.2020.100373.

56. Paknejad. P, Khorsand. R, and Ramezanpour. M, "Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment", ***Future Generation Computer Systems***, Vol. 117, (2021), 12-28, DOI: 10.1016/j.future.2020.11.002.

57. Wei. X, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing", ***Journal of Ambient Intelligence and Humanized Computing***, (2020), DOI: 10.1007/s12652-020-02614-7.

58. Chen. X, Cheng. L, Liu. C, Liu. Q, Liu. J, Mao. Y, and Murphy. J, "A WOA-based optimization approach for task scheduling in cloud computing systems", ***IEEE Systems Journal***, Vol. 14, No. 3, (2020), 3117-3128, DOI: 10.1109/JSYST.2019.2960088.

59. Tubishat. M, Abushariah. M. A. M, Idris. N, and Aljarah. I, "Improved whale optimization algorithm for feature selection in Arabic sentiment analysis", ***Applied Intelligence***, Vol. 49, No. 5, (2019), 1688-1707, DOI: 10.1007/s10489-018-1334-8.

60. Tos. U, Mokadem. R, Hameurlain. A, Ayav. T, and Bora. S, "A performance and profit oriented data replication strategy for cloud systems", 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, (2016), DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0125.

Persian Abstract

چکیده

رایانش ابری منابع محاسباتی مانند سخت‌افزار و نرم‌افزار را به عنوان خدمات از طریق شبکه برای کاربران فراهم می‌کند. زمانبندی کارها یکی از مسائل اصلی برای دستیابی به اجرای مقرون به صرفه است. هدف اصلی زمانبندی کارها اختصاص کارها به منابع است تا بتواند یک یا چند معیار را بهینه کند. از آنجا که مسئله زمانبندی کارها یکی از مسائل زمان چندجمله‌ای غیرقطعی سخت (*NP-hard*) است، الگوریتم‌های فراابتکاری به طور گسترده‌ای برای حل مسئله زمانبندی کار به کارگرفته شده‌اند. یکی از الگوریتم‌های فراابتکاری جدید الهام گرفته از زیست الگوریتم بهینه سازی مرغ دریایی (*SOA*)است. در این مقاله، ما یک الگوریتم آگاه از انرژی و مقرون به صرفه زمانبندی کار مبتنی‌بر *SOA (SOATS)* ارائه می‌کنیم. الگوریتم پیشنهادی قصد دارد با استفاده از تعداد تکرارهای کمتر، بین پنج هدف (یعنی مصرف انرژی، زمان اتمام کار، هزینه، زمان انتظار، و تعادل بار) تعادل ایجاد کند. نتایج آزمایش‌ها با مقایسه با چندین الگوریتم فراابتکاری (یعنی، الگوریتم ژنتیک (*GA*)، بهینه‌سازی ازدحام ذرات (*PSO*)، بهینه‌سازی کلونی مورچه‌ها (*ACO*) و الگوریتم بهینه‌سازی نهنگ‌ها (*WOA*)) نشان می‌دهد که روش پیشنهادی عملکرد بهتری در حل مسئله‌ی زمانبندی کارها دارد. علاوه بر این، ما الگوریتم پیشنهادی را با روش‌های زمانبندی کار مقایسه می‌کنیم: زمانبندی کار مبتنی بر هزینه (*CJS*)، الگوریتم جستجوی پروانه مبتنی بر تکامل تفاضلی (*MSDE*) و *Fuzzy-GA (FUGE)*. درمحیط با بار زیاد، الگوریتم *SOATS* مصرف انرژی را ۱۰٪ و هزینه را ۲۵٪ بهبود می‌بخشد.