Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 9(1): 847–867.
DOI: 10.3934/math.2024043
Received: 30 September 2023
Revised: 15 November 2023
Accepted: 22 November 2023
Published: 04 December 2023

Research article

Boosting task scheduling in IoT environments using an improved golden jackal optimization and artificial hummingbird algorithm

Ibrahim Attiya[1,2], Mohammed A. A. Al-qaness[3,4], Mohamed Abd Elaziz[1,5,6,7] and Ahmad O. Aseeri[8,*]

[1] Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt
[2] Faculty of Computer Science and Engineering, New Mansoura University, New Mansoura, Egypt
[3] College of Physics and Electronic Information Engineering, Zhejiang Normal University, Jinhua 321004, China
[4] Zhejiang Optoelectronics Research Institute, Jinhua 321004, China
[5] Artificial Intelligence Research Center (AIRC), Ajman University, Ajman 346, United Arab Emirates
[6] MEU Research Unit, Middle East University, Amman 11831, Jordan
[7] Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon
[8] Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia

* **Correspondence:** Email: a.aseeri@psau.edu.sa.

Abstract: Applications for the internet of things (IoT) have grown significantly in popularity in recent years, and this has caused a huge increase in the use of cloud services (CSs). In addition, cloud computing (CC) efficiently processes and stores generated application data, which is evident in the lengthened response times of sensitive applications. Moreover, CC bandwidth limitations and power consumption are still unresolved issues. In order to balance CC, fog computing (FC) has been developed. FC broadens its offering of CSs to target end users and edge devices. Due to its low processing capability, FC only handles light activities; jobs that require more time will be done via CC. This study presents an alternative task scheduling in an IoT environment based on improving the performance of the golden jackal optimization (GJO) using the artificial hummingbird algorithm (AHA). To test the effectiveness of the developed task scheduling technique named golden jackal artificial hummingbird (GJAH), we conducted a large number of experiments on two separate datasets with varying data sizing. The GJAH algorithm provides better performance than those competitive task scheduling methods. In particular, GJAH can schedule and carry out activities more effectively than other algorithms to reduce the makespan time and energy consumption in a cloud-fog computing environment.

Keywords: internet of things (IoT); golden jackal optimization (GJO); artificial hummingbird algorithm (AHA); energy consumption; cloud computing
Mathematics Subject Classification: 68M10, 68M11, 68M20, 90C29

1. Introduction

Systems are deemed efficient when all running applications deliver their best performance [1, 2]. As a result, excellent resource management and task scheduling (TS) are required for any system's efficient operation. The same is true in the cloud; if consumers want efficient performance, they must employ appropriate scheduling mechanisms [3, 4]. Scheduling is allocating available resources for applications delivering during a specific time period so that each application can utilize those resources to their fullest potential and provide the highest level of service quality feasible (QoS). Accordingly, various workloads can be distributed using various resource types while observing the constraints set by cloud customers and providers [3].

Cloud and fog computing is necessary for the internet of things (IoT) field. As known, the IoT has recently emerged as one of the most recognized and leading innovations in the field of communication information technology [5]. IoT extends the capabilities of conventional smart devices, such as tablets, laptops and smartphones, by using a variety of products, including new security systems, smart devices, cameras, machines, sensors, autos and others [6]. One of the main objectives of the IoT is to enable traffic control, transportation, healthcare, vehicle networks, energy management, manufacturing, medical treatment and other applications and services.

These applications often generate a huge quantity of data that necessitate processing, storage and analysis in order to derive valuable judgments that meet the user's goals and desires [7]. In recent years, metaheuristic (MH) optimization algorithms have provided efficient solutions for resolving real-world engineering and optimization problems based on naturally inspired behaviors [8, 9]. MH is based on bio-inspired algorithms generally developed by mimicking the behaviors of animals and insects, including ants, termites, birds, fish and others. They can be distinguished by emergent behaviors that originate from local interactions between an individual and the behaviors of the intelligent group. Several swarm intelligence-inspired algorithms have been presented and successfully implemented in a wide variety of applications. Also, recently proposed MH methods are still being researched to determine their efficacy.

The MH methods have been adopted to handle different problems in the IoT industry [8], including feature selection [10], sensing applications [11], intrusion detection systems [12] and others [13, 14]. In task scheduling, they also have been successfully implemented and they achieved prominent performance [15]. For example, the particle swarm optimization algorithm (PSO) has been utilized for scheduling tasks in the cloud and fog applications [16, 17]. Al-Turjman et al. [18] presented a TS technique using two variants of the PSO called the canonical PSO (CPSO) and fully informed PSO (FIPS). The suggested methods were employed to address the resource allocation in heterogeneous and homogeneous cloud applications. The main idea is to meet the requirement of the QoS according to throughput and delay. Simulation results assessed the quality of the two PSO variants, specifically the FIPS, which attained superior results compared to the CPSO. In [19], the authors developed a combined method of a genetic algorithm and ant colony optimization (HGA-ACO) for cloud scheduling. The main objective is to utilize the ACO to boost the GA searchability. This hybrid approach was utilized to consider the optimal scheduling outputs and to obtain the optimal resource on response time. In [20], a new variant of the Henry gas solubility optimizer (HGSO) was developed to optimize the scheduling of cloud requests within IoT environments. This new variant of the HGSO, called Henry gas solubility whale comprehensive (HGSWC), was improved using the operators of the whale optimizer (WOA) and opposition-based learning. In [21], a hybrid approach of the grasshopper optimization algorithm (GOA) and antlion optimization (ALO) was developed to enhance the cloud scheduling process. The target of this approach was to minimize the makespan, energy consumption and task cost, as well as increase the throughput.

In [22], a new version of the arithmetic optimization algorithm (AOA) was developed to address job scheduling in fog computing. The marine predators algorithm (MPA) was used to modify the search process of the traditional AOA. The main objective of the modified AOA algorithm is to maximize the QoS and minimize makespan. In [23], a hybrid moth flame optimization (HMFO) was proposed to address the scheduling problems on the internet of health things (IoHT) applications. The main target of this approach is to boost the QoS and enhance response time, resource allocation and energy consumption. In [24], a new virtual machine scheduling approach was proposed using an MH called the entropy value krill herd optimization algorithm. This method outperformed the traditional krill herd optimization algorithm and several well-known optimization algorithms. In [25], a hybrid PSO with gray wolf optimizer (GWO) was presented to address the task scheduling in clouds. The main objective was to minimize the execution time. The PSO-GWO provided better efficiency than GWO and PSO algorithms. Additionally, in [26], the authors studied the applications of several optimization algorithms for cloud scheduling, such as PSO, the cuckoo search (CS) algorithm, WOA and the bat algorithm (BA). They concluded that WOA achieved the best performance. Moreover, a new variant of the GWO called the multi-objective GWO was developed by [27]. This new variant was applied to determine the optimal task scheduling solutions in the cloud environments.

As noticed from the aforementioned studies, MH methods showed acceptable performance in this field. Accordingly, this study presents an efficient MH method for cloud and fog computing scheduling in IoT applications. The proposed approach seeks to enhance the quality of the solution of task scheduling based on the integration between the golden jackal optimization (GJO) and artificial hummingbird algorithm (AHA). Since the GJO still has some limitations that influence the performance of the final output, this motivated us to apply AHA. In general, the AHA is a recently developed bio-inspired algorithm for tackling engineering and optimization problems. It was inspired by the foraging strategies and special flight skills of hummingbirds. As described in [28], hummingbirds possess three distinct flight abilities, namely, omnidirectional, axial and diagonal. Additionally, there are several foraging strategies implemented in searching for food processes, such as territorial, guided and migrating foraging. Also, a memory table can be generated to build the memory function of the hummingbirds to remember food sources. It showed competitive performance in different optimization problems compared to previous methods. More so, the GJO is a new proposed naturally inspired method developed by [29]. It emulated the hunting characteristics of the Canis aureus. The GJO has three main steps: Prey searching, enclosing, and pouncing. According to those three steps, the mathematical model was built to solve optimization problems. Similar to the AHA, the GJO also showed significant performance in solving optimization and engineering problems. Like other optimization algorithms, each individual MH optimization method faces some shortcomings and challenges while searching for optimal solutions. For example, they can get stuck at local points, which leads to the degradation of the quality of the solution. To overcome those shortcomings and limitations, the hybrid concept can be employed by using the operators of an MH technique to boost the search process of another algorithm.

Following this concept, in this research we develop an alternative task scheduling based on improving the efficiency of GJO using the AHA algorithm in an IoT environment. The AHA is utilized to improve the traditional GJO's search capability. The proposed golden jackal artificial hummingbird (GJAH) starts by generating a set of individuals that represent the solution of task scheduling, then evaluates these individuals using the fitness value to determine the best individual. Therefore, the competition between GJO and AHA is applied to update the current individuals, and this process is performed until reached to the stop conditions.

In short, the main objective of this paper can be presented as:

- Proposing a task scheduling approach in the IoT environment using a modified version of the GJO algorithm.
- Using the operators of AHA to enhance the searching performance of GJO through enhancing its exploitation ability.
- Assessing the developed method's efficiency using various datasets and comparing it to a set of well known task scheduling methods-based MH techniques.

The rest of this work is presented as: The problem definition and basics of MH algorithms are given in section two. In section three, the phases of the designed approach are introduced. The findings and in-depth discussions are provided in section four. Section five brings the conclusion and outlines future directions.

2. Background

2.1. Problem formulation

Task scheduling is considered a critical problem in the environment of cloud-fog computing. The description of the TS can be formulated by assuming there is a set of physical servers that have variant storage capacity, a number of processors, network bandwidth and memory size [7]. In addition, to achieve the required SLAs and QOS, these servers can be scaled up or down. Consider $VM = \{VM_1, VM_2, ..., VM_m\}$ as a set of virtual machines (VMs) that exist in the data center. In addition, each VM_j has its computational ability computed using millions of instructions per second (MIPS), whereas $T = \{T_1, T_2, ..., T_n\}$ refers to a list of demands sent by cloud members to be conducted on the collection of VMs. In addition, the length of T_i is given as TL_i defined in terms of millions of instructions (MI).

Within this paper, to preserve the time expected to conduct the task in VMs, the expected time to compute (ETC) matrix is applied and each element in this matrix ETC_{ij} is defined as [30]:

$$ETC_{ij} = \frac{TL_i}{VMP_j}, \ 1 \leq i \leq n, 1 \leq j \leq m. \tag{2.1}$$

In general, ETC_{ij} refers to the ETC of the task i performed on the VM j, whereas VMP_j is the processing power of the VM j.

In the current research, we aim to tackle the TS problem as a minimization optimization problem by handling two objectives named makespan (MKS) and energy efficiency ($TEng$). This problem can be formulated as:

$$F = \phi \times TEng + (1 - \phi) \times MKS. \tag{2.2}$$

In Eq (2.2), ϕ refers to a value applied to balance between the two objectives. In addition, MKS is formulated as in Eq (2.3), which represents the amount of time required to complete all tasks [31]:

$$MKS = \max_{j \in 1,2,\dots,m} \sum_{i=1}^{n} ETC_{ij}. \tag{2.3}$$

Moreover, $TEng$ refers to the total energy consumption and it is defined as:

$$TEng = \sum_{j=1}^{m} Eng(VM_j), \tag{2.4}$$

where $Eng(VM_j)$ represents the consumed energy (Joules) by VM_j and is defined as [7]:

$$\begin{aligned} Eng(VM_j) &= VMP_j \times (\beta_j \times TE_j + \alpha_j \times (MKS - TE_j)), & (2.5)\\ \alpha_j &= 0.6 \times \beta_j, & (2.6)\\ \beta_j &= 10^{-8} \times VMP_j^2. & (2.7) \end{aligned}$$

Since the two factors (i.e., energy consumption and makespan) have the largest influence on the quality of the cloud-fog system, the main target is to provide the smallest makespan with low energy consumption. According to this formulation, the TS is considered as a bi-objective optimization problem.

2.2. AHA

The AHA algorithm is a novel MH methodology that mimics hummingbird activity [28]. The three flight abilities-axial, diagonal, and omnidirectional-are utilized for foraging mechanisms.

Following [28], the process of creating the initial X population of N hummingbirds is the first step, and this is formulated as:

$$X_j = r \times (U - L) + L, \ \ j = 1, 2, \dots, N, \tag{2.8}$$

in which U and L denote the limits of each value of X_j and $r \in [0, 1]$ denotes the random value. Meanwhile, the visit table related to the X_b is represented as:

$$VT_{ji} = \begin{cases} 0 & if\ j \neq i \\ null & j = i \end{cases}, i, j = 1, \dots, N. \tag{2.9}$$

If $i = j$, then $VT_{ji} = null$ denotes the amount of food a hummingbird found at a particular location. The jth hummingbird for visiting the i source of food is referred to as the $VT_{ji} = 0$.

2.2.1. Guided foraging

The hummingbird is supposed to discover the food that has the maximum visiting rate and then choose the agent that has the highest rate of nectar-refilling from X, which represents the best agent for guided foraging behavior. The three aspects of flight used in this forage are omnidirectional, diagonal and axial. The definition of axial flight ($D_i, i = 1, \ldots, d$) can be presented as follows:

$$D_i = \begin{cases} 1 & if\ i = R \\ 0 & else \end{cases}. \tag{2.10}$$

Moreover, the diagonal flight is given as:

$$D_i = \begin{cases} 1 & if\ i = P(j),\ j \in [1, k],\ P = randperm(k) \\ 0 & else \end{cases}, \tag{2.11}$$

where $randperm(k) \in [1, k]$ is a random integer and $k \in [2, \lceil r_1(d-2) \rceil + 1]$. In addition, the formulation of omnidirectional flight is given as:

$$D_i = 1,\ i = 1, \ldots, d. \tag{2.12}$$

In Eq (2.12), $R \in [1, d]$ denotes a random value and $r_1 \in [0, 1]$ stands to a random value. Additionally, the updating of the solution based on guided foraging is formulated as:

$$V_i(t+1) = X_i(t) + a \times D \times (X_i(t) - X_i(t)), \tag{2.13}$$

where $X_i(t)$ indicates the ith value at iteration t of X. $a \in N(0, 1)$ represents random value. $X_i(t)$ denotes the desired solution explored by X_i. Thus, X_i may be improved as follows:

$$X_i(t+1) = \begin{cases} X_i(t), & if\ f(X_i(t)) \leq f(V_i(t+1)) \\ V_i(t+1), & otherwise, \end{cases} \tag{2.14}$$

whereas f denotes the fitness values.

2.2.2. Territorial foraging

After consuming all of the nectar from a flower, a hummingbird searches for food rather than visiting other flowers. Therefore, the bird might make a rapid migration to an area close to its territory where a new food source might be located in place of the old one. The following formula was introduced to emulate the local foraging of hummingbirds and a potential solution:

$$V_i(t+1) = b \times D \times X_i(t) + X_i(t),\ \ b \in N(0, 1). \tag{2.15}$$

2.2.3. Migration foraging

The hummingbird may migrate to a new spot farther away if its preferred feeding location is in need of food. The hummingbird steps will migrate to a position with one of the worst rates of nectar replenishment selected randomly from the search domain when the number of generations exceeds the predetermined coefficient of migration. Therefore, VT will be enhanced as this hummingbird switches from using the old solution to the new solution. A hummingbird will migrate to a new nectar source established at random from the source with the lowest nectar refill rates.

$$X_w(t+1) = r \times (U - L) + L. \tag{2.16}$$

In Eq (2.16), X_w denotes the worst fitness values. Figure 1 shows the workflow of the AHA.

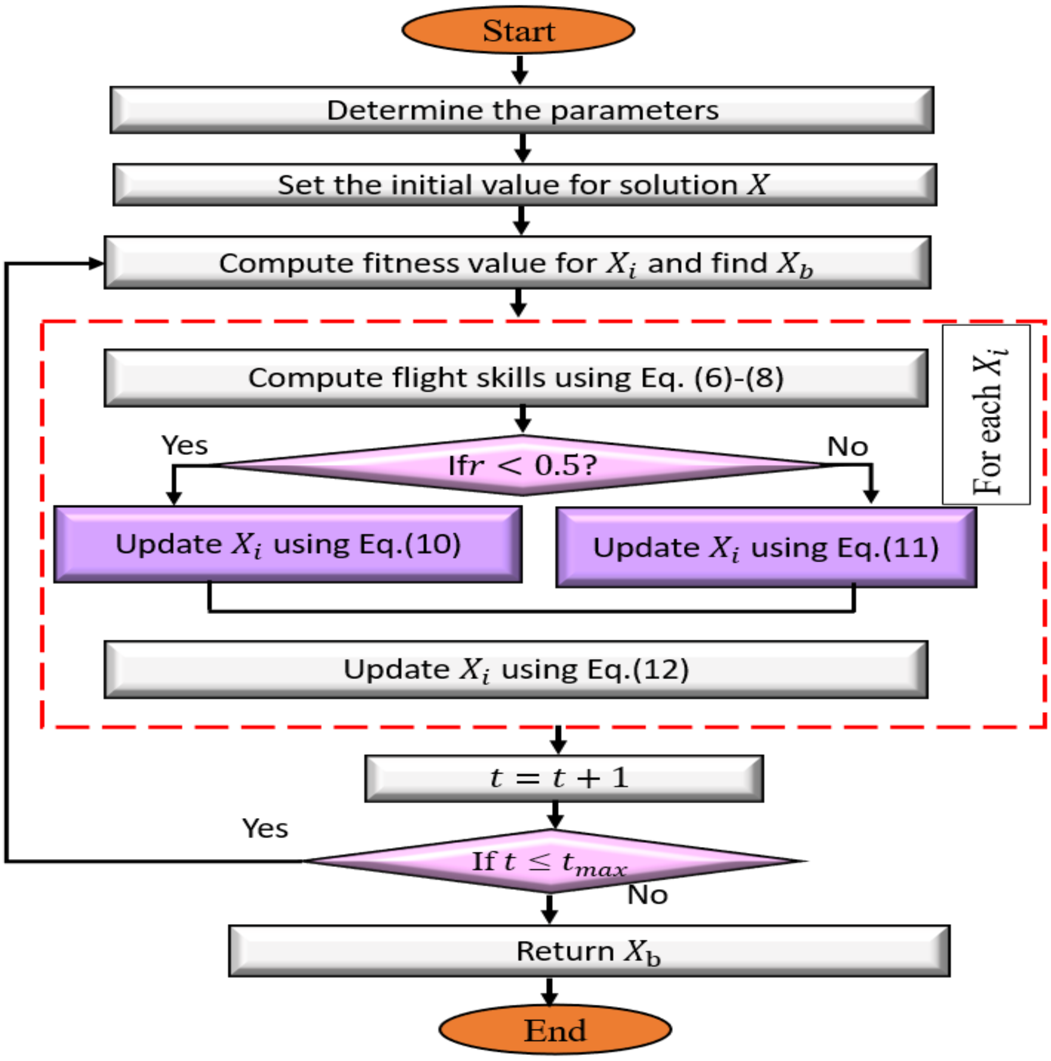
Start
Determine the parameters
Set the initial value for solution X
Compute fitness value for X_i and find X_b
Compute flight skills using Eq. (6)-(8)
For each X_i
Yes
Ifr < 0.5?
No
Update X_i using Eq.(10)
Update X_i using Eq.(11)
Update X_i using Eq.(12)
$t = t + 1$
Yes
If $t \leq t_{max}$
No
Return X_b
End

Figure 1. Steps of AHA.

2.3. *GJO*

The information of the GJO [29] is presented in this section. This algorithm is similar to other swarm algorithms, which starts by producing a population of agents using Eq (2.17):

$$X_i = L + r \times (U - L), \quad 1 = 1, 2, \ldots, N. \tag{2.17}$$

After that, the fitness value of X_i is assessed, then it allocates the best jackal (named male jackal) X_b, and the second one is named the female jackal for prey position. Thereafter, the matrix prey (X_P) is constructed using the following formula as an initial value,

$$X_P = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1d} \\ X_{21} & X_{22} & \ldots & X_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ X_{N1} & X_{N2} & \ldots & X_{Nd} \end{bmatrix}. \tag{2.18}$$

2.3.1. Steps of exploration

Following [29], the hunting process is the responsibility of the male jackal (X_M) while the female (X_{FM}) follows it. This formulates the exploration process of GJO to find the prey, and it can be given as:

$$X_1(t) = X_M(t) - E \times |X_M(t) - r_l \times X_P(t)|, \tag{2.19}$$

$$X_2(t) = X_{FM}(t) - E \times |X_{FM}(t) - r_l \times X_P(t)|, \tag{2.20}$$

where X_1 and X_2 are the enhanced value of X_M and X_{FM}, respectively. In addition, E represents the energy of prey that is computed as:

$$E = E_0 \times E_1, \tag{2.21}$$

where E_0 and E_1 refer to the initial value of energy and decreasing energy value, respectively, which are computed as:

$$E_0 = 2 \times r - 1, \tag{2.22}$$

$$E_1 = c_1 \times (1 - (t/T)), \tag{2.23}$$

where T refers to the total iteration number. Moreover, r_l is a random value obtained using the Levy distribution, and this is represented as:

$$r_l = Levy \times 0.05. \tag{2.24}$$

Finally, the values of jackals are improved by employing Eq (2.25):

$$X(t+1) = \frac{X_1(t) + X_2(t)}{2}. \tag{2.25}$$

2.3.2. Steps of exploitation

In this stage, the exploitation behavior of GJO is introduced. In general, this process is formulated by simulating the hunting of male and female jackals as:

$$X_1(t) = X_M(t) - E \times |r_l \times X_M(t) - X_P(t)|, \tag{2.26}$$

$$X_2(t) = X_{FM}(t) - E \times |r_l \times X_{FM}(t) - X_P(t)|. \tag{2.27}$$

2.3.3. Switching between exploration and exploitation

The process of transition between search phases is made possible by the prey's escape energy. The solutions start their exploration stage if the value $|E| > 1$; otherwise, they start their exploitation stage. Figure 2 provides an illustration of the GJO process.

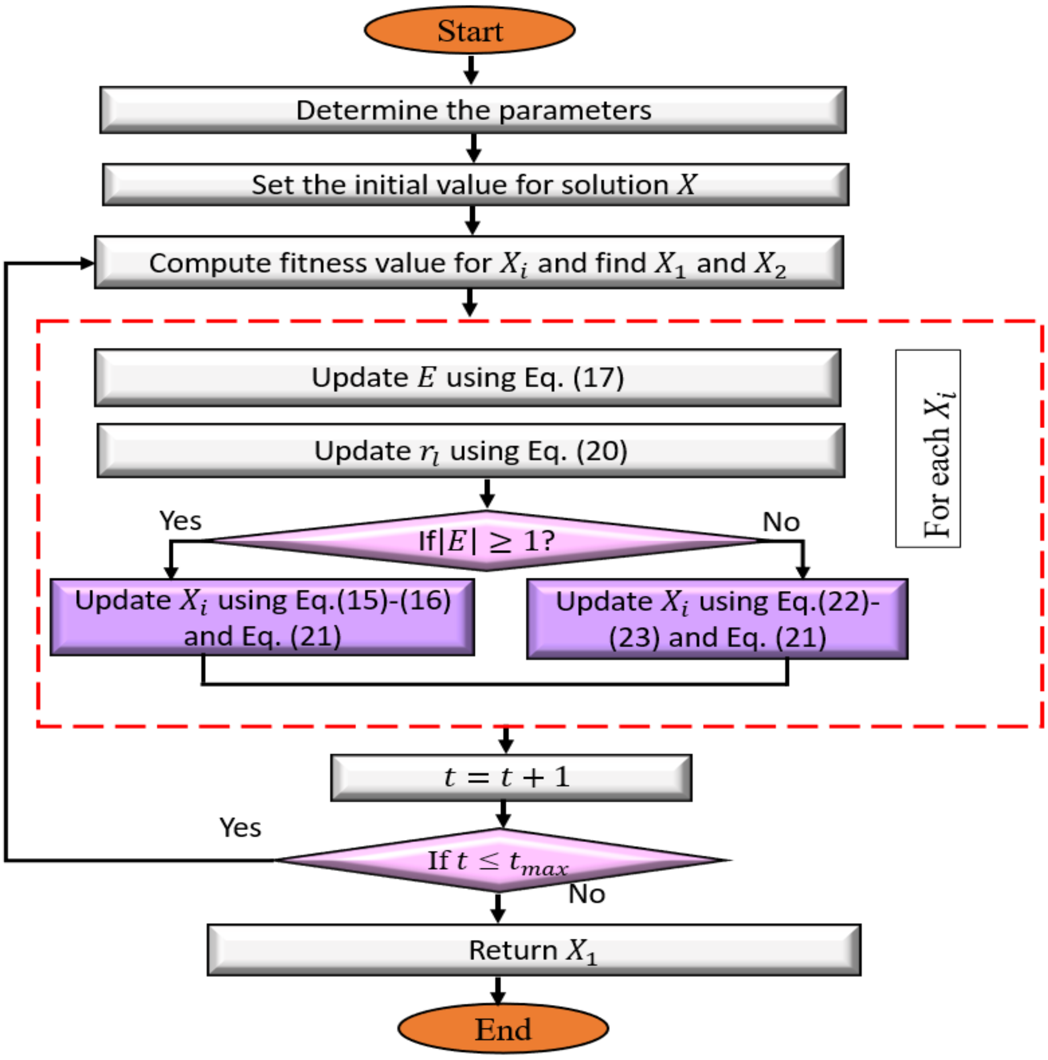
Start
Determine the parameters
Set the initial value for solution X
Compute fitness value for X_i and find X_1 and X_2
Update E using Eq. (17)
Update r_l using Eq. (20)
For each X_i
If$|E| \geq 1$?
Yes
No
Update X_i using Eq.(15)-(16) and Eq. (21)
Update X_i using Eq.(22)-(23) and Eq. (21)
$t = t + 1$
If $t \leq t_{max}$
Yes
No
Return X_1
End

Figure 2. Steps of GJO algorithm.

3. The developed task scheduling method

Figure 3 depicts the main layout of the task scheduler that was designed using the modified GJO. The AHA operators are utilized to boost the GJO's exploitation capability.

In the proposed GJAH, a collection of N agents (X) are first created and each of them is considered as a solution to fix the scheduling problem. More so, the fitness function (Fit_i) is employed to assess the quality of the candidate agents ($X_i, i = 1, 2, ..., N$), then the best solution (X_b) that has the smallest Fit_b is decided. The next step is to enhance X by utilizing the competition between GJO and AHA. This can be performed by using the operators of GJO to discover the feasible regions, while the operators of AHA are used during the exploitation phase. This leads to improving the convergence toward the optimal solution. In Figure 3, the GJAH framework for the IoT scheduling problem is displayed. The stages of GJAH are further illustrated in what follows.

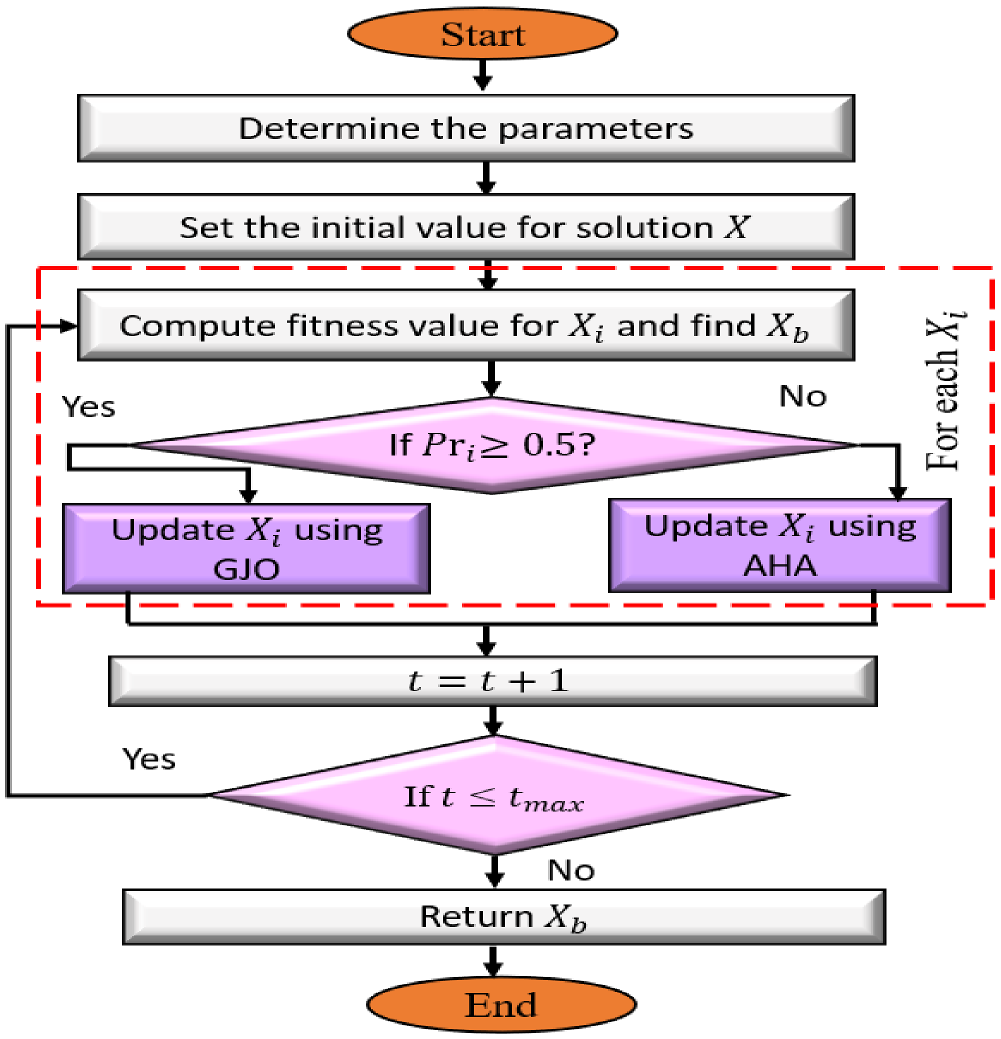

Figure 3. Developed GJAH method.

3.1. Construction population

The main aim of this step is to generate the population X, which denotes the configuration of tasks distributed on machines. This process can be represented as:

$$X_{ij} = floor(L_{ij} + \alpha \times (U_{ij} - L_{ij})),\ \alpha \in [0, 1],\ j = 1, 2, ..., n, \tag{3.1}$$

where the function $floor(.)$ is employed to convert real values into integer ones. L and U are defined as the boundaries of the search problem. This formulation in Eq (3.1) denotes the process of handling task scheduling as a discrete problem.

3.2. Updating population

This step is employed for updating X solutions by applying the operators of GJO and AHA. This can be achieved by using the computed fitness values (Fit_i) of each X_i, as given in Eq (2.2). Moreover, the following step is to determine X_b (the one that recorded the smallest fitness value). After that, we update X_i using either of the operators of the exploration phase of GJO, as defined in Eqs (2.25) and (2.26). Meanwhile, during the steps of exploitation, either GJO or AHA are employed for updating the values of X_i. This process is defined as:

$$X_i(t+1) = \begin{cases} Using\ GJO\ to\ update\ X_i, & if\ \ Pr_i \geq 0.5 \\ Using\ AHA\ to\ update\ X_i, & otherwise, \end{cases}. \tag{3.2}$$

where $Pr_i \in [0, 1]$ denotes a random value that can be employed to manage the enhancement of X based on either GJO or AHA operators. This process of updating the solutions X generates competition between the operators of GJO and AHA. In this case, the GJAH avoids the attraction to the local optima and increases the convergence toward the optimal solutions. The termination condition is then examined and, if it is satisfied, the developed GJAH is stopped and the optimal X_b is returned; otherwise, the GJAH stages are repeated. Algorithm 1 illustrates the developed GJAH's pseudo-code.

Algorithm 1 The developed GJAH.

```
1: Inputs: $n$ IoT tasks, $m$ group of VMs, $T$ iterations, $N$ number of solutions;
2: Form $X$ (initial solutions).
3: $t = 1$.
4: while $t <= T$ do
5:     Evaluate $X_i$ by calculating its $Fit_i$.
6:     Determining $X_b$ that obtained the best $Fit_b$.
7:     for $i = 1 : N$ do
8:         Improving $X_i$ by employing Eq (3.2).
9:     $t = t + 1$.
10: Return $X_b$.
```

4. Experimental studies

In this section, the feasibility of the developed GJAH approach is examined through conducting a series of experiments. We first describe the experimental setup and specifications of the dataset used in the experiments (Subsection 4.1). Thereafter, we explain the metrics defined for the evaluation of our proposal (Subsection 4.2). In the end, we present and analyze the experimental results as well as give some remarks on the key findings (Subsection 4.3).

4.1. Experimental setup

All the results were obtained using a Dell PC with an Intel Core i5-4200U 2.4 GHz CPU and 4 GB of RAM, running Windows 10 and MATLAB R2018a. The employed cloud-fog environment comprises of two data centers and 20 nodes of varying configurations hosted on two host machines. Table 1 shows the configurations of the hosts and computing nodes. The processing power of computing nodes is considered in MIPS. The findings reported below are an average of 20 independent runs.

The performance assessment was performed using synthesized and real datasets produced from real systems. In particular, a total of 1,500 user requests with lengths between 2,000 and 56,000

MI were produced in the synthetic workload using a uniform distribution. The synthetic workload characteristics are presented in Table 2. Likewise, the real workloads comprising HPC2N (High-Performance Computing Center North) and NASA (National Aeronautics and Space Administration) Ames iPCS/860 (Intel Personal SuperComputer) are utilized in the experiments. The NASA Ames iPSC/860 workload captures statistics for 42,264 tasks, while the HPC2N workload captures statistics for 527,371 tasks. Table 3 demonstrates the specifics of those parallel workloads.

Table 1. Cloud-fog infrastructure characteristics.

Entity	Parameter	Default Value
Subscribers	No. of subscribers	100 - 200
Host	Number of hosts	4
Cloud nodes:		
	Number of nodes	10
	Processing capacity	3000 - 5000 MIPS
	RAM	2 GB
	Bandwidth	1 Gb/s
	Storage	10 GB
Fog nodes:		
	Number of nodes	10
	Processing capacity	1000 - 2800 MIPS
	RAM	1 GB
	Bandwidth	1 Gb/s
	Storage	4 GB

Table 2. Characteristics of the synthetic workload.

Parameter	Value
Task's length	[2000, 56000] MI
File size	[400, 600] MB
Number of tasks	[300, 1500]
Output size	[400, 600] MB

Table 3. Characteristics of real workloads.

Workload	Period	Months	Users	Tasks	CPUs
HPC2N	Jul 02–Jan 06	42	257	527,371	240
NASA iPSC	Oct 93–Dec 93	3	69	42,264	128

4.2. *Evaluation measures*

This subsection discusses the performance measures employed to assess the efficiency of the GJAH against its competitors from the literature.

Makespan is one of the most important efficiency evaluation criteria, which is expressed as the completion time of the last performed task. Minimization of makespan ensures better QoS to the cloud users. In general, makespan is computed based on Eq (2.3).

Total energy consumption is another metric that should be minimized to improve the system's performance. It can be calculated by summing the energy consumed by all computing nodes, as defined in Eq (2.4).

The aim of this study is to guarantee a minimum total energy consumption while reducing the makespan.

4.3. *Experimental analysis*

This section exhibits and analyzes the evaluation outcomes achieved by the suggested GJAH algorithm. To validate the success of the GJAH, it was compared to several widely established algorithms in the field. In this paper, GJO [29], AHA [28], Atomic Orbital Search (AOS) [32], MPA [33], AOA [34], Dynamic Jellyfish Simulated Disruption (DJSD) [35] and the chameleon swarm algorithm (CSA) [36] are selected as compared approaches. The parameter settings for the presented GJAH method and other contrasting techniques are listed in Table 4. The common parameters such as the number of iterations and size of the population are assigned to 1,000 and 50, respectively. Each experiment was conducted under 30 repetitions and the achieved results were reported. Furthermore, ϕ is set to 0.7.

Table 4. Parameter settings for the techniques.

Technique	Parameters	Values
AHA	R	[0,1]
	a, b	[0,1]
GJO	c_1	0.5
	r, r_l	[0,1]
MPA	$FADs$	0.2
	P	0.5
DJSD	γ	0.1
	η	4.0
CSA	γ	2
	α	4
	ρ	1
	β	3
AOA	MOP_{Max}	1
	MOP_{Min}	0.2
	α	5
	μ	0.499
GJAH	R	[0,1]
	a, b	[0,1]
	c_1	0.5
	r, r_l	[0,1]
AOS	Foton rate	0.1
	Total number of layers	5

To testify the GJAH performance on different datasets and real applications, the performance curves in terms of the average fitness values are demonstrated in Figures 4–6. In particular, the performance curve of the synthetic workload is presented in Figure 4. We can see that GJAH significantly outperforms other peer approaches in all the instances of the synthetic workload. In a similar manner, the performance curve of the NASA Ames iPSC/860 dataset displayed in Figure 5 reveals that the GJAH method is better than several existing scheduling algorithms. In addition, GJAH performs better than the compared seven techniques for the HPC2N workload when the number of tasks varies between 1,000 and 5,000, as shown in Figure 6. Overall, according to the values of the fitness function, we can see that GJAH offers the most promising performance, followed by AHA and AOS.

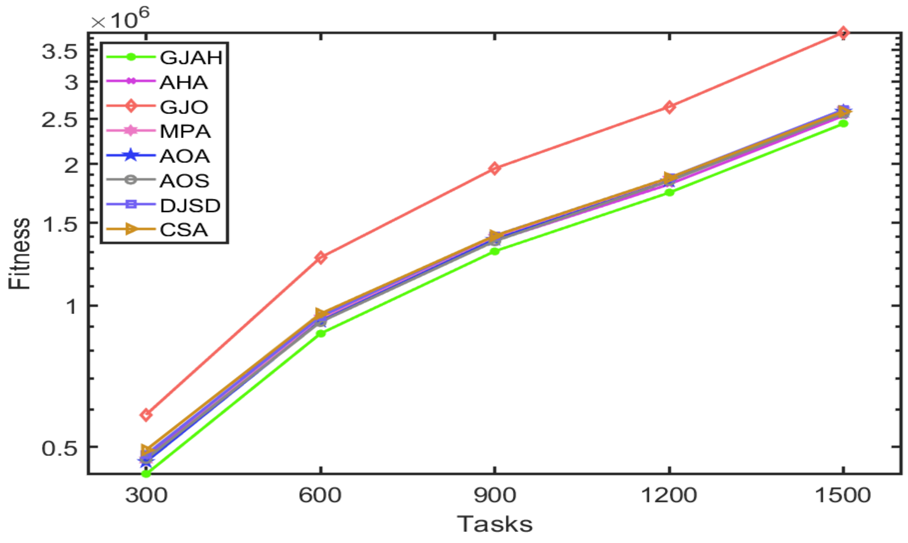

Figure 4. Synthetic workload fitness values.

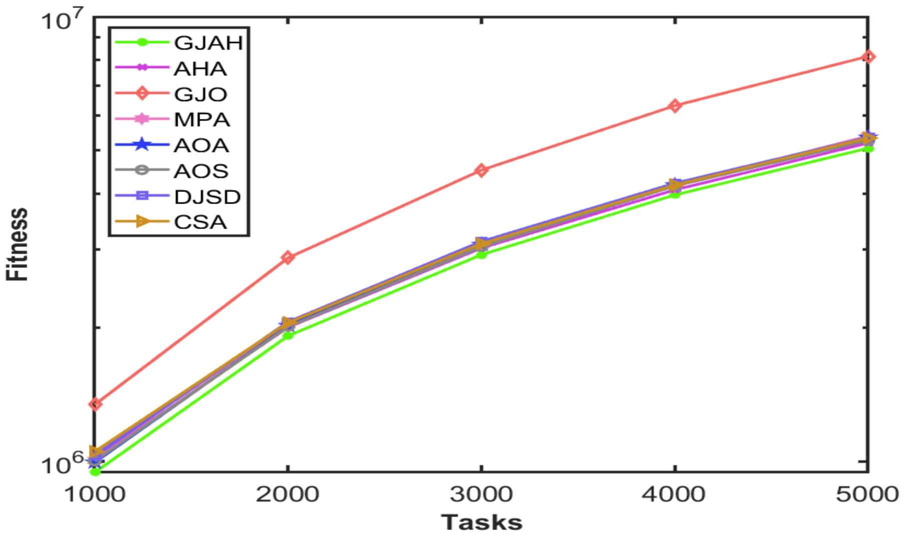

Figure 5. Real workload NASA iPSC fitness values.

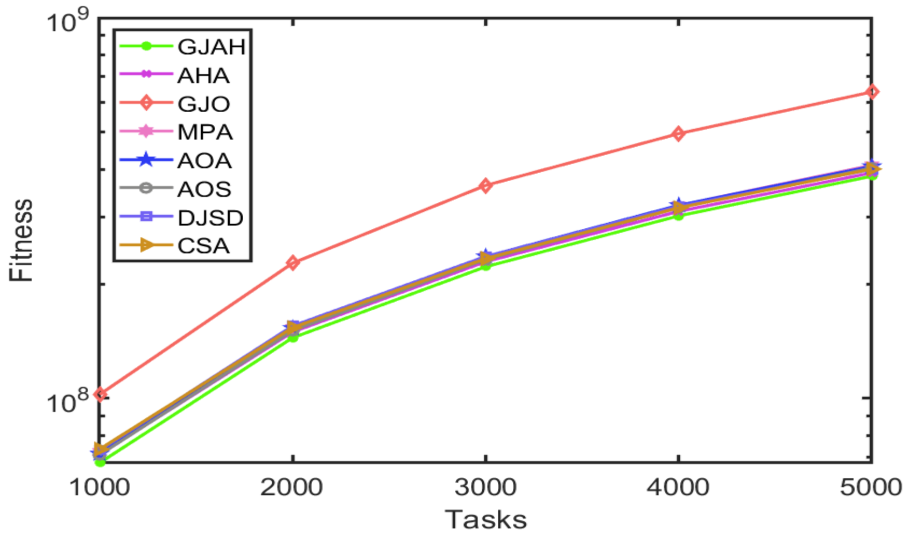

Figure 6. Real workload HPC2N fitness values.

The comparison results on average makespan produced by GJAH, AHA, GJO, MPA, AOA, AOS, DJSD and CSA for the considered workload traces are provided in Figures 7–9. Specifically, GJAH attains the best average makespan on the synthetic dataset compared to the standard GJO algorithm and other approaches, as shown in Figure 7. Similarly, we can see from Figure 8 that GJAH achieved the best results on the NASA Ames iPSC workload in terms of the average makespan. Furthermore, Figure 9 exhibits that GJAH also yields better results than the other seven peer approaches on the HPC2N workload. After all, the obtained results indicate that GJAH demonstrates average improvements in makespan time, ranging from 6.27% to 27.15% for synthetic workloads, 7.23% to 23.42% for NASA iPSC workloads and 5.12% to 23.82% for HPC2N workloads when compared to MPA, AOA, AOS, DJSD and CSA algorithms.

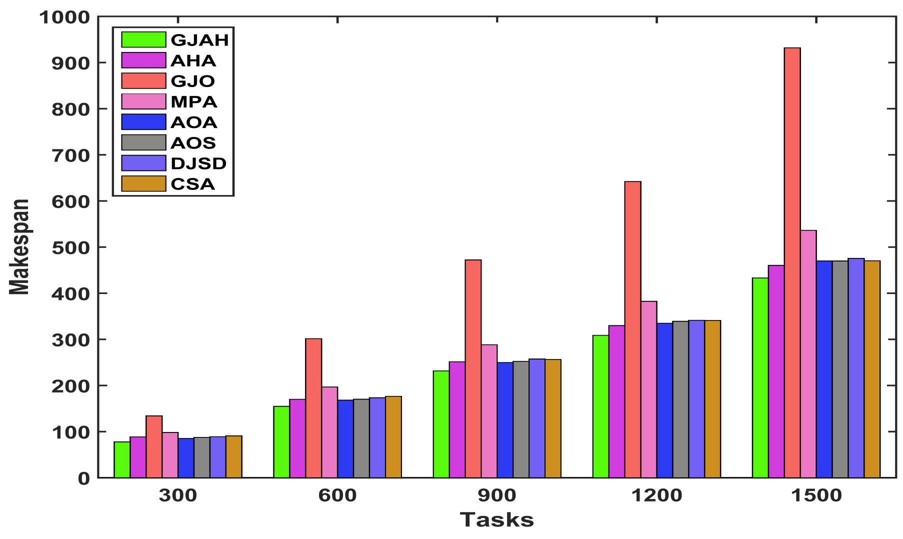

Figure 7. The results of the synthetic workload average makespan.

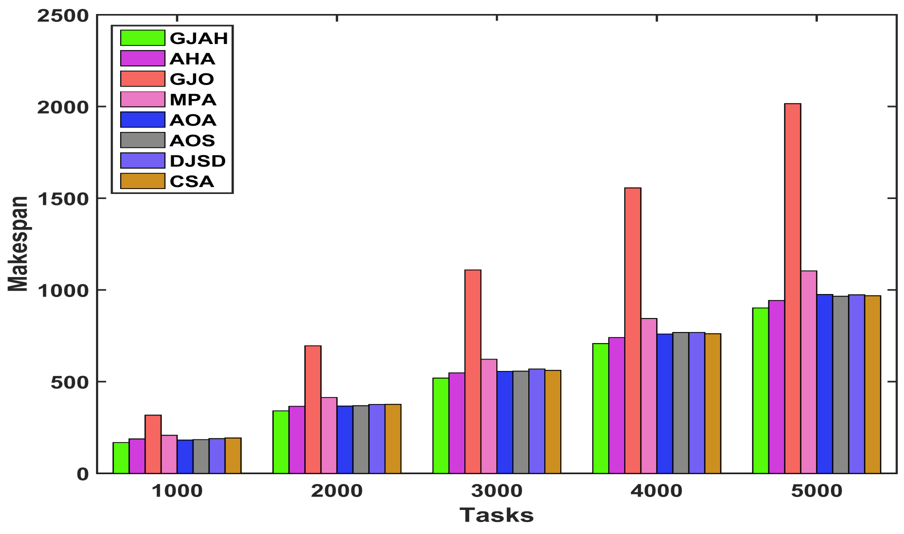

Figure 8. The results of real workload NASA iPSC average makespan.

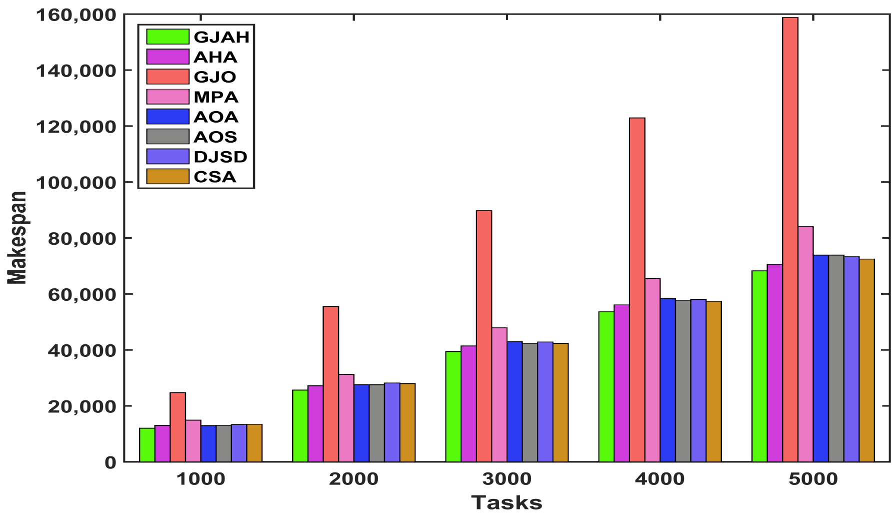

Figure 9. The results of real workload HPC2N average makespan.

Figures 10 to 12 provide the numerical results of the comparisons in terms of the achieved total energy consumption between GJAH, AHA, GJO, MPA, AOA, AOS, DJSD and CSA algorithms using different workload traces. In Figure 10 it can be noticed that the GJAH strategy has the best overall energy consumption compared to the other approaches. Similarly, in Figure 11, the proposed GJAH achieves the best energy consumption for all the test instances of the NASA Ames iPSC workload. In addition to this, GJAH also gives a better consumption of the total energy among computational nodes for the HPC2N workload instances, as can be observed in Figure 12. After all, the results obtained reveal that GJAH exhibits average enhancements in total energy consumption, varying from 3.94% to 56.43% for synthetic workloads, 3.51% to 60.94% for NASA iPSC workloads and 3.12% to 66.46% for HPC2N workloads in comparison to peer scheduling algorithms.

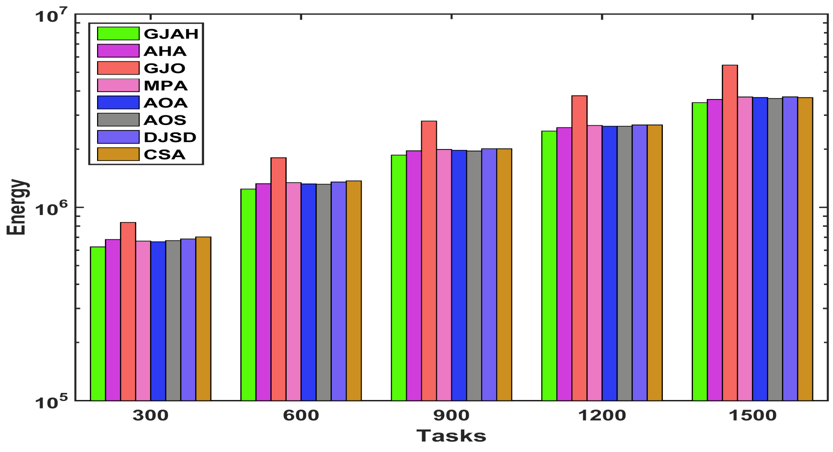

Figure 10. The total energy consumption for the synthetic workload.

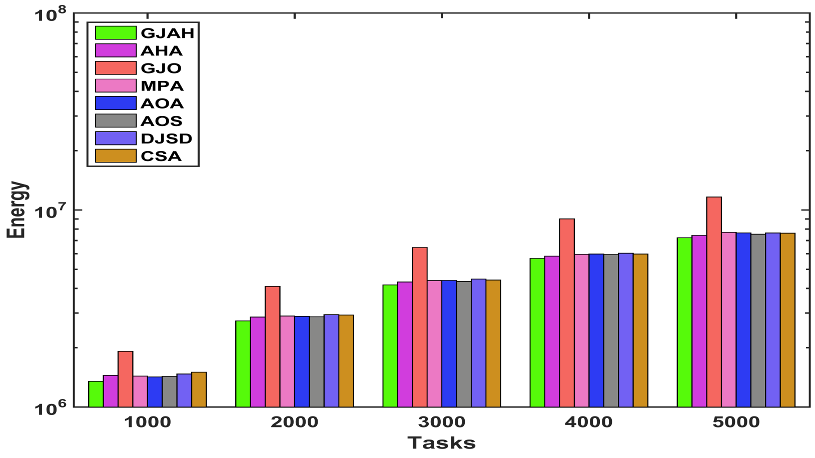

Figure 11. The total energy consumption for the real workload NASA iPSC.

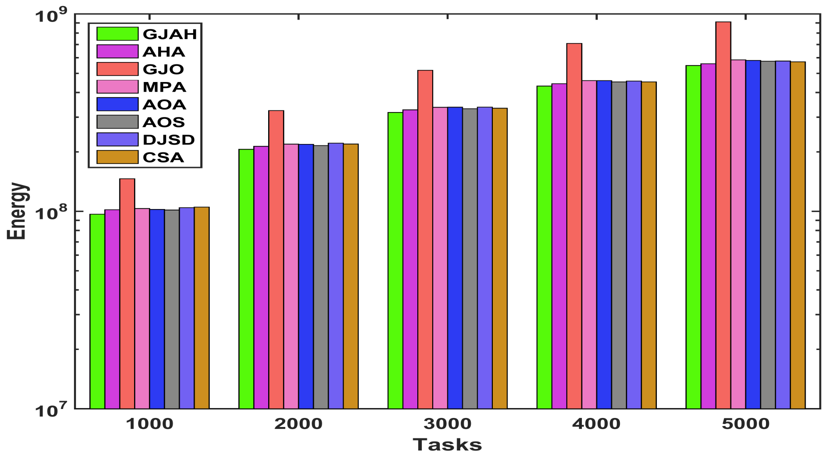

Figure 12. The total energy consumption for the real workload HPC2N.

In summary, the experimental outcomes showed that GJAH outperformed other compared methods on all investigated dataset instances and achieved the most favorable performance in terms of minimizing the makespan while optimizing the energy consumption of computing resources. Thus, incorporating AHA with the GJO can efficiently boost the search capability to produce more efficient solutions for all evaluated instances.

5. Conclusion and future works

In order to enhance the multi-objective functionality of IoT applications and deliver better services in less time while rationalizing energy consumption during task execution, the GJAH algorithm was developed in this paper to address task scheduling issues within cloud-fog computing environments. In our proposal, the GJO has been combined with the AHA to boost the ability of GJO based on the operators of AHA. Comprehensive experiments with different datasets and varying numbers of tasks were carried out to verify the performance of the GJAH. The results have shown the effectiveness of GJAH and superior performance over the basic GJO and other peer scheduling algorithms. The presented GJAH method can be extended in future work to address the IoT task scheduling in cloud-fog environments, considering different objective functions and various workload types. Moreover, GJAH may also be employed in other optimization tasks, such as knapsack problems, job scheduling, traveling salesman problems and advanced industrial engineering problems.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in writing the paper.

Acknowledgments

This study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2023/R/1445).

Conflict of interest

The authors declare no conflict of interest.

References

1. J. B. Hu, J. W. Huang, Z. Y. Li, J. X. Wang, T. He, A receiver-driven transport protocol with high link utilization using anti-ecn marking in data center networks, *IEEE Trans. Netw. Serv. Manag.*, **20** (2022), 1898–1912. https://doi.org/10.1109/TNSM.2022.3218343
2. J. Wang, Y. Liu, S. Y. Rao, X. Y. Zhou, J. B. Hu, A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks, *Ad Hoc Netw.*, **150** (2023), 103284. https://doi.org/10.1016/j.adhoc.2023.103284
3. 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, *Simul. Model. Pract. Theory*, **111** (2021), 102353. https://doi.org/10.1016/j.simpat.2021.102353

4. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Gener. Comp. Syst.*, **25** (2009), 599–616. https://doi.org/10.1016/j.future.2008.12.001

5. B. M. Nguyen, H. T. T. Binh, T. T. Anh, D. B. Son, Evolutionary algorithms to optimize task scheduling problem for the Iot based bag-of-tasks application in cloud-fog computing environment, *Appl. Sci.*, **9** (2019), 1730. https://doi.org/10.3390/app9091730

6. M. A. Elaziz, I. Attiya, L. Abualigah, M. Iqbal, A. Ali, A. Al-Fuqaha, et al., Hybrid enhanced optimization-based intelligent task scheduling for sustainable edge computing, *IEEE Trans. Consum. Electr.*, 2023, 1. https://doi.org/10.1109/TCE.2023.3321783

7. I. Attiya, M. A. Elaziz, L. Abualigah, T. N. Nguyen, A. A. A. El-Latif, An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud, *IEEE Trans. Ind. Inform.*, **18** (2022), 6264–6272. https://doi.org/10.1109/TII.2022.3148288

8. M. R. Raju, S. K. Mothku, Delay and energy aware task scheduling mechanism for fog-enabled iot applications: A reinforcement learning approach, *Comput. Netw.*, **224** (2023), 109603. https://doi.org/10.1016/j.comnet.2023.109603

9. M. A. A. Al-qaness, A. A. Ewees, H. Fan, L. Abualigah, M. A. Elaziz, Boosted ANFIS model using augmented marine predator algorithm with mutation operators for wind power forecasting, *Appl. Energy*, **314** (2022), 118851. https://doi.org/10.1016/j.apenergy.2022.118851

10. T. Li, S. Fong, R. C. Millham, J. Fiaidhi, S. Mohammed, Fast incremental learning with swarm decision table and stochastic feature selection in an iot extreme automation environment, *IT Prof.*, **21** (2019), 14–26. https://doi.org/10.1109/MITP.2019.2900016

11. M. A. A. Al-qaness, A. M. Helmi, A. Dahou, M. A. Elaziz, The applications of metaheuristics for human activity recognition and fall detection using wearable sensors: A comprehensive analysis, *Biosensors*, **12** (2022), 821. https://doi.org/10.3390/bios12100821

12. M. A. Elaziz, M. A. A. Al-qaness, A. Dahou, R. A. Ibrahim, A. A. A. El-Latif, Intrusion detection approach for cloud and iot environments using deep learning and capuchin search algorithm, *Adv. Eng. Softw.*, **176** (2023),103402. https://doi.org/10.1016/j.advengsoft.2022.103402

13. S. N. Ghorpade, M. Zennaro, B. S. Chaudhari, R. A. Saeed, H. Alhumyani, S. Abdel-Khalek, Enhanced differential crossover and quantum particle swarm optimization for iot applications, *IEEE Access*, **9** (2021), 93831–93846. https://doi.org/10.1109/ACCESS.2021.3093113

14. G. Agarwal, S. Gupta, R. Ahuja, A. K. Rai, Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog-cloud computing, *Knowl. Based Syst.*, **272** (2023), 110563. https://doi.org/10.1016/j.knosys.2023.110563

15. W. B. Sun, J. Xie, X. Yang, L. Wang, W. X. Meng, Efficient computation offloading and resource allocation scheme for opportunistic access fog-cloud computing networks, *IEEE Trans. Cogn. Commun. Netw.*, **9** (2023), 521–533. https://doi.org/10.1109/TCCN.2023.3234290

16. B. Jana, M. Chakraborty, T.a Mandal, A task scheduling technique based on particle swarm optimization algorithm in cloud environment, In: *Soft computing: Theories and applications*, Singapore: Springer, **742** (2019), 525–536. https://doi.org/10.1007/978-981-13-0589-4_49

17. A. Pradhan, S. K. Bisoy, A. Das, A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment, *J. King Saud Univ. Comput. Inform. Sci.*, **34** (2022), 4888–4901. https://doi.org/10.1016/j.jksuci.2021.01.003

18. F. Al-Turjman, M. Z. Hasan, H. Al-Rizzo, Task scheduling in cloud-based survivability applications using swarm optimization in Iot, *Trans. Emerg. Telecommun. Technol.*, **30** (2019), e3539. http://doi.org/10.1002/ett.3539

19. A. M. S. Kumar, M. Venkatesan, Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment, *Wireless Pers. Commun.*, **107** (2019), 1835–1848. http://doi.org/10.1007/s11277-019-06360-8

20. M. A. Elaziz, I. Attiya, An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing, *Artif. Intell. Rev.*, **54** (2021), 3599–3637. http://doi.org/10.1007/s10462-020-09933-3

21. A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm, *J. Netw. Syst. Manag.*, **29** (2021), 31. http://doi.org/10.1007/s10922-021-09599-4

22. M. A. Elaziz, L. Abualigah, R. A. Ibrahim, I. Attiya, Iot workflow scheduling using intelligent arithmetic optimization algorithm in fog computing, *Comput. Intel. Neurosc.*, **2021** (2021), 9114113. https://doi.org/10.1155/2021/9114113

23. N. Arivazhagan, K. Somasundaram, D. V. Babu, M. G. Nayagam, R. M. Bommi, G. B. Mohammad, et al., Cloud-internet of health things (IOHT) task scheduling using hybrid moth flame optimization with deep neural network algorithm for e healthcare systems, *Sci. Program.*, **2022** (2022), 4100352. https://doi.org/10.1155/2022/4100352

24. B. B. Naik, D. Singh, A. B. Samaddar, Multi-objective virtual machine selection in cloud data centers using optimized scheduling, *Wireless Pers. Commun.*, **116** (2021), 2501–2524. https://doi.org/10.1007/s11277-020-07807-z

25. N. Arora, R. K. Banyal, Workflow scheduling using particle swarm optimization and gray wolf optimization algorithm in cloud computing, *Concurr. Comput. Pract. Exper.*, **33** (2021), e6281. https://doi.org/10.1002/cpe.6281

26. S. Goyal, S. Bhushan, Y. Kumar, A. ul H. S. Rana, M. R. Bhutta, M. F. Ijaz, et al., An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm, *Sensors*, **21** (2021), 1583. https://doi.org/10.3390/s21051583

27. D. Alsadie, TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers, *IEEE Access*, **9** (2021), 37707–37725. https://doi.org/10.1109/ACCESS.2021.3063723

28. W. Zhao, L. Wang, S. Mirjalili, Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications, *Comput. Methods Appl. Mech. Eng.*, **388** (2022), 114194. https://doi.org/10.1016/j.cma.2021.114194

29. N. Chopra, M. M. Ansari, Golden jackal optimization: A novel nature-inspired optimizer for engineering applications, *Expert Syst. Appl.*, **198** (2022), 116924. https://doi.org/10.1016/j.eswa.2022.116924

30. I. Attiya, L. Abualigah, D. Elsadek, S. A. Chelloug, M. A. Elaziz, An intelligent chimp optimizer for scheduling of Iot application tasks in fog computing, *Mathematics*, **10** (2022), 1100. https://doi.org/10.3390/math10071100

31. I. Attiya, X. Zhang, X. Yang, TCSA: A dynamic job scheduling algorithm for computational grids, In: *2016 First IEEE international conference on computer communication and the internet (ICCCI)*, 2016, 408–412. https://doi.org/10.1109/CCI.2016.7778954

32. Mahdi Azizi, Atomic orbital search: A novel metaheuristic algorithm, *Appl. Math. Modell.*, **93** (2021), 657–683. https://doi.org/10.1016/j.apm.2020.12.021

33. A. Faramarzi, M. Heidarinejad, S. Mirjalili, A. H. Gandomi, Marine predators algorithm: A nature-inspired metaheuristic, *Expert Syst. Appl.*, **152** (2020), 113377. https://doi.org/10.1016/j.eswa.2020.113377

34. L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz, A. H. Gandomi, The arithmetic optimization algorithm, *Comput. Methods Appl. Mech. Engrg.*, **376** (2021), 113609. https://doi.org/10.1016/j.cma.2020.113609

35. I. Attiya, L. Abualigah, S. Alshathri, D. Elsadek, M. A. Elaziz, Dynamic jellyfish search algorithm based on simulated annealing and disruption operators for global optimization with applications to cloud task scheduling, *Mathematics*, **10** (2022), 1894. https://doi.org/10.3390/math10111894

36. M. S. Braik, Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems, *Expert Syst. Appl.*, **174** (2021), 114685. https://doi.org/10.1016/j.eswa.2021.114685