processes

MDPI

*Article*

# An Optimal Scheduling Method in IoT-Fog-Cloud Network Using Combination of Aquila Optimizer and African Vultures Optimization

**Qing Liu [1], Houman Kosarirad [2], Sajad Meisami [3], Khalid A. Alnowibet [4] and Azadeh Noori Hoshyar [5,*]**

[1] School of Artificial Intelligence, Chongqing Creation Vocational College, Yongchuan, Chongqing 402160, China
[2] Durham School of Architectural Engineering and Construction, University of Nebraska-Lincoln, 122 NH, Lincoln, NE 68588, USA
[3] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA
[4] Statistics and Operations Research Department, College of Science, King Saud University, Riyadh 11451, Saudi Arabia
[5] Institute of Innovation, Science and Sustainability, Federation University Australia, Brisbane, QLD 4000, Australia
* Correspondence: a.noorihoshyar@federation.edu.au

**Abstract:** Today, fog and cloud computing environments can be used to further develop the Internet of Things (IoT). In such environments, task scheduling is very efficient for executing user requests, and the optimal scheduling of IoT task requests increases the productivity of the IoT-fog-cloud system. In this paper, a hybrid meta-heuristic (MH) algorithm is developed to schedule the IoT requests in IoT-fog-cloud networks using the Aquila Optimizer (AO) and African Vultures Optimization Algorithm (AVOA) called AO_AVOA. In AO_AVOA, the exploration phase of AVOA is improved by using AO operators to obtain the best solution during the process of finding the optimal scheduling solution. A comparison between AO_AVOA and methods of AVOA, AO, Firefly Algorithm (FA), particle swarm optimization (PSO), and Harris Hawks Optimization (HHO) according to performance metrics such as makespan and throughput shows the high ability of AO_AVOA to solve the scheduling problem in IoT-fog-cloud networks.

**Keywords:** Aquila Optimizer; African Vultures Optimization Algorithm; task scheduling; fog computing; cloud computing; Internet of Things



**Citation:** Liu, Q.; Kosarirad, H.; Meisami, S.; Alnowibet, K.A.; Hoshyar, A.N. An Optimal Scheduling Method in IoT-Fog-Cloud Network Using Combination of Aquila Optimizer and African Vultures Optimization. *Processes* **2023**, *11*, 1162. https://doi.org/10.3390/pr11041162

Academic Editor: Olympia Roeva

Received: 19 February 2023
Revised: 25 March 2023
Accepted: 4 April 2023
Published: 10 April 2023



## 1. Introduction

The Internet of Things (IoT) occupies a special place in today's world as well as in the world of information technology [1]. IoT devices including laptops, tablets, and smartphones have their functions expanded using other smart devices [2]. IoT devices generate a significant amount of information through their respective sensors. This collected information requires different resources for processing and storage so that final decisions can be made based on the result of the processing to achieve the goals and desires of the user [3]. Cloud computing is an efficient computing environment to meet the needs of different users around the world, and the information of IoT devices can be processed and stored in the cloud. The cloud environment provides public access to resources for users [4]. Fog computing is a developed environment of cloud computing that has significantly fewer computing resources than the cloud environment, is geographically closer to users, and has high speed [5]. Therefore, fog computing can provide less traffic and less delay in the IoT-fog-cloud network than cloud computing [6]. The three-layer architecture of the IoT-fog-cloud network is shown in Figure 1.

The highest layer includes servers in the cloud environment that can store and process significant amounts of user data. The middle layer consists of fog nodes and the edge of

the network and includes mini-servers and smart gateways. The lower layer, which is the edge of the IoT-fog-cloud network, includes IoT devices and end systems such as laptops, smartphones, cars, personal computer machines, sensors, etc. [6–8]. Task scheduling is a practical resource management approach in cloud and fog computing to allocate a set of tasks requested by IoT to the most appropriate resources in the cloud and fog [9].

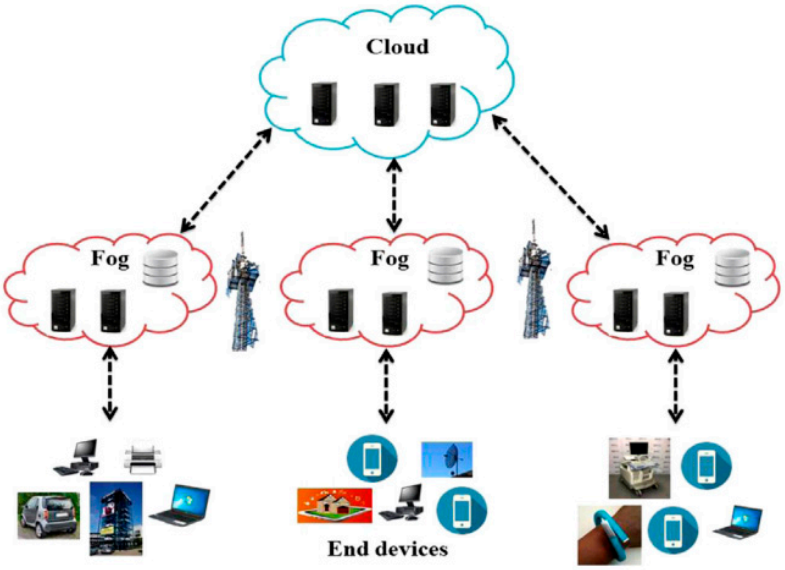



**Figure 1.** The IoT-fog-cloud network structure.

In [10], a hidden Markov model (HMM) is used to predict the availability of fog sources. Additionally, the DO-HHO hybrid algorithm (discrete opposition-based Harris hawk optimization (HHO)) has been used to schedule IoT tasks. In [11], a scalable algorithm is proposed to schedule time-sensitive tasks. In [12], the delay of the processing and communication tasks of IoT devices that are distributed in a cloud and fog network is investigated based on the two criteria of delay and consumed energy. In [13], a blockchain-based protocol has been proposed for the development of e-health programs, in which the Proof of Work (PoW) method is not used. In their method, the cost of the network, i.e., bandwidth and processor usage, is reduced.

The task scheduling in fog and cloud computing environments is considered as an NP-hard problem [14]. For this reason, different task scheduling methods that have been adopted using artificial intelligence algorithms have been proposed in the relevant work section. Recently, meta-heuristic (MH) algorithms such as particle swarm optimization (PSO) [15], Whale Optimization Algorithm (WOA) [16], Moth Flame Optimization (MFO) [17], Artificial Bee Colony (ABC) [18], and Harris hawks optimizer (HHO) [19] have been used to address said problems.

Optimization algorithms are very useful for solving problems in the fields of edge, fog, cloud, and IoT [20–26]. In this paper, the combination of two algorithms, Aquila Optimizer and African Vultures, is used to solve the task scheduling problem. That is, the combination of the ExploRation Phase (ERF) of Aquila with the ExploiTation Phase (ETP) of African Vultures. The innovation of this paper relies on the combination of two algorithms, Aquila Optimizer and African Vultures. The motivation of this paper is the combination of these two algorithms. We have also investigated two algorithms, Aquila Optimizer (AO) and African Vultures Optimization (AVO), separately to solve the task scheduling problem. In the comparison section, the AO_AVOA algorithm is compared with AO, AVOA, Firefly Algorithm (FA), particle swarm optimization (PSO), and Harris hawks optimization (HHO) algorithms and it is superior to the corresponding algorithms. The purpose of using this combination is to solve the task scheduling problem in the IoT-fog-cloud network and reduce the task makespan time, so that tasks are completed in the shortest possible time,

and so that the final solution of the problem is obtained by minimizing the fitness function. In this paper, makespan time is used as the fitness function. Experiments are performed on two datasets. The proposed method is compared with some other algorithms, and the proposed method performs better than the compared algorithms.

The structure of this paper is as follows: Section 2 introduces the related works, Section 3 introduces the system's model and problem formulation, Section 4 includes prerequisites, Section 5 introduces the proposed method, Section 6 includes evaluations and experimental results, and Section 7 includes the conclusion.

## 2. Related Works

In this section, a number of existing works in the field of task/workflow scheduling are briefly described. Nguyen et al. [27] have proposed an evolutionary algorithm to achieve an optimal scheduling in order to create an optimal balance between task execution time and task arrangements in the IoT-fog-cloud network. Boviri et al. [28] have proposed a method in a multiprocessor environment using the improved Ant Colony Algorithm (IACO) in order to optimize the sequence of tasks.

Tong et al. [29] have proposed a hybrid algorithm using a neural network and Q-learning algorithm, in order to address the scheduling of IoT requests in the cloud environment. Yang et al. [30] proposed a multi-objective evolutionary algorithm to solve the task scheduling problem in the fog environment in order to reduce the time and optimally allocate resources to the relevant tasks. Mtshali et al. [31] proposed a method in the fog environment using the visualization method to build a suitable algorithm in order to reduce energy consumption and reduce delay.

Qobaei Qobaei-Arani et al. [32] proposed a method in the fog environment using the MFO algorithm in order to increase the quality of services (QoSs) and reduce the performance time of the total tasks. Abualigah et al. [33] proposed a method in the cloud environment using a gray wolf optimizer in order to find the time cost and optimal allocation of resources to the relevant tasks. Zeng et al. [34] proposed a method in the fog environment to support embedded systems in order to reduce the execution time of tasks to keep users active. The authors of [35] proposed a task scheduling method using the Genetic Algorithm (GA) in order to allocate resources to tasks, taking into account customer needs and resource constraints. The authors' objective was to reduce preparation time and increase customer satisfaction.

Rjoub et al. [36] proposed a new solution based on using four deep reinforcement learning (DRL)-based methods to optimize the process of task scheduling in cloud. The authors' objective was to reduce execution time and maximize resource utilization. Their proposed DRL methods are Deep Q Networks (DQN), reinforcement learning (RL), recurrent neural network long short-term memory (RNNLSTM), and DRL combined with LSTM. Jacob [37] proposed an algorithm for minimizing makespan time using the Bat Algorithm (BA). Raghavan et al. [38] proposed a method using the BA in cloud computing, aiming to reduce the whole cost.

In [39], DRL is used for task scheduling in heterogeneous computing, which implies reducing the task completion time. In [40], in order to reduce energy consumption in fog-cloud, a convolutional neural networks (CNNs) algorithm is used for task scheduling to minimize the cost.

In the IoT-fog network, a task scheduling method was proposed to allocate resources to IoT tasks, which optimally selects the best resources to execute the tasks [41,42]. Ranumayee et al. [43] used the evolutionary learning method to optimize energy, makespan, and cost and schedule tasks in the IoT-fog-cloud network. Mokni et al. [44] have used the multi-objective fuzzy method to offload workflow in the fog-cloud network. Ranumayee et al. [45] have used the WOA in order to allocate optimal resources and schedule efficient tasks in the IoT-fog-cloud network. Panda et al. [46] presented a task scheduling method in the cloud environment using pair. Shukla et al. [47] proposed a Fuzzy-AHP-TOPSIS-Based Task Offloading method for scheduling workflows in the fog-cloud system.

Stewart et al. [48] proposed a bi-objective integer linear programming method in the cloud in order to optimize energy consumption and makespan time.

A task scheduling method based on software-defined networking (SDN) is proposed in the cloud computing environment in order to schedule the tasks of IoT devices using the combination of WOA and Aquila Optimizer (AO), which minimizes the MST [49]. A workflow scheduling method has been proposed in the cloud computing environment using a combination of MFO and the Salp Swarm Algorithm (SSA), which are selected by considering several objectives of energy consumption, makespan time, and throughput time of the most optimal virtual machines (VMs) [50]. Task offloading methods in fog and cloud environments have also been proposed by researchers using Dynamic Service Caching [51] and D2D-Associated Mobile Edge Computing for IoT tasks [52].

The algorithms and methods mentioned above, which have been used to optimize the task/workflow scheduling problem in fog and cloud environments, could not sufficiently select the most optimal resources for the execution of tasks. The WOA, MFO, GA, and ACO algorithms suffer from the weakness they have in each of the exploration and exploitation phases and cannot obtain the most optimal solution.

The African Vultures Optimization Algorithm (AVOA) [53] is a meta-heuristic (MH) algorithm inspired by nature. In AVOA, the ability to ERP is weaker than the ability to ETP it during the process of searching for solutions. This problem reduces the quality of the final output in AVOA. To solve this problem, in this paper, the high ERP ability of AO [54] is combined with the high ETP ability of AVOA. Therefore, in this paper, an intelligent task scheduling method for IoT requests, based on the combination of AO and AVOA in the IoT-fog-cloud computing environment, is proposed, which is called AO_AVOA. Therefore, the advantages of the power of AO ERP and AVOA ETP are combined to obtain the best scheduling method that is stronger than AO and AVOA, which is used in the ETP of one of the AO or AVOA methods. The main contribution of this paper is to present an intelligent algorithm based on the integration of AO and AVOA to improve IoT services in an IoT-fog-cloud computing environment using makespan time minimization.

## 3. System Model and Problem Formulation

This section explains the system model and formulation of the task scheduling problem in this paper.

### 3.1. System Model

In this paper, it is assumed that the used architecture consists of three layers of fog, cloud, and IoT devices (Figure 1). IoT devices may have requests to send as a set of tasks to higher layers of fog and cloud for processing and storage. Tasks that are time-sensitive are stored and processed in the nodes close to the devices, i.e., the fog environment, to reduce the delay [55].

Additionally, compute-intensive tasks are sent to cloud servers because cloud servers provide higher computing and storage capabilities than fog nodes for the corresponding tasks. According to the characteristics of the tasks of IoT devices as well as the capabilities of the resources available in the fog and cloud, the task scheduler schedules the tasks by mapping the corresponding tasks on the corresponding computing nodes.

### 3.2. Problem Formulation

It is assumed that the n1 independent task is T = {T1, T2, T3, . . . , Tn1} from the side of the IoT devices to be executed, which must be sent to the corresponding resources to be processed in the corresponding fog-cloud computing environment. Each task has characteristics such as task length (in million instructions), memory requirements, deadline, and size of input and output files.

Additionally, it is assumed that the fog-cloud system consists of n2 computing nodes as Node = {node1, node2, node3, . . . , noden2}, which includes fog and cloud nodes. Each node includes characteristics such as CPU processing rate (in terms of millions of

instructions per second (MIPS)), memory size, storage capacity, and network bandwidth. Therefore, for n1 tasks and n2 computing nodes, the expected computation time (ECT) for task requests on nodes is represented by using the ECT matrix, which is of size n1 × n2. The task scheduler uses the ECT matrix to make task scheduling decisions in the fog-cloud environment. The $ect_{i,j}$ indicates the expected execution time of the $i$-th task on the $j$-th computing node $ect_{i,j}$ as

$$ect_{i,j} = \frac{Task\_length_i}{node.Pow_j} \tag{1}$$

where $Task\_length_i$ is the length of the $i$-th task, and $node.Pow_j$ is the processing speed of the $j$-th node. In the task scheduling problem, the main objective is to find the ideal schedule in the fog-cloud system that reduces the task completion time or makespan time. In this case, it can be guaranteed that there will be no task that requires a significant amount of time to execute and complete it [6]. The amount of makespan time is obtained from the following equation:

$$Makespan\ time = \max_{j \in 1, 2, \ldots, n2} \sum_{i=1}^{n1} ect_{i,j} \tag{2}$$

## 4. Prerequisites

In this section, the two AO and AVOA algorithms, which are the main prerequisites for the design of the combined AO_AVOA algorithm, are briefly described.

### 4.1. Aquila Optimizer

The Aquila Optimizer (AO) algorithm [54] is a swarm intelligence algorithm in which Aquila provides four different hunting behaviors for different prey. AO, like other MH algorithms, performs the optimization process using both ERP and ETP and finally converges to the final optimal solution. A brief description of the mathematical model of the AO algorithm follows.

Step 1: Explore the search space extensively. The mathematical model of this behavior is as follows:

$$X(t+1) = X_{best}(t) \times (1 - \frac{t}{T}) + (X_m(t) - X_{best}(t) \times rand) \tag{3}$$

$$X_m(t) = \frac{1}{N} \sum_{k=1}^{N} X_i(t) \tag{4}$$

where $X_m(t)$ represents the average location of all agents in the current iteration, $X_{best}(t)$ represents the best position achieved so far, $N$ is the population size, $rand$ is a random number between 0 and 1, and $t$ and $T$ are the current iteration and the maximum iteration number, respectively.

Step 2: Narrow exploration to short attack the prey. The equation for updating positions is as follows:

$$X(t+1) = X_{best}(t) \times LF(D) + X_R(t) + (y - x) \times rand \tag{5}$$

where $D$ is the dimension size, $X_R(t)$ represents the random position of Aquila, and $LF(D)$ represents the Levy flight function, which is defined as follows:

$$LF(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{6}$$

The parameter $\sigma$ is obtained according to the following equation:

$$\sigma = \left[ \frac{r(1+\beta) \times sin(\frac{\pi\beta}{2})}{r\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right] \tag{7}$$

where $s$, $u$ and $v$ are random numbers between 0 and 1, $\beta$ are fixed values equal to 0.01 and 1.5, respectively, $y$ and $x$ are used to represent the spiral shape of the algorithm in searching the search space, which are defined as follows:

$$\begin{cases} x = r \times \sin(\theta) \\ y = r \times \cos(\theta) \\ r = r_1 + 0.00565 \times D_1 \\ \theta = -w \times D_1 + \frac{3 \times \pi}{2} \end{cases} \quad (8)$$

where, $D_1$ is the integers from 1 to $D$ and is equal to 0.005, and $r_1$ means the number of search cycles between 1 and 20 [54].

Stage 3: Extensive exploitation: Aquila vertical landing to attack the prey. This behavior is defined as follows:

$$X(t+1) = (X_{best}(t) - X_m(t)) \times \alpha - rand + ((UB - LB) \times rand + LB) \times \delta \quad (9)$$

where $UB$ and $LB$ are the upper and lower bounds of the desired problem, respectively, and $\alpha$ and $\delta$ are the ETP adjustment parameters that are equal to 0.1 [52].

Stage 4: Limited exploitation: moving and catching prey. The mathematical model of this behavior is as follows:

$$\begin{cases} X(t+1) = QF \times X_{best}(t) - (G_1 \times X(t)) \times rand - G_2 \times LF(D) + rand \times G_1 \\ QF(t) = t^{\left(\frac{2 \times rand - 1}{(1-T)^2}\right)} \\ G_1 = 2 \times rand - 1 \\ G_2 = 2 \times \left(1 - \frac{t}{T}\right) \end{cases} \quad (10)$$

where $X(t)$ represents the current location and $QF(t)$ is the value of the used quality function to creation balance the search strategy. $G_1$ shows the movement parameter of the Aquila when tracking the prey, which is a random number between [–1, 1]. $G_2$ shows the flight slope when chasing prey, which decreases linearly from 2 to 0. AO has good power in the ERP, but it is a little weak in the ETP, and for this reason, it cannot find local optimal solutions with high power and accurately. On the other hand, AVOA is the opposite of AO and is slightly weaker than AO in the ERP.

However, AVOA has very good power in the ETP and can find local optimal solutions with high accuracy compared to AO. For this reason, in this paper, the ERP of AO is combined with the ETP of AVOA, and in this case, the combined AO-AVOA algorithm will be obtained, which has very good power in both ERP and ETP, and in terms of finding final optimal solutions it is better than AO and AVOA. Therefore, AO-AVOA can choose the best and most suitable virtual machines for executing tasks on the user side.

### 4.2. African Vultures Optimization Algorithm

Like AO, the African Vulture optimization algorithm [53] is also formulated in four separate steps.

First phase: determining the best agent in each group

After forming the initial population, the fitness value for all answers is calculated, the best answer of the first Vulture group and the best answer of the second Vulture group are determined, and other answers are selected using Equation (11).

$$R(i) = \begin{cases} BestVulture_1 \ \ if \ p_i = L_1 \\ BestVulture_2 \ \ if \ p_i = L_2 \end{cases} \quad (11)$$

where $L_1$ and $L_2$ are parameters that must be initialized with values between 0 and 1 before the search operation, and the sum of both parameters is equal to 1. The probability of

choosing the best answer using the roulette wheel method for each group is obtained using Equation (12).

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \tag{12}$$

The second phase: the level of hunger of Vultures

For mathematical modeling of this behavior, Equation (14) is used. It used to transition from the ERP to the ETP, which is inspired by the speed of satiety or hunger of Vultures.

$$T = h \times (sin^w (\frac{\pi}{2} \times \frac{iter_i}{max\_iter}) + \cos(\frac{\pi}{2} \times \frac{iter_i}{max\_iter}) - 1) \tag{13}$$

$$F = (2 \times rand_1 + 1) \times z \times (1 - \frac{iter_i}{max\_iter}) + t \tag{14}$$

In Equations (13) and (14), $F$ indicates that the agents are satiated, $iter_i$ is the current iteration number, $max\_iter$ indicates the total number of iterations, and $z$ is a random number between $-1$ and 1 that changes each iteration, while $h$ is a random number between $-2$ and 2. $rand_1$ has a random value between 0 and 1, $w$ is a parameter with a constant number that is set before the optimization operation. When the value of $|F| > 1$, the agents look for food in different areas and AVOA enters the ERP. If the value of $|F| < 1$, AVOA enters the ETP and agents forage in the neighborhood of solutions.

The third phase: exploration

In this step, the parameter $P_1$, which has a value between 0 and 1, is used to choose two different strategies. This parameter must be valued before the search operation. A random number between 0 and 1 is generated to select each of the strategies in the ERP $rand_{p_1}$. If this number is $\geq P_1$, Equation (16) is used. However, if $rand_{p_1} < P_1$, Equation (17) is used, as follows:

$$P1(i+1) = \begin{cases} Equation\ (16)\ \ if\ p_1 \geq rand_{p_1} \\ Equation\ (17)\ \ if\ p_1 \geq rand_{p_1} \end{cases} \tag{15}$$

$$P1(i+1) = R(i) - D(i) \times F \tag{16}$$

$$D(i) = |X \times R(i) - P1(i)| \tag{17}$$

where $P1(i+1)$ is the agent's location vector in the next iteration, and $F$ is the satiation rate of the agent. In Equation (17), $R(i)$ is one of the best agents. Furthermore, $X$ is the agents randomly moving to protect food from other agents and is given by $X = 2 \times rand$ where $rand$ is a random number between 0 and 1. $P1(i)$ is the current location vector of the Vulture.

$$P1(i+1) = R(i) - F + rand_2 \times (UB - LB) \times rand_3 + LB) \tag{18}$$

In Equation (18), $rand_2$ has a random value between 0 and 1 and $rand_3$ takes a number close to 1.

The fourth phase: exploitation

In this step, if $|F| < 1$, AVOA enters the ETP, which also has two phases in which two different strategies are used in each phase. The selection degree of each strategy in each internal phase is determined by two parameters, namely $P_2$ and $P_3$. The $P_2$ parameter is used to select the strategies in the first stage and the $P_3$ parameter is used to select the strategies in the second stage. Both parameters must be set to 0 and 1 before performing the search operation.

Exploitation (first stage):

AVOA enters the first stage in the ETP when $|F|$ it is between 1 and 0.5. In the first stage, two different strategies of turn flight and siege combat are performed. $P_2$ is used to specify the choice of each strategy, which must be evaluated before performing the search

operation, and the value must be between 0 and 1. At the beginning of this step, $rand_{p_2}$, which is a random number between 0 and 1, is generated. If this number is $\geq P_2$, the Siegefight strategy is executed slowly. However, if this random number is $< P_2$, the rotary flight strategy is performed. This method is shown in Equation (19).

$$P1(i+1) = \begin{cases} Equation(21) \ \ if \ p_2 \geq rand_{p_2} \\ Equation(24) \ \ if \ p_2 \geq rand_{p_2} \end{cases} \tag{19}$$

Competition for food:

When $|F| \geq 0.5$, the agents are relatively full and have enough energy. When many agents congregate on a food source, it can cause intense conflicts over food. In such cases, agents with high physical strength prefer not to share food with other agents. The weaker agents try to exhaust and take food from healthy agents by gathering around healthy agents. Equations (20) and (21) are used to model this step.

$$P1(i+1) = D(i) \times (F + rand_4) - d(t) \tag{20}$$

$$d(t) = R(i) - P1(i) \tag{21}$$

$D(i)$ is calculated using Equation (17) and $F$ is the satiety of agents which is calculated using Equation (14). $rand_4$ is a random number between 0 and 1, which is used to increase the randomness factor. In Equation (21), $R(i)$ is one of the best agents of the second category, selected using Equation (11). In the current iteration, $P1(i)$ is the agent's current location vector, by which the distance between the agent and one of the best agents in the second category is obtained.

Circular flight of agents:

Agents often perform a circling flight. A spiral model has been used for mathematical modeling of rotary flight. An equation (spiral) is created between all agents and one of the top two agents. The rotational flight is explained by Equations (22) and (23).

$$S_1 = R(i) \times (\frac{rand_5 \times P1(i)}{2\pi}) \times \cos(P1(i)), \quad S_2 = R(i) \times (\frac{rand_6 \times P1(i)}{2\pi}) \times \sin(P1(i)) \tag{22}$$

$$P1(i+1) = R(i) - (S_1 + S_2) \tag{23}$$

In Equations (22) and (23), $R(i)$ represents the location vector of one of the two best agents in the current iteration, which is obtained by Equation (11). *Cos* and *sin* represent the sine and cosine functions, respectively, $rand_5$ and $rand_6$ are random numbers between 0 and 1. $S_1$ and $S_2$ are obtained using Equation (22). Finally, using Equation (23), the location of the agents is updated.

Exploitation (second stage):

In the second stage of ETP, the movements of two agents gather several types of agents over the food source, and encirclement and aggressive fighting are carried out to find food. If $|F| < 0.5$, this step of the algorithm is executed. At the beginning of this stage, $rand_{p_3}$ is generated, which is a random number between 0 and 1. If $rand_{p_3} \geq P_3$, the strategy is to accumulate several types of agents on the food source. Otherwise, if the generated value is $< P_3$, the siege–fight–offensive strategy is executed. This method is shown in Equation (24).

$$P1(i+1) = \begin{cases} Equation(27) \ if \ p_3 \geq rand_{p_3} \\ Equation \ (28) \ if \ p_3 \geq rand_{p_3} \end{cases} \tag{24}$$

Gathering of several types of agents on the food source:

The movement of all agents toward the food source is monitored. Occasionally, agents become hungry and there is a significant amount of competition over food, which may

crowd several agent species into a food source. Equation (25) is used to formulate this movement of the agents.

$$\begin{aligned} A_1 &= BestVulture_1(i) - \frac{BestVulture_1(i) \times p1(i)}{BestVulture_1(i) - p1(i)^2} \times F, \\ A_2 &= BestVulture_2(i) - \frac{BestVulture_2(i) \times p1(i)}{BestVulture_2(i) - p1(i)^2} \times F \end{aligned} \tag{25}$$

In Equation (25), $BestVulture_1(i)$ is the best agent from the first category in the current iteration and $BestVulture_2(i)$ is the best agent from the second category in the current iteration, while $F$ is the satiety of the agent, which is calculated using Equation (14) and $P1(i)$ is the current vector location of the agent.

$$P1(i+1) = \frac{A_1 + A_2}{2} \tag{26}$$

Finally, the aggregation of all the agents is conducted using Equation (26), where $A_1$ and $A_2$ are obtained using Equation (25) and $P1(i + 1)$ is the agent location vector in the next iteration.

Aggressive competition for food:

When $|F| < 5$, head agents become hungry and weak and do not have enough energy to fight other agents. On the other hand, other agents also become aggressive in search of food. They move in different directions toward the agent head. Equation (27) is used to model this movement.

$$P1(i+1) = R(i) - |d(t)| \times F \times Levy(d) \tag{27}$$

In Equation (27), $d(t)$ represents the distance of the agent to one of the best agents of the second category, which is calculated using Equation (21). Levy flight (LF) patterns have been used to increase the effectiveness of AVOA in Equation (27), and LF has been identified and used in many MH algorithms. $LF$ is calculated using Equation (28).

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{p}}}, \quad \sigma = \left(\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma(1+\beta 2) \times \beta \times 2\left(\frac{\beta-1}{2}\right)}\right)^{\frac{1}{p}} \tag{28}$$

In Equation (28), $d$ represents the dimensions of the problem, $u$ and $v$ are a random number between 0 and 1, and $\beta$ is a fixed number with a default value of 1.5.

## 5. Proposed Task Scheduling with AO_AVOA

The task scheduling in fog and cloud environments using the combination of AO and AVOA algorithms is described in this section. In the task scheduling algorithm using AO_AVOA, the solutions are competitively updated in the ETP using AO or AVOA operators. However, in the ERP, only AO operators are used due to the fact that the AO ERP is stronger than the AVOA. The flowchart of the proposed AO_AVOA algorithm for scheduling in the fog-cloud network is shown in Figure 2.

In the initialization phase, the AO_AVOA algorithm starts by setting the $N$ agents and converting them into integers. This initial population is generated randomly, and distributed appropriately for task scheduling in fog-cloud computing environments. This behavior is modeled according to the following:

In the above equation, $Lb = 1$ and $Ub$ is equal to the number of resources or VMs, $fix(.)$ is the same function used to convert real numbers to integers in AO_AVOA. Then, the fitness function is calculated to evaluate the quality of each solution using the objective function obtained from Equation (2). Considering that the task scheduling is assumed to be a minimization problem in this paper, the solution with the smallest makespan value is determined as the best solution, and in the ERP and ETP, $X$ solutions are updated according to it. In the ERP, the solutions are updated using AO operators according to

Equation (5). However, in the ETP, the solutions are updated using competition between AO and AVOA operators.

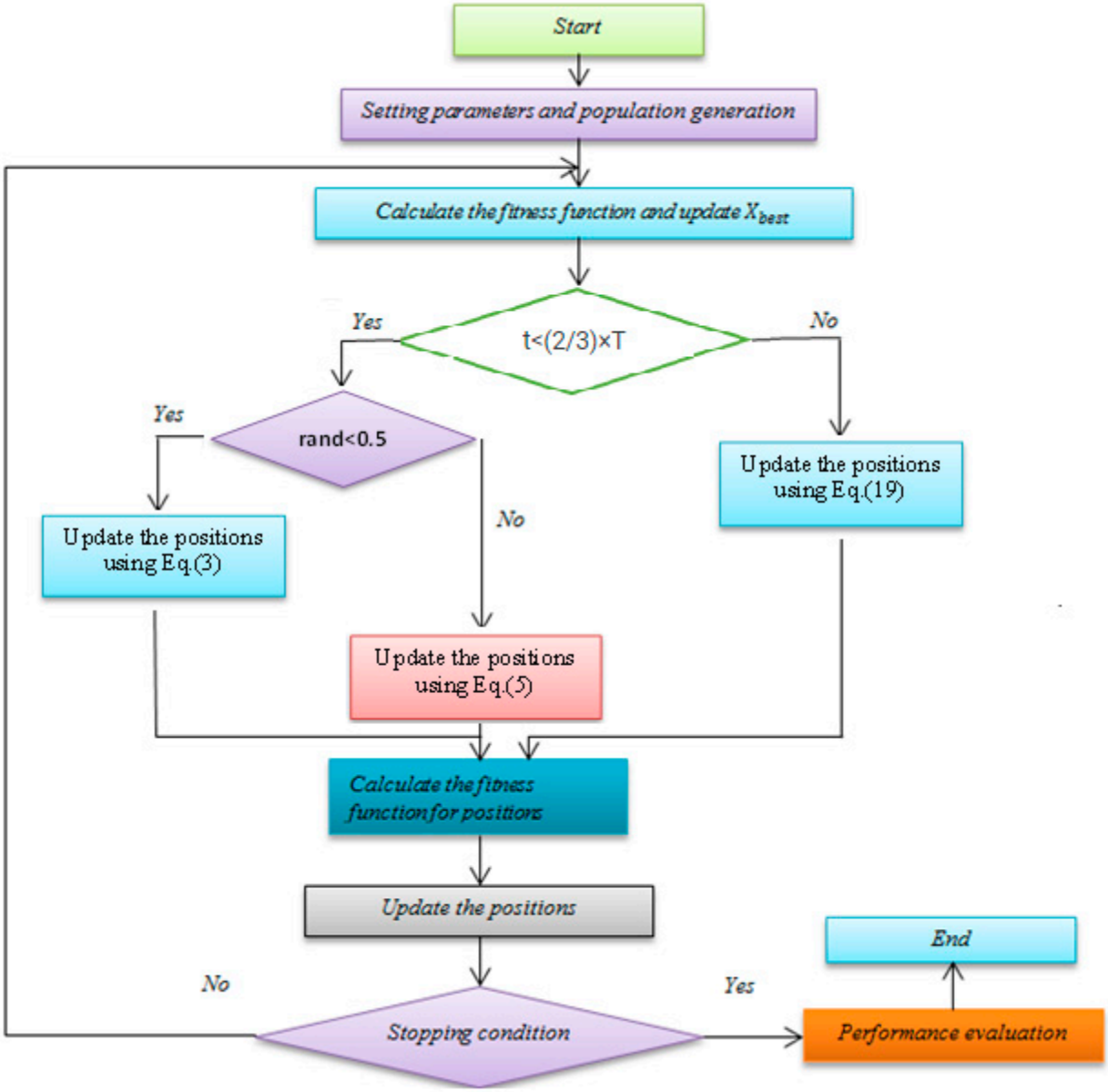



**Figure 2.** Diagram of the proposed AO_AVOA.

In the ETP, the selection of any of the AO or AVOA algorithms to update the solutions is achieved according to the $P_i$ for each solution, which is defined as the following equation.

Therefore, the solution $P_i$ is updated in the ETP using Equation (29). Updating the solutions continues until the stopping condition is reached.

$$P_i = \frac{fitness_i}{\sum_{i=1}^{N} fitness_i} \tag{29}$$

Therefore, the solution $pos_i$ is updated in the ETP using the following equation.

$$pos_i = \begin{cases} operators\ of\ AO & P_i > th \\ operators\ of\ AVOA & otherwise \end{cases} \tag{30}$$

Updating the solutions continues until the stopping condition is reached.

## 6. Evaluation Metrics and Experimental Results

In this paper, all the simulations for the proposed method and other algorithms have been performed using MATLAB R2018b software. An ASUS laptop with a Core i5 processor with 6 GB of memory and a frequency of 2.50 GHz and Windows 64-bit operating system has been used. Six physical machines (hosts) and 25 different VMs are considered for the fog-cloud computing environment. Table 1 shows the characteristics of the physical machines and hypothetical VMs in these experiments. According to Table 1, the slowest and fastest VMs have a speed of 100 MIPS and 5000 MIPS, respectively.

**Table 1.** Experiment parameters.

| | Specification | Amount |
|---|---|---|
| Client | Clients Count | [60, 120] |
| Physical Machine | Hosts Count | 6 |
| | CPU capacity | [100, 5000] |
| | Storage | 1 TB |
| | Network Bandwidth | 10 Gb/s |
| | RAM size | 6 GB |
| Virtual Machine | VMs count | 25 |
| | CPU capacity | [100, 5000] |
| | Storage | 20 GB |
| | RAM size | 1 GB |
| | Network Bandwidth | 1 Gb/s |
| | Processor | Xen |
| | Processors' count | 1 |

Two different datasets, namely High Performance Computing Center North (HPC2N) and NASA Ames iPSC/860 [56], have been used in these tests, each of which has 500, 1000, 1500, and 2000 independent and non-preemptive tasks [56]. The features of the two datasets used are given in Table 2.

**Table 2.** Dataset features of HPC2N and NASA iPSC.

| | Specification | Amount |
|---|---|---|
| NASA iPSC | CPUs | 240 |
| | Users | 257 |
| | Tasks | 202.871 |
| | Utilization | 60.1% |
| | Filename | NASA-iPSC-1993-3.1-cln.swf |
| HPC2N | CPUs | 128 |
| | Users | 69 |
| | Tasks | 18.239 |
| | Utilization | 46.6% |
| | Filename | HPC2N-2002-2.2-cln.swf |

That is, 240 processors or CPUs are needed for the NASA iPSC dataset, which is related to 257 users. The average number of tasks in this dataset is 202.871. A total of 128 processors or CPUs are needed for the HPC2N dataset, which is related to 69 users. The average number of tasks in this dataset is 18.239 [56]. The proposed AO_AVOA scheduling

algorithm is compared with five other algorithms including AVOA [53], AO [54], Firefly Algorithm (FA) [57], PSO [15], and HHO [58]. In order to achieve more stable results for the algorithms, each algorithm has been executed 30 times for each task set, and the average of these 30 executions is considered as the output. Table 3 shows the fixed parameters used in the AO_AVOA algorithm and other algorithms.

**Table 3.** Parameters' settings in the MH algorithms.

| | Specification | Amount |
|---|---|---|
| FA | $E_0$ | $[-1, 1]$ |
| | A | 0.5 |
| PSO | $c_1$ | 1.49 |
| | $c_2$ | 1.49 |
| AO | A | 0.1 |
| | $\Delta$ | 0.1 |
| AVOA | $L_1$ | 0.8 |
| | $L_2$ | 0.2 |
| | W | 2.5 |
| | $p_1$ | 0.6 |
| | $p_2$ | 0.4 |
| | $p_3$ | 0.6 |
| HHO | B | 0.2 |
| | $\Gamma$ | 1 |
| | W | $0.9 \rightarrow 0.4$ |

In the next section, the evaluation metric examined in this paper as well as the comparison process of the AO_AVOA with other algorithms are described.

### *6.1. Evaluation Metrics*

In this section, the evaluation metrics used in this metric, which include fitness function value, makespan time, and throughput time, are explained.

#### 6.1.1. Fitness Function Value

In problems that are solved using MH algorithms, the final solution of the problem is obtained by minimizing or maximizing the value of the fitness function. Therefore, the value of the fitness function can be used as an evaluation metric. In minimization problems, the lower the fitness function, the better the solutions obtained by the corresponding optimizer algorithm. When it comes to comparing several MH algorithms, the MH algorithm and the value of the fitness function are effective in determining the near-optimal solution. In this paper, Equation (2) is used as the fitness function, which is the same as makespan time.

#### 6.1.2. Makespan Time

In task scheduling methods in different fog, cloud, and fog-cloud computing environments, the value of makespan time is an metric to evaluate the method. The value of makespan time is equal to the completion time of the last task (Equation (2)) [6]. The lower value of makespan time indicates the superiority of the desired scheduling algorithm.

6.1.3. Throughput Time

The throughput time metric indicates the execution time of the tasks performed in a metric period [6], which is obtained by using Equation (31). The lower the throughput time, the better the scheduling algorithm.

$$\text{Throughput time} = \sum_{T_k \in T} Exe.Time(T_k) \tag{31}$$

6.1.4. Performance Improvement Rate

The next metric is performance improvement rate (PIR), which obtains the percentage of performance improvement of an algorithm compared to other algorithms using the value of its objective function [6]. The *PIR* value is obtained using Equation (32). The lower the *PIR* value of one algorithm compared to another algorithm, the closer the two algorithms are.

$$\text{PIR}(\%) = \frac{Fit_C - Fit_P}{Fit_P} \times 100 \tag{32}$$

where $Fit_C$ and $Fit_P$ are the objective values obtained by each of the comparative algorithms and AO_AVOA, respectively.

6.1.5. Comparison with Existing Works

In this section, the AO_AVOA is evaluated using fitness function, makespan time (second or sec), and throughput time evaluation metrics and is compared with AO, AVOA, PSO, HHO, and FA to improve the performance of AO_AVOA with AO, AVOA, PSO, HHO, and FA. Table 4 shows the results of makespan time for NASA iPSC and HPC2N datasets.

**Table 4.** Makespan time (sec) for datasets.

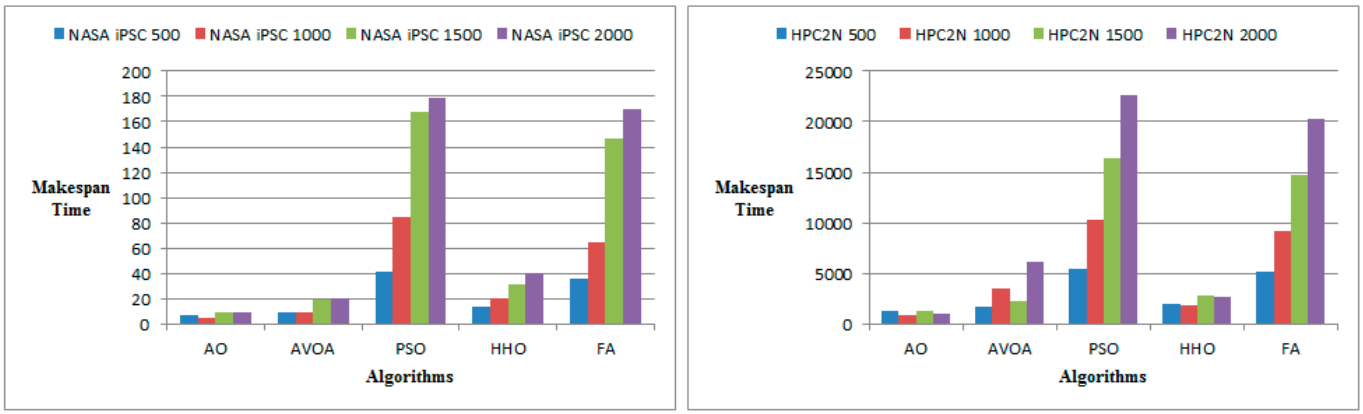
| **Dataset** | **NASA iPSC** | | | | **HPC2N** | | | |
|---|---|---|---|---|---|---|---|---|
| **Tasks Number** | **500** | **1000** | **1500** | **2000** | **500** | **1000** | **1500** | **2000** |
| AO_AVOA | 40.43 | 84.75 | 129.49 | 159.48 | 2566.39 | 9459.75 | 16,051.39 | 24,459.28 |
| AO | 47.61 | 89.38 | 138.29 | 168.41 | 3835.49 | 10,375.79 | 17,266.33 | 25,460.43 |
| AVOA | 49.44 | 93.45 | 148.72 | 179.31 | 4297.52 | 12,949.85 | 18,249.59 | 18,370.32 |
| PSO | 81.62 | 169.32 | 297.56 | 338.25 | 7945.46 | 19,789.69 | 32,459.72 | 47,141.17 |
| HHO | 53.75 | 105.32 | 160.52 | 199.79 | 4574.52 | 11,248.79 | 18,831.39 | 27,080.25 |
| FA | 75.85 | 149.81 | 276.48 | 329.12 | 7763.73 | 18,677.25 | 30,728.65 | 44,751.51 |

According to the results of Table 4, AO_AVOA has a lower makespan time than the comparative algorithms. AO_AVOA outperforms AO, AVOA, PSO, HHO, and FA for both datasets. Additionally, the value of makespan time is equal to the value of fitness, which is the lowest for AO_AVOA.

The reason for the superiority of AO_AVOA compared to the compared algorithms is that AO_AVOA was able to avoid becoming stuck in the local optimum by using the combination of AO exploration phases and AVOA exploitation and obtain the best solutions in a reasonable time for the task scheduling problem. In other words, AVOA is stronger than both AO and AVOA because it takes advantage of the high power of AO's exploration phase and the high power of AVOA's exploitation phase. However, AO is weak in the exploitation phase and AVOA is weak in the exploration phase. Additionally, the HHO algorithm is weaker than the AO, AVOA, and AO_AVOA algorithms, and this is related to the HHO structure, due to the fact that HHO cannot obtain the best solutions like AO, AVOA, and AO_AVOA due to its structure. The PSO and FA algorithms are much weaker than the AO, AVOA, HHO, and AO_AVOA algorithms due to their structure.

For both datasets with 500, 1000, 1500, and 2000 tasks, the diagram of the difference in makespan time value of the AO_AVOA with AO, AVOA, PSO, HHO, and FA methods is shown in Figure 3. The difference value of makespan time is obtained using Equation (33):

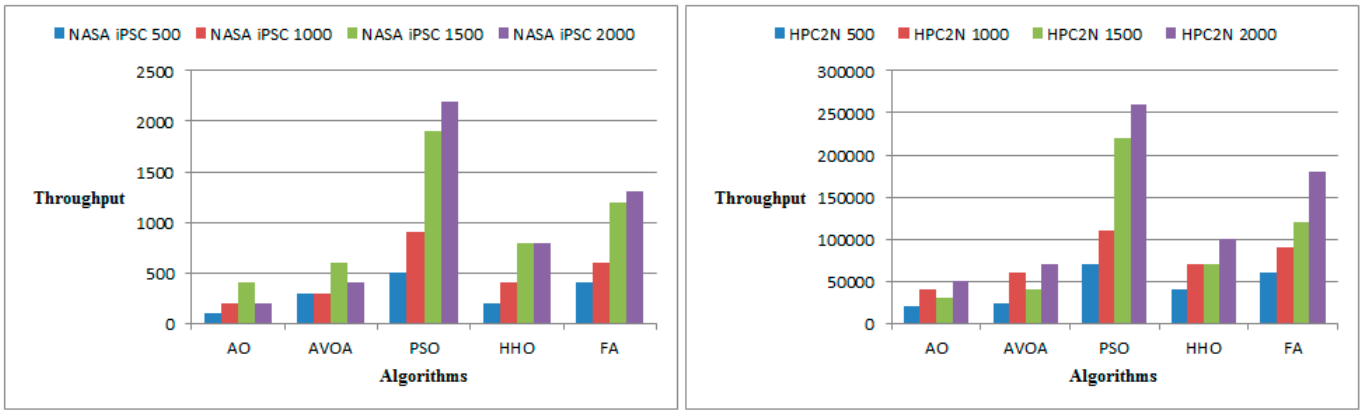
$$difference\ value\ \ of\ makespan\ time = makespan\_time_{AO\_AVOA} - makespan\_time_i \quad (33)$$




(**a**) (**b**)

**Figure 3.** Diagram of the difference in makespan time for AO_AVOA with AO, AVOA, PSO, HHO, and FA for both datasets. (**a**) NASA iPSC dataset. (**b**) HPC2N dataset.

In which $makespan\_time_{AO\_AVOA}$ represents the makespan time of AO_AVOA and $makespan\_time_i$ represents the makespan time of each of the other algorithms. According to Figure 3, AO_AVOA has a large difference with PSO, and it has the least difference with AO. The performance strength of AO is higher than the AVOA, PSO, HHO, and FA algorithms; AVOA is stronger than PSO, HHO and FA; and HHO is stronger than FA and PSO. PSO has the worst performance. The main reason for the difference in the outputs of the algorithms is the difference in their structure. Figure 4 shows the throughput time results of the AO, AVOA, PSO, HHO, FA, and AO_AVOA algorithms for two datasets with 500, 1000, 1500, and 2000 tasks. According to Figure 4, AO_AVOA has lower throughput time than AO, AVOA, PSO, HHO, and FA. This means that the relevant tasks have been completed by AO_AVOA in less time.

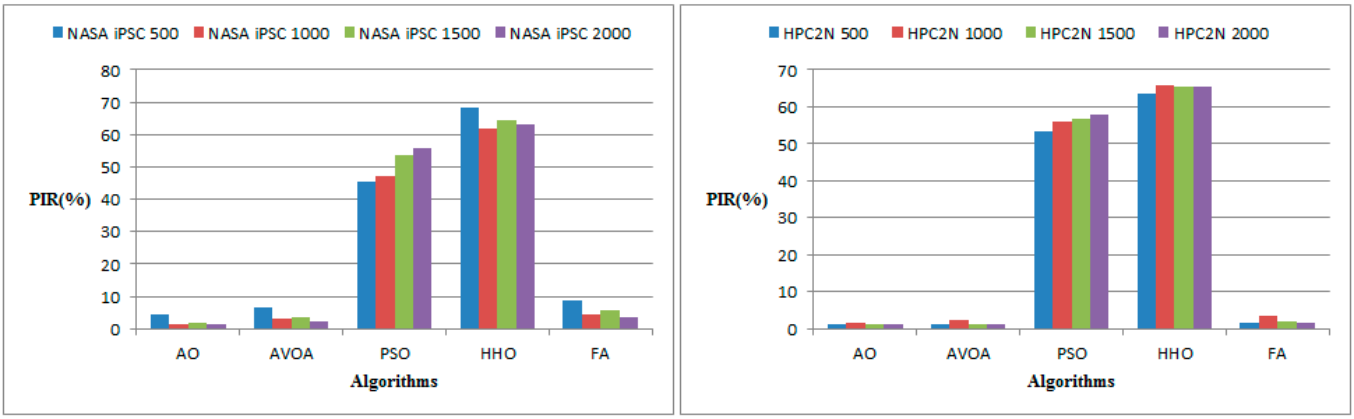



(**a**) (**b**)

**Figure 4.** Diagram of the throughput for AO_AVOA, AO, AVOA, PSO, HHO, and FA for both datasets. (**a**) NASA iPSC dataset. (**b**) HPC2N dataset.

Figure 5 shows the diagram of PIR (%) of the makespan time for the datasets. According to Figure 5, AO has a lower PIR (%) than AVOA, PSO, HHO, and FA. AO executes

the set of tasks in less time than the AVOA, FA, PSO, and HHO algorithms. As it is made clear from Figure 5, AO_AVOA has the least difference compared to AO and the biggest difference compared to PSO.




(**a**) (**b**)

**Figure 5.** PIR (%) diagram of the makespan time for AO, AVOA, PSO, HHO, and FA for both datasets. (**a**) NASA iPSC dataset. (**b**) HPC2N dataset.

## 7. Conclusions

In this paper, the problem of task scheduling in the fog-cloud environment for the implementation of IoT task requests was considered as an optimization problem considering more QoS requirements. The combination of the AVOA and AO algorithms was used to solve the task scheduling problem in the fog-cloud environment, which is called AO_AVOA. The AO_AVOA was used to improve the AVOA exploration process, and the exploration phase of the AO algorithm was combined with the exploitation phase of AVOA. The minimization of the makespan time function was used to evaluate the optimizer algorithm to optimize the scheduling problem and find the best virtual machines in the fog-cloud environment. AO_AVOA was applied to two datasets using the metrics of makespan time, fitness function value, PIR, and throughput time. Then, AO_AVOA was compared with the HHO, FA, PSO, AVOA, and AO algorithms, and it was proved that AVOA performs better than the mentioned algorithms in terms of relevant evaluation metrics. Compared to the AO, AVOA, PSO, HHO, and FA algorithms, AO_AVOA improves the makespan time by 8.36%, 2.61%, 104.38%, 17.56%, and 94.05%, respectively, for both datasets. In the future, it will be attempted to combine deep learning methods with AVOA to solve the task scheduling problem and examine more datasets, or AO_AVOA used as multi-objective.

**Author Contributions:** Conceptualization, Q.L. and H.K.; Methodology, Q.L. and S.M.; Software, S.M.; Validation, Q.L., H.K. and A.N.H.; Resources, H.K.; Data curation, Q.L. and H.K.; Formal analysis, K.A.A. and A.N.H.; Writing—original draft, Q.L.; Writing—review and editing, K.A.A. and A.N.H.; Supervision, A.N.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** Researchers Supporting Project number (RSP2023R305), King Saud University, Riyadh, Saudi Arabia.

**Data Availability Statement:** The data will be available upon reasonable request.

**Acknowledgments:** The authors extend their appreciation to King Saud University, Saudi Arabia for funding this work through Researchers Supporting Project number (RSP2023R305), King Saud University, Riyadh, Saudi Arabia.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
2. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
3. Masdari, M.; Khoshnevis, A. A survey and classification of the workload forecasting methods in cloud computing. *Clust. Comput.* **2020**, *23*, 2399–2424. [CrossRef]
4. Shakarami, A.; Ghobaei-Arani, M.; Shahidinejad, A.; Masdari, M.; Shakarami, H. Data replication schemes in cloud computing: A survey. *Clust. Comput.* **2021**, *24*, 2545–2579. [CrossRef]
5. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.
6. Elaziz, M.A.; Abualigah, L.; Attiya, I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Gener. Comput. Syst.* **2021**, *124*, 142–154. [CrossRef]
7. Yang, M.; Ma, H.; Wei, S.; Zeng, Y.; Chen, Y.; Hu, Y. A multi-objective task scheduling method for fog computing in cyber-physical-social services. *IEEE Access* **2020**, *8*, 65085–65095. [CrossRef]
8. Ghasempour, A.; Moon, T.K. Optimizing the number of collectors in machine-to-machine advanced metering infrastructure architecture for internet of things-based smart grid. In Proceedings of the 2016 IEEE Green Technologies Conference, Kansas City, MO, USA, 6–8 April 2016; pp. 51–55.
9. Chen, Z.G.; Zhan, Z.H.; Lin, Y.; Gong, Y.J.; Gu, T.L.; Zhao, F.; Qia, H. Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. *IEEE Trans. Cybern.* **2018**, *49*, 2912–2926. [CrossRef]
10. Javaheri, D.; Gorgin, S.; Lee, J.A.; Masdari, M. An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing. *Sustain. Comput. Inform. Syst.* **2022**, *36*, 100787. [CrossRef]
11. Ataie, I.; Taami, T.; Azizi, S.; Mainuddin, M.; Schwartz, D. D2FO: Distributed Dynamic Offloading Mechanism for Time-Sensitive Tasks in Fog-Cloud-IoT-based Systems. In Proceedings of the 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, 11–13 November 2022; pp. 360–366.
12. Taami, T.; Krug, S.; O'Nils, M. Experimental Characterization of Latency in Distributed IoT Systems with Cloud Fog Offloading. In Proceedings of the 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS), Sundsvall, Sweden, 27–29 May 2019; pp. 1–4.
13. Meisami, S.; Beheshti-Atashgah, M.; Aref, M.R. Using Blockchain to Achieve Decentralized Privacy in IoT Healthcare. *arXiv* **2021**, arXiv:2109.14812. [CrossRef]
14. Wang, Y.; Wen, X.; Gu, B.; Gao, F. Power Scheduling Optimization Method of Wind-Hydrogen Integrated Energy System Based on the Improved AUKF Algorithm. *Mathematics* **2022**, *10*, 4207. [CrossRef]
15. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
16. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
17. Mirjalili, S. Moth-Flame Optimization Algorithm: A Novel Nature-Inspired Heuristic Paradigm. *Knowl. -Based Syst.* **2015**, *89*, 228–249. [CrossRef]
18. Salehnia, T.; Fathi, A. Fault tolerance in LWT-SVD based image watermarking systems using three module redundancy technique. *Expert Syst. Appl.* **2021**, *179*, 115058. [CrossRef]
19. Raziani, S.; Salehnia, T.; Ahmadi, M. Selecting of the best features for the knn classification method by Harris Hawk algorithm. In Proceedings of the Conference: 8th International Conference on New Solutions in Engineering, Information Science and Technology of the Century, Online, 7–9 June 2021.
20. Cao, B.; Fan, S.; Zhao, j.; Tian, S.; Zheng, Z.; Yan, Y.; Yang, P. Large-Scale Many-Objective Deployment Optimization of Edge Servers. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3841–3849. [CrossRef]
21. Cao, B.; Gu, Y.; Lv, Z.; Yang, S.; Zhao, J.; Li, Y. RFID Reader Anticollision Based on Distributed Parallel Particle Swarm Optimization. *IEEE Internet Things J.* **2021**, *8*, 3099–3107. [CrossRef]
22. Cao, B.; Zhao, J.; Lv, Z.; Yang, P. Diversified Personalized Recommendation Optimization Based on Mobile Data. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2133–2139. [CrossRef]
23. Cao, B.; Li, M.; Liu, X.; Zhao, J.; Cao, W.; Lv, Z. Many-Objective Deployment Optimization for a Drone-Assisted Camera Network. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 2756–2764. [CrossRef]
24. Sun, B.; Li, Y.; Zeng, Y.; Chen, J.; Shi, J. Optimization planning method of distributed generation based on steady-state security region of distribution network. *Energy Rep.* **2022**, *8*, 4209–4222. [CrossRef]
25. Lin, Y.; Song, H.; Ke, F.; Yan, W.; Liu, Z.; Cai, F. Optimal caching scheme in D2D networks with multiple robot helpers. *Comput. Commun.* **2022**, *181*, 132–142. [CrossRef]
26. Zheng, W.; Yin, L. Characterization inference based on joint-optimization of multi-layer semantics and deep fusion matching network. *PeerJ Comput. Sci.* **2022**, *8*, e908. [CrossRef]
27. Nguyen, B.M.; Thi Thanh Binh, H.; The Anh, T.; Bao Son, D. Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud-fog computing environment. *Appl. Sci.* **2019**, *9*, 1730. [CrossRef]
28. Boveiri, H.R.; Khayami, R.; Elhoseny, M.; Gunasekaran, M. An efficient swarm-intelligence approach for task scheduling in cloud-based internet of things applications. *J. Ambient Intell. Humanized Comput.* **2019**, *10*, 3469–3479. [CrossRef]

29. Tong, Z.; Chen, H.; Deng, X.; Li, K.; Li, K. A scheduling scheme in the cloud computing environment using deep Q-learning. *Inform. Sci.* **2020**, *512*, 1170–1191. [CrossRef]
30. Yang, X.; Rahmani, N. Task scheduling mechanisms in fog computing: Review. trends, and perspectives. *Kybernetes* **2020**, *50*, 22–38. [CrossRef]
31. Mtshali, M.; Kobo, H.; Dlamini, S.; Adigun, M.; Mudali, P. Multi-objective optimization approach for task scheduling in fog computing. In Proceedings of the 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems, IcABCD, KwaZulu Natal, South Africa, 5–6 August 2019; pp. 1–6.
32. Ghobaei-Arani, M.; Souri, A.; Safara, F.; Norouzi, M. An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3770. [CrossRef]
33. Abualigah, L.; Shehab, M.; Alshinwan, M.; Alabool, H.; Abuaddous, H.Y.; Khasawneh, A.M.; Diabat, M.A. TS-GWO: IoT tasks scheduling in cloud computing using Grey Wolf optimizer. In *Swarm Intelligence for Cloud Computing, Chapman and Hall*; CRC: Boca Raton, FL, USA, 2020; pp. 127–152.
34. Zeng, D.; Gu, L.; Guo, S.; Cheng, Z.; Yu, S. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Trans. Comput.* **2016**, *65*, 3702–3712. [CrossRef]
35. Jena, T.; Mohanty, J. Ga-based customer-conscious resource allocation and task scheduling in multi-cloud computing. *Arab. J. Sci. Eng.* **2018**, *43*, 4115–4130. [CrossRef]
36. Rjoub, G.; Bentahar, J.; Wahab, O.A.; Bataineh, A.S. Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurr. Comput. Pract. Exp.* **2020**, *33*, e5919. [CrossRef]
37. Jacob, L. Bat algorithm for resource scheduling in cloud computing. *Int. J. Res. Appl. Sci. Eng. Technol.* **2014**, *2*, 53–57.
38. Raghavan, S.; Sarwesh, P.; Marimuthu, C.; Chandrasekaran, K. Bat algorithm for scheduling workflow applications in cloud. In Proceedings of the 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), Shillong, India, 29–30 January 2015; pp. 139–144.
39. Lin, Z.; Li, C.; Tian, L.; Zhang, B. A scheduling algorithm based on reinforcement learning for heterogeneous environments. *Appl. Soft Comput.* **2022**, *130*, 109707. [CrossRef]
40. Iftikhar, S.; Ahmad, M.M.M.; Tuli, S.; Chowdhury, D.; Xu, M.; Gill, S.S.; Uhlig, S. HunterPlus: AI based energy-efficient task scheduling for cloud-fog computing environments. *Internet Things* **2022**, *21*, 100667. [CrossRef]
41. Wadhwa, H.; Aron, R. Optimized task scheduling and preemption for distributed resource management in fog-assisted IoT environment. *J. Supercomput.* **2023**, *79*, 2212–2250. [CrossRef]
42. Shaheen, Q.; Shiraz, M.; Hashmi, M.U.; Mahmood, D.; Zhiyu, Z.; Akhtar, R. A Lightweight Location-Aware Fog Framework (LAFF) for QoS in Internet of Things Paradigm. *Mob. Inf. Syst. Hindawi* **2020**, *2020*, 8871976. [CrossRef]
43. Sing, R.; Bhoi, S.K.; Panigrahi, N.; Sahoo, K.S.; Bilal, M.; Shah, S.C. EMCS: An Energy-Efficient Makespan Cost-Aware Scheduling Algorithm Using Evolutionary Learning Approach for Cloud-Fog-Based IoT Applications. *Sustainability* **2022**, *14*, 15096. [CrossRef]
44. Mokni, M.; Yassa, S.; Hajlaoui, J.E.; Omri, M.N.; Chelouah, R. Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog–Cloud computing. *Simul. Model. Pract. Theory* **2023**, *123*, 123–102687. [CrossRef]
45. Sing, R.; Bhoi, S.K.; Panigrahi, N.; Sahoo, K.S.; Jhanjhi, N.; AlZain, M.A. A Whale Optimization Algorithm Based Resource Allocation Scheme for Cloud-Fog Based IoT Applications. *Electronics* **2022**, *19*, 3207. [CrossRef]
46. Panda, S.K.; Nanda, S.S.; Bhoi, S.K. A pair-based task scheduling algorithm for cloud computing environment. *J. King Saud Univ.—Comput. Inf. Sci.* **2022**, *34*, 1434–1445. [CrossRef]
47. Shukla, P.; Pandey, S.; Hatwar, P.; Pant, A. FAT-ETO: Fuzzy-AHP-TOPSIS-Based Efficient Task Offloading Algorithm for Scientific Workflows in Heterogeneous Fog–Cloud Environment. *Proc. Natl. Acad. Sci. India Sect. A Phys. Sci.* **2023**, *18*, 1–15. [CrossRef]
48. Stewart, R.; Raith, A.; Sinnen, O. Optimising makespan and energy consumption in task scheduling for parallel systems. *Comput. Oper. Res.* **2023**, *154*, 106212. [CrossRef]
49. Salehnia, T.; Montazerolghaem, A.; Mirjalili, S. An SDN-based optimal task scheduling method in Fog-IoT network using the combination of Aquila and Whale otimization ogorithms. *Compr. Metaheuristics Algorithms Appl.* **2023**, *2*, 48–65.
50. Salehnia, T.; Naderi, S.; Ahmadi, M.; Mirjalili, S. A workflow scheduling in cloud environment using a combination of Moth-Flame and Salp Swarm algorithms. *Appl. Soft Comput.* **2023**, *18*, 135–153.
51. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.S.; Min, G.; Dustdar, S.; Liu, J. Task Offloading for Cloud-Assisted Fog Computing With Dynamic Service Caching in Enterprise Management Systems. *IEEE Trans. Ind. Inform.* **2023**, *19*, 662–672. [CrossRef]
52. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.S.; Dus, S.; Liu, J. Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2023**, *19*, 480–490. [CrossRef]
53. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
54. Abualigah, L.; Yousri, D.; Elaziz, M.A.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization Algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
55. Cao, B.; Sun, Z.; Zhang, J.; Gu, Y. Resource Allocation in 5G IoV Architecture Based on SDN and Fog-Cloud Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3832–3840. [CrossRef]

56. Parallel Workloads Archive. Available online: http://www.cse.huji.ac.il/labs/parallel/workload/logs.html (accessed on 31 July 2020).
57. Yang, X.S. Multiobjective firefly algorithm for continuous optimization. *Eng. Comput.* **2013**, *29*, 175–184. [CrossRef]
58. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]