mathematics

MDPI

Article

An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing

Ibrahim Attiya [1], Laith Abualigah [2,3], Doaa Elsadek [1], Samia Allaoua Chelloug [4,*] and Mohamed Abd Elaziz [5,6,7]

1 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt; ibrahimateya@zu.edu.eg (I.A.); daudy.kaushy@gmail.com (D.E.)
2 Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan; Aligah.2020@gmail.com
3 School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, George Town 11800, Malaysia
4 Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
5 Faculty of Computer Science & Engineering, Galala University, Suze 435611, Egypt; abd_el_aziz_m@yahoo.com
6 Artificial Intelligence Research Center (AIRC), Ajman University, Ajman 346, United Arab Emirates
7 Faculty of Science, Zagazig University, Zagazig 44519, Egypt
* Correspondence: sachelloug@pnu.edu.sa

Abstract: The cloud computing paradigm is evolving rapidly to address the challenges of new emerging paradigms, such as the Internet of Things (IoT) and fog computing. As a result, cloud services usage is increasing dramatically with the recent growth of IoT-based applications. To successfully fulfill application requirements while efficiently harnessing cloud computing power, intelligent scheduling approaches are required to optimize the scheduling of IoT application tasks on computing resources. In this paper, the chimp optimization algorithm (ChOA) is incorporated with the marine predators algorithm (MPA) and disruption operator to determine the optimal solution to IoT applications' task scheduling. The developed algorithm, called CHMPAD, aims to avoid entrapment in the local optima and improve the exploitation capability of the basic ChOA as its main drawbacks. Experiments are conducted using synthetic and real workloads collected from the Parallel Workload Archive to demonstrate the applicability and efficiency of the presented CHMPAD method. The simulation findings reveal that CHMPAD can achieve average makespan time improvements of 1.12–43.20% (for synthetic workloads), 1.00–43.43% (for NASA iPSC workloads), and 2.75–42.53% (for HPC2N workloads) over peer scheduling algorithms. Further, our evaluation results suggest that our proposal can improve the throughput performance of fog computing.

Keywords: chimp optimization algorithm; marine predators algorithm; cloud computing; fog computing; task scheduling; makespan; metaheuristics

Citation: Attiya, I.; Abualigah, L.; Elsadek, D.; Chelloug, S.A.; Abd Elaziz, M. An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing. *Mathematics* **2022**, *10*, 1100. https://doi.org/10.3390/math10071100

Academic Editor: María Purificación Galindo Villardón

Received: 15 January 2022
Accepted: 21 March 2022
Published: 29 March 2022

1. Introduction

The Internet of Things (IoT) consists of massive distributed nodes, in which each node comprises various sensors [1]. Each sensor inside a node realizes the differences that happened in the enclosing area inside its authorized bandwidth [2,3]. Subsequently, after clustering and evaluating the gathered data, the traditional decisions and the arranged actions are made [4]. IoT is a system discovered in different applications, such as manufacturing, healthcare, agriculture, urban area, etc. Cloud computing (CC) is a data access stage for data backup and processing because it has high storage and processing [5–7]. The vast data produced by IoT devices (workload) is presented to CC for processing (task offloading) [8]. However, transferring the data straight to the cloud may cause network costs or, in another term, bandwidth overhead. The cloud typically delays the uploaded

data processing latency because of the long response time [9]. Consequently, a new alternative, namely, fog computing (FC), was introduced to address the abilities of the IoT devices [10,11].

Cisco launched FC technology in 2012, which has appeared as an architecture to promote the quality of services afforded to employers by CC [12,13]. Fog nodes are not valuable in computing applications; thus, it is utilized together with CC. Lately, CC presents the ideal alternative for various employers to handle, archive, and process their data in internet data centers. Regrettably, the need for CC devices has grown to utilize tremendous computing devices with a convenient pricing paradigm that simulates pay-on-use [14]. The extreme demand for CC benefits increases latency and needs a wide-range network. The huge latency produced by the cloud can transform refinement into a blemish [15]. For illustration, creating a trustworthy real time is essential to the science of medicine, as it follows patients' health. Hence, assistance is needed and must produce new data centers to satisfy their users' objectives and guarantee the quality of services (QoSs) provided [16,17]. The FC structure is presented in Figure 1.

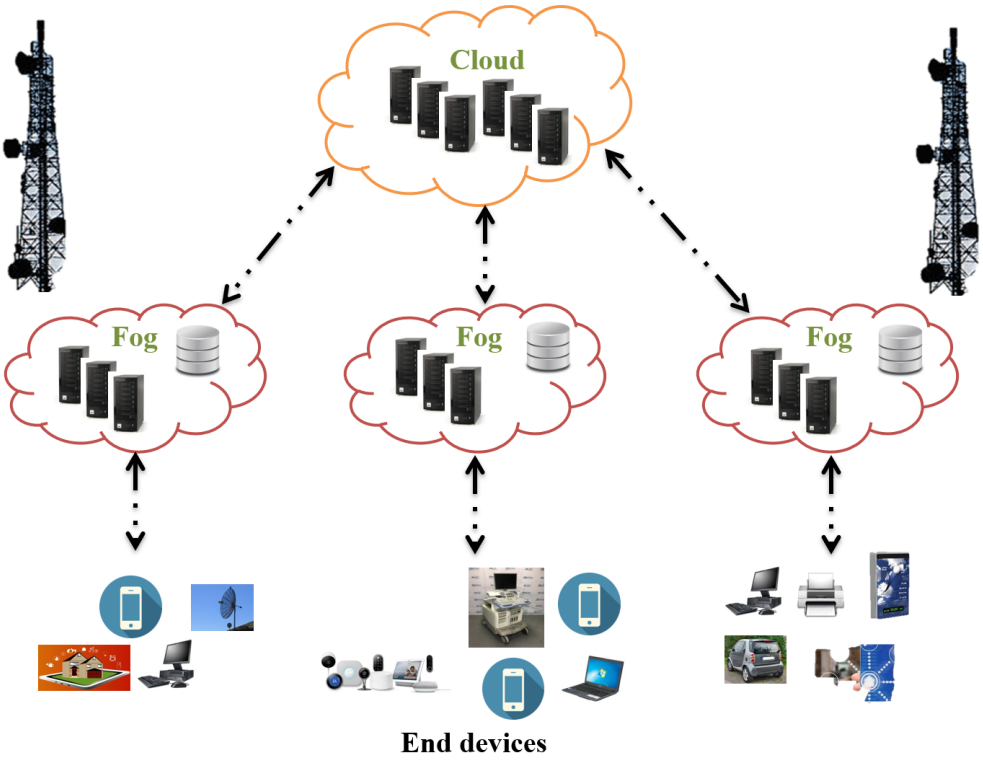

Figure 1. The fog computing structure.

Task scheduling in modern environments, such as CC, has a large-scale processing system [18]. Still, some aspects affect assignment execution, such as nonequilibrium load and little aid usage [19]. This procedure consumes considerable uncertainty and increases the cloud data center load; thus, the appearance of FC is crucial. In FC, task scheduling is the most fundamental issue to balance the requests and resources [20,21]. In IoT backgrounds, the user's requirements are different. Simultaneously, the FC platform's assistance arrangement is much higher than the CC platform's service performance. Hence, assigning the restricted resources effectively in FC and distributing them according to the consumers' needs to obtain the optimal scheduling aim is a critical problem that should be tackled.

Merging fog with CC is a new case in modern IoT investigation [22,23]. As is known to all, CC, inclusive hybrid, and public clouds are devices and waitpeople linked to the Internet, usually assisting users in a wide area. Recently, the literature presented several methods attempting to address task scheduling problems in IoT environments [24,25]. For

example, Wang et al. in [21] presented a novel task scheduling technique utilizing a new firework algorithm (I-FA) for decreasing the processing time of the task and resource in FC environments. The explosion radius discovery method is provided to the conventional firework algorithm in the proposed I-FA. Mtshali et al. in [26] introduced a task scheduling method using an intelligent approach to obtaining a practical method that can find the optimal power price with a lower uncertainty in FC backgrounds. The results of the given intelligent approach showed that it obtained better performance than other methods. Ghobaei-Arani et al. in [27] proposed a task scheduling approach based on the moth-flame optimizer to locate various tasks to FC nodes optimally. This approach attempted to obtain the QoSs satisfaction by reducing the total execution time. Yang et al. in [28] introduced a multi-objective strategy for solving task scheduling problems in FC environments. This method attempted to enhance the task computational needed time and find the optimal cost. In [29], a new intelligent task scheduling method was proposed in CC environments. The proposed method combined the Q-learning approach and deep neural network powers. The results of the proposed Q-learning approach are promising, and it obtained better performance than other comparative methods.

Chimp optimization algorithm (ChOA) is a novel search technique suggested by Khishe et al. [30]. The ChOA primary motivation is the general hunting performed by intelligent chimps. ChOA is devoted to solving several optimization problems according to these behaviors. For example, Khishe et al. [31] proposed a modified neural network (NN) using ChOA, which aims to determine the optimal weights for NN. This algorithm was evaluated by using it to classify the underwater acoustical dataset. However, this traditional ChOA algorithm suffers from some restrictions, such as being stuck in the local optimum and slow convergence, which impacts the quality of the solutions. Therefore, Kaur et al. [32] improved the behavior of ChOA by combining it with the sine–cosine algorithm. However, the modifications performed on ChOA still require further improvements. Additionally, several other optimizers can be used to solve these problems [33–36].

The proposed model depends on improving ChOA using the operators of the marine predators algorithm (MPA) [37] along with a disruption operator (DO). In the developed model, each of the MPA and DO has its own function, for example, MPA is used to enhance the local search process of ChOA and keep it from becoming stuck in the local point, whereas the DO is applied to improve the exploration ability of ChOA while maintaining the diversity of agents. In general, MPA has demonstrated its efficiency in solving different optimization problems, for example, forecasting cases of COVID-19 [38], power resources [39], photovoltaic array reconfiguration [40], and others [41].

In this study, an intelligent IoT task scheduling-based approach in FC environments is proposed. The proposed method improved the performance of the conventional ChOA, using external operators from the MPA and DO, so it is named CHMPAD. As the original method (ChOA) is safer from some issues during the optimization process in some cases, these issues are the low diversity of the candidate solutions. This problem hinders the development process from increasing the similarity in the available solutions, making it more difficult to find new and different solutions. The reason behind utilizing these operators is to incorporate their advantages together in order to develop an efficient method that can effectively address various task-scheduling-based problems. Comprehensive experiments with different numbers of tasks are conducted to evaluate the given CHMPAD method's performance. The achieved results reveal that the presented CHMPAD obtained better results in overall test scenarios. It is a superior task scheduling method compared to other optimization-based intelligent scheduling methods.

The contributions stated in this research are presented as follows.

- An intelligent task scheduling method is proposed, using an improved ChOA that depends on MPA and DO for IoT utilization in FC environments.
- The makespan, performance improvement ratio, and throughput time measures are considered to confirm IoT devices' QoSs requirements.

- We validate the performance of the proposed CHMPAD method in terms of the evaluation measures by conducting a set of experiments with various numbers of tasks, using a comprehensive CloudSim toolkit platform.

The structure of this paper is given as follows: in Section 2, the background for the problem mathematical presentation, the chimp optimization algorithm and the marine predators algorithm are presented. In Section 3, the proposed IoT-based TS method utilizing the improved chimp optimizer (CHMPAD) is presented. The evaluations and comparisons are shown in Section 4. We seal the outcomes of this research and suggest future directions in Section 5.

2. Background

This section aims to present the essential information for the problem formulation, the mathematical concepts of the chimp optimization algorithm, the marine predators algorithm, and disruption operator.

2.1. Problem Formulation

The task scheduling in IoT is considered a decision-making optimization problem that aims to allocate appropriate resources to the users required to process/task during a specific time [42]. The techniques used to tackle this problem in FC must consider users and cloud providers' perspectives, simultaneously.

The mathematical formulation of the IoT task scheduling (TS) in the fog environment can be defined by assuming there is a data center (DC) that consists of N_{ph} number of physical servers (PS) that are connected by using links which have a high-speed transmission. Each of those PSs contains N_{vm} virtual machines (VMs). These VMs are used to provide the services to the cloud customers' tasks; each VM has its hardware and software configurations. Moreover, the availability of the VM and characteristics of the tasks of users are passed to the cloud broker, who has the responsibility to determine the optimal allocation of those tasks to suitable VMs. This process is formulated by considering that there are N_T IoT tasks requested by users. Each of them has a specific length (T_{len}), which represents the processing requirement, and millions of instructions (MI) measure it. Then the execution time for completing (ETC) a certain task on a specific VM is computed as

$$ETC_{ij} = \frac{T^i_{len}}{VM^j_{Pow}}, i = 1, 2, ..., N_T, \; j = 1, 2, ..., N_{VM} \tag{1}$$

In Equation (1), the ETC_{ij} denotes the expected time to compute the task i on VM_j. VM^j_{Pow} denotes the processing speed of the jth VM. Thereafter, the ETC for each task overall the VMs is given as

$$ETC = \begin{bmatrix} ETC_{11} & ETC_{12} & ... & ETC_{1N_{VM}} \\ ETC_{21} & ETC_{22} & ... & ETC_{2N_{VM}} \\ ... & ... & ... & ... \\ ETC_{N_T1} & ETC_{N_T2} & ... & ETC_{N_TN_{VM}} \end{bmatrix} \tag{2}$$

The scheduling of IoT tasks in FC is considered an optimization problem that reduces the completion time (or makespan). Then, the formulation of this problem is given as

$$Fit = \min\left(\max_{j \in 1,2,...,m}\left(\sum_{i=1}^{n} ETC_{ij}\right)\right) \tag{3}$$

2.2. Chimp Optimization Algorithm

The chimp optimization algorithm emulates the intelligent behaviors of chimps [30]. Mathematical models of an autonomous community are presented: driving, blocking, chasing, and attacking. It would then define the corresponding ChOA algorithm.

2.2.1. Chasing and Driving

During discovery and extraction processes, the prey is hunted. To model the chasing and driving of the prey mathematically, Equations (4) and (5) are suggested, respectively.

$$D = |c \times X_b(t) - m \times X_{chimp}(t)| \tag{4}$$

$$X_{chimp}(t+1) = X_b(t) - a \times D \tag{5}$$

where t denotes the current iteration, X_{chimp} stands for the position of chimp and X_b denotes the prey position. a, m, and c denote the coefficients and they are updated as

$$a = 2 \times f \times r_1 - f,\ c = 2 \times r_2 \tag{6}$$

$$m = Chaotic\_value \tag{7}$$

In Equation (6), f is decreased non-linearly through the iteration course, from 2.5 to 0. r_1 and r_2 are random vectors. m stands for a chaotic value formed using different chaotic maps that expresses the impact of chimp sexual motivation in the hunting phase in this vector.

2.2.2. Attacking Method (Exploitation Phase)

The behavior of the chimp to attack the prey can be simulated by exploring the search space (using driving, blocking and chasing) about X_b followed by encircling it. Therefore, the mathematical modeling of these behaviors is performed using the best four positions and forcing the other chimps to follow them. This is formulated using Equations (8)–(10).

$$\begin{aligned} D_A &= |c_1 X_b - m_1 x|,\ D_B = |c_2 X_B - m_2 x|, \\ D_C &= |c_3 X_C - m_3 x|,\ D_D = |c_4 X_D - m_4 x| \end{aligned} \tag{8}$$

$$\begin{aligned} x_1 &= X_b - a_1(D_A),\ x_2 = X_B - a_2(D_B), \\ x_3 &= X_C - a_3(D_C),\ x_4 = X_D - a_4(D_D) \end{aligned} \tag{9}$$

$$X(t+1) = (x_1 + x_2 + x_3 + x_4)/(4) \tag{10}$$

2.2.3. Prey Attacking (Utilization)

The chimps will strike the prey in the final stage and end the chase at the time the prey stops running. To simulate the process of fighting, the f value is decreased and the a is updated according to the value of f inside $[-2f, 2f]$.

2.2.4. Searching for Prey (Exploration)

The mechanism of exploration between chimps is primarily carried out. They are diverging to hunt for the prey and to strike the prey in aggregate. This is simulated (i.e., divergence behavior) by using either $(a > 1)$ or $(a < -1)$. This technique displays the process of exploration since the search solutions are forced to diverge and to separate themselves from the prey. Moreover, the value of parameter $c \in [0, 2]$ affects the exploration ability of chimps. The value of c represents random weights, which aims to increase $(c > 1)$ or decrease $(c < 1)$, the influence of the prey, to determine the distance as in Equation (10).

2.2.5. Social Incentive (Sexual Motivation)

In the final stage, gaining meetings and resulting social motivation (sex and grooming) allows chimps to alleviate the hunting duties. Following [30], there is a probability of (50 percent) to update the position either normally or using chaotic maps as formulated in Equation (11).

$$X_{chimp}(t+1) = \begin{cases} X_b(t) - a \times D & if\ \mu < 0.5 \\ Chaotic\_value & if\ \mu \geq 0.5 \end{cases} \tag{11}$$

where $\mu \in [0, 1]$ is a random number.

2.3. Marine Predators Algorithm

The marine predators algorithm (MPA) [37], similar to other metaheuristics, is a population-based system. It begins by forming the population X as

$$X = Lb_{ij} + rand \times (Ub_{ij} - Lb_{ij}) \tag{12}$$

where Lb_{ij} and Ub_{ij} are the variables' lower and upper limits. *rand* stands for the random vector, ranging from 0 to 1. The elite matrix is formed as

$$Elite = \begin{bmatrix} X_{11}^1 & X_{12}^1 & \dots & X_{1d}^1 \\ X_{21}^1 & X_{22}^1 & \dots & X_{2d}^1 \\ \dots & \dots & \dots & \dots \\ X_{n1}^1 & X_{n2}^1 & \dots & X_{nd}^1 \end{bmatrix}, \tag{13}$$

2.3.1. Scenarios for MPA Optimization

The process of updating the solution using MPA is divided into three primary stages, which depends on a variant set of velocity ratios and parallel optimization phases as follows:

1. A high velocity ratio or whether X_b runs quicker than the predator.
2. In the ratio of unit velocity or where both predator and X_b travel at approximately the same time.
3. At a low velocity ratio, as the predator moves faster than X_b.

A particular cycle of iteration is assigned for each given process. These measures are illustrated using the basis of rules regulating the essence of predator and prey motion while emulating predator and prey movement in nature.

Phase 1: High-speed proportion or whether the predator is running faster than the prey. In the initial optimization iterations, where the exploration occurs, this situation happens. With a high-speed ratio of $(v \geq 10)$, the strongest predator approach does not move at all. In this rule, the mathematical model is implemented as

$$\begin{aligned} &while\,(iter < \frac{1}{3}Max - Iter) \\ &stepsize_i = R_B \otimes (Elite_i - R_B \otimes X_i), \\ &X_i = X_i + P \times R \otimes stepsize_i \end{aligned} \tag{14}$$

where $\otimes$ displays entry-wise multiplications, R_B contains random numbers according to the Brownian movement representing the normal distribution. The R_B emulates the movement of X_b. $P = 0.5$ represents a constant integer, and $R \in [0, 1]$ represents a random uniform number. In case the step size or movement for elevated exploration capacity occurs, velocity is high in the first third of iterations. The current iteration is Iter, while the limit is $Max - iter$.

Phase 2: In this phase, both exploitation and exploration are performed using half of the population. Brownian is the best tactic for the predator in the unit velocity ratio $(v \approx 1)$, as prey moves in Lévy. This is formulated as

$$While\,\frac{1}{3}Max - Iter < Iter < \frac{2}{3}Max - Iter \tag{15}$$

For the first half of X

$$\begin{aligned} stepsize_i &= R_L \otimes (Elite_i - R_L \otimes X_i),\ i = 1, ..., \frac{N}{2} \\ X_i &= X_i + P \times R \otimes stepsize_i \end{aligned} \tag{16}$$

A vector of Lévy distribution-based random numbers reflecting the Lévy movement is R_L. In the Lévy way, the multiplication of R_L and prey represents the prey movement. In

contrast, the prey movement is simulated by using the phase scale to the prey position. Since the large step size of the Levy distribution is correlated with small measures, exploitation is supported by this section. The second half of X is calculated by the following equation:

$$\begin{gathered} stepsize_i = R_B \otimes (R_B \otimes Elite_i - X_i),\ i = \frac{N}{2}, ..., N. \\ X_i = Elite_i + P \times CF \otimes stepsize_i \\ CF = (1 - \frac{Iter}{Max - Iter})^{(2\frac{Iter}{Max - Iter})} \end{gathered} \tag{17}$$

While CF is considered to monitor the phase size for predator motion as an adaptive parameter. The multiplication of R_B and elite simulates a Brownian-manner predator movement when the prey changes its location based on the predator movement in Brownian motion.

Phase 3: Low-speed ratio or whether the predator moves quicker than the prey. This represents the last stage of MPA, which is often correlated with high exploitation potential. Lévy is the perfect predator tactic for the low-speed ratio of $(v = 0.1)$. This process is seen as

$$\begin{gathered} while\ (iter > \frac{2}{3} Max - Iter) \\ stepsize_i = R_L \otimes (R_L \otimes Elite_i - X_i),\ i = 1, ..., N \\ X_i = Elite_i + P \times CF \otimes stepsize_i \end{gathered} \tag{18}$$

In the Lévy mechanism, the multiplication of elite and R_L emulates the predator's movement, while trying to apply the stage scale to the elite position emulates the movement of the predator to update the position of the predator further. The predator does not move at all in Phase 1, although it moves in Brownian motion in Phases 2 and 3, and it demonstrates the Lévy technique.

2.3.2. Eddy Formation and FAD's Effect

Fish aggregating devices' (FADs) impacts, as an environmental concern, are another point that induces a behavioral shift in marine predators. The FADs are treated as local optima with its trapping effect.Consideration of these longer hops eliminates inertia in local optima during the simulation. The influence of FADs is introduced as

$$X_i = \begin{cases} X_i + CF \times \left(Lb_{ij} + R \otimes RA_{ij}\right) \otimes U & if(r < FADs) \\ X_i + (FADs \times (1 - r) + r) \times D_r & if(r > FADs) \end{cases} \tag{19}$$

where $RA_{ij} = Ub_{ij} - Lb_{ij}$ and $D_r = X_{r1} - X_{r2}$. U represents a binary vector with values of zero and one. r is the uniform random number for $[0, 1]$. The vectors Lb_{ij} and Ub_{ij} contain the lower and upper dimensional boundaries. The subscripts $r1$ and $r2$ denote random prey matrix indexes.

2.3.3. Marine Memory

In remembering the location where they were active in foraging, marine predators have a clear memory. This capacity is simulated by the MPA memory storage. This matrix stands for the assessment of fitness to upgrade the elite after updating the prey and applying the FADs effect. The fitness of the current solution is contrasted with its value at the previous iteration. If it is fitter, the current one substitutes the solution. This approach also increases the efficiency of the solution with the iteration lapse. Additionally, it stimulates the return of predators with good foraging to the areas of the large prey area.

2.4. Disruption Operator

This section introduces the basics of the disruption operator (DO). In general, the DO is inspired by the astrophysics, and the mathematical formulation of DO is given as [43]

$$DO = \begin{cases} dist_{i,j} \times \delta(\frac{-1}{2}, \frac{1}{2}) & if \ \ dist_{i,b} \geq 1 \\ 1 + dist_{i,b} \times \delta(\frac{-10^{-16}}{2}, \frac{10^{-16}}{2}) & otherwise \end{cases} \quad (20)$$

In Equation (20), $dist_{i,j}$ denotes the Euclidean distance between the agent i and its nearest one j. Whereas the distance between i and X_b is given by $dist_{i,b}$. $\delta(a, b)$ is used to generate random value from $[a, b]$.

The disruption operator aims to improve the MH techniques' searching process by making them emphasize the diversity of solutions. This leads to improving the exploration of the solutions and the quality of the final solution.

3. The Proposed Task Scheduler

The general structure of the developed task scheduler based on the modified chimp optimization algorithm (CHMPAD) is given in Figure 2. The chimp optimizer's improvement is achieved by using the operators of MPA to enhance its exploitation ability. Simultaneously, the disruption operator is also applied to maintain the solution's diversity during the searching process. This leads to increase the convergence rate and the efficiency of the final solution for the task scheduling problem.

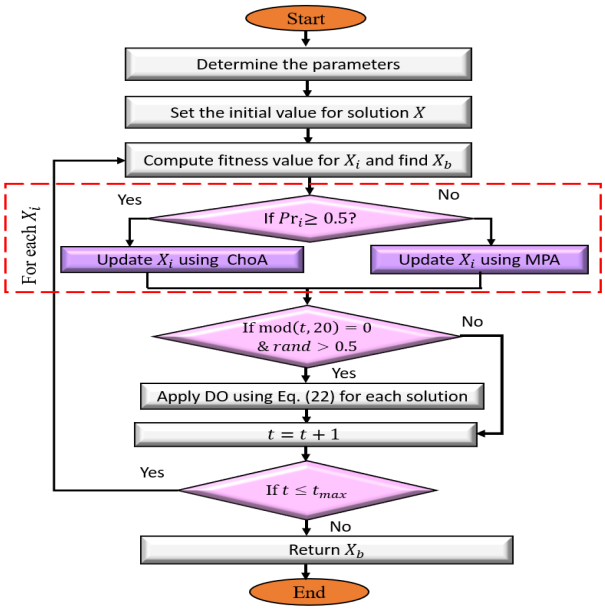

Figure 2. Developed CHMPAD method.

The proposed CHMPAD begins by generating a set of N agents (X). Each one refers to a solution for the task scheduling problem. Next, the fitness function (Fit_i) is used to assess the candidate solutions' quality $X_i, i = 1, 2, ..., N$. After that, the best solution (X_b) is determined based on the smallest Fit_b value.

The next step is to update X, using MPA operators to improve the exploitation ability and use the disruption operator (DO) to enhance the exploration ability. This process starts by updating the current agent (X_i), using either the MPA steps or steps of ChOA according to random probability. This strategy allows the MPA and ChOA to be competitive in updating their agents. The last step in updating the agents is applying the DO to maintain the diversity of the solutions. The framework of CHMPAD for the IoT scheduling problem is shown in Figure 2. The following subsections illustrate with more stages of CHMPAD.

3.1. Initial Stage

Within this stage, a population (X) is formed using Equation (21).

$$X_{ij} = floor(Lb_{ij} + \alpha \times (Ub_{ij} - Lb_{ij})),\ \alpha \in [0,1],\ j = 1,2,...,n \tag{21}$$

where, the $floor(.)$ function is used to convert the real values to integer values, which belong to the interval $[Lb, Ub]$. $Lb = 1$ and $Ub = m$ are the limits of the boundary. Note that Equation (21) represents the process of converting the original representation of ChOA from a continuous form to discrete form to be suitable for the task scheduling problem.

3.2. Updating Stage

The developed CHMPAD begins by evaluating the fitness value (Fit_i) (as defined in Equation (3)) for each agent and then finding X_b that has the smallest fitness value. This is followed by updating the value of each agent (i.e., X_i) according to the MPA (as in Equations (14)–(19)) or ChOA (as in Equations (4)–(11)) according to Equation (22).

$$X_i(t+1) = \begin{cases} Update\ X_i\ using\ ChOA & if\ \ Pr_i \geq 0.5 \\ Update\ X_i\ using\ MPA & if\ \ otherwise \end{cases}. \tag{22}$$

In Equation (22), $Pr_i \in [0,1]$ represents a random value used to control the updating of X according to either MPA and ChOA. After that, the DO is applied to the current population (X) by using Equation (20). However, since this process is time consuming, we apply the DO over 20 iterations and when $\delta > 0.5$, where $\delta \in [0,1]$ is a random number. Lastly, check the termination condition when it is satisfied, then stop the execution of the developed CHMPAD and return the best solution (X_b); otherwise, repeat the steps of CHMPAD. The pseudo-code of the developed CHMPAD is illustrated in Algorithm 1.

Algorithm 1: CHMPAD scheduler

```
Input: n No. of IoT tasks, m No. of group of VMs, t_max No. of iterations, and number of solutions (N).
Form the initial solutions (X).
Put t = 1.
while t <= t_max do
    Assess X_i by computing its Fit_i.
    Determine X_b that has the smallest Fit_b.
    for i = 1 : N do
        Enhance X_i using Equation (22).
    t = t + 1.
Return X_b.
```

3.3. Complexity of the Proposed CHMPAD

The time complexity of the proposed task scheduler-based CHMPAD depends on some main factors, such as number of iterations t_{max}, number of used solutions N, dimension of problem D, and complexity of ChOA, MPA, and DO. Thus, the complexity of the proposed CHMPAD method is given as follows:

$$O(CHMPAD) = (N) + t \times (O(ChOA) + O(MPA) + O(DO)) \tag{23}$$

The time complexities of the ChOA, MPA, and DO are given in Equations (24)–(26).

$$O(ChOA) = O(N_{CH} \times (Dim + 1)) \tag{24}$$

$$O(MPA) = O(N_{MP} \times Dim) \tag{25}$$

$$O(DO) = O(N \times Dim) \tag{26}$$

Finally, the complexity of the proposed CHMPAD method is presented as follows.

$$O(CHMPAD) = N + t \times (N_{CH}(Dim + 1)) + N_{MP} \times Dim + N \times Dim) \quad (27)$$

where N_{CH} and N_{MP} are the number of solutions updated using ChOA and MPA operators, respectively.

4. Experimental Results

This section describes the experimental configuration, results analysis, and discussions.

4.1. Experimental Settings

This sub-section describes the experimental environment, parameter configurations, and details of the workloads utilized in our experiments.

To investigate whether the development of CHMPAD was effectively accomplished, the performance evaluations of our algorithm were implemented using CloudSim toolkit [44]. CloudSim was chosen for our experiments because of its support for the modeling and simulation of large and complex computing environments. The experimental tests were carried out on a PC with an Intel Core i5 @ 2.40 GHz, 4 GB RAM, running Ubuntu 14.04 and CloudSim 3.0.3 toolkit. Table 1 offers a detailed configuration of the presented cloud–fog environment.

Table 1. Configuration of empirical parameters.

Entity	Parameter	Value
Datacenter	No. of datacenters	2
Host	Number of hosts	4
Cloud nodes:		
VM	No. of VMs	10
	CPU power	3000–5000 MIPS
	RAM	1 GB
	Storage	20 GB
	Bandwidth capacity	0.5 Gb/s
Fog nodes:		
VM	No. of VMs	15
	CPU power	100–2800 MIPS
	RAM	0.5 GB
	Storage	10 GB
	Bandwidth capacity	1 Gb/s
	No. of CPUs	1

In the conducted experiments, the CHMPAD algorithm's effectiveness was assessed, using synthetic and real workload instances. The synthetic workload utilized herein comprises 1000 tasks with lengths varying from 1000 to 20,000 MI that were generated from a uniform distribution. The description of the synthetic workload is presented in [45]. Additionally, the real datasets generated by the "Parallel Workload Archives" consisting of NASA Ames iPCS/860 and HPC2N were utilized in the experimentation. In particular, we utilized the cleaned versions of the real workload traces (i.e., NASA-iPSC-1993-3.1-cln.swf and HPC2N-2002-2.2-cln.swf) [46].

4.2. Performance Metrics

The metrics utilized to compare the CHMPAD algorithm against similar scheduling strategies in the literature are presented in this subsection.

Makespan is the most well-known criterion for assessing the quality and efficacy of scheduling strategies. It is defined as the finishing time of the last completed task [47].

Throughput refers to the total number of tasks that are implemented successfully within a given period of time. The cloud system must reach a high level of throughput to ensure

that it takes a minimum amount of time to produce the results of the given job. Hence, the throughput can be computed by using the following equation:

$$Throughput = \sum_{t_i \in Tasks} ExeTime(t_i) \tag{28}$$

where $ExeTime(t_i)$ indicates the execution time of ith task.

Performance improvement rate (PIR) is a measure that estimates the percentage of improvement in performance for every developed method with regard to the comparison methods from the related literature as presented in Equation (29).

$$PIR(\%) = \frac{(Y - Y')}{Y'} \times 100 \tag{29}$$

where Y' signifies the fitness value attained by CHMPAD, and Y signifies the fitness value achieved by either of the comparison methods.

4.3. Results and Discussions

This subsection describes and summarizes the experiments' findings to validate our CHMPAD scheduling algorithm's performance optimization. The criteria that are considered for performance assessment are the makespan time, throughput, and PIR rate. The results obtained are contrasted with the results of five popular scheduling algorithms, including the conventional ChOA algorithm [30], MPA algorithm [37], whale optimization algorithm (WOA) [48], HGSWC [49], chaos game optimization (CGO) [50], atomic orbital search (AOS) [51], and chameleon swarm algorithm (CSA) [52]. The specific parameter settings of HGSWC are given in [49], while those of CHMPAD and other comparative approaches are summarized in Table 2. The swarm size is set equal to 100 for all algorithms to ensure a fair comparison.

Table 2. Parametric details of the comparison algorithms.

Algorithm	Parameter	Value
CGO	α, β	[0, 1]
AOS	Maximum number of Layers	5
	Foton Rate	0.1
ChOA	a	$[-1, 1]$
	f	$2 \rightarrow 0$
WOA	a	2
	b	1
MPA	$FADs$	0.2
	P	0.5
CSA	γ	2
	α	4
	ρ	1
	β	3
CHMPAD	a	$[-1, 1]$
	f	$2 \rightarrow 0$
	$FADs$	0.2
	P	0.5

To assess the efficacy of the comparison algorithms, the results obtained by CHMPAD are compared and contrasted with those of the ChOA, MPA, CGO, AOS, CSA, WOA, and HGSWC algorithms, as given in Tables 3–5. The best, average and worst makespan values obtained by the comparison algorithms under 30 repetitions are listed in these Tables. Based on the tables' findings, it is obvious that CHMPAD generates better results in most

of the tested instances, compared to the other seven comparison scheduling algorithms. Specifically, in terms of the average makespan achieved, CHMPAD outperforms AOS in 14 out of 15 instances, outperforms MPA in 13 out of 15 instances, and significantly outperforms ChOA, WOA, HGSWC, CSA, and CGO in all 15 test instances. In a similar manner, CHMPAD outperforms both MPA and AOS in 9 out of 15 instances and outperforms ChOA and the other four existing metaheuristics in all the test instances for the best makespan gained. The trend when makespan is considered suggests that the presented CHMPAD strategy is an advancement over the traditional ChOA approach. Additionally, it outperforms its competitors in most instances. Hence, it can be concluded that CHMPAD is a very competitive method for solving medium to large cloud-task-scheduling problems.

Table 3. Makespan values of CHMPAD algorithm and compared algorithms—synthetic workload.

Tasks	Statistics	CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA	CHMPAD
200	Best	38.66	36.15	38.28	44.93	37.39	32.60	42.93	36.32
	Average	42.43	38.27	39.74	52.77	42.58	36.42	47.61	37.85
	Worst	45.73	40.46	41.75	68.65	46.57	45.93	52.84	38.95
400	Best	78.68	69.75	73.15	86.23	78.03	66.65	83.24	71.41
	Average	84.47	74.42	76.80	101.10	84.32	79.35	94.25	73.37
	Worst	92.38	78.74	80.85	125.45	93.12	94.93	103.57	75.41
600	Best	111.73	106.00	106.76	120.37	117.18	103.66	118.09	103.94
	Average	124.50	110.75	111.81	147.89	127.13	119.94	135.69	106.84
	Worst	137.15	116.58	116.98	180.13	138.02	143.57	154.78	108.34
800	Best	148.87	136.54	143.31	161.34	152.45	149.91	151.84	134.94
	Average	161.72	144.18	148.93	193.77	165.53	166.32	172.45	139.07
	Worst	170.69	150.08	154.16	260.97	177.09	193.23	199.96	145.28
1000	Best	186.38	170.66	172.85	214.48	188.53	178.89	183.20	167.95
	Average	200.69	176.80	179.69	244.36	201.45	209.50	204.59	170.64
	Worst	223.70	185.76	190.90	329.45	219.51	242.37	240.83	172.17

Table 4. Makespan values of CHMPAD algorithm and compared algorithms—NASA iPSC workload.

Tasks	Statistics	CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA	CHMPAD
500	Best	53.43	48.68	50.88	58.85	53.21	44.61	60.32	50.76
	Average	58.71	51.68	55.09	71.53	58.45	50.71	65.70	51.78
	Worst	62.49	54.64	58.34	92.19	67.50	61.26	73.56	53.70
1000	Best	107.52	95.22	102.15	125.42	103.84	92.28	113.75	97.59
	Average	114.40	101.06	106.67	142.66	116.52	111.20	128.18	100.06
	Worst	121.24	106.57	112.31	176.66	125.86	129.78	147.94	106.88
1500	Best	156.73	143.49	147.98	174.49	157.36	161.31	162.41	145.40
	Average	173.07	151.75	157.99	205.26	178.16	181.69	182.64	147.90
	Worst	191.08	159.50	167.97	250.88	194.67	204.35	202.27	150.41
2000	Best	208.67	193.65	197.06	234.45	210.43	222.25	219.86	191.07
	Average	234.59	203.37	207.24	276.32	235.84	249.71	244.14	195.49
	Worst	254.45	218.14	215.31	303.73	256.30	284.06	268.32	200.08
2500	Best	271.69	244.97	246.94	306.25	266.64	298.14	269.32	240.66
	Average	295.32	255.79	257.85	350.09	299.53	331.69	296.97	244.09
	Worst	322.06	267.81	267.50	426.29	329.11	363.05	337.96	250.27

Table 5. Makespan values of CHMPAD algorithm and compared algorithms—HPC2N workload.

Tasks	Statistics	CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA	CHMPAD
500	Best	5587.41	5173.63	5347.63	6189.20	5843.56	5073.24	5784.75	5253.96
	Average	6142.08	5518.09	5654.95	7445.34	6353.31	5959.25	6775.71	5370.14
	Worst	6633.14	5764.36	5874.67	8626.60	7253.51	8121.81	8068.42	5455.81
1000	Best	12,322.75	11,220.68	11,664.67	14,026.45	12,498.23	12,755.55	12,484.81	11,195.54
	Average	13,586.98	11,783.58	12,133.40	15,864.37	13,753.92	14,314.34	13,974.44	11,392.63
	Worst	15,323.38	12,238.52	12,467.56	18,423.01	16,124.69	16,058.39	16,018.29	11,560.03
1500	Best	20,085.68	17,837.49	17,925.15	21,381.87	20,151.27	20,463.79	19,880.14	17,730.03
	Average	21,269.73	18,841.35	18,698.31	25,571.57	21,790.42	24,319.46	22,143.31	17,940.78
	Worst	23,218.31	20,001.50	19,361.54	31,233.22	24,243.83	28,299.33	26,616.21	18,160.03
2000	Best	29,022.40	25,452.85	25,116.20	30,542.26	29,316.27	29,628.07	27,733.14	25,087.30
	Average	31,316.53	26,775.45	26,711.07	35,579.61	30,981.60	34,678.12	31,104.27	25,484.77
	Worst	33,143.37	28,080.38	27,662.21	43,729.87	33,945.03	40,197.73	36,867.45	26,122.76
2500	Best	36,804.59	33,825.37	34,020.78	40,877.95	36,593.99	42,897.24	36,529.28	33,048.68
	Average	40,444.64	35,404.83	35,198.99	47,468.26	40,595.96	47,010.57	41,190.84	33,445.82
	Worst	42,521.72	36,716.87	36,756.72	60,882.06	43,628.67	51,432.00	47,738.72	34,176.51

Figures 3–5 compare the achieved throughput between the CHMPAD, ChOA, MPA, CGO, AOS, CSA, WOA, and HGSWC algorithms using the synthetic and real workloads. From these figures, it can be observed that CHMPAD achieves better throughput than the other evaluated metaheuristic approaches for all the task sizes and datasets. This proves the CHMPAD algorithm's stability and robustness in finding near-optimal solutions not only when synthetic tasks are used, but for real tasks as well.

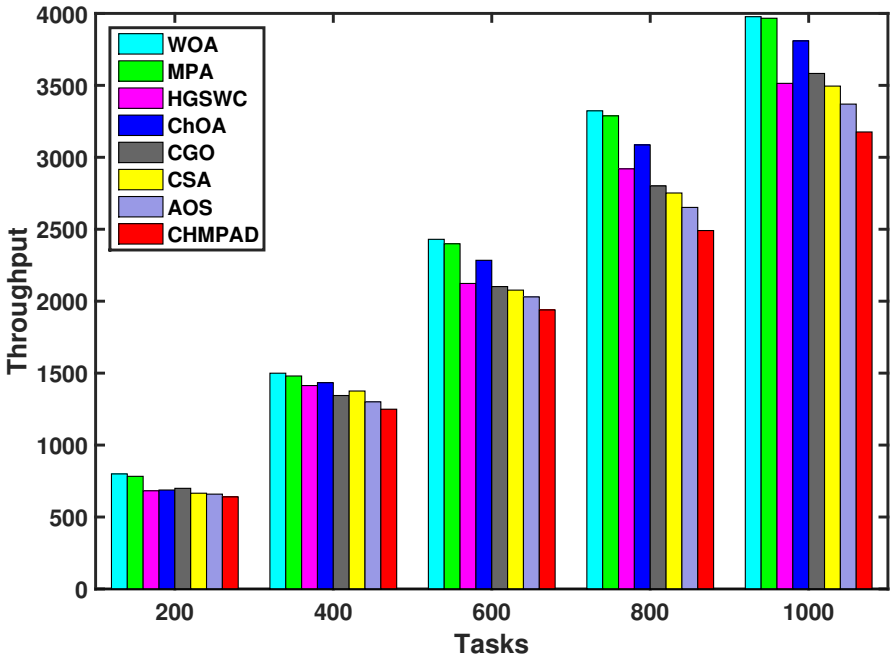

Figure 3. Average throughput time—synthetic workload.

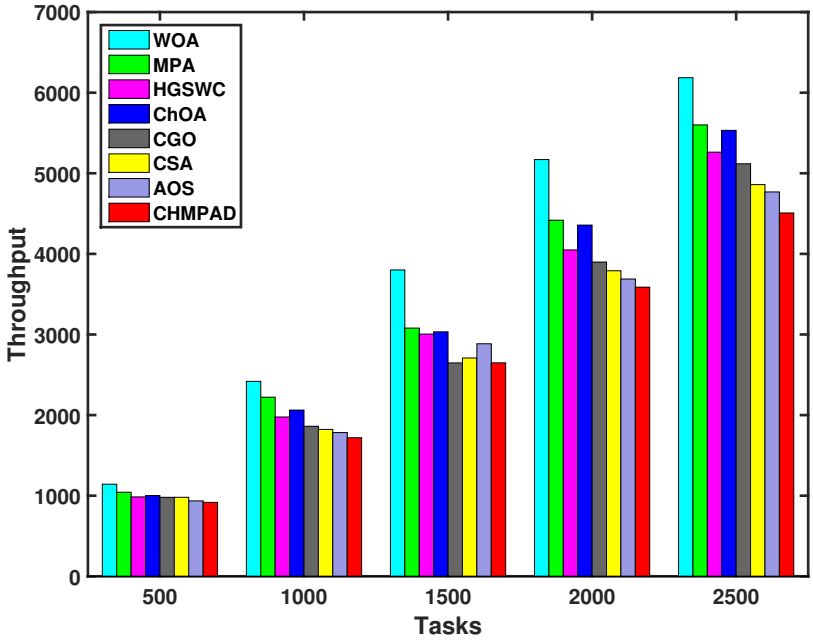

Figure 4. Average throughput time—NASA iPSC workload.

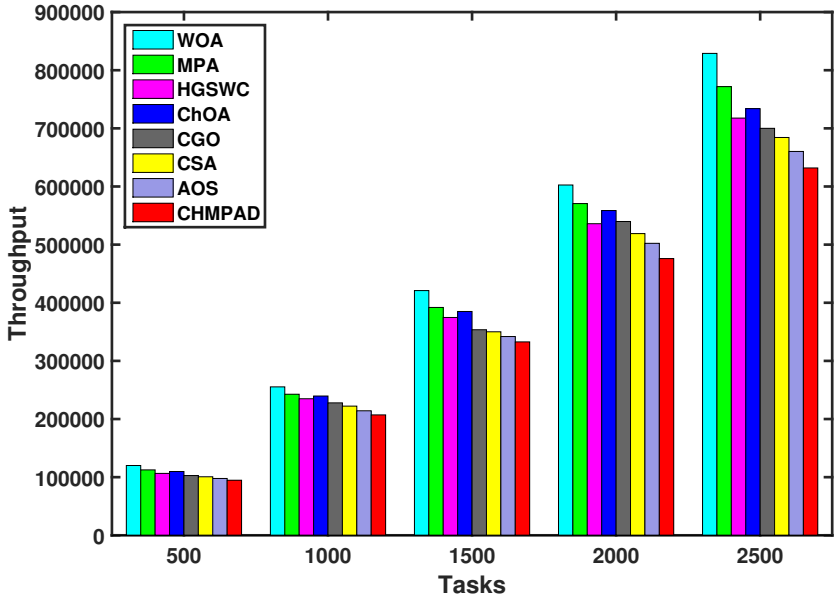

Figure 5. Average throughput time—HPC2N workload.

In order to deliver a more accurate assessment of the achieved results, improvements of the average makespan (given in percentage) attained by CHMPAD method over its competitors are shown in Table 6. From Table 6, the results show that the CHMPAD can accomplish 19.89–28.46% average makespan time improvements over the ChOA algorithm when considering the synthetic workload. Similarly, the CHMPAD significantly outperforms the standard ChOA by approximately 21.66–28.10% for executing the NASA iPSC real workload. Moreover, the results also state that the performance improvement of the CHMPAD is 22.05–26.17% over the ChOA for the execution of the HPC2N real workload.

To further validate our proposed approach's superiority, improvements of the average makespan achieved by CHMPAD over other existing approaches are also presented in Table 6. From the table, the results demonstrate that CHMPAD can acquire an improvement of 37.81–43.20% over the WOA, 12.12–17.61% over the CGO, 12.50–19.02% over the HGSWC, 4.65–7.09% over the CSA, and 1.12–3.61% over the AOS when the synthetic workload

is considered. Additionally, CHMPAD produces performance improvement over MPA ranging 8.15–22.77% across the synthetic workload, except the test instance with 200 tasks. In this test instance, the average makespan found by MPA is 3.77% lower than that of CHMPAD. For the NASA iPSC workload, the reported results indicate that the CHMPAD algorithm can accomplish 38.15–43.43%, 13.40–20.99%, 12.89–22.71%, and 5.64–6.82% makespan time improvements over the WOA, CGO, HGSWC, and CSA approaches, respectively. We can also see that CHMPAD outperforms MPA and AOS on almost all of the NASA Ames workload instances, except the instance with 500 tasks. Furthermore, it is clear from Table 6 that the average makespan time found by CHMPAD is 38.64–42.53% lower than the WOA, 14.37–22.88% lower than the CGO, 18.31–21.57% lower than the HGSWC, 4.22–6.50% lower than the CSA, 10.97–40.56% lower than the MPA, and 2.75–5.86% lower than the AOS for the execution of HPC2N workload.

Table 6. The $PIR(\%)$ of the average makespan for different workload traces.

Instance	No. of Tasks	PIR (%) Over						
		CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA
Synthetic	200	12.12	1.12	4.99	39.44	12.50	−3.77	25.78
	400	15.13	1.44	4.69	37.81	14.94	8.15	28.46
	600	16.52	3.66	4.65	38.42	18.99	12.26	27.00
	800	16.29	3.67	7.09	39.33	19.02	19.60	24.00
	1000	17.61	3.61	5.30	43.20	18.06	22.77	19.89
NASA iPSC	500	13.40	-0.18	6.40	38.15	12.89	−2.05	26.89
	1000	14.33	1.00	6.60	42.57	16.45	11.13	28.10
	1500	17.02	2.60	6.82	38.78	20.46	22.85	23.49
	2000	20.00	4.03	6.01	41.34	20.64	27.73	24.88
	2500	20.99	4.80	5.64	43.43	22.71	35.89	21.66
HPC2N	500	14.37	2.75	5.30	38.64	18.31	10.97	26.17
	1000	19.26	3.43	6.50	39.25	20.73	25.65	22.66
	1500	18.56	5.02	4.22	42.53	21.46	35.55	23.42
	2000	22.88	5.06	4.81	39.61	21.57	36.07	22.05
	2500	20.93	5.86	5.24	41.93	21.38	40.56	23.16

To further analyze the results obtained by the developed method, we utilized both the Wilcoxon rank test and Friedman test as non-parametric tests. Table 7 shows p-values obtained using the Wilcoxon rank test. It can be observed that most of the provided p-values are less than 0.05, and this indicates that there is a significant difference between the developed CHMPAD and each of the other comparison methods. However, there is no significant difference between CHMPAD and AOS for synthetic 400 and NASA iPSC 500. In addition, there is no significant difference between CHMPAD and MPA for NASA iPSC at 500.

Furthermore, the findings of performing the statistical Friedman test are given in Table 8. From this table, it can be noticed that CHMPAD has the best mean rank among the competitive algorithms. It is followed by AOS and CSA, which have the second and third best mean ranks of makespan of all the compared algorithms.

In conclusion, the experimental results reported here indicate a substantial improvement in the makespan time and throughput on most of the benchmark instances. Our proposed approach's promising performance is mostly due to the integration of the MPA operators along with the disruption operator within the ChOA algorithm. We conclude that CHMPAD has proved to be a powerful and effective method for solving large-size and challenging task scheduling problems.

Table 7. Wilcoxon rank test.

Instance	Tasks	CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA
Synthetic	200	5.46×10^{-11}	0.122301	4.97×10^{-9}	3.02×10^{-11}	8.89×10^{-10}	0.003501	3.02×10^{-11}
	400	3×10^{-11}	0.025081	2.38×10^{-8}	3.02×10^{-11}	3.02×10^{-11}	7.2×10^{-5}	3.02×10^{-11}
	600	3×10^{-11}	1.1×10^{-6}	1.17×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.82×10^{-9}	3.02×10^{-11}
	800	3×10^{-11}	4.43×10^{-7}	1.2×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	1000	3×10^{-11}	1.69×10^{-9}	3×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
NASA iPSC	500	3.32×10^{-11}	0.579241	4.1×10^{-7}	3.02×10^{-11}	3.34×10^{-11}	0.325527	3.02×10^{-11}
	1000	3×10^{-11}	0.03914	2.14×10^{-10}	3.02×10^{-11}	3.34×10^{-11}	5.97×10^{-9}	3.02×10^{-11}
	1500	3×10^{-11}	0.002751	1.2×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	2000	3×10^{-11}	7.66×10^{-8}	4.94×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	2500	3×10^{-11}	9.7×10^{-10}	5.46×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
HPC2N	500	3×10^{-11}	7.72×10^{-6}	8.84×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	5.27×10^{-5}	3.02×10^{-11}
	1000	3×10^{-11}	5.84×10^{-6}	3×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	1500	3×10^{-11}	3.14×10^{-10}	1.46×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	2000	3×10^{-11}	1.6×10^{-10}	5.94×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	2500	3×10^{-11}	6.03×10^{-11}	4.94×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}

Table 8. Friedman test.

CGO	AOS	CSA	WOA	HGSWC	MPA	ChOA	CHMPAD
4.6667	2.2667	2.9333	8	5.2667	5.2667	6.4000	1.2000

5. Conclusions

In this paper, we proposed a new intelligent algorithm that combines the chimp optimization algorithm (ChOA) with the marine predators algorithm (MPA) to defeat the fog scheduling problem. The presented CHMPAD method was intended to boost the exploitation ability of the basic ChOA. The performance of CHMPAD was evaluated under different evaluation criteria and contrasted with the conventional ChOA and other existing metaheuristics, such as MPA, whale optimization algorithm (WOA), Henry gas solubility optimization based on comprehensive opposition-based learning (HGSWC) algorithms, chaos game optimization (CGO), atomic orbital search (AOS), and chameleon swarm algorithm (CSA). The experimental findings demonstrate the efficiency and consistency of the CHMPAD approach. In particular, the simulation findings reveal that CHMPAD can provide average makespan time improvements of 1.12–43.20% (for synthetic workloads), 1.00–43.43% (for NASA iPSC workloads), and 2.75–42.53% (for HPC2N workloads) over peer scheduling algorithms. Moreover, the results also indicate that our approach can boost fog computing's throughput. In the future, we plan to implement the CHMPAD scheduling algorithm in a real FC system. CHMPAD may also be improved further and extended to solve more complicated optimization problems, such as quadratic assignment, vehicle routing, assembly line balancing, and health care facility planning.

Author Contributions: Conceptualization, I.A., L.A., D.E., S.A.C. and M.A.E.; Methodology, I.A., L.A., D.E., S.A.C. and M.A.E.; Software, I.A., L.A., D.E. and M.A.E.; Validation, I.A., L.A., D.E. and M.A.E.; Formal Analysis, I.A., L.A., D.E. and M.A.E.; Investigation, I.A., L.A., D.E., S.A.C. and M.A.E.; Writing—original draft, I.A., L.A., D.E., S.A.C. and M.A.E.; Writing—review and editing, I.A., L.A., D.E., S.A.C. and M.A.E.; Visualization, I.A., L.A., D.E., S.A.C. and M.A.E.; Supervision, M.A.E.; Project administration, I.A., L.A., D.E., S.A.C. and M.A.E.; and Funding acquisition, S.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R239), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia

Acknowledgments: The authors would like to thank the support of the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University. This research project was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R239), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

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

References

1. Abualigah, L.; Diabat, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust. Comput.* **2021**, *24*, 205–223. [CrossRef]
2. 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. Humaniz. Comput.* **2019**, *10*, 3469–3479. [CrossRef]
3. Forestiero, A. Metaheuristic algorithm for anomaly detection in Internet of Things leveraging on a neural-driven multiagent system. *Knowl. Based Syst.* **2021**, *228*, 107241. [CrossRef]
4. Fu, J.S.; Liu, Y.; Chao, H.C.; Bhargava, B.K.; Zhang, Z.J. Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4519–4528. [CrossRef]
5. Darwish, A.; Hassanien, A.E.; Elhoseny, M.; Sangaiah, A.K.; Muhammad, K. The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: Opportunities, challenges, and open problems. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 4151–4166. [CrossRef]
6. Anuradha, M.; Jayasankar, T.; Prakash, N.; Sikkandar, M.Y.; Hemalakshmi, G.; Bharatiraja, C.; Britto, A.S.F. IoT enabled cancer prediction system to enhance the authentication and security using cloud computing. *Microprocess. Microsyst.* **2021**, *80*, 103301. [CrossRef]
7. Gill, S.S.; Tuli, S.; Xu, M.; Singh, I.; Singh, K.V.; Lindsay, D.; Tuli, S.; Smirnova, D.; Singh, M.; Jain, U.; et al. Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. *Internet Things* **2019**, *8*, 100118. [CrossRef]
8. Shanthan, B.; Kumar, A.D.V.; Govindrajan, E.; Arockian, L. Scheduling for internet of things applications on cloud: A review. *Imp. J. Interdiscip. Res.* **2017**, *3*, 1649–1653.
9. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A survey on the edge computing for the Internet of Things. *IEEE Access* **2017**, *6*, 6900–6919. [CrossRef]
10. Hossain, M.R.; Whaiduzzaman, M.; Barros, A.; Tuly, S.R.; Mahi, M.J.N.; Roy, S.; Fidge, C.; Buyya, R. A scheduling-based dynamic fog computing framework for augmenting resource utilization. *Simul. Model. Pract. Theory* **2021**, *111*, 102336. [CrossRef]
11. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. CCSA: Conscious neighborhood-based crow search algorithm for solving global optimization problems. *Appl. Soft Comput.* **2019**, *85*, 105583. [CrossRef]
12. Yi, S.; Li, C.; Li, Q. A survey of fog computing: Concepts, applications and issues. In Proceedings of the 2015 Workshop on Mobile Big Data, Hangzhou, China, 22–25 June 2015; pp. 37–42.
13. El Kafhali, S.; Salah, K. Efficient and dynamic scaling of fog nodes for IoT devices. *J. Supercomput.* **2017**, *73*, 5261–5284. [CrossRef]
14. Li, S.; Li, W.; Liu, H.; Sun, W. A Stackelberg Game Approach toward Migration of Enterprise Applications to the Cloud. *Mathematics* **2021**, *9*, 2348. [CrossRef]
15. Tan, K.; Zhang, Y.; Tong, X. Cloud extraction from Chinese high resolution satellite imagery by probabilistic latent semantic analysis and object-based machine learning. *Remote Sens.* **2016**, *8*, 963. [CrossRef]
16. Abdel-Basset, M.; El-Shahat, D.; Elhoseny, M.; Song, H. Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. *IEEE Internet Things J.* **2021**, *8*, 12638–12649. [CrossRef]
17. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
18. Attiya, I.; Elaziz, M.A.; Abualigah, L.; Nguyen, T.N.; Abd El-Latif, A.A. An Improved Hybrid Swarm Intelligence for Scheduling IoT Application Tasks in the Cloud. *IEEE Trans. Ind. Inform.* **2022**. [CrossRef]
19. Chen, Z.G.; Zhan, Z.H.; Lin, Y.; Gong, Y.J.; Gu, T.L.; Zhao, F.; Yuan, H.Q.; Chen, X.; Li, Q.; Zhang, J. Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. *IEEE Trans. Cybern.* **2018**, *49*, 2912–2926. [CrossRef]
20. Arri, H.S.; Khosa, R.S.; Jha, S.; Prashar, D.; Joshi, G.P.; Doo, I.C. Optimized Task Group Aggregation-Based Overflow Handling on Fog Computing Environment Using Neural Computing. *Mathematics* **2021**, *9*, 2522. [CrossRef]
21. Wang, S.; Zhao, T.; Pang, S. Task scheduling algorithm based on improved firework algorithm in fog computing. *IEEE Access* **2020**, *8*, 32385–32394. [CrossRef]
22. Lin, B.; Guo, W.; Xiong, N.; Chen, G.; Vasilakos, A.V.; Zhang, H. A pretreatment workflow scheduling approach for big data applications in multicloud environments. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 581–594. [CrossRef]
23. Forestiero, A.; Mastroianni, C.; Meo, M.; Papuzzo, G.; Sheikhalishahi, M. Hierarchical approach for green workload management in distributed data centers. In *European Conference on Parallel Processing*; Springer: Berlin, Germany, 2014; pp. 323–334.
24. Cheng, Y.; Xie, Y.; Wang, D.; Tao, F.; Ji, P. Manufacturing Services Scheduling With Supply–Demand Dual Dynamic Uncertainties Toward Industrial Internet Platforms. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2997–3010. [CrossRef]
25. Abd Elaziz, M.; Abualigah, L.; Ibrahim, R.A.; Attiya, I. IoT Workflow Scheduling Using Intelligent Arithmetic Optimization Algorithm in Fog Computing. *Comput. Intell. Neurosci.* **2021**, *2021*, 9114113. [CrossRef] [PubMed]
26. 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), Winterton, South Africa, 5–6 August 2019; pp. 1–6.

27. 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]
28. 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]
29. Tong, Z.; Chen, H.; Deng, X.; Li, K.; Li, K. A scheduling scheme in the cloud computing environment using deep Q-learning. *Inf. Sci.* **2020**, *512*, 1170–1191. [CrossRef]
30. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]
31. Khishe, M.; Mosavi, M. Classification of underwater acoustical dataset using neural network trained by Chimp Optimization Algorithm. *Appl. Acoust.* **2020**, *157*, 107005. [CrossRef]
32. Kaur, M.; Kaur, R.; Singh, N.; Dhiman, G. SChoA: An newly fusion of sine and cosine with chimp optimization algorithm for HLS of datapaths in digital filters and engineering applications. *Eng. Comput.* **2021**, 1–29. [CrossRef]
33. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
34. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization Algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
35. Abualigah, L.; Abd Elaziz, M.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
36. Biondi, G.; Franzoni, V. Discovering correlation indices for link prediction using differential evolution. *Mathematics* **2020**, *8*, 2097. [CrossRef]
37. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
38. Sahlol, A.T.; Yousri, D.; Ewees, A.A.; Al-Qaness, M.A.; Damasevicius, R.; Elaziz, M.A., COVID-19 image classification using deep features and fractional-order marine predators algorithm. *Sci. Rep.* **2020**, *10*, 15364. [CrossRef]
39. Eid, A.; Kamel, S.; Abualigah, L. Marine predators algorithm for optimal allocation of active and reactive power resources in distribution networks. *Neural. Comput. Appl.* **2021**, *33*, 14327–14355. [CrossRef]
40. Yousri, D.; Babu, T.S.; Beshr, E.; Eteiba, M.B.; Allam, D. A robust strategy based on marine predators algorithm for large scale photovoltaic array reconfiguration to mitigate the partial shading effect on the performance of PV system. *IEEE Access* **2020**, *8*, 112407–112426. [CrossRef]
41. Ramezani, M.; Bahmanyar, D.; Razmjooy, N. A new improved model of marine predator algorithm for optimization problems. *Arab J. Sci. Eng.* **2021**, *46*, 8803–8826. [CrossRef]
42. Attiya, I.; Zhang, X. D-Choices Scheduling: A Randomized Load Balancing Algorithm for Scheduling in the Cloud. *J. Comput. Theor. Nanosci.* **2017**, *14*, 4183–4190. [CrossRef]
43. Liu, H.; Ding, G.; Wang, B. Bare-bones particle swarm optimization with disruption operator. *Appl. Math. Comput.* **2014**, *238*, 106–122. [CrossRef]
44. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
45. Abd Elaziz, M.; 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]
46. Parallel Workloads Archive. Available online: http://www.cse.huji.ac.il/labs/parallel/workload/logs.html (accessed on 1 October 2021).
47. Attiya, I.; Zhang, X.; Yang, X. TCSA: A dynamic job scheduling algorithm for computational grids. In Proceedings of the 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI), Wuhan, China, 13–15 October 2016; pp. 408–412.
48. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
49. Abd Elaziz, M.; Attiya, I. An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing. *Artif. Intell. Rev.* **2021**, *54*, 3599–3637. [CrossRef]
50. Talatahari, S.; Azizi, M. Chaos Game Optimization: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2021**, *54*, 917–1004. [CrossRef]
51. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model.* **2021**, *93*, 657–683. [CrossRef]
52. Braik, M.S. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Syst. Appl.* **2021**, *174*, 114685. [CrossRef]