symmetry

MDPI

Article

Fog Computing Task Scheduling of Smart Community Based on Hybrid Ant Lion Optimizer

Fengqing Tian *, Donghua Zhang, Ying Yuan, Guangchun Fu, Xiaomin Li and Guanghua Chen

Henan Institute of Science and Technology, Xinxiang 453000, China; zdh18439027037@163.com (D.Z.); 15237613225@163.com (Y.Y.); fuguchun@hist.edu.cn (G.F.); xiaomin@hist.edu.cn (X.L.); chengh@hist.edu.cn (G.C.)
* Correspondence: fengqingtian@hist.edu.cn

Abstract: Due to the problem of large latency and energy consumption of fog computing in smart community applications, the fog computing task-scheduling method based on Hybrid Ant Lion Optimizer (HALO) is proposed in this paper. This method is based on the Ant Lion Optimizer (ALO. Firstly, chaotic mapping is adopted to initialize the population, and the quality of the initial population is improved; secondly, the Adaptive Random Wandering (ARW) method is designed to improve the solution efficiency; finally, the improved Dynamic Opposite Learning Crossover (DOLC) strategy is embedded in the generation-hopping stage of the ALO to enrich the diversity of the population and improve the optimization-seeking ability of ALO. HALO is used to optimize the scheduling scheme of fog computing tasks. The simulation experiments are conducted under different data task volumes, compared with several other task scheduling algorithms such as the original algorithm of ALO, Genetic Algorithm (GA), Whale Optimizer Algorithm (WOA) and Salp Swarm Algorithm (SSA). HALO has good initial population quality, fast convergence speed, and high optimization-seeking accuracy. The scheduling scheme obtained by the proposed method in this paper can effectively reduce the latency of the system and reduce the energy consumption of the system.

Keywords: smart community; fog computing; task scheduling; ant lion optimizer; latency; energy consumption

Citation: Tian, F.; Zhang, D.; Yuan, Y.; Fu, G.; Li, X.; Chen, G. Fog Computing Task Scheduling of Smart Community Based on Hybrid Ant Lion Optimizer. *Symmetry* **2023**, *15*, 2206. https://doi.org/10.3390/sym15122206

Academic Editors: Alexander Zaslavski and Tomohiro Inagaki

Received: 11 October 2023
Revised: 30 November 2023
Accepted: 15 December 2023
Published: 17 December 2023

1. Introduction

In recent years, the rapid development of the Internet of Things (IoT) has promoted the development of smart cities [1]. The smart community is one of the key elements as a basic component to realize a smart city. The IoT devices involved in smart communities are located at the edge of the Internet and are of diverse types. The number and the amount of data of IoT devices are growing exponentially year by year [2]. Therefore, the transmission latency and energy consumption in the cloud computing of a smart community has been highlighted directly. The bandwidth of a network has become a key factor in limiting the development of cloud computing. Then, fog computing was born [3]. Fog nodes have limited and different computing capabilities in the application of fog computing. How to coordinate and schedule the fog computing tasks between the fog nodes and IoT devices to reduce data transmission time and energy consumption and improve response speed [4], and enhance the quality of the user experience [5], has become the focus of research in fog computing.

Currently, a large number of fog computing task-scheduling algorithms have been proposed for the scheduling of fog computing tasks [6–8]. The hardware utilization efficiency of fog computing systems has been improved through task scheduling [9–12]. Verba et al. [13] proposed a real-time service model for community energy management, and cost-effective energy management was achieved for electricity by three scenarios for the energy consumption in smart community fog computing. In order to meet the

latency requirements of multiple services in the electric power IoT scenario, a BRT (Balanced initialization, Resource allocation, and Task allocation) algorithm was proposed by Niu et al. [14]. In the paper, the original problem was decomposed into three subproblems, and the computational resources were allocated by improved particle swarm optimization to minimize the service latency. Amit et al. [15] studied the latency decrease in the resource-constrained IoT devices in smart cities that require low-latency processing by a fog computing task-scheduling algorithm based on ant colony optimization. Dai et al. [16] improved the fog computing architecture for smart factories and reduced the task latency and kept the resource balance by improved genetic algorithm for the characteristics and needs of smart factories. Cen et al. [17] proposed a computing resource allocation method for smart distribution network fog computing devices to improve the adaptability of a distribution network automation system. The latency of the fog computing system by task scheduling was optimized in the literature [14–17], but the system energy consumption was not considered. Mohamed et al. [18] proposed an improved artificial ecosystem algorithm for computing task scheduling in order to improve the performance of IoT systems. The improved artificial ecosystem algorithm was evaluated using synthetic and real datasets of different sizes, and the method worked best in reducing latency and energy consumption. Wang et al. [19] investigated the limited computing resources and high energy consumption of terminal devices in smart factories based on a hybrid heuristic algorithm. The tasks of terminal devices were able to be processed in real time and efficiently with the method in the paper. Abdel-Basse et al. [20] proposed a fog computing task-scheduling algorithm based on the Modified Marine Predator Algorithm (MMPA) to solve the problem of high latency and energy consumption of fog computing in IoT applications. The latency and energy consumption of the system were reduced effectively in the literature [18–20], but the priority of IoT task processing was not considered.

Hussein et al. [21] proposed two different scheduling algorithms to improve the quality of service of fog computing. The load of IoT tasks on fog nodes was balanced effectively by ant colony algorithm (ACO) and particle swarm optimization in which the communication cost and response time were considered. Rafique et al. [22] proposed a novel bio-inspired hybrid algorithm (NBIHA) task-scheduling algorithm based on modified particle swarm optimization (MPSO) and modified cat swarm optimization (MCSO) to reduce the response time of IoT fog computing system and improve the fog computing resource utilization. Movahedi et al. [23] investigated a task-scheduling algorithm based on the Opposition-based Chaotic Whale Optimization Algorithm (OppoCWOA), and the latency and energy consumption of IoT tasks were decreased by having fewer offloading requests and fog node resource constraints in the smart city. Xu et al. [24] proposed a task-scheduling algorithm based on slackness and the ant colony algorithm to reduce the energy consumption for scheduling complex tasks with priority constraints in IoT applications, but the latency of fog computing was not considered.

Table 1 compares the different works in terms of application scenario, optimization method, optimization objective and priority. To sum up, the existing fog computing scheduling research mainly focused on manufacturing [6,16,19] and smart grids [13,14,17]. There were few research papers on fog computing task-scheduling in smart communities. The smart communities have characteristics that distinguish them from other smart subjects: (1) The smart communities include smart grids [25,26], smart water networks [27], smart parking [28,29], smart buildings [30] and other subjects, in which a variety of IoT devices are deployed, and different devices with different requirements of latency, energy consumption and different priorities of IoT tasks; (2) The smart communities are non-profit organizations, which invest less in the deployment of fog nodes than other subjects, and have insufficient fog node resources; (3) Irrational task scheduling can lead to some nodes running at full capacity, and the system will generate a lot of latency and energy consumption, which does not meet the requirements of delay-sensitive tasks in smart communities. Therefore, the traditional optimization algorithms have no way to meet the fog computing task-scheduling requirements in smart communities. Ant Lion Optimizer [31] (ALO) is a new intelligent

algorithm proposed by Mirjalili, an Australian scholar, in 2015. Given that ALO has the advantages of diverse populations, fast convergence, and few adjustment parameters [32], in this paper, a fog computing task-scheduling algorithm based on the improved Ant Lion Optimizer is proposed by exploiting the features of intelligent communities and the Ant Lion Optimizer. The main contribution of this paper is embodied in three aspects.

(1) Modeling smart community fog computing architecture with the specifics of a smart community;
(2) Chaotic mapping is adopted to initialize the population, and the quality of the initial population is enhanced;
(3) The random walk is adopted to improve the solving efficiency of ALO;
(4) The improved DOLC strategy is embedded in the generation-hopping process of ALO, the population diversity is enriched and the ability to jump out of the local optimum is enhanced.

Table 1. Task scheduling in fog computing.

Ref.	Application Scenario	Optimization Method	Optimization Objective	Priority
[13]	smart microgrids	GA	service delivery and longevity	No
[14]	smart grids	BRT	latency	No
[15]	-	SACO	latency	No
[16]	smart factory	IDGSA	latency and resource balance	No
[17]	smart distribution transformer area	a sequential logic-based evolution algorithm	configured capacities	No
[18]	-	SSA	latency and energy	No
[19]	smart factory	Hybrid Heuristic Algorithm	latency and energy	No
[20]	-	Energy-Aware Marine Predators Algorithm	latency and energy	No
[21]	-	ACO	response time and load balance	Yes
[22]	-	NBIHA	efficient resource utilization	No
[23]	smart city	OppoCWOA	latency and energy	No
[24]	-	LBP-ACS	energy	Yes
Our work	smart community	HALO	latency and energy	Yes

2. Fog Computing Scheduling Model for Smart Communities

2.1. Fog Computing Architecture Model

The fog computing architecture model with three main layers for smart communities is established based on the current research. The system architecture model consists of an IoT layer, fog layer and cloud layer, as shown in Figure 1.

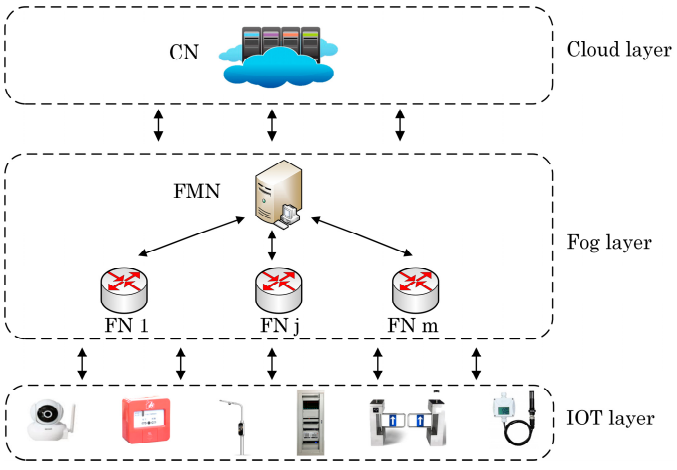


Figure 1. Fog computing architecture model for smart communities.

IoT layer: The IoT devices are located on this layer and make up the smart subjects in the smart community (power system monitoring, smart street lights, community access control, environmental monitoring, fire alarm, etc.), which have very limited computing power and need to upload data to the fog layer or cloud layer for processing.

Fog layer: The fog layer is located between the IoT layer and cloud layer and is composed of fog nodes and fog management nodes mainly. The fog nodes are distributed in different areas near the IoT devices and have certain data processing capability, but the computing resources are limited and are mainly used to handle delay-sensitive tasks (power system fault alarm, fire alarm, dangerous behavior warning) in the fog computing system of the smart community. The fog management node is able to maintain fog nodes information, fully dispatch fog computing layer resources, and improve the equipment utilization rate of the fog computing system.

Cloud layer: Located at the top layer of the network architecture, it consists of cloud servers with scalable computing capabilities, which are used to store and analyze core data and provide high-performance computing services for smart community tasks. The cloud layer cannot guarantee the quality of service for all applications due to network bandwidth and is mainly used for computing-intensive tasks with low real-time requirements.

The processing of kinds of tasks can be well solved by fog-layer layering. But, due to the limited resources of fog nodes in smart communities, unreasonable task scheduling will lead to some nodes running at full load, and the system will generate a lot of latency and energy consumption, which does not meet the requirements of delay-sensitive tasks. The task-scheduling problem of the fog computing layer still needs to be solved. This paper mainly studies the task scheduling of the fog computing layer for smart communities.

2.2. Fog Computing Task-Scheduling Model

The fog computing task-scheduling process of smart community can be described as follows: n IoT tasks generated in a certain period in the smart community are mapped to m fog nodes by executing a task scheduler.

The fog computing task-scheduling process is shown in Figure 2. IoT devices send task-execution requests and device information to the fog management node; fog nodes send the current state information of the nodes to the fog management node; the fog management node is responsible for scheduling and distributing fog computing tasks, and sends the result to IoT devices and fog nodes for execution; IoT devices communicate with the corresponding fog nodes to complete computing tasks.

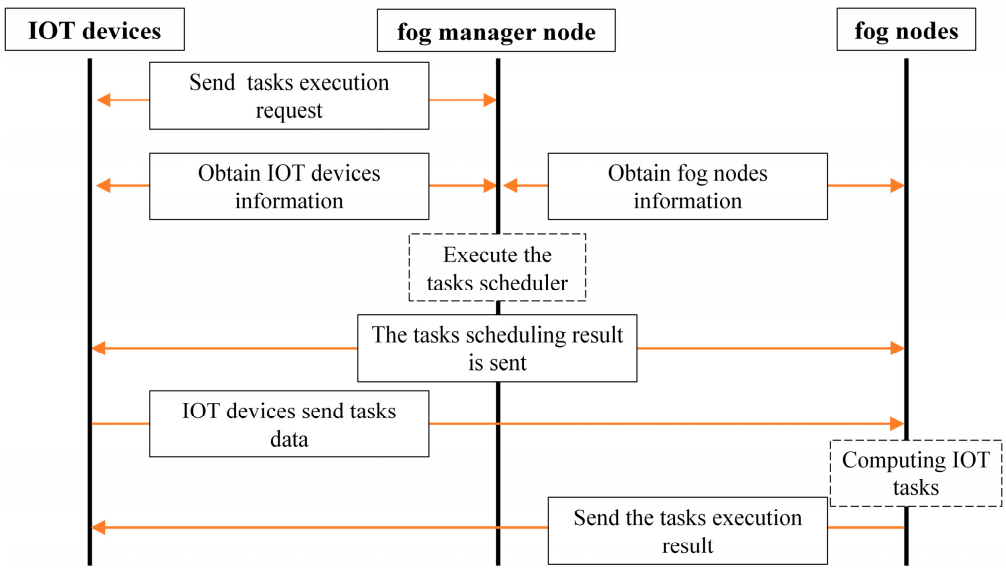


Figure 2. Fog computing task-scheduling process.

The parameters of the mathematical model for the fog computing tasks scheduling in smart communities are shown in Table 2.

Table 2. Fog computing task-scheduling parameters.

Parameters	Meaning
S	Number of IoT devices
s_o	The oth IoT device
N	Tasks collection $N = \{V_1 \dots V_i \dots V_n\}$
TA_i	The ith task
D_i	The amount of data in TA_i (Mb)
P_i	Transmission power of IoT devices of TA_i (W)
DM_i	The maximum delay time of TA_i (s)
TT_i	The type of tasks for TA_i
DU_i	The degree of urgency for TA_i
θ_i	The computational density of TA_i (cycles/bit)
F	Fog nodes collection $F = \left\{f_1 \dots f_j \dots f_m\right\}$
f_j	The jth fog node
PC_j	The calculated power of f_j (W)
C_j	The computing capacity of f_j (G cycles/bit)
SC_j	Storage capacity of f_j (Mb)
SF_j	The state of f_j (0 or 1)
B_j	The bandwidth of f_j (MHz)

The latency of a task in a fog computing system consists of the transmission time of the task $T_{ij}^{transtime}$ and the execution time of the task $T_{ij}^{executes}$. The transmission time of the ith task to the jth fog node is expressed as:

$$T_{ij}^{transtime} = \frac{D_i}{r_{ij}}, \tag{1}$$

where r_{ij} denotes the transmission rate [33] of the ith task to the jth fog node, expressed as:

$$r_{ij} = B_j \times log_2\left(1 + \frac{h_{ij} \times P_i}{\sigma^2}\right), \tag{2}$$

where P_i denotes the transmission power of the IoT device, h_{ij} is the channel power gain, and σ is the noise power.

The computation time for the ith task is expressed as:

$$T_{ij}^{executes} = \frac{D_i \times \theta_i}{C_j} \tag{3}$$

The latency of the ith task is expressed as:

$$T_{ij} = T_{ij}^{transtime} + T_{ij}^{executes} \tag{4}$$

The energy consumption for the ith task consists of the transmission energy consumption and the computational energy consumption, respectively, are expressed as:

$$E_{ij}^{transtime} = T_{ij}^{transtime} \times P_i \tag{5}$$

$$E_{ij}^{executes} = T_{ij}^{executes} \times PC_j \tag{6}$$

where P_i indicates the IoT device transmission power of the ith task, and PC_j indicates the computed power of the jth fog node.

The energy consumption of the ith task is expressed as:

$$E_{ij} = E_{ij}^{transtime} + E_{ij}^{executes} \tag{7}$$

The total latency and total energy consumption of the system are expressed as:

$$T_{total} = \sum_{i=1}^{n} \sum_{j=1}^{m} S_{ij} \times T_{ij} \tag{8}$$

$$E_{total} = \sum_{i=1}^{n} \sum_{j=1}^{m} S_{ij} \times E_{ij} \tag{9}$$

Since the two optimization objectives of latency and energy consumption differ significantly in value, they are normalized, and this paper uses the MIN–MAX normalization process as follows:

$$obj1 = \frac{T_{total} - T_{MIN}}{T_{MAX} - T_{MIN}} \tag{10}$$

$$obj2 = \frac{E_{total} - E_{MIN}}{E_{MAX} - E_{MIN}} \tag{11}$$

$obj1$ is the latency after normalization and $obj2$ is the energy consumption after normalization, and T_{MIN}, T_{MAX}, E_{MIN}, E_{MAX} are the minimum and maximum values of system latency and energy consumption in the ideal state.

In order to achieve a smart community for $obj1$ and $obj2$, two objectives of different weights in the practical application, this paper uses a linear weighting method to establish the objective function, as follows, in Equation (12):

$$\begin{aligned} & U = min(\omega \times obj1 + (1-\omega) \times obj2) \\ & s.t. C1: \sum_{j=1}^{m} S_{ij} = 1, i = 1, 2, \ldots, n \\ & \quad C2: \sum_{i=1}^{n} S_{ij} \times D_i < SC_j, j = 1, 2, \ldots, m \\ & \quad C3: \sum_{i=1}^{n} S_{ij} \times \theta_i < C_j, j = 1, 2, \ldots, m \\ & \quad C4: T_{ij} < DM_i, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m \\ & \quad C5: S_{ij} \in \{0, 1\}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m \\ & \quad C6: DU_{i-1} < DU_i, i = 1, 2, \ldots, n \\ & \quad C7: 0 < \omega < 1 \end{aligned} \tag{12}$$

$C1$, $C5$ indicate that tasks can only be executed on a fog node, $C2$, $C3$ indicate that a set of tasks executed by the fog node cannot consume more resources than the storage and computational resources of that node, $C4$ indicates that the execution time of a task cannot be greater than the maximum completion time, $C6$ indicates that the priority of the ith task is greater than that of the $(i-1)$th task, and $C7$ indicates the degree of user preference for latency and energy consumption.

3. Traditional Ant Lion Optimizer

The ant lion optimizer achieves global optimization by simulating the idea of ant lions capturing ants in nature. Firstly, the ant lion optimizer randomly generates the position of the initial population in the search space, denoted as:

$$X_i = r1 \times (ub_i - lb_i) + lb_i, i = 1, 2, \ldots, n \tag{13}$$

$r1$ is a random number between [0, 1], where ub_i, lb_i are the upper and lower bounds of the search space and X_i denotes the ith dimensional variable of the population individuals.

Ants wander randomly in the search space with different random wandering steps, denoted as:

$$X(Iter) = \left[0, cumsum(s(1)) \ldots cumsum(s(Iter)) \ldots cumsum(s(Max\_Iter))\right] \quad (14)$$

$X(Iter)$ is the set of steps in which the ant randomly wanders, cumsum is the calculation of the cumulative sum, and *Iter* and *Max_Iter* are the current and maximum number of iterations, respectively;

$s(Iter)$ is the random function of -1 and 1, denoted as:

$$s(Iter) = \begin{cases} 1, r2 > 0.5 \\ -1, r2 < 0.5 \end{cases} \quad (15)$$

$r1$ is a random number between [0, 1].

A normalization process is applied to each dimension of $X(Iter)$ so that it is in the search space to generate new ants, denoted as:

$$X_i^{Iter} = \left(\frac{(X_i^{Iter} - \min(X(Iter)))}{(\max(X(Iter)) - \min(X(Iter)))} \times (ub_i^{Iter} - lb_i^{Iter}) + lb_i^{Iter} \right) \quad (16)$$

a_i and b_i are the minimum and maximum values of ants randomly wandering, respectively, and ub_i^{Iter} and lb_i^{Iter} are the upper and lower boundaries of the search space of the ith dimension of an individual ant at the $Iter$th generation, respectively.

The search space of ants randomly wandering is related to the behavior of ants captured by the ant lion, around which the trap is first constructed as follows:

$$\begin{cases} lb_i^{Iter} = PAL_y^{Iter} + c^{Iter} \\ ub_i^{Iter} = PAL_y^{Iter} + d^{Iter}, \end{cases} \quad (17)$$

PAL_y^{Iter} indicates the position of the $Iter$th generation of the yth ant lion, and c^{Iter} and d^{Iter} indicate the minimum and maximum values of the $Iter$th generation of all variables, respectively.

Secondly, as the number of iterations increases, the scope of the search space in which the ants randomly roam diminishes, denoted as:

$$\begin{cases} c^{Iter} = \frac{c^{Iter}}{I} \\ d^{Iter} = \frac{d^{Iter}}{I} \end{cases} \quad (18)$$

$\mathrm{I} = 10^w(Iter/Max\_Iter)$ indicates the ratio of the current algebra to the largest algebra, and w is a constant.

Finally, two approaches were used to jointly determine the random wandering of the ants; the ants randomly wandered around the ant lions chosen by the roulette strategy; and the ants randomly wandered around the elite ant lions, elite ant lions being the ant lions with the best fitness values during each iteration of the process, denoted as:

$$PA_k^{Iter} = \frac{R_A^{Iter} + R_E^{Iter}}{2} \quad (19)$$

R_A^{Iter}, R_E^{Iter} are separately the random wanderings of the ant lion and elite ant lion selected by the $Iter$th generation of ants around the roulette wheel.

When the ant has a better fitness value than the ant lion, the ant lion captures the ant and the ant lion updates to the location of the captured ant, as follows:

$$PAL_y^{Iter} = PA_k^{Iter} \; if \left(PA_k^{Iter}\right) > U\left(PAL_y^{Iter}\right) \quad (20)$$

PAL_y^{Iter} indicates the position of the $Iter$th generation of the yth ant lion, and PA_k^{Iter} indicates the location of the $Iter$th generation of the kth ant.

4. HALO

As mentioned earlier, the fog computing task scheduling in the smart community is a high-dimensional discrete problem, and the Ant Lion Optimizer has poor initial population quality, slow iteration speed and easily falls into a local optimum when solving complex high-latitude problems. This paper makes the following improvements to address the above problems of the Ant Lion Optimizer.

4.1. Chaotic Mapping Initialization

The Ant Lion Optimizer generates initial populations randomly in the search space, resulting in uneven distribution of initial populations in the search space [34]; the poor ergodicity of the solutions affects the pre-searching ability of the Ant Lion Optimizer. Chaotic mapping has the characteristics of ergodicity, regularity and orderliness, which can improve the diversity of the initial population of the algorithm [35–37].

The Circle Chaos Map (CCM) has chaotic properties such as unpredictability and non-periodicity and can generate uniformly distributed random numbers in the range of [0, 1].

The Circle Chaos Map can be expressed as

$$x_{i+1} = mod\left(x_i + 0.2 - \left(\frac{0.5}{2\pi}\right) \times \sin(2\pi \times x_i), 1\right) \tag{21}$$

Figure 3a,b show the distribution of random and circle chaos mapping separately; it can be seen that circle chaos mapping has a more uniform and ergodic distribution of points compared to random distribution. By introducing the circle chaos mapping into the ALO algorithm, the initial population is more evenly distributed in the search space, which enriches the population diversity and improves the quality of the initial solution of the algorithm.

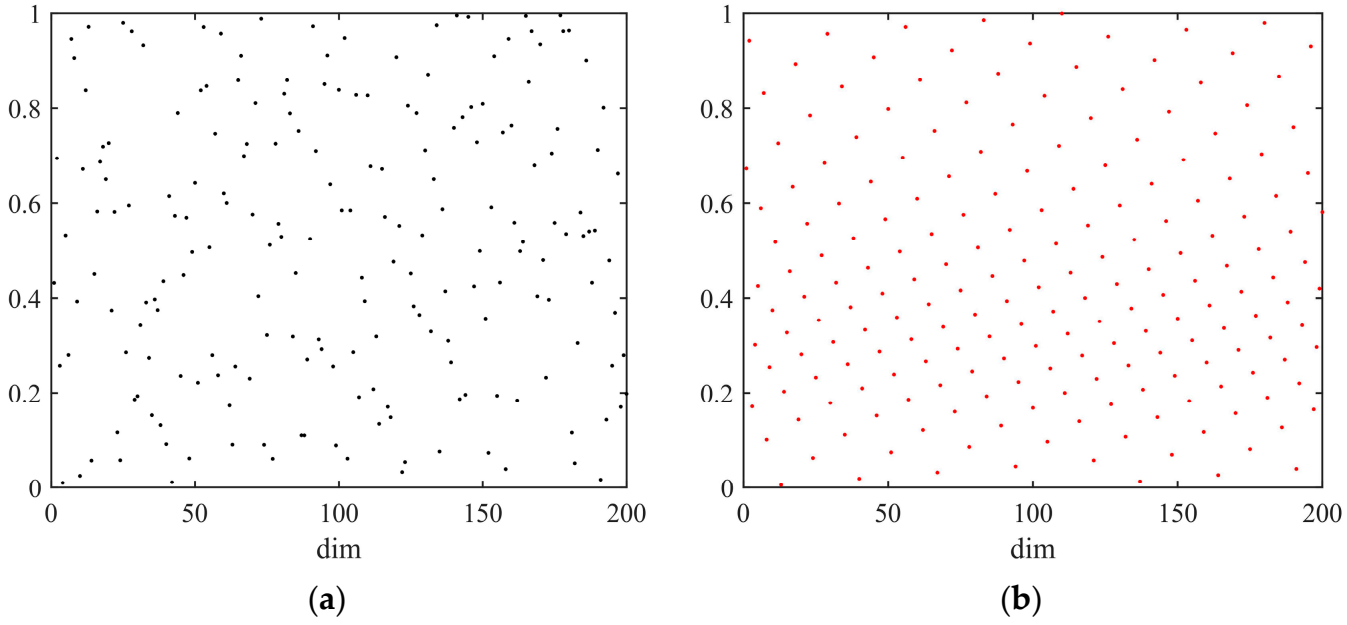


Figure 3. Scatter plot comparison: (**a**) Scatterplot of random distribution; (**b**) Circle chaos mapping scatter plot.

Suppose $X = (X_1, \ldots, X_i, \ldots, X_n)$ is an n-dimensional individual in the population. Apply the circle chaos mapping to the population initialization, denoted as

$$X_i = x_i \times (ub_i - lb_i) + lb_i, i = 1, 2, \ldots, n \tag{22}$$

where $ub_i$and lb_i are the upper and lower bounds of the search space, $x_i$is the chaos factor of the circular chaos mapping and X_i denotes the ith dimensional variable of the population individuals. Mapping the chaos factor x_i onto the solution space of the fog computing task scheduling according to Equation (22) replaces the original random distribution, avoids duplicate values, and improves the quality of the initial population.

4.2. Adaptive Random Wandering

The Ant Lion Optimizer uses the cumulative sum method to simulate the random wandering of ants during each iteration according to Equations (14) and (15). The cumulative sum method ensures the superiority-seeking ability of the ants, but the excessive use of this technique increases the running time and complexity of the algorithm during the iteration process and inevitably generates duplicate values. In order to improve the solving efficiency of the Ant Lion Optimizer and to guarantee the merit-seeking ability of the ants in the search space, this paper designs an adaptive random walk (ARW) to simulate the results of the ant's random walk T times; during each iteration, randomized wandering results are generated directly according to Equations (23) and (24), avoiding the complexity of cumulative sums:

$$X(Iter) = sa \times (r3 - \gamma) \tag{23}$$

$X(Iter)$ is the set of steps for an ant to wander randomly, $r3$ is a random number between [0, 1], γ is the conditioning parameter, and sa is the adaptive factor, denoted as:

$$sa = Max\_Iter - \sin\left(\frac{Iter \times \pi}{4 \times Max\_Iter}\right) \times Iter \tag{24}$$

$Iter$ and Max_Iter are the current and maximum number of iterations, respectively.

Comparison of the two wandering methods is shown in Figure 4, where Figure 4a shows the results of ants randomly wandering according to ALO after 200 iterations. Figure 4b shows the result of the ant after 200 iterations according to the ARW proposed in this paper.

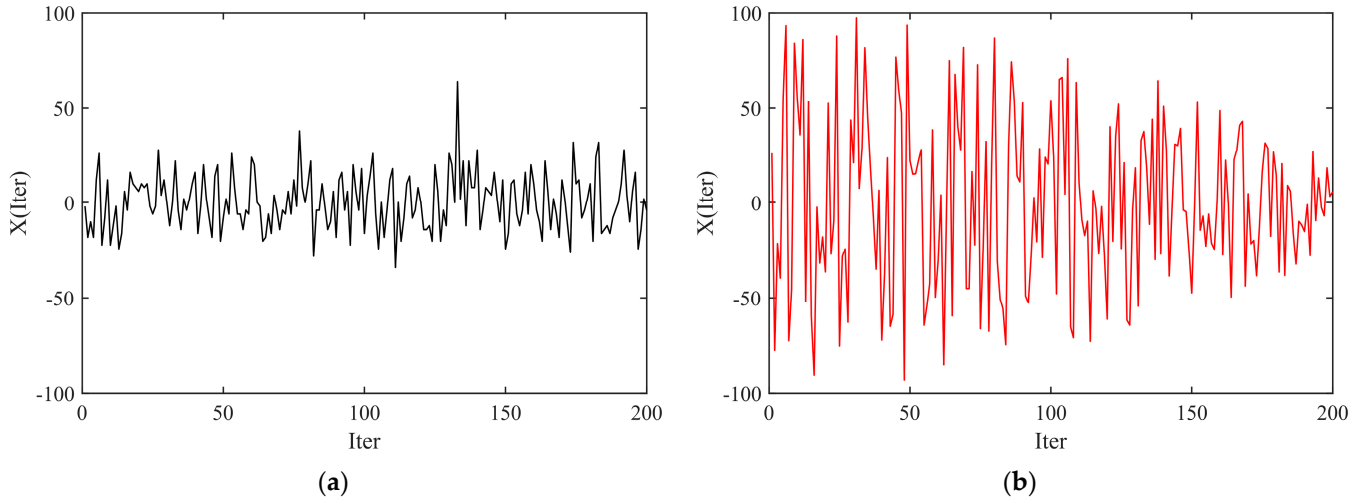


Figure 4. Wandering mode comparison: (**a**) Random wandering; (**b**) Adaptive random wandering.

From Figure 4a, it can be seen that the range of random wandering in the search space is not sufficient, and the phenomenon of duplicate values exists regardless of any stage of the algorithm iteration, and the diversity after wandering is poor. Figure 4b shows that,

due to the use of ARW strategy, through the control of *sa*, the wandering results of the ants maintain the corresponding characteristics; in the iteration of the algorithm in the early stage, the ants use ARW in the solution space; the search range is wider compared to the random wandering, wandering after the location diversity, in favor of the algorithm's global search for the optimal, and in the iteration of the algorithm in the late stage, through the adaptive factor of the control of the ants, the ants of the random wandering results gradually reduced, gradually approaching the optimal solution, in favor of the algorithm's global search for optimality. In the late iteration of the algorithm, through the control of the adaptive factor, the result of the ant's random walk gradually decreases and approaches the optimal solution, which is conducive to the local optimization of the algorithm in the late stage.

4.3. Improving Dynamic Oppositional Learning Strategies

Dynamic Oppositional Learning (DOL) is that Xu et al. [38] proposed an improved version for the problem of unavoidable convergence to locally optimal positions existing in the space from the current number to the opposite number for Oppositional Learning (OBL) [39]. DOL has the features of search space asymmetry and random dynamic adjustment of the number of opposites and is commonly used in the generation-hopping process of the algorithm to improve the ability of the algorithm to jump out of the local optimum, denoted as:

$$X^O = lb + ub - X \tag{25}$$

$$X^{RO} = r4 \times X^O \tag{26}$$

$$X^{DO} = X + \varphi \times r5 \times (X^{RO} - X) \tag{27}$$

where $r4$, $r5$ are random numbers between [0, 1], X^O is the dyadic number of X in the search space, X^{RO} is the asymmetric search space controlled by random numbers, and X^{DO} is the dynamic dyadic number in the space of X and X^{RO}. φ is the weighting factor used to balance the region and diversity of the search space.

Expanding the dynamic dyadic numbers in n dimensional space, then, the dynamic dyadic points are expressed as:

$$X_i{}^O = lb_i + ub_i - X_i, i = 1, 2, \ldots, n \tag{28}$$

$$X_i{}^{RO} = r6 \times X_i{}^O, i = 1, 2, \ldots, n \tag{29}$$

$$X_i{}^{DO} = X_i + \varphi \times r7 \times (X_i{}^{RO} - X_i), i = 1, 2, \ldots, n \tag{30}$$

where $r6$ and $r7$ are random numbers between [0, 1], $X_i{}^O$ is the opposition point of X_i in the search space, $X_i{}^{RO}$ is the asymmetric search space controlled by random numbers, and $X_i{}^{DO}$ is the dynamic opposition point in the space of X_i and $X_i{}^{RO}$.

The asymmetric search space of DOL applied in the Ant Lion Optimizer [32] is shown in Figure 5. The search space for X^{DO} is located between X^{RO} and X [38]. Where (a) is the location distribution of X^{RO}, (b) is the search space of X^{DO} when the value of X^{RO} is smaller than X^O, (c) is the search space of X^{DO} when the value of X^{RO} is greater than X^O, and (d) is the search space of X^{DO} when the value of X^{RO} exceeds the boundary of the Ant Lion trap. To prevent $X_i{}^{DO}$ from exceeding the search space, it is necessary to perform a boundary check according to Equation (31), as follows:

$$X_i{}^{DO} = r8 \times (ub_i - lb_i) + lb_i, i = 1, 2, \ldots, n \tag{31}$$

where $r8$ is a random number between [0, 1], and ub_i and lb_i are the upper and lower bounds of the search space.

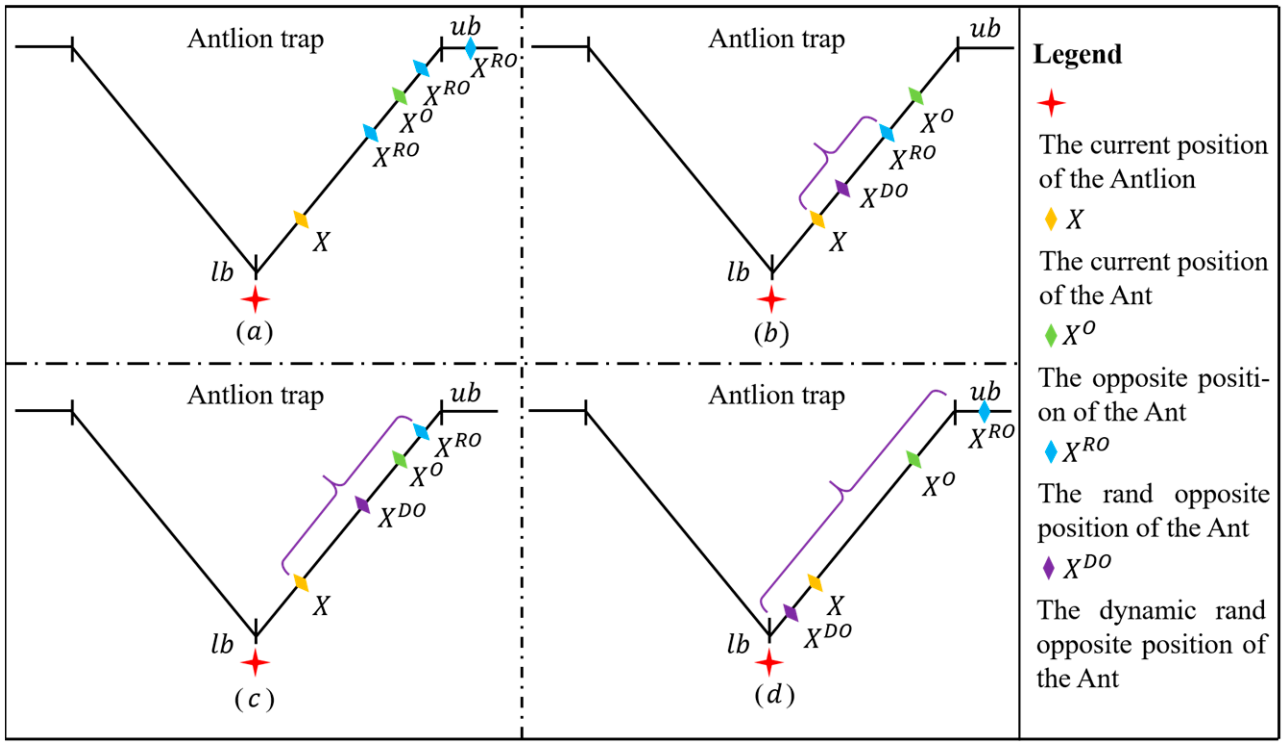


Figure 5. DOL asymmetric search space. (**a**) The location distribution of X^{RO}; (**b**) The search space of X^{DO} when the value of X^{RO} is smaller than X^{O}; (**c**) The search space of X^{DO} when the value of X^{RO} is greater than X^{O}; (**d**) The search space of X^{DO} when the value of X^{RO} exceeds the boundary of the Ant Lion trap.

In order to further improve the performance of the DOL strategy in the high-dimensional discrete problem of smart community fog computing, this paper embeds the dynamic oppositional learning crossover (DOLC) strategy designed in this paper in the generation-hopping process of the ant-lion algorithm. Firstly, the Ant Lion Optimizer generates new ant $DOLPA_{\mathrm{i}}{}^{Iter}$ by the DOL strategy. Secondly, the original individuals and $DOLPA_{\mathrm{i}}{}^{Iter}$ as parents are subjected to a two-point crossover process to generate new offspring, CPA_1^{Iter} and CPA_2^{Iter}; finally, the better-adapted individual ants are selected based on the idea of greed $DOLCPA_{\mathrm{i}}{}^{Iter}$. After solving the dynamic opposite solution, DOLC generates offspring individuals by fusing the superior genes of two individuals, which in turn enriches the population diversity and improves the ability of the algorithm to jump out of local optima. A jump rate factor jump is introduced to prevent the population from staying in a local optimum due to overuse of the dynamic pairwise crossover technique.

4.4. Coding Method

Fog computing task scheduling in a smart community is a discrete problem, and this paper adopts a real number encoding approach. Taking the location of ants in the Ant Lion Optimizer as an example, the location of each ant can be represented by an n dimensional array, and each ant represents a task-scheduling scheme, as shown in Figure 6.

3	7	10	10	14	7	11

Figure 6. Ant_k^{Iter} Example of coding.

Ant_k^{Iter} is the location of the kth ant of the $Iter$th generation, the location of the ants is represented as an n-dimensional array corresponding to n IoT tasks, and the value of each dimension of the array represents the assignment of the corresponding task to the fog node f_j(in the interval [1,m]). Figure 6 shows a simple coding example

$Ant_k^{Iter} = \{3, 7, 10, 10, 14, 7, 11\}$, which indicates that task 1 is assigned to fog node 3, task 2 and task 6 are assigned to fog node 7, task 3 and task 4 are assigned to fog node 10, task 5 is assigned to fog node 14, and task 7 is assigned to fog node 11.

4.5. Fog Computing Task Scheduling Algorithmic Process

In this section, the steps of HALO are summarized and the HALO flowchart is drawn, as shown in Figure 7. (*Iter* denotes the current iteration number, *Max_Iter* denotes the maximum iteration number) .

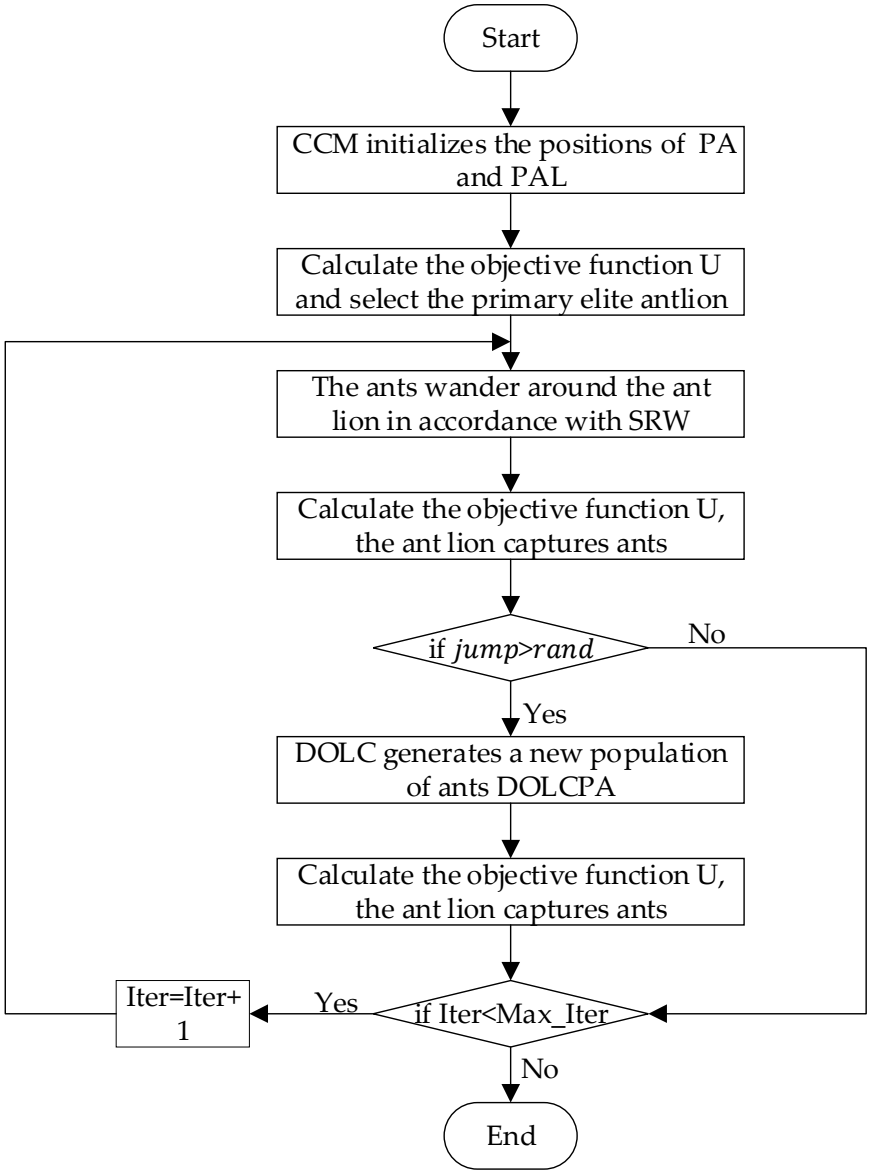


Figure 7. HALO flow chart.

5. Experimental Studies

5.1. Experimental Environment and Data Set

In order to verify the effectiveness of HALO in the scheduling of fog computing tasks for smart communities, a lot of research has been conducted for two objectives: latency and energy consumption in the fog computing environment. The experimental environment in this paper is Windows 10, 64-bit operating system, Intel(R) Celeron(R) CPU G3900, 8 GB memory; all programs run in MATLAB2018b software, and each experiment is repeated 20 times independently. Taking the example of the Henan Xinxiang Shi He Fu Smart

Community and synthesizing nine data sets (Task Data) with different numbers of IoT tasks (Task 50~Task 450), the main parameters are shown in Table 3. In all experiments, it is assumed that all fog nodes have been deployed and can cover all IoT devices.

Table 3. Description of experimental parameters.

Parameters	Description	Parameter Value
n	Number of tasks	50~450
D_i	Amount of data for the TA_i	1~5 M
DU_i	The degree of urgency the TA_i	First, Second, Third
DM_i	Maximum delay time of the TA_i	0.5~1 s
θ_i	Computational density of TA_i	100~200
P_i	Transmission power of IoT devices	0.1~0.5 W
m	Number of fog nodes	15
C_j	Fog node computing capacity	2~4 G cycles/s
B_j	Fog node bandwidth	2000 MHz
CP_j	Fog node calculated power	5~15 W

Each dataset generates 15 fog nodes with different configurations, and the slowest and fastest fog nodes have computational capacities of 2 and 4 G cycles/s, respectively. Each IoT task in the experiments is independent with different task priorities. The IoT task data size of each dataset ranges from 1 to 5 M, and IoT tasks of different urgency are randomly generated.

5.2. Evaluation Indicators

In order to evaluate the effectiveness of the HALO algorithm proposed in this paper in the fog computing system of smart communities, this paper evaluates the following metrics.

$Mean_T^O$ and $Mean_E^O$ are the average latency and average energy consumption results, respectively, of the original task-scheduling algorithm after running it independently 20 times, and are expressed in Equations (32) and (33).

$$Mean_T^O = \left(\sum_{fr=1}^{20} T_{total}^O(fr) \right) \div 20 \tag{32}$$

$$Mean_E^O = \left(\sum_{fr=1}^{20} E_{total}^O(fr) \right) \div 20 \tag{33}$$

$T_{total}^O(fr)$ and $E_{total}^O(fr)$ are the latency and energy consumption results after the frth run of the original task-scheduling algorithm, respectively, as calculated by Equations (8) and (9).

$Mean_T^C$ and $Mean_E^C$ are the average latency and average energy consumption results of the current task-scheduling algorithm after running it independently 20 times, respectively, and are represented by Equations (34) and (35).

$$Mean_T^C = \left(\sum_{fr=1}^{20} T_{total}^C(fr) \right) \div 20 \tag{34}$$

$$Mean_E^C = \left(\sum_{fr=1}^{20} E_{total}^C(fr) \right) \div 20 \tag{35}$$

$T_{total}^C(fr)$ and $E_{total}^C(fr)$ are the latency and energy consumption results after the frth run of the current task-scheduling algorithm and are calculated by Equations (8) and (9).

PIR_T and PIR_E are the average latency and average energy consumption improvement rates of the current task-scheduling algorithm compared to the original task-scheduling algorithm after 20 independent runs and are represented by Equations (36) and (37).

$$PIR_T(\%) = \left(\frac{Mean_T^O - Mean_T^C}{Mean_T^C} \right) \times 100 \tag{36}$$

$$PIR_E(\%) = \left(\frac{Mean_E^O - Mean_E^C}{Mean_E^C} \right) \times 100 \tag{37}$$

$Mean_T^O$ and $Mean_E^O$ are the average latency and average energy consumption results of the original task-scheduling algorithm, and $Mean_T^C$ and $Mean_E^C$ are the average latency and average energy consumption results of the current task-scheduling algorithm.

5.3. Sensitivity Analysis

This paper uses linear weighting to construct the objective function as Equation (12); the different values of ω determine the optimization weights of the objective. In this section, simulation experiments are conducted for HALO algorithms under different ω to obtain the optimal ω solutions to balance the weights of the optimization objective in the objective function under the same experimental environment.

In the set case, the data set with 200 tasks is selected for the experiment, the population size is set to 40, the maximum number of iterations is 200 [40], the value of ω is from 0.1 to 0.9, and the experiment is run 20 times independently to take the average value. The results are shown in Table 4.

Table 4. Analysis of optimization target weights.

HALO	ω = 0.1	ω = 0.2	ω = 0.3	ω = 0.4	ω = 0.5	ω = 0.6	ω = 0.7	ω = 0.8	ω = 0.9
*obj*1	0.206	0.199	0.196	0.194	0.193	0.191	0.190	0.187	0.184
*obj*2	0.234	0.236	0.241	0.243	0.244	0.246	0.249	0.254	0.256

Figure 8 shows the average values of *obj*1 and *obj*2 for different weighting factors. It can be seen that with the increasing of ω, *obj*1 shows a decreasing trend and *obj*2 shows an increasing trend. By adjusting the value of ω, the learning of different objectives can be achieved. As ω increases, the importance of *obj*1 increases in the system, *obj*2 decreases, and when $\omega = 0.4$, the two optimization objectives reach the equilibrium optimum.

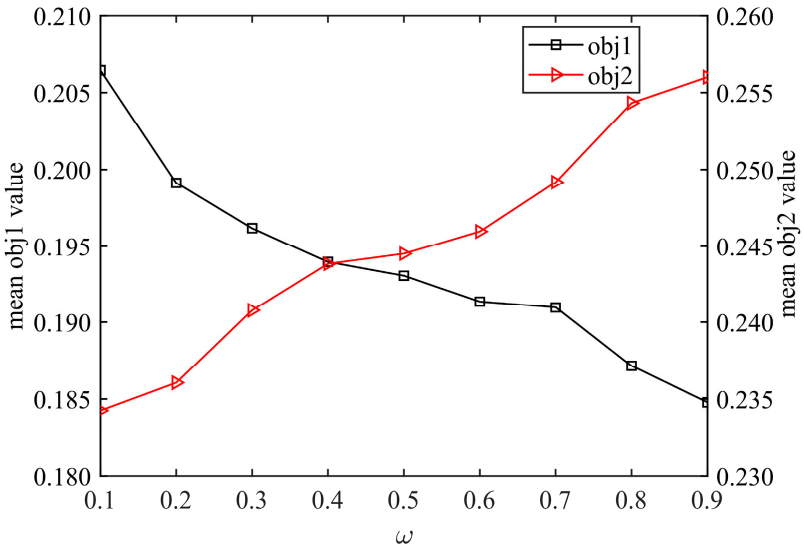


Figure 8. *obj*1 and *obj*2 under different ω.

5.4. Analysis of the Effectiveness of the Proposed Strategy

Three strategies, Circle Chaotic Mapping (CCM) initialization, Adaptive Random Wandering (ARW), and Dynamic Opposite Learning Crossover (DOLC), are used to improve the efficiency of HALO in solving the smart community fog computing task-scheduling problem. Among them, the CCM strategy and ARW strategy are added to ALO, named HALO-1, and the DOLC strategy is added to HALO-1, named HALO. For fair comparison, each algorithm is run 20 times independently in each experiment, the same population size of 40 is set, and the maximum number of iterations is 200.

From Figures 9 and 10, it can be seen that HALO has the smallest latency and energy consumption results compared to other algorithms for different numbers of tasks, which proves the effectiveness of the algorithm proposed in this paper. Tables 5 and 6 show the results of PIR_T and PIR_E for different algorithms under Task Data. HALO-1's latency and energy consumption results on Task Data are generally better than ALO; ARW not only reduces the complexity of ALO, but also ensures the optimization capability of ALO. The HALO algorithm adds the DOLC strategy on the basis of HALO-1, which is more powerful in Task Data, and the results of latency and energy consumption for high-dimensional test problems such as Task 200–Task 450 are improved by 2.85~3.33% and 4.65~5.39%, respectively, compared with ALO, and they are better than HALO-1, proving that the proposed HALO algorithm still has a stronger performance than ALO in solving high-dimensional complex task-scheduling problems.

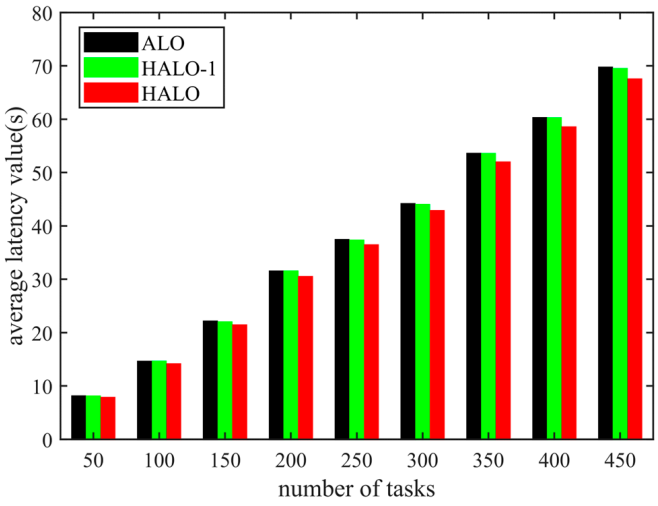


Figure 9. Latency results of three algorithms with different Task Data.

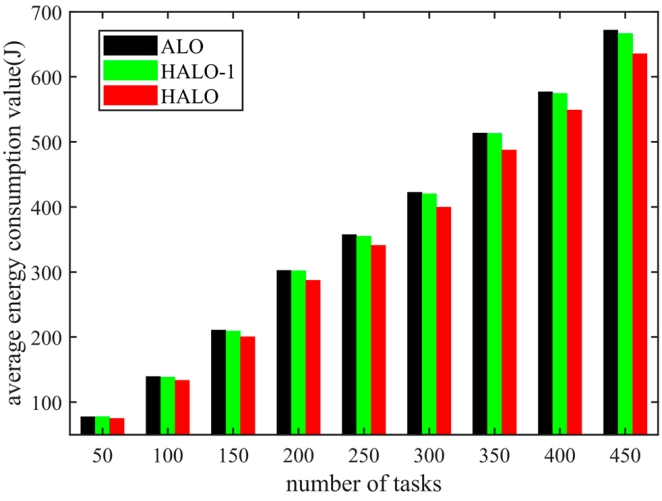


Figure 10. Energy consumption results of three algorithms with different Task Data.

Table 5. PIR_T of two algorithms on latency over ALO with different Task Data.

	Task 50	Task 100	Task 150	Task 200	Task 250	Task 300	Task 350	Task 400	Task 450
HALO-1 *PIR* (%)	0.47	0.16	0.70	0.07	0.32	0.27	0.11	0.04	0.40
HALO *PIR* (%)	3.15	3.48	3.22	3.33	2.85	2.92	3.06	2.85	3.20

Table 6. PIR_E of two algorithms on energy consumption over ALO with different Task Data.

	Task 50	Task 100	Task 150	Task 200	Task 250	Task 300	Task 350	Task 400	Task 450
HALO-1 *PIR* (%)	0.28	0.65	0.84	0.20	0.75	0.51	0.09	0.45	0.79
HALO *PIR* (%)	3.45	4.26	4.87	4.98	4.65	5.35	5.03	4.89	4.89

To visualize the performance of the algorithms, the average convergence curves of the three algorithms run independently 20 times on Task 350 are shown in Figure 11. The initial values of HALO-1 and HALO are better than those of ALO, because HALO-1 and HALO are initialized with CCM, which generates high-quality initial populations and improves the convergence speed of the algorithm in the first iteration. Meanwhile, at the late stage of the algorithm iteration, it can be seen that HALO-1 has a smaller average fitness value compared to ALO, which proves that ARW can still guarantee the algorithm's merit-seeking ability at the late stage. HALO has a stronger ability to find the optimal compared to ALO and HALO-1; due to the DOLC strategy, it enriches the diversity of ant populations and is able to jump out of the local optimum very well.

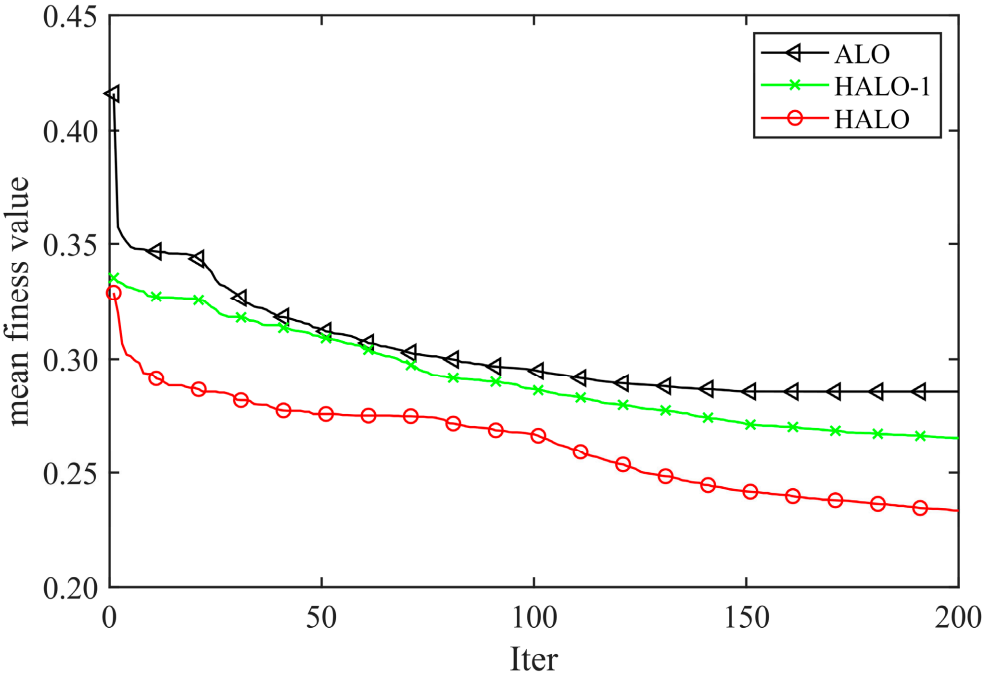


Figure 11. Comparison of different algorithms on Task 350.

5.5. Comparison with Other Algorithms

In this section, HALO is tested using Task Data, and the results of HALO are compared with algorithms proposed in recent years such as GWO, SSA, ALO and other researchers' applications of GA, PSO, WOA in fog computing [31,41–44]. To reflect fairness, the population size and the number of iterations of the algorithms involved in computing are the same as HALO at 40 and 200, respectively, and run 20 times independently, with the minimum and average values of latency and energy consumption and the performance improvement rate as evaluation criteria.

From Figures 12 and 13, it is intuitively clear that the ant lion algorithm improved in this paper is optimized to have the smallest latency and energy consumption relative to the original algorithm for different numbers of tasks. Tables 7 and 8 give the results of PIR_T and PIR_E for different algorithms under Task Data. It can be seen that HALO outperforms the results obtained by other algorithms on Task Data. Latency and energy consumption fall by 0.12~4.21% and 0.50~6.76%, respectively, compared to GA, latency and energy consumption, which fall by 1.93~3.47% and 3.20~4.60%, respectively, compared to WOA, latency and energy consumption, which fall by 2.15~3.62% and 3.11~5.50%, respectively, compared to GWO, latency and energy consumption, which fall by 4.46~5.48% and 7.00~9.69%, respectively, compared to SSA; compared to PSO, latency and energy consumption fall by 1.86~4.11% and 1.02~6.08%, respectively, and compared to ALO, latency and energy consumption fall by 2.68~3.25% and 2.35~5.37%, respectively.

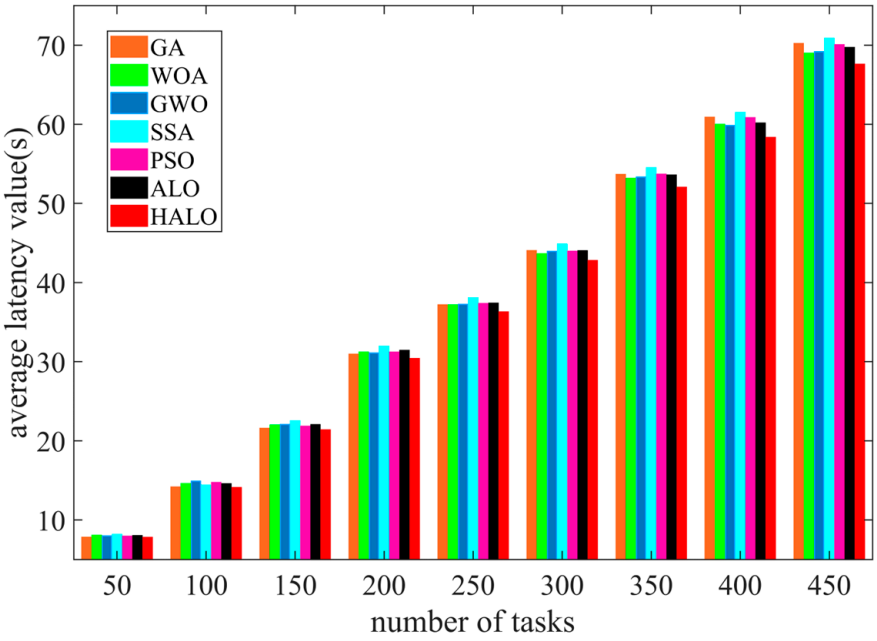


Figure 12. Latency results of different algorithms with different Task Data.

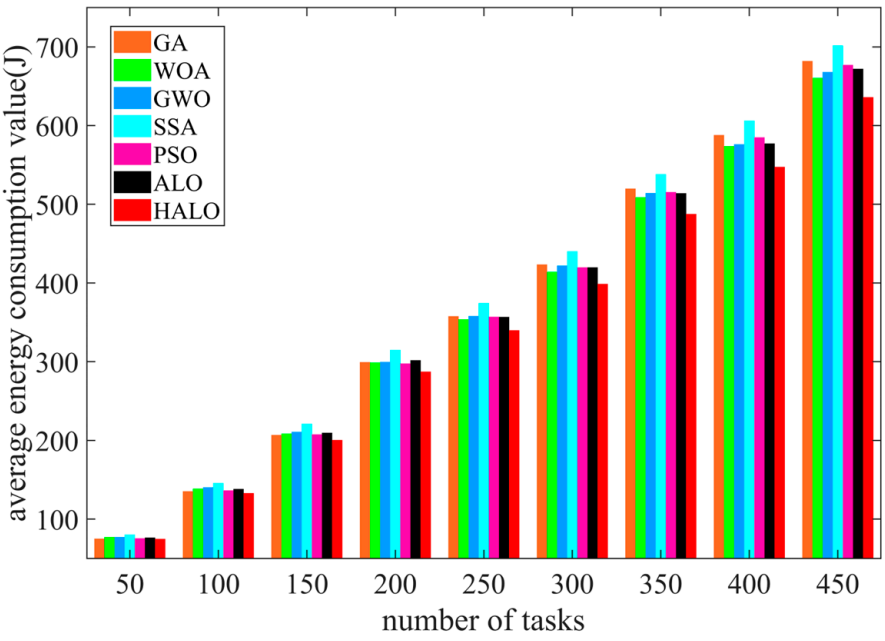


Figure 13. Energy consumption results of different algorithms with different Task Data.

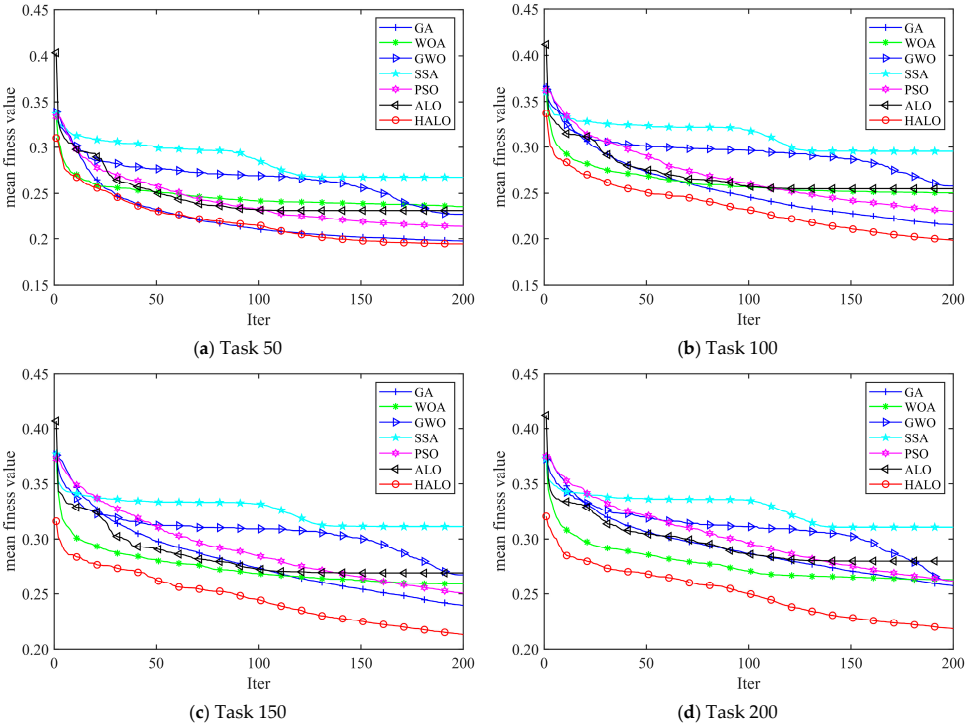
Table 7. PIR_T of different algorithms on latency with different Task Data.

	Task 50	Task 100	Task 150	Task 200	Task 250	Task 300	Task 350	Task 400	Task 450
GA	0.12%	0.65%	0.98%	1.76%	2.43%	2.81%	3.02%	4.21%	3.71%
WOA	3.16%	3.47%	2.84%	2.58%	2.35%	1.93%	2.10%	2.79%	2.02%
GWO	2.15%	3.62%	3.13%	2.26%	2.53%	2.61%	2.41%	2.55%	2.25%
SSA	4.46%	5.48%	5.14%	4.79%	4.63%	4.58%	4.52%	5.13%	4.61%
PSO	1.86%	2.07%	2.11%	2.57%	2.81%	2.67%	3.09%	4.11%	3.51%
ALO	2.68%	3.36%	3.05%	3.25%	2.93%	2.79%	2.89%	3.05%	3.02%

Table 8. PIR_E of different algorithms on energy consumption with different Task Data.

	Task 50	Task 100	Task 150	Task 200	Task 250	Task 300	Task 350	Task 400	Task 450
GA	0.50%	1.83%	3.08%	4.03%	5.01%	5.80%	6.21%	6.91%	6.76%
WOA	3.20%	4.22%	3.88%	3.90%	3.94%	3.71%	4.19%	4.60%	3.71%
GWO	3.11%	5.31%	4.90%	4.14%	5.05%	5.50%	5.17%	4.98%	4.82%
SSA	7.00%	8.96%	9.34%	8.78%	9.24%	9.40%	9.40%	9.69%	9.39%
PSO	1.02%	2.57%	3.43%	3.68%	4.80%	5.00%	5.43%	6.43%	6.08%
ALO	2.35%	3.90%	4.36%	4.83%	4.75%	5.16%	5.13%	5.15%	5.37%

To visualize the performance of the algorithm, the convergence curves of the comparison algorithm on Task Data are shown in Figure 11. The figure shows that the initial values of HALO are better compared to those of GA, WOA, GWO, SSA, PSO, and ALO. As shown in Figure 14f–i, it can be seen that for high-dimensional problems, GWO inevitably falls into a local optimum; SSA starts to converge only after 100 generations, while GA, WOA, PSO can guarantee convergence performance, but the optimization-seeking accuracy is not as high as HALO for high-latitude problems. HALO can effectively jump out of the local optimum while guaranteeing convergence speed, which proves that the HALO algorithm improved by CCM, ARW and DOLC in this paper is effective in scheduling fog computing tasks in smart communities and can effectively reduce the system latency.



(**a**) Task 50 (**b**) Task 100

(**c**) Task 150 (**d**) Task 200

Figure 14. *Cont.*

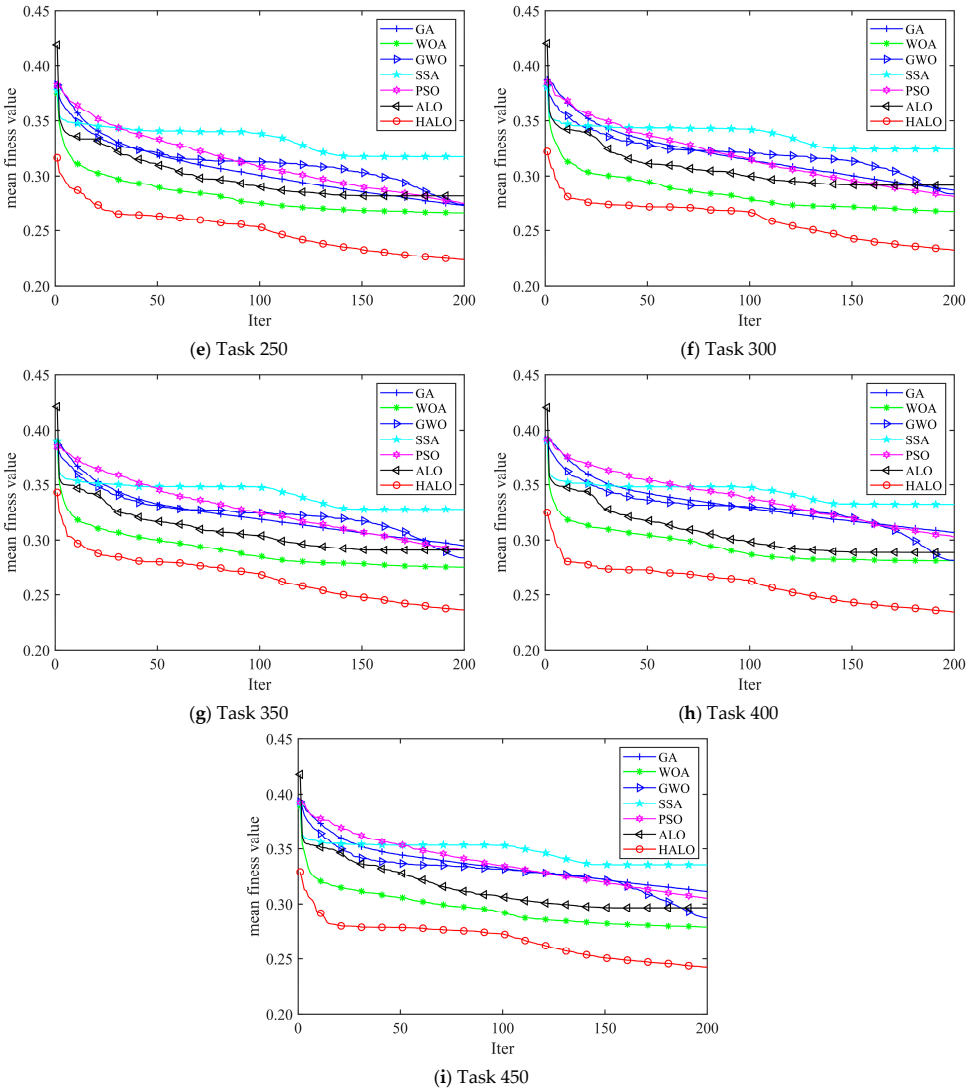


Figure 14. Comparison of different algorithms on Task Data.

6. Conclusions

A mathematical model of fog computing task scheduling is established for smart communities. The HALO fog computing task-scheduling algorithm is proposed to improve the quality of service for users under the condition of satisfying the goal of latency and energy consumption to minimize the same for the smart community fog computing system. The improvement of ALO in this paper is mainly in three aspects: (1) The initial population is initialized by the chaotic mapping, which makes the initial values more evenly distributed; (2) The adaptive random wandering is used for Ant Lion wandering, which improves the solving speed; (3) The dynamic opposite learning crossover strategy is embedded in the ALO, which improves the optimization-seeking ability of ALO. Simulation experiments show that compared with GA, WOA, GWO, SSA, PSO, and ALO algorithms, HALO has the best computing capability in high dimensional fog computing task scheduling. Using the method proposed in this paper, compared with GA, WOA, GWO, SSA, PSO, and ALO, the latency fell, respectively, by 4.21%, 2.79%, 2.55%, 5.13%, 4.11%, and 3.05%, and the energy consumption fell, respectively, by 6.76%, 3.71%, 4.82%, 9.39%, 6.08%, and 5.37% for the large-scale smart community fog computing task scheduling. The proposed method can effectively reduce the energy consumption and latency in the smart community fog computing.

Author Contributions: F.T.: the corresponding author, provided the idea and other technical guidance required for completing the study. D.Z. completed the practical design, case study and drafted the manuscript. Y.Y. and G.F. were responsible for the visualization of the manuscript. X.L. and G.C. were responsible for the supervision of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Science and Technology Development of Henan [Grant No. 232102211071]; Doctoral project of Henan Institute of Science and Technology [Grant No. 103020222001/066].

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are grateful to other project participants for their cooperation and endeavor.

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

References

1. Xu, X.; Huang, Q.; Yin, X.; Abbasi, M.; Khosravi, M.R.; Qi, L. Intelligent Offloading for Collaborative Smart City Services in Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 7919–7927. [CrossRef]
2. Perera, C.; Qin, Y.R.; Estrella, J.C.; Reiff-Marganiec, S.; Vasilakos, A.V. Fog Computing for Sustainable Smart Cities: A Survey. *ACM Comput. Surv.* **2017**, *50*, 1–43. [CrossRef]
3. Salaht, F.A.; Desprez, F.; Lebre, A. An Overview of Service Placement Problem in Fog and Edge Computing. *ACM Comput. Surd.* **2020**, *53*, 1–35. [CrossRef]
4. De Donno, M.; Tange, K.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access* **2019**, *7*, 150936–150948. [CrossRef]
5. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24. [CrossRef]
6. 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]
7. Zhang, G.; Shen, F.; Liu, Z.; Yang, Y.; Wang, K.; Zhou, M.T. FEMTO: Fair and Energy-Minimized Task Offloading for Fog-Enabled IoT Networks. *IEEE Internet Things J.* **2019**, *6*, 4388–4400. [CrossRef]
8. La, Q.D.; Ngo, M.V.; Dinh, T.Q.; Quek, T.Q.; Shin, H. Enabling intelligence in fog computing to achieve energy and latency reduction. *Digit. Commun. Netw.* **2019**, *5*, 3–9. [CrossRef]
9. Baniata, H.; Anaqreh, A.; Kertesz, A. PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. *Inf. Process. Manag.* **2021**, *58*, 102393. [CrossRef]
10. Tuli, S.; Ilager, S.; Ramamohanarao, K.; Buyya, R. Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments Using A3C Learning and Residual Recurrent Neural Networks. *IEEE Trans. Mob. Comput.* **2022**, *21*, 940–954. [CrossRef]
11. Adhikari, M.; Mukherjee, M.; Srirama, S.N. DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing. *IEEE Internet Things J.* **2020**, *7*, 5773–5782. [CrossRef]
12. Luo, J.; Yin, L.; Hu, J.; Wang, C.; Liu, X.; Fan, X.; Luo, H. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Gener. Comput. Syst.-Int. J. Escience* **2019**, *97*, 50–60. [CrossRef]
13. Verba, N.; Nixon, J.D.; Gaura, E.; Dias, L.A.; Halford, A. A community energy management system for smart microgrids. *Electr. Power Syst. Res.* **2022**, *209*, 107959. [CrossRef]
14. XNiu, D.; Shao, J.S.; Xin, C.; Zhou, J.; Guo, S.; Chen, X.; Qi, F. Workload Allocation Mechanism for Minimum Service Delay in Edge Computing-Based Power Internet of Things. *IEEE Access* **2019**, *7*, 83771–83784.
15. Kishor, A.; Chakarbarty, C. Task Offloading in Fog Computing for Using Smart Ant Colony Optimization. *Wirel. Pers. Commun.* **2022**, *127*, 1683–1704. [CrossRef]
16. Zhou, M.T.; Ren, T.F.; Dai, Z.M.; Feng, X.-Y. Task Scheduling and Resource Balancing of Fog Computing in Smart Factory. *Mob. Netw. Appl.* **2023**, *28*, 19–30. [CrossRef]
17. Cen, B.; Hu, C.; Cai, Z.; Wu, Z.; Zhang, Y.; Liu, J.; Su, Z. A configuration method of computing resources for microservice-based edge computing apparatus in smart distribution transformer area. *Int. J. Electr. Power Energy Syst.* **2022**, *138*, 107935. [CrossRef]
18. Elaziz, M.A.; Abualigah, L.; Attiya, I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Gener. Comput. Syst.-Int. J. Escience* **2021**, *124*, 142–154. [CrossRef]
19. Wang, J.; Di, L. Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing. *Sensors* **2019**, *19*, 1023. [CrossRef]
20. Abdel-Basset, M.; Mohamed, R.; Elhoseny, M.; Bashir, A.K.; Jolfaei, A.; Kumar, N. Energy-Aware Marine Predators Algorithm for Task Scheduling in IoT-Based Fog Computing Applications. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5068–5076. [CrossRef]
21. Hussein, K.; Mousa, M.H. Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization. *IEEE Access* **2020**, *8*, 37191–37201. [CrossRef]

22. Rafique, H.; Shah, M.A.; Islam, S.U.; Maqsood, T.; Khan, S.; Maple, C. A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing. *IEEE Access* **2019**, *7*, 115760–115773. [CrossRef]
23. Movahedi, Z.; Defude, B.; Hosseininia, A.M. An efficient population-based multi-objective task scheduling approach in fog computing systems. *J. Cloud Comput.-Adv. Syst. Appl.* **2021**, *10*, 53. [CrossRef]
24. Xu, J.; Hao, Z.; Zhang, R.; Sun, X. A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling. *IEEE Access* **2019**, *7*, 116218–116226. [CrossRef]
25. Chen, H.; Wang, X.; Li, Z.; Chen, W.; Cai, Y. Distributed sensing and cooperative estimation/detection of ubiquitous power internet of things. *Prot. Control Mod. Power Syst.* **2019**, *4*, 13. [CrossRef]
26. Junior, W.L.R.; Borges, F.A.; Veloso, A.F.d.S.; Rabêlo, R.d.A.; Rodrigues, J.J. Low voltage smart meter for monitoring of power quality disturbances applied in smart grid. *Measurement* **2019**, *147*, 106890. [CrossRef]
27. Amaxilatis, D.; Chatzigiannakis, I.; Tselios, C.; Tsironis, N.; Niakas, N.; Papadogeorgos, S. A Smart Water Metering Deployment Based on the Fog Computing Paradigm. *Appl. Sci.* **2020**, *10*, 1965. [CrossRef]
28. Ammad, M.; Shah, A.M.; Islam, S.U.; Maple, C.; Alaulamie, A.A.; Rodrigues, J.J.P.C.; Mu, S.; Tariq, U. A Novel Fog-Based Multi-Level Energy-Efficient Framework for IoT-Enabled Smart Environments. *IEEE Access* **2020**, *8*, 150010–150026. [CrossRef]
29. Nguyen, S.; Salcic, Z.; Zhang, X.; Bisht, A. A Low-Cost Two-Tier Fog Computing Testbed for Streaming IoT-Based Applications. *IEEE Internet Things J.* **2021**, *8*, 6928–6939. [CrossRef]
30. Li, Q.; Wang, X.; Wang, P.; Zhang, W.; Yin, J. FARDA: A fog-based anonymous reward data aggregation security scheme in smart buildings. *Build. Environ.* **2022**, *225*, 109578. [CrossRef]
31. Mirjalili, S. The Ant Lion Optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
32. Abualigah, L.; Diabat, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust. Comput.-J. Netw. Softw. Tools Appl.* **2021**, *24*, 205–223. [CrossRef]
33. Hosny, K.M.; Awad, A.I.; Khashaba, M.M.; Mohamed, E.R. New Improved Multi-Objective Gorilla Troops Algorithm for Dependent Tasks Offloading problem in Multi-Access Edge Computing. *J. Grid Comput.* **2023**, *21*, 21. [CrossRef]
34. Ahmad, S.; Sulaiman, M.; Kumam, P.; Zubaira, H.; Jan Muhammadd, A.; Khand, M.W.; Masiha, U. A novel population initialization strategy for accelerating Levy flights based multi-verse optimizer. *J. Intell. Fuzzy Syst.* **2020**, *39*, 1–17. [CrossRef]
35. Zhang, H.; Liu, T.; Ye, X.; Heidari, A.A.; Liang, G.; Chen, H.; Pan, Z. Differential evolution-assisted salp swarm algorithm with chaotic structure for real-world problems. *Eng. Comput.* **2023**, *39*, 1735–1769. [CrossRef] [PubMed]
36. Bandyopadhyay, R.; Basu, A.; Cuevas, E.; Sarkar, R. Harris Hawks optimisation with Simulated Annealing as a deep feature selection method for screening of COVID-19 CT-scans. *Appl. Soft Comput.* **2021**, *111*, 107698. [CrossRef] [PubMed]
37. Zhang, X.Z.; Wang, Z.Y.; Lu, Z.Y. Multi-objective load dispatch for microgrid with electric vehicles using modified gravitational search and particle swarm optimization algorithm. *Appl. Energy* **2022**, *306*, 118018. [CrossRef]
38. Xu, Y.; Yang, Z.; Li, X.; Kang, H.; Yang, X. Dynamic opposite learning enhanced teaching-learning-based optimization. *Knowl.-Based Syst.* **2020**, *188*, 104966. [CrossRef]
39. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; pp. 695–701.
40. Arshed, J.U.; Ahmed, M.; Muhammad, T.; Afzal, M.; Arif, M.; Bazezew, B. GA-IRACE: Genetic Algorithm-Based Improved Resource Aware Cost-Efficient Scheduler for Cloud Fog Computing Environment. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 6355192. [CrossRef]
41. Aburukba, R.O.; AliKarrar, M.; Landolsi, T.; El-Fakih, K. Scheduling Internet of Things requests to minimize latency in hybrid Fog-Cloud computing. *Future Gener. Comput. Syst.-Int. J. Escience* **2020**, *111*, 539–551. [CrossRef]
42. Ghobaei-Arani, M.; Shahidinejad, A. A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment. *Expert Syst. Appl.* **2022**, *200*, 117012. [CrossRef]
43. Mirjalili, S.; Lewis, S.M.M.A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
44. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]