Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.042340

ARTICLE

Using Metaheuristic OFA Algorithm for Service Placement in Fog Computing

Riza Altunay[1,2,*] **and Omer Faruk Bay**[3]

[1]Department of Information Systems, Gazi University, Ankara, 06680, Turkey

[2]Department of Computer Technologies, Ondokuz Mayis University, Samsun, 55100, Turkey

[3]Department of Electrical-Electronics Engineering, Gazi University, Ankara, 06560, Turkey

*Corresponding Author: Riza Altunay. Email: riza.altunay@omu.edu.tr

Received: 26 May 2023 Accepted: 19 October 2023 Published: 26 December 2023

ABSTRACT

The use of fog computing in the Internet of Things (IoT) has emerged as a crucial solution, bringing cloud services closer to end users to process large amounts of data generated within the system. Despite its advantages, the increasing task demands from IoT objects often overload fog devices with limited resources, resulting in system delays, high network usage, and increased energy consumption. One of the major challenges in fog computing for IoT applications is the efficient deployment of services between fog clouds. To address this challenge, we propose a novel Optimal Foraging Algorithm (OFA) for task placement on appropriate fog devices, taking into account the limited resources of each fog node. The OFA algorithm optimizes task sharing between fog devices by evaluating incoming task requests based on their types and allocating the services to the most suitable fog nodes. In our study, we compare the performance of the OFA algorithm with two other popular algorithms: Genetic Algorithm (GA) and Randomized Search Algorithm (RA). Through extensive simulation experiments, our findings demonstrate significant improvements achieved by the OFA algorithm. Specifically, it leads to up to 39.06% reduction in energy consumption for the Elektroensefalografi (EEG) application, up to 25.86% decrease in CPU utilization for the Intelligent surveillance through distributed camera networks (DCNS) application, up to 57.94% reduction in network utilization, and up to 23.83% improvement in runtime, outperforming other algorithms. As a result, the proposed OFA algorithm enhances the system's efficiency by effectively allocating incoming task requests to the appropriate fog devices, mitigating the challenges posed by resource limitations and contributing to a more optimized IoT ecosystem.

KEYWORDS

Internet of Things; cloud computing; fog computing

1 Introduction

With the rapid proliferation of technology, encompassing smartphones, wearable devices, industrial tools, and various other domains, the number of internet-connected devices has experienced significant growth. This surge in interconnectedness has given rise to the concept of the IoT, which was initially introduced by a group of academics in 1991. As IoT devices often face processing capacity and resource limitations, the adoption of cloud services has become a necessity [1]. Nevertheless,

the escalating utilization of cloud services and devices has brought forth challenges, including issues related to distributed data arising from diverse geographical locations, increased bandwidth usage, and end-to-end delays. To effectively tackle these challenges, fog computing has emerged as a viable solution [2]. Fog computing is strategically positioned between the cloud and IoT devices, serving as a decentralized infrastructure comprised of diverse fog nodes that offer management, storage, and communication capabilities [3]. Moreover, fog computing brings additional advantages, including data protection and secure communication [4,5]. However, due to the heterogeneity and limited resources of fog nodes, achieving effective service placement poses a significant challenge. In response to this issue, several metaheuristic algorithms have been proposed to address task scheduling in the IoT system. These algorithms encompass the Fireworks Algorithm (FWA) [6], Marine Predators Algorithm (MPA) [7], Bee Algorithm (BA) [8], Particle Swarm Optimization (PSO) [9], and Genetic Algorithm (GA) [10]. Abohamama et al. proposed a semi-dynamic real-time task scheduling algorithm that leverages a modified version of the genetic algorithm to optimize task scheduling [11]. Ghobaei-Arani et al. utilized the meta-intuitive whale optimization algorithm to allocate tasks on fog devices and enhance service quality [12]. On the other hand, Dubey et al. introduced a fog device framework that improves delays, computational costs, load balancing, and energy consumption in the IoT system, utilizing Cuckoo and PSO algorithms [13]. Maiti et al. proposed a fog node layout to minimize delays using algorithms such as Simulated Annealing (SA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). Their findings indicated that the GA-SA algorithm proved to be the most efficient in minimizing delays [14].

In their study on service placement, Mohamed et al. [15] combined two distinct algorithms to determine the most optimal and shortest path while selecting the nearest fog node for users. Their approach resulted in successful improvements in network usage, distance optimization, load balancing, and data transmission. Zare et al. [16] employed the Asynchronous Advantage Actor-Critic (A3C) algorithm, developed in the field of deep and reinforcement learning, to ensure high-quality service delivery. Skarlat et al. [17] used the Genetic Algorithm (GA) to showcase the efficient utilization of fog resources, leading to reductions in communication delays within the network. Additionally, Saif et al. [18] employed the Grey Wolf Optimizer algorithm to address energy consumption and minimize delays in the system. The literature review highlights that fog computing is adopted to address the growing number of objects and their demands in cloud computing. However, due to the limited resources of fog clouds, they are unable to accommodate all requests. As a result, the increasing number of requests adversely impacts bandwidth usage, energy consumption, processing costs, and operation time. To tackle these challenges, numerous studies have been conducted, focusing on areas such as resource sharing, task scheduling, and load balancing to mitigate the issues associated with limited fog cloud resources. In current studies, the service placement problem remains a critical issue that requires improvement, particularly concerning the placement of applications from objects onto fog clouds with limited resources. The primary objective of this study is to implement the Optimal Foraging Algorithm (OFA) to efficiently allocate services onto application modules. The proposed Optimal Foraging Algorithm (OFA) has been subjected to comparison with various algorithms, focusing on basic comparison functions and optimization problems [19]. The algorithm's strong performance in optimization problems, its novelty in IoT applications, and its promising potential for success due to its unique structure were regarded as a novel perspective, leading to its application in this study. The working logic and pseudo code of the algorithm are elaborated upon in Section 3. In the deployment of IoT applications on fog nodes, it is crucial to ascertain the requirements of the applications and the desired service quality to ensure compatibility with the existing fog nodes and applications. During the service distribution stage, where services are allocated to suitable fog clouds

based on application types, the OFA metaheuristic algorithm is employed to generate appropriate solutions [19].

The obtained solutions are first internally compared using various metrics and subsequently pitted against the GA and RA algorithms in terms of energy consumption, bandwidth usage, processing cost, and execution time. Through these comparisons, the advantages of the proposed algorithm are demonstrated.

The contributions of this study to the literature can be summarized as follows: (1) Optimization of efficient service placement between fog devices in fog cloud-based IoT using a three-layer architecture; (2) Introduction of the Optimal Foraging Algorithm (OFA) for service placement, marking its application for the first time in IoT; (3) Demonstration through simulation results that the proposed algorithm outperforms other algorithms in terms of energy consumption, CPU usage, bandwidth usage, and runtime. The study is structured into six parts. The first part introduces the topic and provides a comprehensive literature review. The second part presents detailed information about the system architecture and formulates the problem being addressed. In the third part, the proposed optimization algorithm is thoroughly explained, along with its operational principles. The fourth part showcases the performance values obtained from the experimentation. In the fifth part, a numerical analysis of the algorithm is presented, including relevant data and insights. Finally, the sixth part concludes the study, summarizing the findings and discussing the overall status of the manuscript.

2 System Architecture and Problem Formulation

The system operates on a layered architecture as shown in Fig. 1. The highest layer is referred to as the cloud layer, which is a distributed data center accessible to all internet-connected devices and not used at a specific location [20]. The middle layer, on the other hand, is where fog computing is implemented, and where fog devices are located, providing real-time computing that considers current operations [21]. The bottom layer is where various smart and non-smart objects with a distributed structure, such as computers, smartphones, and machines, are connected [22].

In IoT applications, the limited resources and distributed nature of the heterogeneous and decentralized fog nodes, as well as their changing processing capacities over time, pose challenges in the service placement stage. To determine when and where services will be placed, service placement is required. The proposed algorithm addresses these challenges and enables the efficient allocation of application modules to fog nodes. The architecture used in the system includes entities and services. Entities are composed of components such as cloud, client, and fog nodes, while services are application modules that are received from objects to be run in the system. The system possesses all the necessary information about fog clouds and incoming mission requests during its operation. As a result, the services are efficiently directed to the relevant fog cloud. Each node runs the tasks assigned by the system. Applications can be sent to fog nodes as modules or sets of modules. Each application module has properties such as processing power, memory, storage, bandwidth, etc. Service modules are categorized into three different types: client, global, and regular modules. Client modules can only be processed on client nodes, while global modules can be processed on nodes other than client nodes. Modules other than these can be processed on both types of nodes. The proposed decision-making algorithm in the system aims to improve system performance by placing services on nodes with limited hardware capabilities under a certain service quality constraint. In this study, energy consumption, processing cost, bandwidth, and execution time are determined as performance criteria. These performance criteria and the quality of service are formulated and calculated separately as shown below.

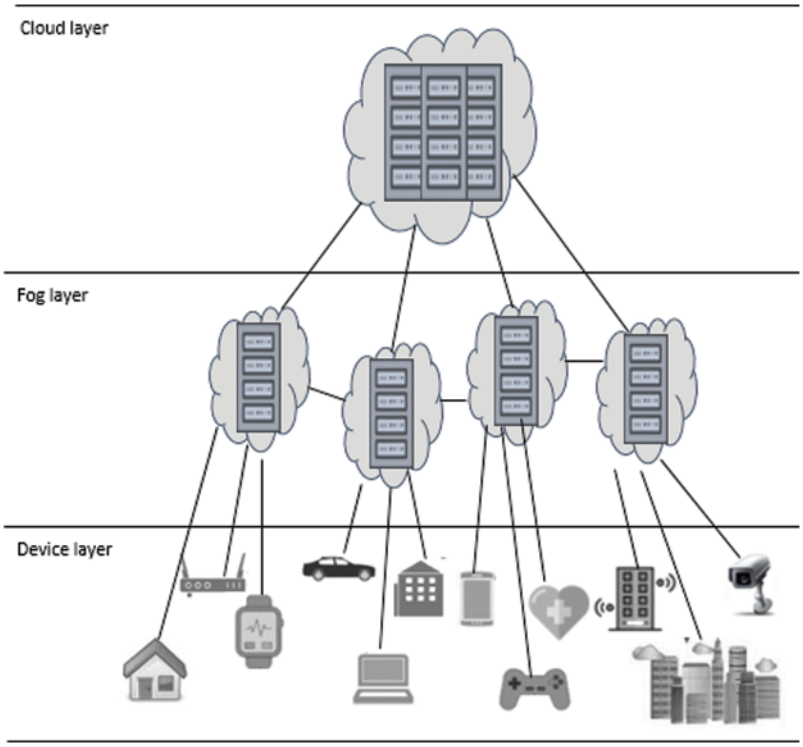

Figure 1: System architecture

2.1 Quality of Service

After including application cycles in the system, it is checked if the task is completed on time while providing Quality of Service (QoS). Operation delays and transmission delays that arise while transferring the application between two nodes are calculated during this process. The importance of service quality has increased due to the growing number of devices and the emergence of diverse services [23]. QoS is considered to be fulfilled if the application cycles are processed and their tasks are completed. Eq. (1) is used to calculate QoS, when representing the cost of C_Q performance, Q, denotes the total number of application loops. A_q^A represents each loop within the application, and A_q^D shows the completion times of tasks in the application. q refers to the total delay in application loops, L_q^P indicates the total processing delays between two nodes, and L_q^T represents the total transmission delays between two nodes. By taking these delays into account, it is checked whether QoS is ensured during the transfer of an application between two nodes [24].

$$C_Q(x) = \sum_{a \in A} e_a x_a, \; e_a = \min\left(\sum_{q \in Q} e_q, 1\right), \; A_q^A = a, \tag{1}$$

$$e_q = \begin{cases} 1, & \text{if } L_q^p + L_q^T > A_q^D; \\ 0, & \text{otherwise.} \end{cases} \quad , \forall q \in [0, Q]$$

2.2 Power Cost

The equation in Eq. (2) is used to calculate the cost of energy consumption by taking into account the transaction cost used during the application and the cost spent on network transmission. It is

important to note that in fog computing, sending data to distant clouds can lead to excessive energy consumption [25]. The power consumption cost (C_{Pw}) is obtained by summing up the processing cost (C_P) and the transmission cost (C_B). To calculate C_P, the difference between the power consumption while active (f^{bPw}) and the power consumption while idle (f^{iPw}) is multiplied by the connection usage percentage. To calculate C_B, the power consumption is multiplied by the connection usage percentage at full transmission capacity (f^{Tx}) [24].

$$C_{Pw}(x) = C_P\left(x. \times \left(f^{bPw} - f^{iPw}\right)\right) + C_B\left(x . \times f^{Tx}\right) \tag{2}$$

2.3 Processing Cost

The system's processing capacity is increased by using a low-cost processor. When operating the system, it is important to consider the processing capacity used by each resource and to take processing costs into account when allocating tasks [26]. Eq. (3) calculates the processing cost (C_P) is the sum of the ratio of the total amount of processing capacity to the utilized capacity multiplied by f_n^{Fog}. $P_n \times m^{Mips}$ represents the total amount of processing capacity required for all modules placed in node n. $a^p \times f_n^{Mips}$ indicates the capacity utilised by multiplying the processing capacity of node n by the resource utilisation percentage (ap). f^{Fog} indicates whether the node is a fog node. If $n = 0$, it is a client node [24].

$$C_P(x) = \sum_{n \in N} f_n^{Fog} x_n \frac{P_n \times m^{Mips}}{a^p \times f_n^{Mips}}. \tag{3}$$

2.4 Bandwidth Cost

While processing the applications, if the necessary bandwidth is provided, the system will operate fast. The number of edges on the system increases the bandwidth cost [27]. Eq. (4) calculates bandwidth. C_B is calculated as the ratio of each value of the tuple routing map $R_{z,e}$ to the bandwidth used ($a^b \times E_e^{Bw}$). The products of these ratios with f_i^{Fog} are summed. The sums obtained are added to the amount of bandwidth ($m_{l_z^S, l_z^D}^{BW}$) required between the destination and source nodes to obtain the result. f^{Fog} indicates whether the nodes are fog nodes or not [24].

$$C_B(x) = \sum_{z \in Z} m_{l_z^S, l_z^D}^{Bw} \sum_{e \in E} f_i^{Fog} x_i \frac{R_{z,e}}{\alpha^b \times E_e^{Bw}}, \quad i = E_e^S \quad . \tag{4}$$

2.5 Execution Time

It is the time in milliseconds from the start of the system to the completion of the optimization phase.

3 Proposed Optimization Algorithm

OFA is a global optimization algorithm that monitors the foraging behaviour of animals and determines their position accordingly. In addition, the OFA algorithm is used to solve optimization problems in many fields such as classification and regression problems [28], ranking problems in industry [29], and healthcare [30,31]. It is a known fact that animals in nature need food in order to survive and continue their generations. Foraging is a fairly common phenomenon to meet these needs. Animals consider three situations while foraging [19]. Where to look for food, When to look for new food, What type of food to choose? In the proposed algorithm, individuals prefer foraging areas according to the rules they have determined. When an area is determined, the first thing to

consider is how valuable the food is. If the food found is valuable, more individuals in the community move there. The search for valuable food continues as long as they live. When the OFA algorithm is summarized mathematically, the following stages emerge.

3.1 OFA Algorithm Steps

First of all, for the d-dimensional array in N number of food-seeking individuals and R constraint space, each individual is specified as $X = [x_1, x_2, x_3, \ldots, x_d]^T$. In this direction, the objective function is defined as in Eq. (5) [19].

$$F(\times^*) = \min_{X \in R} f(x), \; R = \left\{ x_i^L \le x_i \le x_i^U, \; i = 1, 2, 3, \ldots, d \right\} \tag{5}$$

F(x), objective function value and $F(\times^*)$ is the optimal objective function value, x^* is the optimal vector. Here, maximum optimization problem of F(x) is equivalent to the minimum optimization problem of −F(x), with x_i^L, x_i^U lower and upper limit values, respectively. In order to create individuals, N number of individuals are produced by using Eq. (6) [19].

$$x_{ji}^1 = x_j^L + \text{rand}(0, 1) \times \left(x_i^U - x_i^L \right) \tag{6}$$

When the recurrence rates are accepted as t for the current number and as t + 1 for the next iteration, the best individuals are shown with Eq. (7) and the remaining individuals are shown with Eq. (8). k variable is shown as $k = t/t_{max}$, t shows the number of valid iterations, t_{max} shows the maximum number of iteration, x_{random}^t shows the randomly selected individual, x_{worst}^t shows the worst individual of the population, r_{1i} and r_{2i} show the random numbers distributed between 0 and 1 [19].

$$x_{ji}^{t+1} = x_{ji}^t - k \times r_{1ji} \times \left(x_{ji}^t - x_{worst}^t \right) + k \times r_{2ji} \times \left(x_{ji}^t - x_{worst}^t \right) \tag{7}$$

$$x_{ji}^{t+1} = x_{ji}^t - k \times r_{1ji} \times \left(x_{random}^t - x_{ji}^t \right) + k \times r_{2ji} \times \left(x_{random}^t - x_{ji}^t \right) \tag{8}$$

While individuals are deciding on whether to update their current location, they use prey selection model [32]. Prey energy uses F_j^t, advantageous prey energy uses F_j^{t+1} variables. In Eq. (9), t and t + 1 positions are compared from each iteration result, if it is efficient, the current position is changed and if it is not, it continues a new iteration with the current position [19].

$$\frac{\text{lambta} \times F_j^{t+1}}{1 + \text{lambta} \times (t + 1)} < \frac{F_j^t}{t} \tag{9}$$

3.2 OFA Algorithm Pseudo Code

General operation logic of the algorithm is given above and the pseudo-code is shown in Algorithm 1.

Algorithm 1: Pseudo code of the OFA algorithm

```
Procedure OFA
    iter ← 0;
    bestSolution ← null;
```

(Continued)

Algorithm 1 (continued)

```
4:          for i ∈ populationsize do
5:                  populationsize[i] ← GenerationRandomSolution
6:          end for
7:          population ← sort(population);
8:          bestSolution ← population[0];
9:          while iter ≤ MAX_ITER do
10:                 prevBestSolution ← bestSolution;
11:                 bestSolution ← compare(bestSolution, createSolution);
12:                 iter ← iter + 1;
13:         end while
14:         return bestSolution;
15:     end procedure
```

Firstly, a population is created, consisting of a specific number of individuals. For each individual in the population, a random solution is generated. Subsequently, a module placement map is created, indicating feasible locations for modules within the solution. Based on this map, a tuple routing map and a module transition map are established, facilitating node selection for transferring modules to their destination nodes. The solutions obtained for each individual are then ranked according to their costs, with the individual having the lowest cost selected as the best solution. When creating a new individual, each one tends to move towards the best position. As individuals search for solutions close to their current location, the newly produced solutions remain in proximity to the previous ones. Each new solution is compared with the previous best solution. Following the comparison, if the location (i.e., the cost) of the newly generated individual is better than the previous best solution, the next search continues with that solution. Conversely, if the generated solution is worse than the previous one, the current solution is retained. In this manner, solutions are generated for a number of iterations, and each time, the best solutions are compared with the previous one. Once the specified iteration is completed, the best solution obtained is selected as the final best solution. The simulation is then initiated, and the results are obtained and evaluated based on the chosen final best solution.

4 Performance Evaluation

There are entities consisting of cloud and fog nodes on physical topology in the operation of the system. Devices such as sensors and actuators that can receive data and apply the result after processing can be connected to these nodes. After the system is installed, the control unit performs the necessary checks and the system operates. Application packages sent on the system are distributed among the existing fog nodes with the proposed algorithm as data bundles and the system is operated. Each node has its properties.

4.1 Components of the System

During the system setup, there is one cloud, a proxy connected to the cloud, a varying number of fog nodes connected to the proxy, and objects have been used. The numbers used have been explicitly specified during the comparison stage. The processing capacities that can be used for all devices on the system, the idle power consumed by the nodes, and the amount of power consumed during operation are defined in Table 1. The results obtained during the operation of the system are produced based on these values. To evaluate the proposed system, a fog computing environment was created and simulated

using the iFogSim [33] simulator, which is commonly used in academic publications. The algorithm was coded using the Java programming language. The experiments were conducted on a desktop computer with an Intel Core i7-6700 processor with a clock speed of 3.40 GHz and 8 GB of memory, using the EEG and DCNS, which are commonly used in academic publications.

Table 1: Components of the system

Parameters	Value
Cloud	
Processing capacity	44,800,000 MIPS
Idle power consumption	1,648 W
Busy power consumption	1,332 W
Proxy	
Processing capacity	1,000,000 MIPS
Idle power consumption	107,339 W
Busy power consumption	83,433 W
Fog Node	
Processing capacity	75,000 MIPS
Idle power consumption	50 W
Busy power consumption	38 W

4.1.1 EEG Beam Tractor Game

EEG Beam Tractor Game is a type of game that takes place between two humans and uses augmented brain-computer interaction. The players' brain activities are measured using EEG devices and these measurements are transferred to a computer program that allows the players to play the game. Thus, players' brain activities enable them to control the objects in the game. The application consists of 5 modules: EEG sensor, screen, client, concentration calculator, and coordinator [34]. EEG parameters are shown in Table 2.

Table 2: Application edges of the EEG tractor beam game

Edge	Processing (MIPS)	Data (B)
Eeg	300	500
Sensor	350	500
Player game state	100	1000
Concentration	1.4	500
Global game state	2.8	1000
Global state update	0	500
Self state update	0	500

Many experiments have been performed using the OFA algorithm on EEG with systems containing different numbers of fog clouds. The results of the experiments are shown in Table 3. In Table 3, the Area column shows the number of fog clouds connected to the Proxy, the Device column shows the number of devices connected to the fog clouds, and the Fog Node column shows the total number of nodes on the system. Respectively, Energy (J) indicates the amount of energy consumed by the system, processed (MI) indicates the amount of data processed through the system, transferred (B) indicates the bandwidth used by the system, and Execution time (ms) indicates the operating time of the system.

Table 3: EEG simulation results

Cloud	Proxy	Area	Device	Fog node	Energy (J)	Processed (MI)	Transferred (B)	Execution time (ms)
1	1	2	4	12	2,808,224	13,329,900	9,226,000	1186
1	1	2	5	14	2,883,287	42,649,600	12,259,000	2023
1	1	2	6	16	2,887,039	43,088,400	15,724,000	2397
1	1	3	4	17	3,682,045	38,802,820	18,508,000	1900
1	1	3	5	20	3,744,886	45,505,900	32,377,500	2243
1	1	3	6	23	4,205,220	57,867,010	39,752,000	2558
1	1	4	4	22	4,806,614	51,716,220	37,525,000	2369
1	1	4	5	26	4,807,274	71,003,150	45,879,000	2943
1	1	4	6	30	5,142,440	90,349,090	50,776,500	3603

The horizontal axes of Fig. 2 show the total number of nodes. In Fig. 2a, the energy consumed by the system exhibits a linear increase. While Figs. 2b–2d also show a general linear increase, there is a noticeable decrease in the values when the number of fog nodes is 17 and 22. This decrease, as observed in Table 3 in the Area domain, occurs due to an initial increase in the number of optimized fog clouds. The rise in the number of fog nodes results in additional resources for distributing services, thereby leading to these reductions. This increase has a positive impact on the system's CPU usage, network usage, and runtime performance.

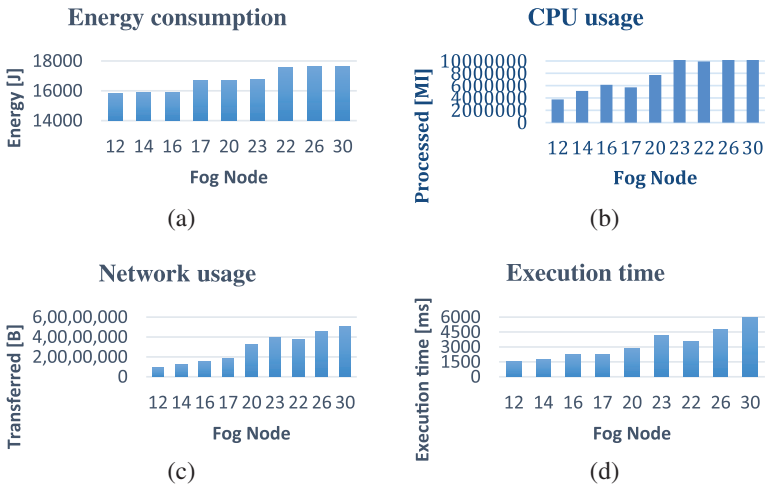

Figure 2: (a) Energy consumption (b) CPU usage (c) Network usage (d) Execution time

4.1.2 DCNS (Intelligent Surveillance through Distributed Camera Networks)

DCNS refers to the deployment of multiple cameras across an area to monitor and analyze activities in real-time. The cameras capture video footage and detect moving objects by analyzing the images. A designated object is tracked by PTZ cameras and the images are transmitted to the user over the internet. The application consists of 5 modules: Motion Detector, Object Detector, Object Tracker, PTZ Control, and User Interface [33]. DCNS parameters are shown in Table 4. The results of the experiments with the OFA algorithm on DCNS are shown in Table 5.

Table 4: Application edges of the DCNS

Edge	Processing (MIPS)	Data (B)
Camera	100	2000
Motion video stream	200	2000
Detected object	50	2000
Object location	100	100
PTZ params	0	100

Table 5: DCNS simulation results

Cloud	Proxy	Area	Device	Fog node	Energy (J)	Processed (MI)	Transferred (B)	Execution time (ms)
1	1	2	4	12	15,853.45418	3,754,233	47,141,400	1529
1	1	2	5	14	15,865.15359	5,151,415	44,976,600	1741
1	1	2	6	16	15,874.51023	6,138,460	85,679,000	2224
1	1	3	4	17	16,704.77644	5,729,339	114,924,400	2228
1	1	3	5	20	16,717.35024	7,327,834	124,112,600	2982
1	1	3	6	23	16,744.33693	10,475,190	157,069,500	4118
1	1	4	4	22	17,575.02128	9,918,626	161,867,000	3517
1	1	4	5	26	17,601.93762	13,087,850	166,051,100	4757
1	1	4	6	30	17,623.96209	15,818,720	252,093,700	6564

The horizontal axes of Fig. 3 show the total number of nodes. In Figs. 3a and 3c, a linear increase in energy consumption and network usage on the system is observed as the number of fog nodes increases. However, in Figs. 3b and 3d, a decrease is noticed when the number of fog nodes is 17 and 22. This decrease is evident in Table 5, which corresponds to the optimization of fog nodes and occurs when the number of fog nodes is initially increased. The increase in the number of fog nodes provides additional resources for distributing services, resulting in positive effects on CPU usage and system runtime.

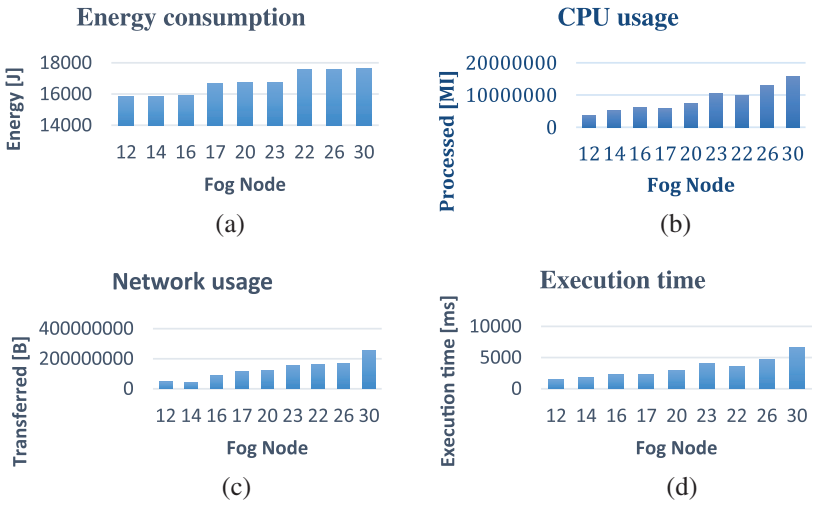

Figure 3: (a) Energy consumption (b) CPU usage (c) Network usage (d) Execution time

4.2 Algorithm Comparison

The EEG and DCNS were run on a layered architecture with the OFA algorithm, GA, and RA algorithms, respectively. Comparisons were made over values such as energy consumption, operation costs, bandwidth cost and execution time. Parameter settings determined for each algorithm to be run on the system are shown in Table 6 below.

Table 6: Algorithm parameters

Parameters	Value
Genetic algorithm	
Population size	12
Maximum iteration	1000
Maximum iteration convergence	25
Random algorithm	
Maximum iteration	1000
Maximum iteration convergence	130
Optimal foraging algorithm	
Population size	12
Maximum iteration	1000
Bound	−5.12, 5.12

4.2.1 Genetic Algorithm

Initially, a random population is generated. The genetic algorithm generates heuristic solutions using the previously generated solution information. In each iteration, the number of convergence is checked and new individuals are created according to the situation. The new individual formation is performed using three genetic operators. The first one is the selection operator, which controls fitness and ensures that the genes are transferred to the next generation. In this way, it transfers a certain percentage of genes of the best individuals to the next generation. Secondly, the crossover operator performs the process of crossing the two best individuals selected by the selection operator, and finally, new individuals are produced by adding random genes by the mutation operator in order not to produce the same solutions. The process is continued for the number of iterations and the best solution is obtained.

4.2.2 Random Algorithm

The RA algorithm generates random solutions without using the values generated in the previous solution for the specified number of iterations. The first generated solution is taken as the best solution. Each generated solution is compared with the best solution and if it is better, the better solution is selected as the best solution. On the other hand, if a certain number of identical solutions are generated by the convergence number, the loop is broken and the best solution is selected. Table 7 shows the EEG results.

Table 7: Simulation results in EEG

	Energy (J)	Processed (MI)	Transferred (B)	Exe. time (ms)
14 Fog device (1 Cloud + 1 Proxy + 2 Area + 5 Device)				
OFA	2,746,065	3,793,497	4,805,274	1499
GA	2,921,447	3,795,804	5,099,065	1404
RA	2,920,742	3,972,147	5,604,187	1620
20 Fog device (1 Cloud + 1 Proxy + 3 Area + 5 Device)				
OFA	25,931,500	51,531,800	71,003,150	2224
GA	26,067,730	51,474,960	64,615,760	2292
RA	25,961,480	51,883,940	82,787,920	2226
26 Fog device (1 Cloud + 1 Proxy + 4 Area + 5 Device)				
OFA	11,093,000	29,440,000	45,879,000	2943
GA	11,771,000	29,841,500	46,555,500	2930
RA	15,427,000	30,700,000	55,160,000	3352

Fig. 4 shows the comparison of energy consumption, processing costs, bandwidth cost and execution time of OFA algorithm with other algorithms. In the comparisons made on systems with different numbers of fog devices; Fig. 4a shows that in terms of energy consumption, it improves 6%, 0.5%, and 6% compared to the GA algorithm, 6%, 0.1%, 39% compared to RA respectively in the experiment using 14, 20, 26 fog devices. Fig. 4b shows that in the experiment using 14, 20, 26 fog

devices, there is an increase of 0.06%, −0.1%, 1%, respectively, in terms of cpu usage compared to GA, and an improvement of 4%, 0.6%, 4%, respectively, compared to RA. Fig. 4c shows that the bandwidth usage using 14, 20, and 26 fog devices improved by 6%, −8%, and 1% compared to GA, 16%, 16%, and 20% compared to RA, respectively. Fig. 4d shows that in the experiment performed using 14, 20, and 26 fog devices on the runtimes, it has improved by −6%, 3%, −0.4% according to GA, 8%, 0.08%, 13% according to RA, respectively. The experiments conducted on the EEG application demonstrate that the OFA algorithm outperforms both the GA and RA algorithms in terms of energy consumption, bandwidth usage, CPU usage, and runtime.

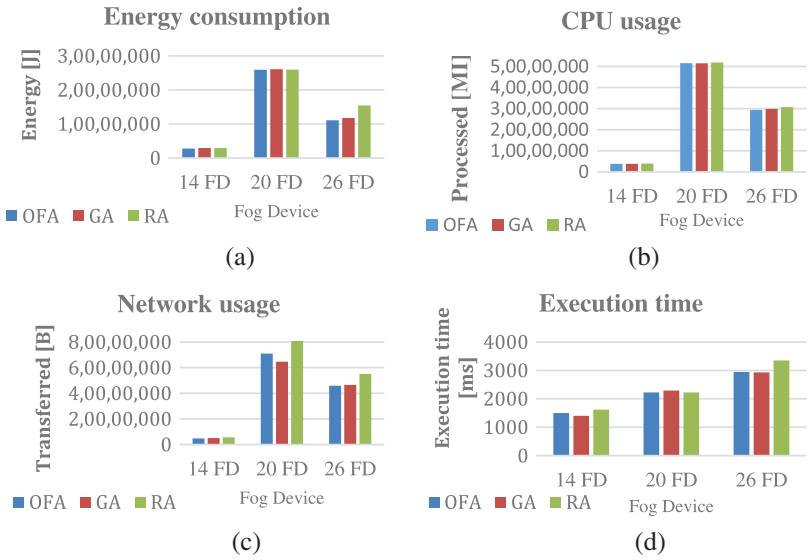

Figure 4: (a) Energy consumption (b) CPU usage (c) Network usage (d) Execution time

However, in certain trials, the OFA algorithm shows a slight lag compared to the GA algorithm. Table 8 shows the DCNS results.

Table 8: Simulation results in DCNS

	Energy (J)	Processed (MI)	Transferred (B)	Exe. time (ms)
14 Fog device (1 Cloud + 1 Proxy + 2 Area + 5 Device)				
OFA	15,861.56	4,563,130	36,850,500	2057
GA	15,864.51	5,142,063	68,814,200	2189
RA	15,869.62	5,743,583	54,250,000	1911
20 Fog device (1 Cloud + 1 Proxy + 3 Area + 5 Device)				
OFA	16,721.59	7,721,936	112,463,900	2887
GA	16,732.39	8,913,689	118,733,500	3230
RA	16,729.46	8,675,054	177,628,500	3575

(Continued)

Table 8 (continued)

	Energy (J)	Processed (MI)	Transferred (B)	Exe. time (ms)
	26 Fog device (1 Cloud + 1 Proxy + 4 Area + 5 Device)			
OFA	17,601.94	13,087,850	166,051,100	4757
GA	17,604.81	13,454,690	192,021,800	4764
RA	17,610.94	14,448,780	198,902,700	5162

Fig. 5 shows the comparison of energy consumption, operation costs, bandwidth cost, and execution time of the OFA algorithm with other algorithms. In DCNS, the OFA algorithm has better results than the others. In the comparisons made on systems with different numbers of fog devices; Fig. 5a shows that the OFA algorithm consumes less energy than other algorithms, even if it is a small amount. Fig. 5b shows that the OFA algorithm improves the CPU usage by 12%, 15%, and 2% compared to GA and 25%, 12%, and 10% compared to RA in the experiments with 14, 20, and 26 fog devices, respectively. Fig. 5c shows that the bandwidth usage improved by 86%, 5%, 15% according to GA, 47%, 57%, and 19% according to RA, respectively, in the experiment using 14, 20, 26 fog devices. Fig. 5d shows that it improves 6%, 11%, 0.14% compared to GA, −7%, 23%, and 8% compared to RA, respectively, in the experiment using 14, 20, 26 fog devices on run times. As a result, it is shown that the OFA algorithm is more efficient than both GA and RA in terms of energy consumption, bandwidth usage, and runtime, while it is slightly behind the RA algorithm in the experiment using 26 devices only in runtimes.

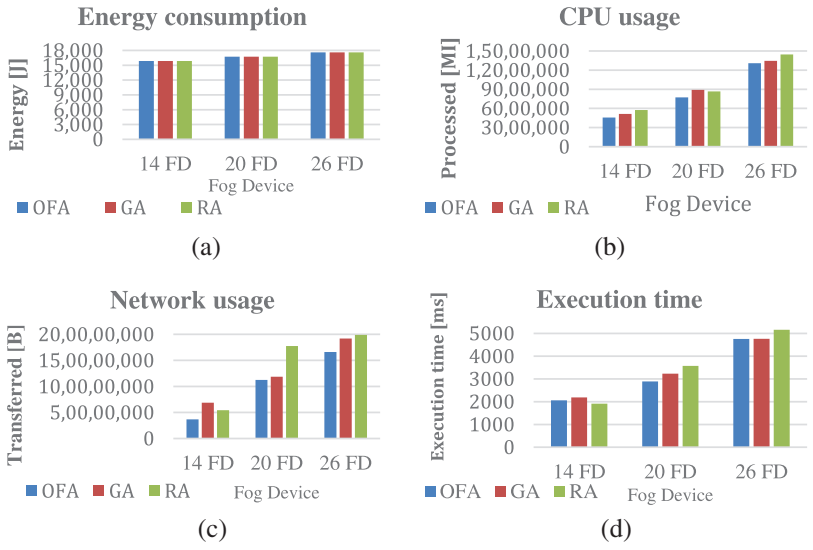

Figure 5: (a) Energy consumption (b) CPU usage (c) Network usage (d) Execution time

5 Discussion

The proposed algorithm is compared with the study conducted by Arora and Singh [32], where heterogeneous modules were sent to fog devices in the network. The comparison is made on the

"Intelligent Surveillance" application using the same configuration settings in the iFogSim [29] environment, with results presented in Table 9. The findings indicate that the average network usage improved by 36% in MB, and the average runtime improved by 37% in milisecond. In another study by Gavaber and Rajabzadeh [35], which focused on the efficiency of the system by adding fog devices in the fog layer, a comparison was made on the "EEG" application in terms of average energy consumption and application cycle delay. The results showed that the average energy consumption was reduced by 71% in MJ, and the application cycle delay decreased by 69% in milisecond. These values are also presented in detail in Table 9. It is evident that the proposed algorithm consistently reduces network usage across different applications, contributing to energy consumption reduction and affecting runtime positively. However, it is important to note that the increase in the number of nodes can lead to higher overall consumption values in any system. Since the constraints of the system used have not been tested in real-life scenarios, the extent to which it may impact the results differently remains unknown. Nevertheless, the authors state that future studies will include real-world trials.

Table 9: Literature comparisons

	OFA	Arora and Singh (2021) [32]	Gavaber and Rajabzadeh (2021) [35]
Network usage (MB)	3.82	6	–
Execution time (ms)	1015	1632	–
Energy consumption (MJ)	5.69	–	19.89
Latency of control loop (ms)	8.24	–	26.85

6 Conclusion

In this study, the OFA algorithm is proposed to facilitate efficient service placement in a fog computing network. The algorithm achieves an effective distribution of applications among fog nodes based on their respective application types. The main objective of this research is to enhance the system's performance in terms of energy consumption, network usage, processing cost, and runtime. To evaluate the proposed algorithm, comparative analyses were conducted with the GA and RA algorithms using the iFogSim simulation tool. Two different configurations were simulated to assess the algorithm's performance comprehensively. The results obtained through simulations demonstrate the effectiveness and significance of the proposed algorithm in outperforming the alternatives concerning energy consumption, network usage, processing cost, and runtime. The algorithm exhibits stability and maintains its efficiency even as workloads increase, displaying linear growth patterns. This study showcases the promising performance and capabilities of the proposed algorithm in the context of service placement within the fog computing environment. Future work aims to assess the algorithm's real-world application performance and further enhance the system's data security to ensure the confidentiality and integrity of shared data.

Acknowledgement: As author, I would like to thank my Ph.D. advisor of Gazi University and his advisory is Prof. Dr. Omer Faruk Bay for their support and assistance. This research has been produced from a part of Riza Altunay's Ph.D. dissertation in Gazi University, Institute of Informatics, and Department of Information Systems.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Study conception and design: R. Altunay, O. F. Bay; data collection: R. Altunay; analysis and interpretation of results: R. Altunay, O. F. Bay; draft manuscript preparation: R. Altunay. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available within the article or its supplementary materials. The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. L. Truong and S. Dustdar, "Principles for engineering IoT cloud systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, 2015.
[2] Y. Liu, J. E. Fieldsend and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, pp. 25445–25454, 2017. https://doi.org/10.1109/ACCESS.2017.2766923
[3] W. N. Hussein, H. N. Hussain, H. N. Hussain and A. Q. Mallah, "A deployment model for IoT devices based on fog computing for data management and analysis," *Wireless Personal Communications*, pp. 1–13, 2023. https://doi.org/10.1007/s11277-023-10168-y
[4] Y. I. Alzoubi, V. H. Osmanaj, A. Jaradat and A. Al-Ahmad, "Fog computing security and privacy for the Internet of Thing applications: State-of-the-art," *Security Privacy*, vol. 4, no. 2, pp. 1–26, 2021.
[5] Y. I. Alzoubi, A. Al-Ahmad and H. Kahtan, "Blockchain technology as a Fog computing security and privacy solution: An overview," *Computing Communucation*, vol. 182, pp. 129–152, 2022.
[6] A. M. Yadav, K. N. Tripathi and S. C. Sharma, "A Bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4236–4260, 2022.
[7] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei *et al.,* "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5068–5076, 2021.
[8] M. Keshavarznejad, M. H. Rezvani and S. Adabi, "Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms," *Cluster Computing*, vol. 24, no. 3, pp. 1825–1853, 2021.
[9] A. Mseddi, W. Jaafar, H. Elbiaze and W. Ajib, "Joint container placement and task provisioning in dynamic fog computing," *IEEE Internet Things Journal*, vol. 6, no. 6, pp. 10028–10040, 2019.
[10] J. U. Arshed, M. Ahmed, T. Muhammad, M. Afzal, M. Arif *et al.,* "GA-IRACE: Genetic algorithm-based improved resource aware cost-efficient scheduler for cloud fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2022, no. 7, pp. 1–19, 2022.
[11] A. S. Abohamama, A. El-Ghamry and E. Hamouda, "Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment," *Journal of Network and Systems Management*, vol. 30, no. 4, pp. 1–35, 2022.
[12] M. Ghobaei-Arani and A. Shahidinejad, "A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment," *Expert Systems with Applications*, vol. 200, pp. 117012, 2022. https://doi.org/10.1016/j.eswa.2022.117012
[13] K. Dubey, S. C. Sharma and M. Kumar, "A secure IoT applications allocation framework for integrated fog-cloud environment," *Journal of Grid Computing*, vol. 20, no. 1, pp. 5, 2022.
[14] P. Maiti, B. Sahoo and A. K. Turuk, "Low latency aware fog nodes placement in Internet of Things service infrastructure," *Journal of Circuits, Systems and Computers*, vol. 31, no. 1, pp. 2250017, 2022.
[15] A. A. Mohamed, L. Abualigah, A. Alburaikan and H. A. E. W. Khalifa, "AOEHO: A new hybrid data replication method in fog computing for IoT application," *Sensors*, vol. 23, no. 4, pp. 2189, 2023.

[16] M. Zare, Y. Elmi Sola and H. Hasanpour, "Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm," *Journal of King Saud University–Computer and Information Sciences*, vol. 35, no. 1, pp. 368–381, 2023.
[17] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski and P. Leitner, "Optimized IoT service placement in the fog," *Soca*, vol. 11, no. 4, pp. 427–443, 2017.
[18] F. A. Saif, R. Latip, Z. M. Hanapi and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.
[19] G. Y. Zhu and W. B. Zhang, "Optimal foraging algorithm for global optimization," *Applied Soft Computing Journal*, vol. 51, pp. 294–313, 2017. https://doi.org/10.1016/j.asoc.2016.11.047
[20] B. Costa, J. Bachiega, L. R. de Carvalho and A. P. F. Araujo, "Orchestration in fog computing: A comprehensive survey," *ACM Computing Survey*, vol. 55, no. 2, pp. 1–34, 2023.
[21] S. Sahil, S. K.Sood and V. Chang, "Fog-cloud-IoT centric collaborative framework for machine learning-based situation-aware traffic management in urban spaces," *Computing*, vol. 2022, pp. 1–33, 2022. https://doi.org/10.1007/s00607-022-01120-2
[22] L. Guo, "Application of blockchain based on deep learning algorithm in enterprise Internet of Things system," *Mobile Information Systems*, vol. 2022, pp. 9943452. https://doi.org/10.1155/2022/9943452
[23] M. Sirisha and P. Abdul Khayum, "Using a software-defined air interface algorithm to improve service quality," *Intelligent Automation and Soft Computing*, vol. 35, no. 2, pp. 1627–1641, 2023.
[24] J. C. R. Vieira, "Fog and cloud computing optimization in mobile IoT environments," M.S. Thesis, University of Tecnico Lisboa, Protekiz, 2019.
[25] V. Jain and B. Kumar, "QoS-aware task offloading in fog environment using multi-agent deep reinforcement learning," *Journal of Network and Systems Management*, vol. 31, no. 7, pp. 1–32, 2023.
[26] N. Mohan and J. Kangasharju, "Placing it right!: Optimizing energy, processing, and transport in edge-fog clouds," *Annales des Telecommunications/Annals of Telecommunications*, vol. 73, no. 7–8, pp. 463–474, 2018.
[27] A. Gupta, J. J. Cherukara, D. Gangadharan, B. G. Kim, O. Sokolsky *et al.,* "E-PODS: A fast heuristic for data/service delivery in vehicular edge computing," in *2021 IEEE 93rd Vehicular Technology Conf. (VTC2021-Spring)*, Helsinki, Finland, pp. 1–6, 2021.
[28] G. I. Sayed, M. Soliman and A. E. Hassanien, "Modified optimal foraging algorithm for parameters optimization of support vector machine," *Advances in Intelligent Systems and Computing*, vol. 723, pp. 23–32, 2018. https://doi.org/10.1007/978-3-319-74690-6_3
[29] W. B. Zhang and G. Y. Zhu, "Drilling path optimization by optimal foraging algorithm," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 2847–2856, 2018.
[30] G. I. Sayed, M. Solyman and A. E. Hassanien, "A novel chaotic optimal foraging algorithm for unconstrained and constrained problems and its application in white blood cell segmentation," *Neural Computing and Applications*, vol. 31, no. 11, pp. 7633–7664, 2019.
[31] G. I. Sayed, "A novel multilevel thresholding algorithm based on quantum computing for abdominal CT liver images," *Evolutionary Intelligence*, vol. 16, no. 2, pp. 439–483, 2023.
[32] U. Arora and N. Singh, "IoT application modules placement in heterogeneous fog–cloud infrastructure," *İnternational Journal of Information Technology*, vol. 13, no. 5, pp. 1975–1982, 2021.
[33] H. Gupta, A. V. Dastjerdi, S. K. Ghosh and R. Buyya, "Ifogsim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Software-Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.
[34] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Mendez, C. E. Chung *et al.,* "Augmented brain computer interaction based on fog computing and linked data," in *2014 Int. Conf. on Intelligent Environments*, Shanghai, China, pp. 374–377, 2014.
[35] M. D. Gavaber and A. Rajabzadeh, "MFP: An approach to delay and energy-efficient module placement in IoT applications based on multi-fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 7, pp. 7965–7981, 2021.