SN Computer Science (2024) 5:708
https://doi.org/10.1007/s42979-024-03036-4

ORIGINAL RESEARCH

An Efficient Approach to Reduce Energy Consumption in a Fog Computing Environment Using a Moth Flame Optimization Algorithm

Razieh Asgarnezhad[1]

Received: 27 May 2023 / Accepted: 4 June 2024

Abstract

After decades of growth in the computer computing field, cyber-physical systems (CPS), a combination of physical and tangible hardware and virtual and supernatural tools and concepts, distributed. Today, fog and cloud computing are the most complex cyber-physical systems available. Theretofore, the cloud data centers developed to provide the resources needed by users, home, and industrial businesses. Cloud computing has provided the possibility of providing services near the occurrence of requests with processing in proximity (PP). However, edge or fog computing will supply a possible solution to improve the quality of service delivery compared to using cloud computing. Applications in cyber-physical fog systems utilize different services provided by diverse resources in fog colonies based on criteria and restriction rules. Since internet of things (IoT) applications executed in real-time and sensitive to time, the problem of delay in providing service to application requests in cloud computing distributed resources is very challenging. Fog computing supply an ideal platform for CPSs with fully geographically distributed features. At first, by placing services on resources are located in the edge layer, the cloud decrease the volume of requests sent to the cloud. Moreover, the response and the average delay time be solved in the proposed method. As a result, it makes the problem of placing services in cloud computing more complicated than in other areas like cloud computing and standard distributed systems. In addition, to balance the load and ensure the quality of services, requested services in the fog system can be freely processed by any of the resources (nodes) available in the fog computing. According to the characteristics of the geographic distribution of fog nodes, the complexity of placing services to provide services to reduce energy consumption will be very high. In this study, we offer a solution based on the meta-heuristic algorithm of moth flame optimization (MFO) to place efficient energy and efficient delay of IoT services. The simulation results with iFogSim have revealed that the performance of the suggested solution has enhanced by 21% compared to the basic solutions in terms of energy consumption and service delivery delay by 15%.

Keywords Fog computing · Service placement · MFO algorithm · Node · Physical cyber · IoT

Introduction

After decades of development in the computer computing world, cyber-physical systems (CPSs) are more than ever distributed, which is a combination of both physical and discernible hardware and virtual and intangible tools and concepts. Cloud computing is known as one of the most complex physical cyber-sociability. The cloud and cloud data centers have already been widely welcomed to supply resources required by users and household and industrial jobs. Cloud computing presents services near the need for requests by proximity processing (PP). Regardless, on the edge or in fog, computing will provide a possible solution to enhance the quality of service compared to the use of cloud computing. Applications in physical cyber-sociability employ different resources provided by different sources in fog colonies based on criteria and constraints. Since Internet applications are most immediately executed and sensitive to time (time deadline), the delay issue is challenging in offering services to the request of fog computing holding containing. Fog can provide an ideal platform for the CPS with all distributed features in geography. First, the placement of services on the resources located on the edge layer near a place where the data is produced, fog reduces the number of requests sent to the cloud. By

✉ Razieh Asgarnezhad
razyehan@gmail.com

[1] Department of Computer Engineering, Aghigh Institute of Higher Education, Shahinshahr, Isfahan 8314678755, Iran

Published online: 10 July 2024

using this method, the question of the response time and the delay mean in service delivery will be resolved in fog.

As a result, the issue of service placement in cloud computing makes it more challenging to place more than other areas such as cloud computing, cloud computing, and standard distributed systems. in addition, to balance workload and guarantee the service quality, demand services in fog be system processed freely by any of the resources (nodes) degree of. Because of the geographic distribution characteristics of the nodes, the complexity of service placement to provide services to reduce energy consumption is very high. Here, we present a strategy based on the moth flame optimization (MFO) algorithm to locate the efficient energy and efficient delay of the internet services. simulation results with ifogsim have shown improvement progress of the proposed strategies compared to basic strategies in energy consumption by 21% and service delivery delay of 15%.

Cloud computing has made it possible to process requests and provide cloud services to customers through centralized cloud data centers. But the long distance and the bottleneck of Internet are two major problems of using cloud services because millions of IoT devices will anticipate to receive services from cloud data centers, which led to the overall disruption of the network infrastructure and ineffectiveness. Cloud computing will be in this matter.

The concept of fog computing was presented as an unavoidable infrastructure for managing the requests and workload of the IoT [1]. Today, fog and cloud computing is known as one of the most complex cyber-physical systems available. Before this, the cloud and cloud data centers have been widely welcomed in order to supply the resources needed by users and home and industrial businesses. But the cloud alone will not be able to support the future requirements of IoT devices [2].

In order to evaluate the efficiency and performance of the proposed method in this research and compare it with previous works, the following scenarios have been selected for testing and evaluation, which will be examined further. The compared methods In this research, we show a method called relax, which uses an energy-efficient service placement solution that takes into account the fog rate, load balancing, and service duplication [3].

The main motivations and contributions for conducting research on the problem of placing services in the fog with the aim of reducing energy consumption are as follows:

- Considering energy consumption and power consumption distribution in CPS environment and fog computing.
- Modeling the problem of energy consumption in fog computing in CPS infrastructure using MFO algorithm.
- Optimizing power consumption by considering placement issues under the conditions of guaranteeing service quality requirements.
- Providing algorithms with a light computational load in order to solve the complex problem of service placement in order to reduce energy consumption.

The rest of this paper is organized as follow. The author describes background and related work, respectively. And the author describes proposed framework and results and discussion. Finally, describes conclusion.

Background

Basic Concepts in Cloud Computing

In this section, we explain and describe the basic concepts used to place services in the cloud platform. Cloud computing is a highly virtualized platform that is capable of supplying services such as processing, storage space, and network services between end devices and traditional cloud computing data centers. According to Cisco's definition, fog computing is regarded as a form of cloud computing paradigm that has moved from the network core to the network edge. But in practice, the definition of fog computing is a scenario in which a large number of heterogeneous devices (wireless and sometimes independent), common (ubiquitous), and decentralized with the potential to collaborate within the network. In order to carry out storage and processing tasks without the intervention of third parties, they are connected with each other. These tasks can be to support general network operations or new services and applications that run in a secure environment. Users rent a part of their devices to host this service in exchange for receiving incentives.

Fog Computing and Related Components

Fog computing has many advantages over cloud computing by providing the capacity of local computing resources. Among the most important features of fog computing are low latency, spatial awareness, wide geographical distribution, mobility and portability, a very high number of nodes, an important role in wireless access, a strong presence in the field of content streaming and real-time application software. and heterogeneity pointed out [4]. Figure 1 shows a view of the structure and location of a fog computing system.

The place of cloud computing is somewhere between the cloud and the end user of the service. In fact, fog acts as a bridge and communication tool between the cloud and the end user, which can handle many computing and storage tasks (see Fig. 2).

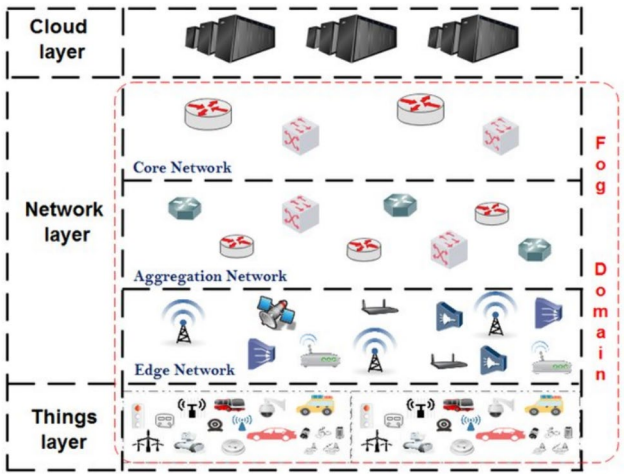


Fig. 1 Multi-layer framework of service delivery with the help of fog infrastructure [5]

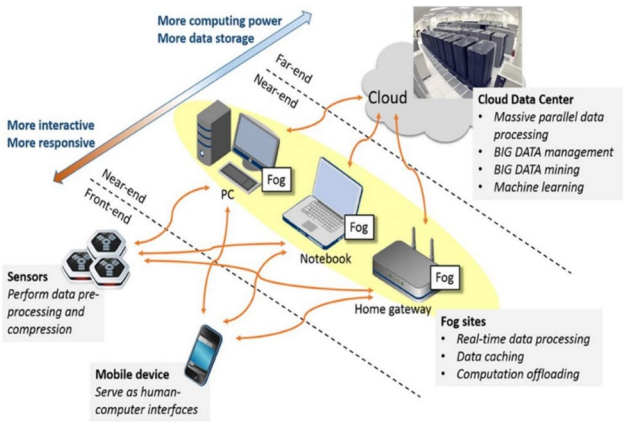


Fig. 2 Schematic of cloud, fog, and users [6]

As described in Fig. 2, in the Cloud-Fog-End structure, the more users move towards the cloud, the far-end of this structure increases the processing power and storage space. We are getting more and more and it becomes possible to process huge processing processes such as big data management and data mining and machine learning. But instead, we will face problems such as more delay, lack of awareness of the location and lack of mobility. But the more we move towards the fog, the speed of response in operations increases and the delay decreases, on the other hand, we will have more mobility. Also, application components such as real-time processing of data, caching them, and processing in the fog are possible.

Fog computing has been proposed to cover some of the limitations of cloud computing. For cloud computing, there are problems such as unreliable delay time, lack of mobility and lack of location awareness, and IoT programs, as one of the main users of cloud services, always support device mobility, geographical distribution. They require location awareness and low latency; which cloud computing does not offer. Fog computing has emerged to solve these problems [5].

Cyber-Physical Systems

Cyber-physical systems are online networks of common parts, equipment and machines that are connected to each other in the form of a virtual network. This communication can be established through physical interfaces such as copper cable and optical fiber or wirelessly. By connecting information technology to mechanical and electronic components, these systems provide the possibility of connecting components and machines with each other through the provided network [7].

Therefore, at this stage of industrialization, we are facing phenomena such as smart factory, smart power grid, smart house and building as important and central elements of the fourth industrial revolution. In practice, physical systems in the IoT platform communicate with each other as well as with humans in real time, share efforts and cooperate, and through the Internet platform, the information and services needed within the organization and outside the organization are sent to the systems. Mechanized management and employees inside and outside the factory send and make available to use for decisions [8].

CPSs are intelligent systems that include engineered networks with the ability to interact with physical and computational components (based on algorithms) [2]. These systems are highly interconnected and integrated, provide new functions to improve and improve the quality of life and lead to the advancement of technology in critical areas such as individual health care, emergency response, traffic flow management, smart manufacturing and national security. and defense and energy production and consumption.

The real value of the IoT is determined when it is possible to receive the data generated by the sensors, devices, machines and terminals of the IoT through the predicted systems. interpreted and processed and finally gave the necessary commands to the appropriate operators [1, 9]. In other words, the real value of the IoT for manufacturers is in the analysis that results from the cyber-physical models of machines and systems. In the fourth generation industry, the systems that can add this added value to the IoTs are the CPSs.

The cyber-physical system usually refers to the systems consisting of computing components that work together and cooperatively to use the feedback that they usually receive from the monitored sensors or the feedback that is given as a command to operators send to control the physical world.

The MFO Algorithm

The MFO algorithm is a new optimization algorithm inspired by nature, which was first proposed by Seyed Ali Mirjalili in 2015 at the University of Australia [10]. The main inspiration of this optimizer is the method of aviation (flight) of moths in nature called transverse orientation. Moths fly at night keeping a constant angle to the moon, which is a very effective mechanism for moving in a straight line for long distances. However, these extraordinary insects are trapped in a useless and deadly spiral around artificial light. The MFO algorithm is compared with known nature-inspired algorithms in 29 benchmarks and 7 real engineering problems. The statistical results in benchmark functions show that this algorithm is capable of providing very promising and competitive results. In addition, the results of real problems show the advantages of this algorithm in solving challenging problems with limited and unknown search spaces. Figure 3 shows the rotational movement of moths around a flame.

At the end, if the exit condition of the algorithm is satisfied, the best result that is available to the insect or the flame with the best performance in terms of the efficiency function will be selected and returned as the answer. Then, based on the placement of the desired node on the selected computing node, the insect matrices and flames are updated so that it can work accurately for other nodes in the remaining network without a resource.

The moth flies around the flame at a fixed angle at night. If the moths see the flame, they continue their flight in a straight path towards the flame. When the moth gets too close to the flame source, it flies around the flame source. Figure 4 shows the movement of the moth in search of flame.

Another interesting point of this solution is the ability of the algorithm to focus over time on the best solutions found,

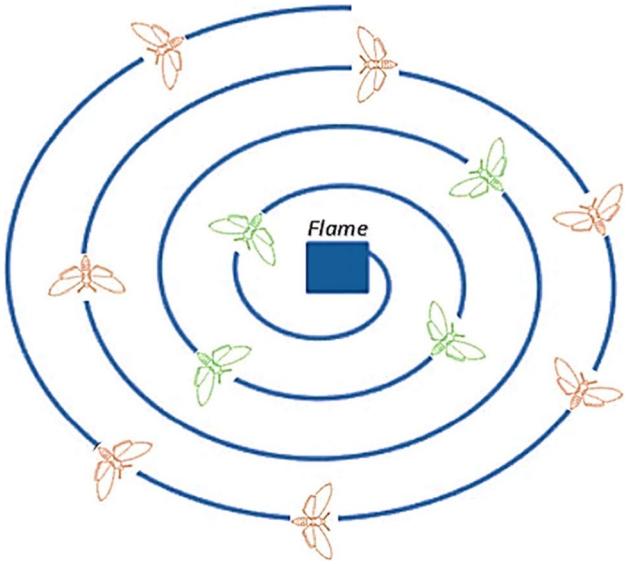


Fig. 4 The position of a moth in the first and last intervals of the MFO algorithm cycle [12]

in such a way that the number of propellers decreases in each sequence and the speed of moving to the optimal path increases. Also, with this technique, in the last interval of the algorithm execution sequence, we will have only one moth, which actually includes the optimal solution. Figure 5 shows how to change the number of active blades in the overall cycle of the MFO algorithm. In this example, which has a cycle with 100 rounds of different sequences, it is shown that in the first round of the algorithm, it starts working with 20 moths and over time the number of searching

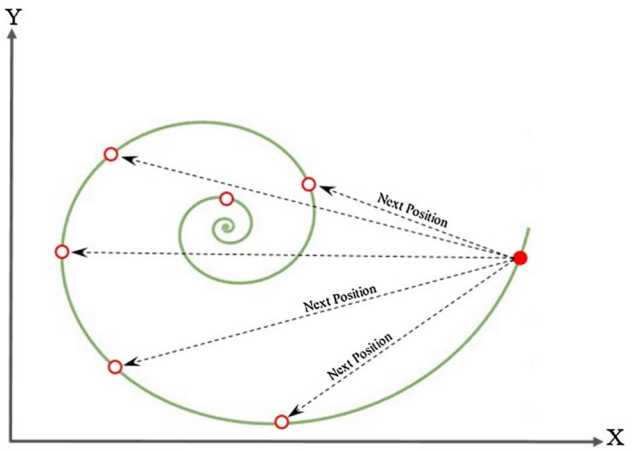


Fig. 3 Schematic of the rotational movement of moths around a flame [11]

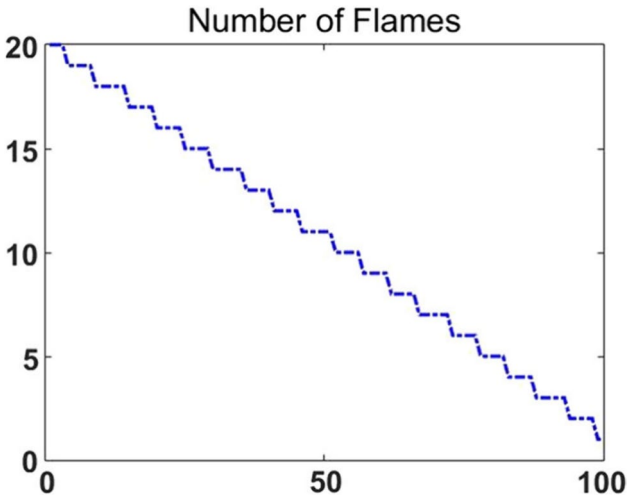


Fig. 5 How to change the number of active blades in the overall cycle of the MFO algorithm [11]

moths decreases, and finally in the last round. There is only one search engine that has an optimal solution.

Moths are search agents and flames are the best position obtained so far, so all the population consider this position as the answer. The MFO algorithm has a set of moths that can be displayed in a matrix as follows.

$$M = \begin{bmatrix} M_{1,1} & \cdots & M_{1,d} \\ \vdots & \vdots & \vdots \\ M_{n,1} & \cdots & M_{n.d} \end{bmatrix} \quad (1)$$

In Eq. (1), n shows the number of moths and d shows the number of variables of the problem. Another matrix for storing the fit values of answers, this matrix shows the level of service quality. OM matrix is expressed in Eq. (2).

$$OM = [OM_1 OM_2 ... OM_n]^T \quad (2)$$

On the other hand, flames are also other components in this algorithm. They can be displayed using the matrix presented in Eq. (3).

$$F = \begin{bmatrix} F_{1,1} & \cdots & F_{1,d} \\ \vdots & \vdots & \vdots \\ F_{n,1} & \cdots & F_{n.d} \end{bmatrix} \quad (3)$$

In Eq. (3), n is the number of moths and d is the dimensions of the problem or the number of variables of the problem. It should be noted that the dimensions of the matrices M and F are equal to each other. The OF matrix is also used as the fitting values for the propellers mentioned in Eq. (4).

$$OF = [OF_1 OF_2 ... OF_n]^T \quad (4)$$

The overall structure of the MFO algorithm is determined using an approximation function with three parameters:

$$\mathrm{MFO} = (\mathrm{I},\ \mathrm{P},\ \mathrm{T}) \quad (5)$$

In Eq. (6), I is a function that initializes the initial population of moths.

$$\mathrm{I} : \emptyset - \rightarrow \{\mathrm{M},\ \mathrm{OM}\} \quad (6)$$

In Eq. (7), P is also a function that moves the propellers based on the relation presented in Eq. (5).

$$\mathrm{P} \ : \ \mathrm{M} \rightarrow \mathrm{M} \quad (7)$$

The last function used in Eq. (5) which T is function. This function returns True value if the end condition of the algorithm is satisfied. Equation (8) shows the T function.

$$\mathrm{T} \ : \ \mathrm{M} - \rightarrow \{\mathrm{true},\ \mathrm{false}\} \quad (8)$$

Moths and flames are the main parts of the MFO algorithm. The moths move around the search space, while the flames show the best position obtained by the moths. Moths fly around the flames and update their position by finding better answers.

Related Work

In the field of distributed computing, fog infrastructure is considered as a new concept for providing processing services, so far, many researches have been done on the architecture and framework of fog infrastructure, some of the recent researches are stated in this section.

The cloud is closer to the user to perform computing, communication and storage tasks on network edge devices. The capabilities of its services are close to the end users. This is the most basic feature of fog computing and its most important advantage compared to other traditional computing models [3].

The main ideas and theoretical foundations about computing in the fog, such as dealing with the basic fog architecture, application and data communication, have already been established, but there is a need for strong solutions to provide resources and virtualize these resources and how to distribute the service. IoT is based on available resources in fog computing [13].

Bonomi et al. [14] placed computing resources at the edge of Internet networks instead of transferring requests to remote cloud resources. By moving cloud services to the edge of the network, it is possible to use local infrastructure and equipment to share the workload in the cloud. The authors in 2018 concentrated on optimal selection of devices. This work showed a non-cooperative comprehensive model to devaluation of response time. Additionally, they produce a model to improve the battery life of the computational task receivers. The proposed model performed backward induction technique. They estimated the performance of the suggested model upon response time, end-users utility and memory utilizations [15]. Zhou et al. [16] have presented a solution for using fog and cloud computing to provide a management and health care system for the elderly. By using cloud computing and fog computing for health care and elderly care using a The practical scenario implemented in the OpSIT-Project in Germany has been presented.

Vaquero et al. [17], by examining the compatibility conditions of users, made it possible to improve website performance by using the edge instead of central cloud servers. For this purpose, by examining and using the current network, the situation, the workload, the authors have provided this possibility in an adaptable way according to the users' conditions. Do et al. [18] presented a new solution to minimize the production of greenhouse gases by solving the common problem of resource allocation in fog computing. A proposed solution showed that by

transferring the requests to the fog sources, it is possible to prevent the excessive production of greenhouse gases.

Wang et al. [19] propose a solution for moving dynamic services at the edge by migrating dynamic services among edge nodes in order to design optimal service moving policies to reduce cost. Deng et al. [20] mathematically formulated the workload allocation problem and created an approximate solution to maintain telecommunications and reduce propagation delay. Vaquero et al. [17] paid attention to different concepts for distinguishing fog architectures, including centralized and decentralized and peer-to-peer approaches. In particular, these authors introduced the concept of edge cloud, which includes private fog and, like the concept of fog colony that we have presented, covers IoT equipment.

Colistra et al. [21] used consensus algorithms to allow devices to cooperate in the problem of distributed resource allocation so that they can adequately share resources. Yang et al. [22] proposed a method based on three phases: criterion specification, evaluation, and testing presented using DEBTS algorithm. The results of the research showed that the authors' solution helps the developers to do their work based on the application requirements and optimize the performance and the amount of use of the available resources. Scarlett et al. [23] presented the concept of fog colonies for the optimization process of order series. GA algorithm is used. Each colony uses a GA to make decisions regarding the services that are placed in the colony and determine which of these services are transferred to the adjacent colonies.

Tanizha and Davi et al. [24] attempted to present the service placement algorithm for efficient use of the network and power consumption, using the OWN algorithm. This algorithm sequentially allocated the application modules with the highest needs to the nodes with the highest capacities, which examined the amount of network consumption and the amount of power consumption and delay in the network. Brogi and his colleagues [25] presented a model for using QoS systems in multi-part IoT applications in fog architecture. The results of the authors' research showed that the presented solutions compared to the previous solutions in order to allocate resources and the amount of resource consumption have led to the improvement of energy consumption.

Azam et al. [26] proposed a more complex resource provisioning mechanism based on forecasting resource needs. In this work, the dynamic allocation of resources is done during the system design time. This approach is based on cost optimization and the allocation of resources depends on possible fluctuations in demand by users, types of services and pricing models. Han et al. [27] have mentioned an approach that affects the conceptual architecture of the fog computing environment and the modeling of IoT applications.

Wagler et al. [28] presented a policy-based approach to optimize topologies running on edge devices. This approach offers a flexible application implementation by defining a "hot pool" of resources for each on-demand application service to enable incremental scaling. Hong et al. [29] present a scheduling model including a simple resource provisioning strategy that focuses on the workload threshold; That is, if the usage of a particular fog cell exceeds a certain value, another fog cell will be used.

In 2019 [30], Lin et al. presented an energy-efficient task placement algorithm with the help of nodes on the edge of fog computing in order to provide green fog computing. In this research, the authors have addressed the problem of minimizing energy consumption costs in order to model the problem and solve it. Also, the authors simultaneously addressed the issues of resource allocation, service migration, and energy consumption scheduling, and the presented solution has led to the improvement of electricity consumption costs. In 2019, Manoz et al. [31] addressed the issue of placing sensitive services in cyber-physical applications. The authors tried to use and present an effective plugin for the Android operating system for the optimal use of energy consumed by cyber-physical applications on mobile devices, which led to a reduction in power consumption.

Zeng et al. [3] formulated workload allocation problems and developed an accurate resource allocation algorithm. The solution provided by the authors leads to less bandwidth consumption and thus reduces the request execution delay. In 2018, Mahmoud et al. [32] reviewed the best practices in the field of cyber-physical fog computing and an efficient energy heuristic algorithm for placing services and tasks of cyber-physical applications on fog resources before use from cloud sources. The proposed solution to the implementation of all requested tasks of applications on the cloud led to a reduction in energy consumption.

In 2019 [33], Demai et al. presented an energy-efficient algorithm for placing IoT services on the cloud platform using the distributed PSO algorithm. The proposed solution compared to the standard PSO algorithm led to the improvement of energy consumption of fog resources in the fog infrastructure. Mehran et al. [34] attempted multi-objective modeling of the problem of placing IoT services in the cloud platform in such a way that an infinite number of active fog resources are available, objectives such as work completion time, costs Considered financial, energy consumption, and other communication criteria. Then the authors presented a multi-objective algorithm with the priority of reducing operational costs, which has led to a greater reduction of operational costs compared to other single-objective algorithms.

Since the proposed model will be secure and sustainable for applying in distributed environments.

Digital twin (DT) authorized IoT develops a huge amount of data sent to the edge servers. But, unpredictable

public communication channels and lack of trust among participating entities yields diverse types of attacks on the continuous communication [35]. The authors presented a blockchain and Deep learning (DL) combined framework for providing decentralized data processing and learning in IoT network. Their framework offered a new DT model to emulate and replicate security-critical processes of IoT. Then, they suggested a blockchain-based data transmission scheme that utilizes smart contracts to guarantee integrity and authenticity of data. Ultimately, the DL scheme designed to apply the intrusion detection system (IDS) against valid data recovered from blockchain. In this scheme, a long short term memory-sparse autoencoder (LSTMSAE) technique offered to learn the spatial–temporal representation. The extracted characteristics are further employed by the suggested multi-head self-attention (MHSA)-based bidirectional gated recurrent unit (BiGRU) algorithm to learn long-distance features. The empirical implementation of their suggested framework establishes significant enhancement of communication security and data privacy in DT empowered IoT network [36].

The smart villages can promote real-time data analytic and automate decision making for local villagers. Nevertheless, all wireless sensing devices exchange information utilizing public network and may not be able to resist all forms of attacks [37]. To address this problem, authors proposed a new network architecture called distributed fog computing (DFC) be designed and integrated with IoT-based smart villages deployment. Also, they designed and evaluated the performance of an intrusion detection system (IDS) in DFC-based smart village environment (Table 1). Eventually, they discussed several open security issues and challenges regarding fog-to-things enabled smart villages [37].

Proposed Framework

In this research, an energy-efficient method is presented for the placement of fog services. The suggested method in this research uses the Moth-Flame meta-heuristic algorithm. The purpose of this research is to optimize energy consumption in the problem of placing customer services. In the following, we will state the problem and its formulation, and then present the features of Moth-Flame meta-heuristic and population-based algorithm. After that, we will present and express the proposed solution. All the symbols used to express the problem and solve it are collected in Table 2.

Formulation and Model of the Problem System

In this section, the mathematical expression and model of the problem system are discussed. For this purpose, the system model of the problem will be expressed first, then the mathematical expression of the problem, inputs, outputs, constraints and the optimization function of the problem will be presented.

A fog colony F connected to cloud computing environment C is assumed. One of the most important and basic elements in the assumed system is the fog colony. Each fog colony contains a subset of IoT devices, which include two different categories:

(A) "thin" devices of the IoT that lack any computing power and are only sensors or actuators.
(B) "Fat" IoT equipment, i.e. fog cells and fog coordination control nodes that have the computing power and can be virtualized.

It controls the functional fog cells known as Res(F) sets, which also include IoT devices capable of executing requests.

The assumed fog environment has n fog colonies in such a way that we denote the ith fog colony by Fi. Each fog colony Fi has a head cell denoted by $f_{i,H}$ and a set of fog computing cells that $f_{i,j} \in Res(F_i)$, $1 \leq j \leq Cf_i$ is All communication in the fog colony is done through fog coordination control cells (fog cells). The communication links between the fog coordination control cell and a computing cell in the fog colony have been provided with a delay.

The head fog cell in each colony is responsible for managing, optimizing, and controlling other fog computing nodes and scalability of fog computing resources, communications between fog colonies, and placement of IoT devices services.

In this research, the issue of placing services of IoT devices in colonies is considered the main issue. However, for this purpose, the placement and use of cloud computing resources and communication between fog colonies are also considered to increase the optimality of the proposed solution in such a way that the connection delay between each computing cell and the head cell in the colony is determined by $d_{i,j}$ has been Also, the delay rate of both fog cells $f_{i,j}$ are equipped with sensors and actuators that bring the ability to be programmed. In the assumed network, all fog cells in a colony have network connections, a complete mesh (which means direct communication between all cells is possible). The efficiency of the processor, main memory, and secondary memory of the fog cell $f_{i,j}$ in the current time interval with the vectors $Uf_{i,j}(\Delta t)$, $Mf_{i,j}(\Delta t)$ and $Sf_{i,j}(\Delta t)$ we show. The average productivity of the processor, main memory and secondary memory of each fog colony is determined by $Uf_i(\Delta t)$, $Mf_i(\Delta t)$ and $Sf_i(\Delta t)$, respectively. The communication delay between two fog colonies i1 and i2 is equal to the amount of data transmission delay between the head cells of two fog colonies, which is represented by $d_{i1}(i2)$. Therefore, the amount of data transmission delay from computing cell

Table 1 The comparison of the reviewed works in this context

Ref./Year/Authors	Approach
[13]/Ali and Ghazal/2017	Proposing a strong solutions to provide resources and virtualize these resources and how to distribute the service in fog computing
[15]/Tiwary et al./2018	Providing a non-cooperative comprehensive model to devaluation of response time and improve the battery life of the computational task receivers
[16]/Zhu et al./2013	Presenting a solution for using fog and cloud computing to provide a management and health care system
[17]/Vaquero and Rodero-Merino/2014	Improving website performance by using the edge instead of central cloud servers
[19]/Wang et al./2015	Proposing a solution for moving dynamic services at the edge by migrating dynamic services among edge nodes in order to design optimal service moving policies to reduce cost
[20]/Deng et al./2015	Formulation the workload allocation problem and created an approximate solution to maintain telecommunications and reduce propagation delay
[22]/Yang et al./2018	Proposing a method based on three phases: criterion specification, evaluation, and testing presented using DEBTS algorithm
[23]/Skarlat et al./2017	Presenting the concept of fog colonies for the optimization process of order series using GA algorithm
[25]/Brogi et al./2017	Presenting a model for using QoS systems in multi-part IoT applications in fog architecture to the improvement of energy consumption
[30]/Gu et al./2019	Presenting an energy-efficient task placement algorithm with the help of nodes on the edge of fog computing in order to provide green fog computing. The authors have addressed the problem of minimizing energy consumption costs in order to model the problem, resource allocation, service migration, and energy consumption scheduling
[31]/Munoz et al./2019	Considering the issue of placing sensitive services in cyber-physical applications and having reduction in power consumption
[3]/Zeng et al./2020	Formulation workload allocation problems, Developing an accurate resource allocation algorithm, Providing less bandwidth consumption and reducing the request execution delay
[32]/Mahmoud et al./2018	Providing an efficient energy heuristic algorithm for placing services and tasks of cyber-physical applications in the field of cyber-physical fog computing and reduction in energy consumption
[33]/Djemai et al./2019	Presenting an energy-efficient algorithm for placing IoT services on the cloud platform using the distributed PSO algorithm to the improvement of energy consumption of fog resources in the fog infrastructure
[34]/Mehran et al./2019	Proposing multi-objective modeling of the problem of placing IoT services in the cloud platform in conjunction with work completion time, costs considered financial, energy consumption, and other communication criteria
[35]/Kumar et al./2022	Developing a huge amount of data sent to the edge servers. Unpredictable public communication channels and lack of trust among participating entities yields diverse types of attacks on the continuous communication
[36]/Kumar et al./2022	Presenting a blockchain and DL combined framework for providing decentralized data processing and learning in IoT network
[37]/Aljuhani et al./2022	Promoting real-time data analytic and automate decision making for local villagers. Proposing a new network architecture and integrated with IoT-based smart villages deployment

j1 in the colony Fog i1 and calculation cell j2 in fog colony i2 is calculated as follows:

$$d_{i1,j1}(i2,j2) = d_{i1,j1} + d_{i1}(i2) + d_{i2,j2} \tag{9}$$

Also, the communication cost between Fog and Cloud is shown by the symbol d_C. In each time interval Δt, the ith fog colony receives m(Δt) IoT cyber-physical applications:

$$A = \left\{A_1, A_2, A_3, \ldots, A_{m(\Delta t)}\right\} \tag{10}$$

where A_k shows the IoT program number k, which has a deadline of $w(A_k)$ and has a set of requests (services) to execute:

$$A_k = \left\{a_{k,1}, a_{k,2}, a_{k,3}, \ldots, a_{k,mr(\Delta t)}\right\} \tag{11}$$

So that mr(Δt) is the number of requests (services) of A_k in the time interval Δt. The amount of processor, main memory, and storage of request $a_{k,s}$ is specified by $U(a_{k,s})$, $M(a_{k,s})$ and $S(a_{k,s})$, respectively. Also, the sum of the total processor, main memory and disk requested by the kth application services are specified by $U(a_k)$, $M(a_k)$ and $S(a_k)$. Every request service is assumed that the type of service can be one of three: (a) read and write service, (b) startup service, (c) processing service. The service type of request $a_{k,s}$ is specified by $T(a_{k,s})$.

Based on the specifications of requested services, an A_k program can be placed only on a subset of fog computing cells. In order to display the appropriate subset for placing the service $a_{k,s}$ of the vector including binary variables

Table 2 The used symbols

Symbol	Explain
C	Cloud computing environment
F	Fog environment
Res(F)	All fog cells
N	All colony cells
Fi	The ith colony of the fog
Cf_i	The number of fog computing cells in the ith colony
$f_{i,j}$	The jth of fog computing cells in the ith colony
$Uf_{i,j}(\Delta t)$	CPU efficiency of the jth computing cell in the ith colony
$Mf_{i,j}(\Delta t)$	The efficiency of the main memory of the jth computing cell in the ith colony
$Sf_{i,j}(\Delta t)$	The efficiency of the disk of the jth computing cell in the ith colony
$Uf_i(\Delta t)$	The average processor efficiency of fog computing cells in the ith colony
$Mf_i(\Delta t)$	The average efficiency of the main memory of fog computing cells in the ith colony
$Sf_i(\Delta t)$	The average disk efficiency of fog computing cells in the ith colony
$d_{i1}(i2)$	Delay between two colonies i1 and i2
$d_{i1,j1}$	The communication delay between the head cell and the fog computing cell j1 in colony i1
$d_{i1,j1}(i2,j2)$	Delay between two computing cells j1 in colony i1 and computing cell j2 in colony i2
A_k	kth cyber-physical application in the ith fog Colony
$m(\Delta t)$	The number of cyber-physical applications in the ith fog colony in the time interval Δt
$a_{k,mr(\Delta t)}$	The last requested service in the kth cyber-physical applications in the ith colony in the time interval Δt
$U(a_{k,s})$	The requested processor of the sth service in the kth application in the ith colony
$M(a_{k,s})$	The main memory of the requested service type of the sth service in the kth application in the ith colony
$S(a_{k,s})$	The requested disk of the sth service in the kth application in the ith colony
$T(a_{k,s})$	The requested service type of the sth service in the kth application in the ith colony
Ex_{A_k}	The running time of the kth application

$X_{a_{k,s}} = \left\{ X_{a_{k,s}}^{F_{i,1}}, X_{a_{k,s}}^{F_{i,2}}, X_{a_{k,s}}^{F_{i,3}}, \ldots, X_{a_{k,s}}^{F_{i,Cf_i}} \right\}$ we use. The binary value of each member of the vector $X_{a_{k,s}}^{F_{i,j}}$ is equal to:

$$X_{a_{k,s}}^{F_{i,j}} = \begin{cases} 1 \ if \ U(a_{k,s}) \le Uf_{i,j}(\Delta t) \ and \ M(a_{k,s}) \le Mf_{i,j}(\Delta t) \ and \ S(a_{k,s}) \le Sf_{i,j}(\Delta t) \\ 0 \ otherwise \end{cases} \quad (12)$$

Now, if the value of $X_{a_{k,s}}^{F_{i,j}}$ for $a_{k,s}$ is equal to 1, it is possible to place the desired service on fog cell $F_{i,j}$. Therefore, if the service $a_{k,s}$ is located on the fog cell $F_{i,j}$, the element $X_{a_{k,s}}^{F_{i,j}}$ will be equal to 1, otherwise, it will be equal to 0.

The amount of energy consumed by a fog cell is divided into two parts: static energy consumption and dynamic energy consumption, in such a way that when the fog cell is active, static energy consumption will be used. In addition, dynamic energy will be consumed based on the number of requests assigned to the fog cell and the efficiency of the processor and other resources of the fog cell. The amount of energy consumption of fog cell j on the ith colony is calculated as follows:

$$E_{i,j}(\Delta t) = ES_{i,j}(\Delta t) + ED_{i,j}(\Delta t) \quad (13)$$

$$ES_{i,j}(\Delta t) = \begin{cases} StaticEnergy(F_{i,j}) \ if \ F_{i,j}(\Delta t) \ is available \\ 0 \ otherwise \end{cases} \quad (14)$$

The amount of dynamic energy consumption of each fog cell is calculated based on the efficiency of its processor in the form of a linear equation based on the ceiling of dynamic energy consumption of the desired cell. Equation (15) shows the dynamic energy consumption of the jth fog cell in the ith fog colony.

$$ED_{i,j}(\Delta t) = Uf_{i,j}(\Delta t).MDE(f_{i,j}) \quad (15)$$

Equation (15), $MDE(f_{i,j})$ shows the maximum dynamic energy consumption of the jth fog cell in colony i, if the processor efficiency of this fog cell is 100%, the energy consumption will be the maximum. Otherwise, it will be calculated based on the productivity of $Uf_{i,j}(\Delta t)$ in the current period.

In addition to the energy consumption cost of the fog cells, based on the placements and communication cost between the fog cells and cyber-physical applications, we calculate the communication cost:

$$EC(\Delta t) = \sum_{a_{k,s}=a_{k,1}}^{a_{k,mr(\Delta t)}} X_{a_{k,s}}^{F_{i,j}}.CD_{a_{k,s}}^{F_{i,j}}.CE_i \tag{16}$$

As shown in Eq. (16), if the sth IoT service belonging to the kth application is placed on the jth cell in the ith colony in the time interval Δt, the value of the variable $X_{a_{k,s}}^{F_{i,j}}$ will be equal to 1 and otherwise will be equal to 0. Therefore, if the placement has been done, with the help of the delay of the transmission of the sth Internet of Things service belonging to the kth application to the jth cell in the colony i, the time interval Δt is calculated and by multiplying it by the basic unit of energy consumption from Exchanges in the ith colony network CE_i will be calculated for the communication cost of placement and transmission of the service in question.

Based on the amount of energy of each colony in the time interval Δt, it is possible to calculate the total energy consumption of each fog cell as well as the total energy consumption of all fog cells and finally the total cost of the fog infrastructure, taking into account the cost of communication in the entire interval calculated as follows:

$$E_{i,j} = \sum_{\Delta t} E_{i,j}(\Delta t) \tag{17}$$

Equation (17) shows the total energy consumption of the jth cell in the ith colony in all the time intervals of its activity.

$$TEF = \sum_{i} \sum_{j} E_{i,j} \tag{18}$$

Equation (18) calculates the total energy consumption of all fog cells in all fog colonies for all time intervals of fog cell activity.

$$TEC = \sum_{\Delta t} EC(\Delta t) \tag{19}$$

Equation (19) calculates the total energy consumption of communications resulting from the placement and transmission of IoT services on fog cells for all time intervals of fog cell activity.

Finally, the total amount of energy consumed by the implementation of fog cells and communications within the fog infrastructure is calculated based on Eq. (20):

$$E = TEF + TEC \tag{20}$$

Based on what was stated above, the objective function of the stated problem is to minimize the amount of energy consumed while maximizing the number of services placed on the fog computing cells in a way that satisfies the application deadline. Do:

$$MIN(E) and MAX\left(\sum_{\Delta t=1}^{x} \sum_{A_k=a_1}^{a_{m(\Delta t)}} \sum_{a_{k,s}=a_{k,1}}^{a_{k,mr(\Delta t)}} \sum_{F_i=1}^{n} \sum_{j=1}^{Cf_i} X_{a_{k,s}}^{F_{i,j}}\right) \tag{21}$$

$$Ex_{A_k} \leq w(A_k) \forall A_1 \leq A_k \leq A_{m(\Delta t)} \text{ and } \Delta t \geq 1 \tag{22}$$

$$\sum_{a_{k,s}=a_{k,1}}^{a_{k,mr(\Delta t)}} X_{a_{k,s}}^{F_{i,j}} * U(a_{k,s}) < CpuCapacity(F_{i,j}), \forall i, j, \Delta t \tag{23}$$

$$\sum_{a_{k,s}=a_{k,1}}^{a_{k,mr(\Delta t)}} X_{a_{k,s}}^{F_{i,j}} * M(a_{k,s}) < RamCapacity(F_{i,j}), \forall i, j, \Delta t \tag{24}$$

$$\sum_{a_{k,s}=a_{k,1}}^{a_{k,mr(\Delta t)}} X_{a_{k,s}}^{F_{i,j}} * D(a_{k,s}) < DiskCapacity(F_{i,j}), \forall i, j, \Delta t \tag{25}$$

$$1 \leq i \leq n \tag{26}$$

$$1 \leq j \leq Cf_i \tag{27}$$

$$a_1 \leq A_k \leq a_{m(\Delta t)} \tag{28}$$

$$a_{k,1} \leq a_{k,s} \leq a_{k,mr(\Delta t)} \tag{29}$$

System Model and Framework Used

In this section, the framework used to place services in the cloud infrastructure is presented. The framework of the proposed method is based on the ring of two management cores to check the status of the fog colony and the fog resource allocation table for IoT services. This section will first examine the architecture of the fog and cloud surface and the arrangement and tasks of fog cells and IoT devices, and then examine the presented framework.

Figure 6 shows the framework of the desired problem. As it has been determined, our assumed system includes N different fog colonies that will be connected in the fog layer. Each fog colony has a management cell (head) that is responsible for managing, monitoring, and controlling

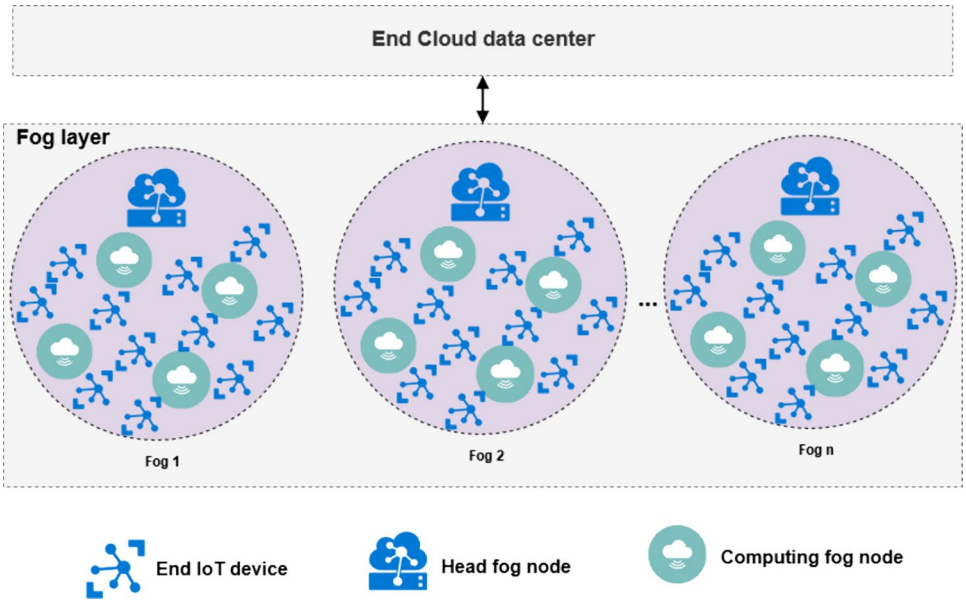


Fig. 6 Fog and cloud architecture in the problem of placement of cyber-physical application services

requested services and active cells in the colony. Each head cell contains two main sections, check status and allocation map, which are shown in detail in Fig. 7. As it is clear in the figure, every cyber-physical application will be able to receive services from various fog cells, and the task of managing and coordinating the use of fog cells will be determined by the head fog node.

Figure 7 shows the framework used to place fog services in the fog infrastructure using the Moth-Flame algorithm. In Fig. 7, one of the colonies of head fog assumed in Fig. 3 is depicted. As it has been determined, the head fog node places the services using the moth flame algorithm. For this purpose, at first, requests are received in the form of service from cyber-physical applications through the gateway. Then, the system status and active services are checked (check status). After checking the situation with the energy efficient MFO technique, services are placed on fog resources using allocation MAP.

For this purpose, at first, the head cell receives input data from cyber-physical applications including requested services. It will also receive the state of fog cells in its colony and the state of other neighboring colonies in the observation phase.

Analysis of the current state of the current colony and other nearby colonies will be done based on the received data. A set of fog cells are selected to serve the requests received from the IoT based on the analysis of the existing fog situation. After performing several generations of Moth Flame population-based algorithm implementation and changing the location of insects and creating and updating luminous points, the best insects will be selected as the chosen solution for placing services.

The Presented Solution of Energy-Efficient Placement of Fog Service with MFO Algorithm

In this section, the presented algorithms are compiled based on the presented framework. In this way, Algorithm 1 shows the general framework of managing and placing fog services with the MFO algorithm. As it is known, the desired algorithm checks the status of fog cells, cyber-physical applications, and requested services in check status and is expressed as the inputs of the problem.

Then the energy-efficient MFO algorithm, to minimize the energy consumption of fog cells and the energy consumption of communication, determines the placement destination of IoT services based on the inputs and the state of the infrastructure, and the output and service placement map. IoT is returned as the output of the solution in line 4.

In the end, in line 5, the operation of placing the IoT services based on the allocation MAP obtained with the MFO algorithm is done, and each of the IoT services is placed on one of its colony fog cells or clone fog cells. Neighboring fogs or on the cloud bed will be executed.

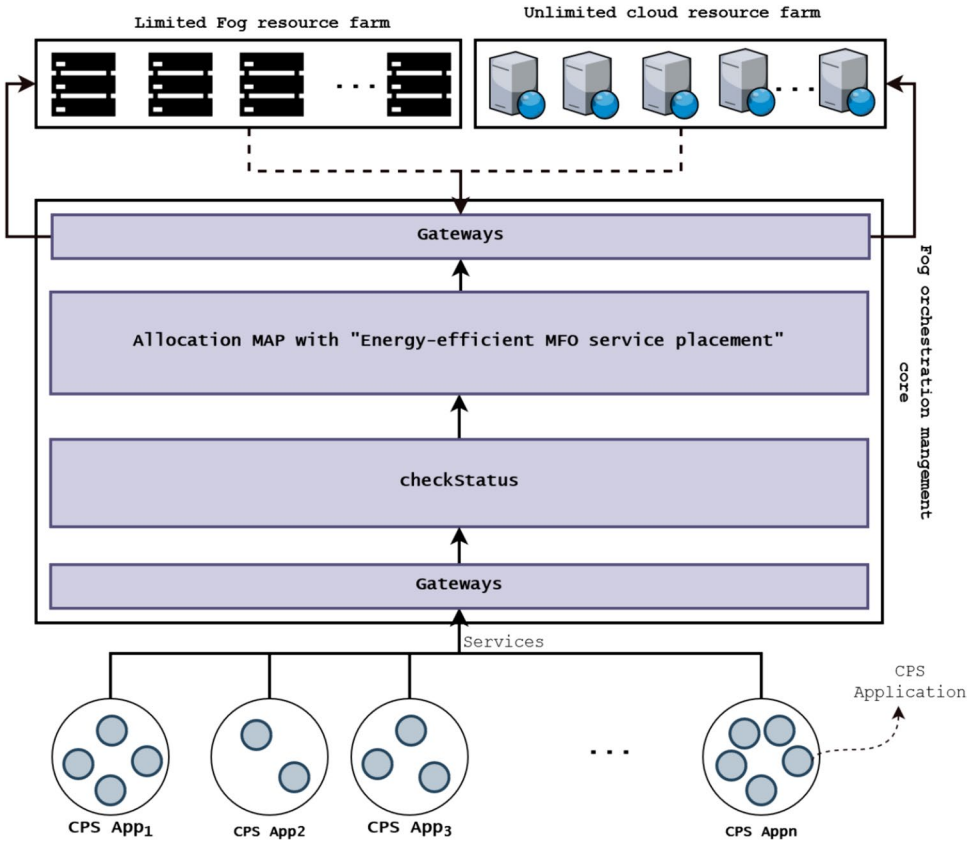


Fig. 7 The framework used to place fog services in fog infrastructure using the MFO algorithm

Algorithm 1 Energy-efficient MFO service placement

```
1:Begin
2:  for each (Time interval Δt in execution time) do
3:      inputs=check Status()
4:      allocation Map=MFO(inputs)
5:      Allocate services to fog nodes based on allocation Map
7: end for each
8:End
```

In this section, the algorithm used for use in the observation phase is described. Based on what was stated in the formulation section, two important elements are the observation of requested fog services by cyber-physical applications and the observation of the status of fog cells and head fog cells.

Algorithm 2 receives the input and the current state of the infrastructure. In lines 4 and 5 of the pseudo-code, the status and productivity of all cyber-physical applications in the fog colony Fi are checked and observed in the time interval Δt. In lines 7 and 8, the status of all the fog cells in fog colony Fi is observed in the current time frame, and also the resource efficiency of all fog cells, including the processor, main memory, and secondary memory, is observed. In the ninth line, all the received inputs and states of cyber-physical applications and fog cells are returned as the output of the observation phase algorithm.

Algorithm 2 Check Status()

Input: Fog colony F_i
Output: monitored Inputs
1:***Begin***
2: ***for each*** (*Fog colony* F_i at interval Δt) ***do***
3: ***for each*** (*CPS Application* A_k at interval Δt) ***do***
4: ***Monitor*** (all fog services for CPS application A_k) /* Application Monitoring*/
5: ***Monitor*** $U(a_k)$ ، $M(a_k)$ و $S(a_k)$ /*Performance Monitoring of IoT applications*/
6: ***end for each***
7: ***for each*** (Fog cell $f_{i,j}$ in fog colony F_i at time interval Δt) ***do***
8: ***Monitor*** ($f_{i,j}$ status at interval Δt) /* Fog cell monitoring */
9: ***Monitor*** $Uf_{i,j}(\Delta t)$,$Mf_{i,j}(\Delta t)$, and $Sf_{i,j}(\Delta t)$ /*Performance Monitoring of fog cells*/
10: ***end for each***
11:***end for each***
12: ***return*** monitored Inputs
13:***End***

After receiving the state of the cloud and fog infrastructure and the requested services from the cyber-physical applications side, the implementation of the proposed placement algorithm based on the meta-heuristic algorithm of MFO should be performed to reach the destination of each IoT service. To be determined, Algorithm 3 is a community-based meta-heuristic solution that simulates the behavior of a moth in nature. This algorithm simulates the behavior of the moth around the flame at night. Moths are search agents and flames are the best position obtained so far, so the population considers this position as the answer. The MFO algorithm has a set of moths that can be displayed in a matrix as follows.

$$M = \begin{bmatrix} M_{1,1} & \cdots & M_{1,d} \\ \vdots & \vdots & \vdots \\ M_{n,1} & \cdots & M_{n.d} \end{bmatrix} \tag{30}$$

In Eq. (30), n shows the number of butterflies and d shows the number of variables of the problem. Another matrix is for storing the fit values of the answers, this matrix shows the quality level of the services. OM matrix is expressed in Eq. (31).

$$OM = [OM_1 OM_2 ... OM_n]^T \tag{31}$$

On the other hand, flames are also other components in this algorithm. They can be displayed using the matrix presented in Eq. (32).

$$F = \begin{bmatrix} F_{1,1} & \cdots & F_{1,d} \\ \vdots & \vdots & \vdots \\ F_{n,1} & \cdots & F_{n.d} \end{bmatrix} \tag{32}$$

In Eq. (32), n is the number of moths, and d is the dimensions of the problem or the number of variables of the problem. It should be noted that the dimensions of the matrices M and F are equal to each other. The OF matrix is also used as the fitting values for propellers mentioned in Eq. (33).

$$OF = [OF_1 OF_2 ... OF_n]^T \tag{33}$$

The overall structure of the MFO algorithm is determined using an approximation function with three parameters:

$$\text{MFO} = (\text{I},\ \text{P},\ \text{T}) \tag{34}$$

In Eq. (34), I is a function that initializes the initial population of moths.

$$\text{I} : \emptyset - \rightarrow \{\text{M},\ \text{OM}\} \tag{35}$$

In Eq. (34), P is also a function that moves the propellers based on the relation presented in the quasi-code.

$$\text{P} : \text{M} \rightarrow \text{M} \tag{36}$$

The last function used in Eq. (34) is T function, this function returns True value if the end condition of the algorithm is satisfied. Equation (37) shows the T function.

$$\text{T} : \text{M} - \rightarrow \{\text{true},\ \text{false}\} \tag{37}$$

Moths and flames are the main parts of the MFO algorithm. The moths move around the search space, while the flames show the best position obtained by the moths. Moths fly around the flames and update their position by finding better answers.

In this section, the presented algorithm will be discussed in order to allocate the fog computing cell to the fog services with the moth optimization algorithm. For this purpose, based on the concepts expressed about the basic algorithm, it acts as follows:

1- At first, the number of variables (d) is considered as the migrant fog services (the fog services that require resources).
2- The number of dimensions (n) is considered equal to the number of fog computing cells available.
3- In order to solve the problem with the moth optimization algorithm, the OM matrix will generally show how good or bad the found solution is, while OMi is equal to the fit function of the ith solution (Moth i).) is in the problem space.

Based on the above explanations, our problem solving matrices will be as follows:

$$M = \begin{bmatrix} M_{1,1} & \cdots & M_{1,d} \\ \vdots & \vdots & \vdots \\ M_{n,1} & \cdots & M_{n.d} \end{bmatrix} \tag{38}$$

It is shown in Eq. (38) that in the Mi matrix, each row will be a solution in such a way that Mi, 1 represents the selected fog computing cell for the first (no source) fog service. In this way, the number of fog services is equal to d.

In this way, the best solutions found for each element of the population are stored as a flag (or flame dot) in the matrix F, which can be obtained based on Eq. (39):

$$F = \begin{bmatrix} F_{1,1} & \cdots & F_{1,d} \\ \vdots & \vdots & \vdots \\ F_{n,1} & \cdots & F_{n.d} \end{bmatrix} \tag{39}$$

Although the two matrices belonging to moths and flames are used in completely different ways and are updated differently; both contain solutions for deploying cloud services. Insects search for the most appropriate destination in the state space that contains all the available resources, while the mutual flames of that moth store the best solution found by it.

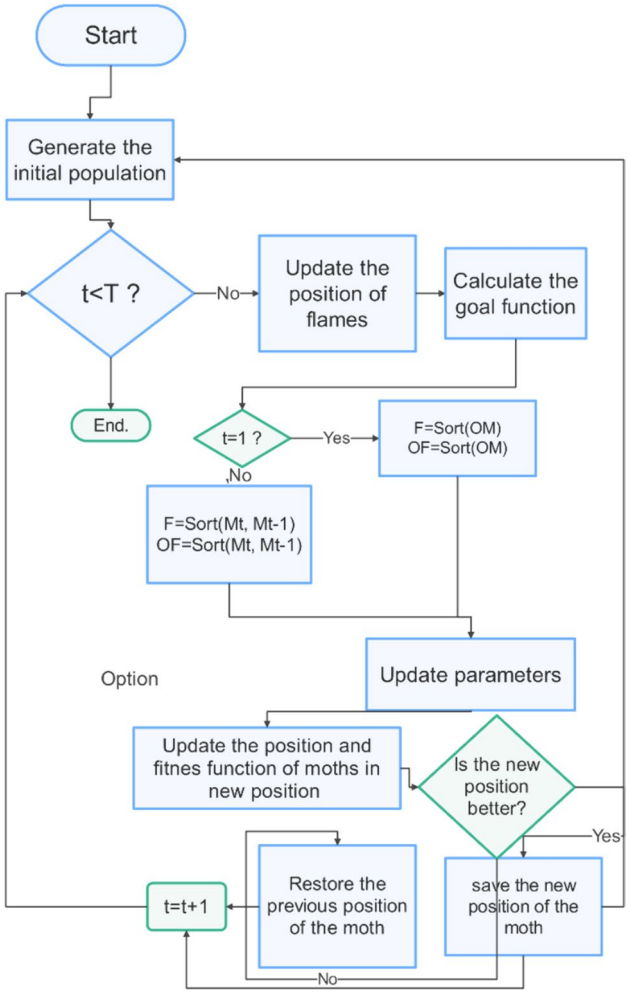


Fig. 8 Flowchart of the proposed solution for allocating fog computing cells to fog services with moth optimization algorithm

Therefore, moths search around the found flames (which are flags placed by moths). Thus, if a set of cloud services that require cloud computing cells for allocation are given to the cloud manager system; all of them are considered as a set of requests at once as input to the algorithm. Based on the set of received fog services, the initial population of the algorithm can be created randomly.

Then the position of the moths is determined randomly (each position is actually a physical host or server). Moths regularly move in the space of the problem state (hosts). This movement continues until the end condition of the algorithm is not satisfied. The main condition for the exit is that, in the first place, the number of each insect has been able to

Table 3 Moth-Flame algorithm parameter settings for Moth services placement algorithm

Parameters	Explain
Decision-making criteria	Position of moths in each interval
solution	Moths
Initial population	Random positions in which moths are placed
The previous solution	Previous position of moths
The current solution	current position of moths
The best solution	Positions of flames
Fitness function	Distance between moths and flames
The process of generating a new solution	Rotational movements of moths around flames

move and move in the problem space at least the number of iterations determined for the algorithm. The second condition is that there is at least one possible OMi solution. One of the challenges of using the moth algorithm is to find a possible solution. For example, it is possible that a fog cell is selected as the destination of several fog services, and as a result, the resource is used by fog services beyond its capacity, which will lead to the creation of an "impossible" solution. In the presented algorithm, a possibility is provided in order to reduce the number of cases of creating an impossible solution.

At the end, after updating and moving the insects, the OM matrix, which contains the fitting function of the found solutions, will also be updated based on the new findings.

Function I, as shown in relation 3–11, produces the initial population, which is actually the selection of computing cells for fog services. At the same time as the initial population is created, the fitness function of each of the generated insects (set of answers) is also calculated and stored in the OM matrix. In order to create the initial population, first, each set of fog services xi should have a selection set $SSi = \{S_{lb_i}, \ldots, S_{ub_i}\}$. Thus, in order to allocate fog computing cells, we will have $(S_{ub_i} - S_{lb_i}) + 1$ selection. Therefore, the initialization function will randomly select a resource for each moth:

$$(S_{ub_i} - S_{lb_i}) + 1 \tag{40}$$

In Eq. (40), S_{ub_i} and S_{lb_i} indicate the lower and upper limits of the calculation cells suitable for the desired fog services.

Figure 8 shows the flowchart presented for placing fog services in fog computing cells using the Moth-flame algorithm. According to the flowchart, at the beginning of each time period, it is checked whether there are still cloud services without computing resources (due to start or relocation) or not. If there is a cloud computing service without a computing resource, in the second step, suitable active resources for the desired cloud service are identified in the cloud data center. Then, based on the active resources, the Moth-flame algorithm setup function is executed, whereby an initial population of moths is generated.

Then, at the beginning of the execution cycle of Moth-flame algorithm generations, it is checked whether the exit condition of the algorithm is satisfied or not. The condition of leaving the execution of the algorithm generations will be satisfied either due to the completion of the limitation of the execution of the generations (Iterations) or by reaching the optimal solution (Optimum). If this condition is not satisfied, the moth matrix and the flame point matrix will be updated as well. Also, in the updating function, the moths will move first towards the flames and then around the flames in a circular manner (as in Fig. 8).

Now, in this section, based on the concepts expressed for the Moth-flame algorithm, as well as based on the stated problem and the flowchart presented to solve the problem of placing fog services with the algorithm of moths and flames, the variables and parameters of the Moth-algorithm Flame is set and considered based on the desired problem as follows, and based on the following concepts, the presented algorithm will be checked (Table 3).

Therefore, based on the above assumptions and the flowchart presented in Fig. 8, we will examine step by step the solution presented in Algorithm 3.

Algorithm 3: The proposed MFO-based Fog service placement algorithm

Input: applications
Output: Solution_map /*Fog services mapped to fog cells*/
1: ***for each*** (*Moth* M_j) ***do***
2: ***for each*** (CPS application A_k at interval Δt) ***do***
3: ***for each*** (Fog service $A_{k,s}$) ***do***
4: $M(j, A_{k,s}) = [(S_{ub_{j,MAX}} - S_{lb_{j,MIN}}) * rand() + S_{lb_{j,MIN}}]$ /*Initialize M(j, $A_{k,s}$)*/
5: **end for**
6: **end for**
7: **end for**
8: ***for each*** (*Moth* M_j) ***do***
9: $Fitness(M_j) = \left[\sum_{Fog\ cell\ i} Static(F_i) + Dynamic(F_i)\right] + \left[\sum_{Fog\ application\ k} \sum_{Fog\ service\ s} Communication(A_{k,s})\right]$ /*Based on energy */
10: ***end for***
11: ***for each*** (*Moth* M_j) ***do***
12: $Fitness(M_j) = Fitness(M_j)/MAX(Fitness(M))$ /*Normalizing fitness values*/
13: ***end for***

14: **while** (Itr<Max_Iteration) **do** /* *T(M)*==false */
15: $Number\_of\_flames = round\left(n - Itr \times \frac{n-1}{Max_{Itr}}\right)$ /*Update D : number of flames*/
16: OM= Fitness(*M*) /*Based on equations in Lines 10-15*/
17: **if** Itr ==1 **then**
18: F=Sort(M)
19: OF=Sort(OM)
20: **else**
21: F=Sort(M_{Itr-1},M_{Itr})
22: OF=Sort(OM_{Itr-1},OM_{Itr})
23: **end if**
24: **for** i=1 to *n* **do**
25: **for** j=1 to *d* **do**
26: $M_i = S(M_i, F_j)$ /*Update mechanism: Update *r* and *t**/
27: $S(M_i, F_j) = D_i . e^{bt} . cos(2\pi t) + F_j$ /* Spiral movement of moths*/
28: **end for**
29: **end for**
30: **end while**
31: ***for each*** (*Fog colony* F_i) ***do***
32: ***for each*** (*CPS Application* A_k) do
33: ***for each*** (fog service $A_{k,s\ in}\ A_k$) ***do***
34: Service Placement $A_{k,s}$ ***into*** *Fog resource i*
35: ***end for each***
38: ***end for each***
39: ***Itr == Itr + 1***
40: ***end for each***
41: **End**

Table 4 Statistical knowledge of resources required for cyber-physical programs

Service	U_{ai} (MIPS)	M_{ai} (MB)	S_{ai} (MB)	D_{ai} (s)
Sensor	50	30	10	0.9
Process control	200	10	30	0.10
Smart grid	200	20	30	0.10
Communicator	100	30	30	0.25
Actuate	50	20	10	0.50

Table 5 Statistical details of programs

CPS application	D_{Ak} (s)	W_{Ak} (s)
A1	120	60
A2	300	0
A3	300	60
A4	360	60
A5	240	0

The input of the proposed algorithm is fog services that need to receive service from one of the active computing cells. These fog services are from one of the two groups of fog services from the IoT devices of the current fog colony or fog services from the IoT devices of the neighboring fog colonies. The output of the algorithm will also be a map of placing fog services on one of the fog computing cells. At the beginning of the startup phase, we will create moths and flames randomly (lines 1–7). In these lines, each fog service is randomly placed on one of the fog cells in the current colony.

After generating the initial population of moths, their fitting function should be calculated. (lines 8–13). In this research, the fitting function is considered based on the static and dynamic energy consumption of fog cells and the communication energy consumption of all IoT services.

The selected fitting function is minimization or in other words a negative fitting function. The positive fit function means that the more the evaluated criterion is, the more positive it is, and vice versa in the negative fit function, the lower the evaluated criterion is, the more favorable it is. The main loop of the proposed algorithm to find the best solutions for placing services with the MFO algorithm is presented in (lines 14–30). This block of pseudo-code shows the Moth-flame algorithm execution and update loop. At the beginning of the loop, the number of moths for the current cycle of the algorithm is updated in such a way that in each iteration the number of butterflies decreases based on a fixed formula. Then the function of the insect matrix is calculated and placed in the OM matrix.

Table 7 Details of static and dynamic energy cost of fog cells

Type of communication	Dynamic energy consumption	Static energy consumption
fog head node	0.3–0.6	0.4–0.7
fog cells	0.5–0.8	0.2–0.5

In line 15, the number of flames is updated based on the current cycle of the algorithm, which will decrease the number of flames and moths in the time axis and execute algorithm cycles as time passes by this equation. Line 16 calculates the quality of the propellers after their last movement and displacement using the fit function.

In the first cycle of the MFO loop, the matrix F and its fitting function, which is equal to OF, will be equal to the first generation of propellers produced (lines 17–19) and otherwise between the best performance of the propellers and the last performance of the propeller. The best option will be selected based on the fit function (lines 20–23).

Then, the mechanism of updating and the movement of moths will be implemented. In this way, in line 26, the moth matrix is updated, and in line 27, the insects will circularly move around the flames. The sequence of execution of the main loop of the MFO algorithm is repeated until the optimal answers are found around the flames by the propellers. In the end, the best propeller will survive until the last cycle of the MFO loop execution, where its flag (its flame dot) will be the best solution found by the algorithm.

After finding the best solution, we place the services based on the found solution in lines 31–39.

Results and Discussion

In this section, you evaluate the proposed method by considering several parameters. The parameters that are discussed for evaluation are processing cost, response time and deviation from deadlines compared to previous methods,

Table 6 Details of possible communication types in the simulated infrastructure along with communication speed and energy consumption

Type of communication	Energy consumption rate	Transmission speed range
The node of the head of the fog with the cloud	1	1
Head node with other neighboring head nodes	0.6–0.9	1
Fog cells with head nodes	0.6–0.8	0.6–0.9
IoT devices with fog cells	0.6–1	0.6–0.9

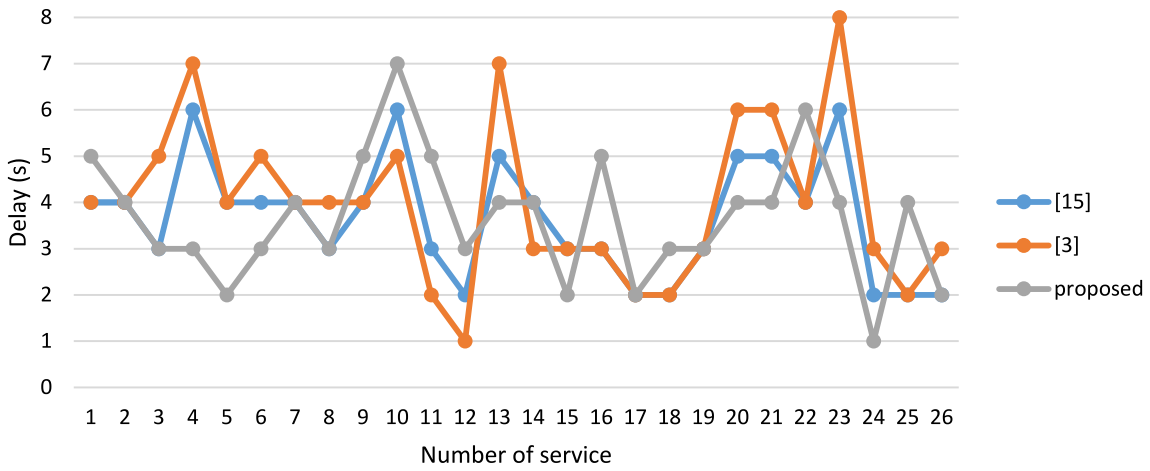


Fig. 9 Delay in responding to the service in the cloud layer of different solutions

implementation in cloud computing and also the most important criterion is the amount of energy consumed.

Considering that the implementation and implementation of the proposed method in operational frameworks in the real world is not possible, tools for simulating real environments have been provided, among which iFogSim tool can be mentioned. The iFogSim tool provides the possibility of modeling and simulating the fog computing environment. Using this tool, we can evaluate resource management and scheduling policies at the edge and cloud computing under different scenarios. This simulator is able to evaluate resource management policies focusing on their impact on latency, energy consumption, network traffic and operational cost. This tool supports edge devices, cloud computing, network links for performance evaluation. The most important application model supported in iFogSim is Sense-Process-Actuate. In this model, the sensors monitor the data generated in the IoT, follow and process the programs that are executed in the fog tools, finally, the final decisions are transferred to the environment through the arms is given [38].

Configuration

To simulate the Internet of Things computing environment and evaluate the proposed method, we need a set of services and basic configurations. Table 4 shows the details of the required resources of the assumed simulation programs. Table 5 shows the statistical details of the program.

In this research, each fog colony consists of ten fog cells, and each cell is connected to a fog colony controller node. We directed the observations in such a way that the placement of the service in a fog colony is visible.

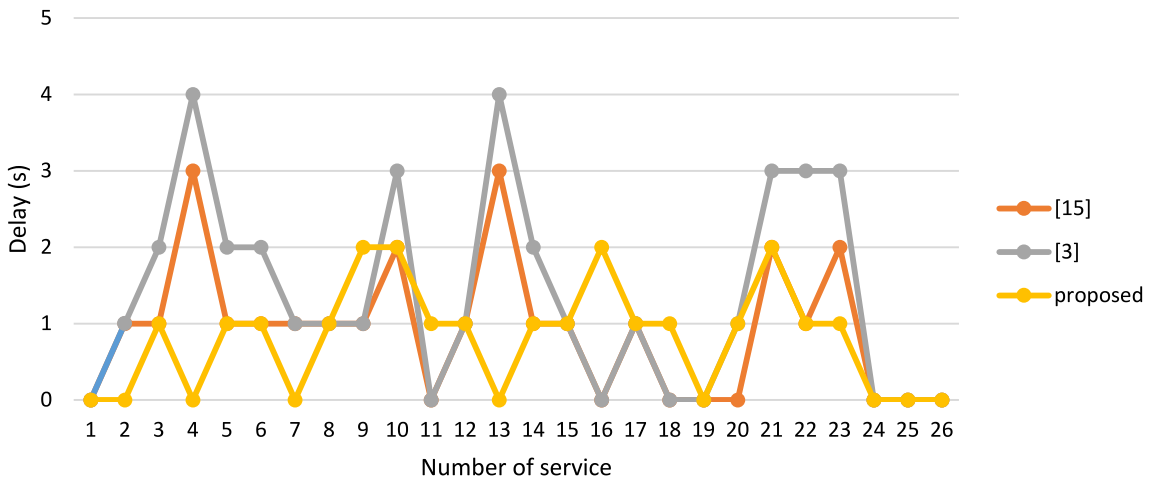


Fig. 10 Service response delay in fog of different solutions

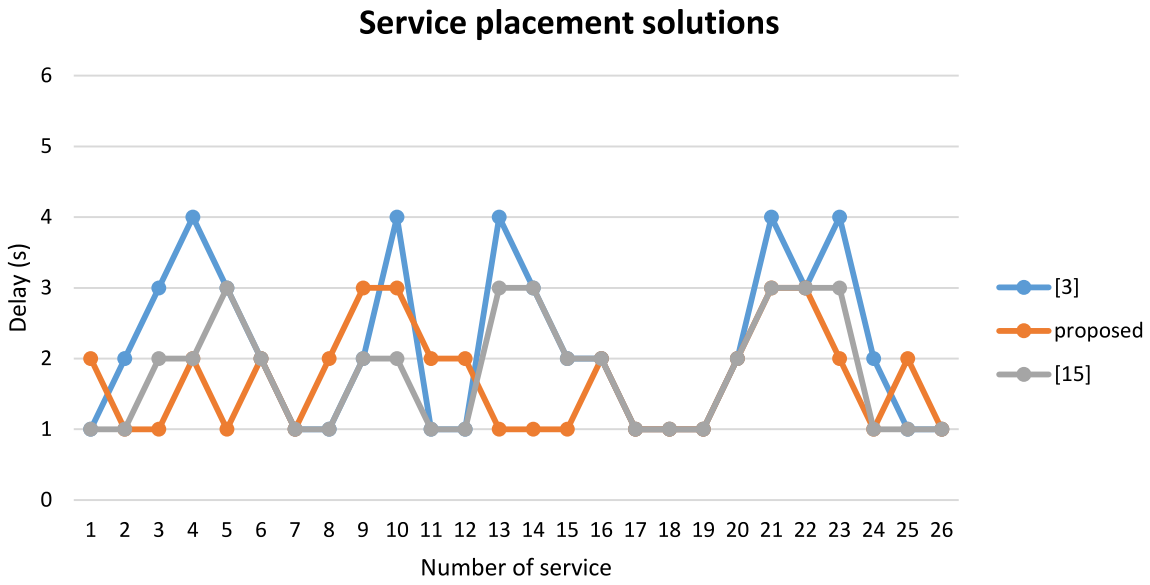
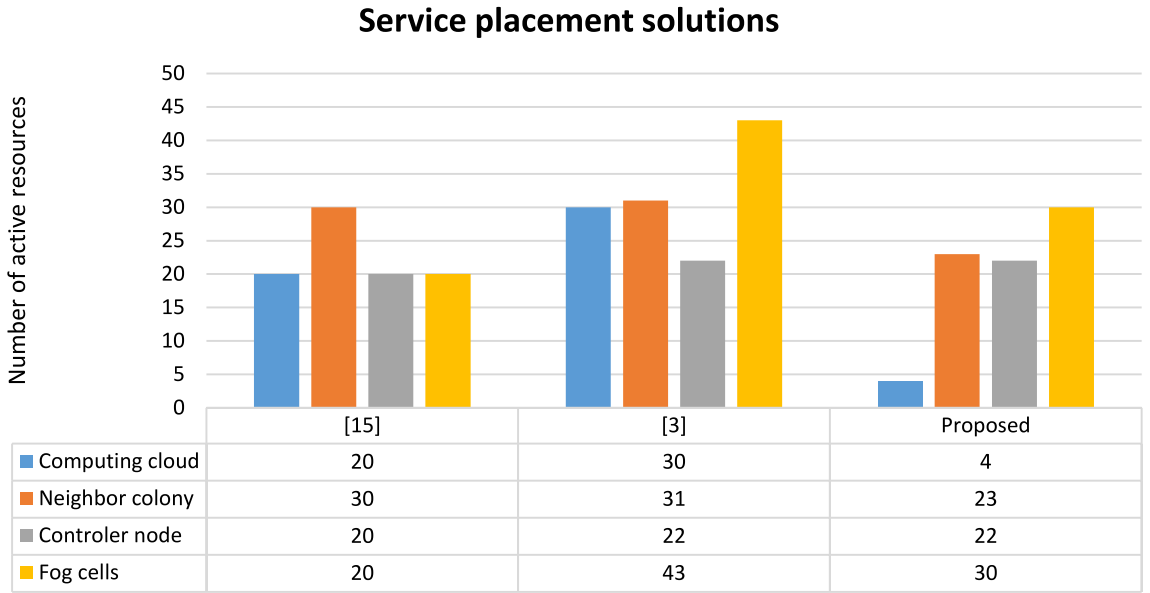


Fig. 11 Service response delay in cloud and fog layers of different solutions

Table 6 shows the energy consumption and communication cost in the simulation. As it is assumed, the cost of communication between the head of the cloud and the center of the cloud has the highest speed and the highest cost of energy consumption. Also, the cheapest type of communication between the fog cells is with the fog head node, and then the cost of communication between the internet of things devices that request to run the service with the fog cells has a variable speed and energy consumption cost depending on the device. The internet of things and the desired fog cell are available.

Table 7 shows the details of the static and dynamic energy consumption of the fog cells, such as the head node and the used fog cells. As it has been determined, in this research, a different range of static and dynamic energy consumption has been used for different experiments, which gives us the ability to test and evaluate the proposed solution in different environmental and physical conditions.



	[15]	[3]	Proposed
Computing cloud	20	30	4
Neighbor colony	30	31	23
Controler node	20	22	22
Fog cells	20	43	30

Fig. 12 Average number of active sources in different layers of cloud bed and fog with different methods

Evaluation Scenarios

To evaluate the efficiency and performance of the proposed method in this research and compare it with previous work, the following scenario have been chosen for testing and evaluation, which will be examined in the following. The compared methods in this research is the references [3, 15], which uses a service placement solution in an energy-efficient way, taking into account the fog rate, load balancing, and service duplication.

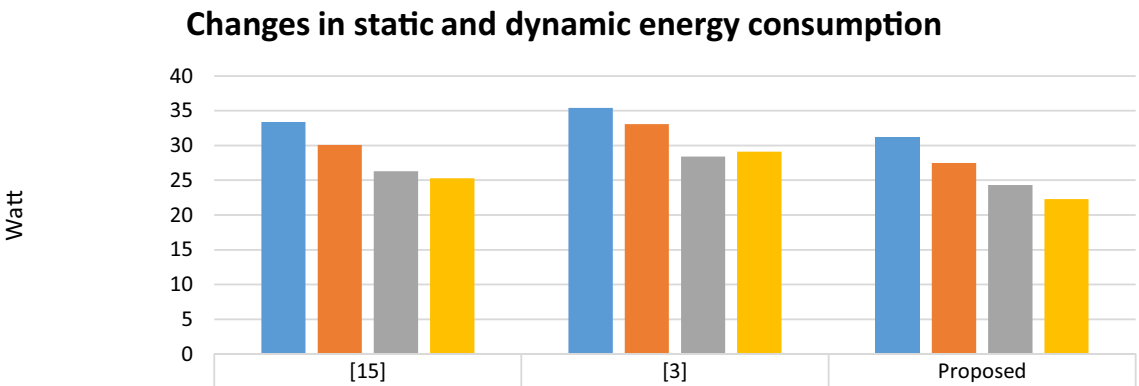
Delay of Services

In this section, the current author will analyze the delay performance of providing resources to IoT services by the proposed solution and the compared basis. As it was said before, the delay of providing the service after the cost of the energy consumed by the resources in the infrastructure is very important. Figure 9 shows the delay in responding to the service in the cloud layer of different solutions in the time axis. As it has been determined, the more time-critical requests are sent to the cloud, which has unlimited capacity but with a longer delay, the greater the delay situation resulting from the execution on the cloud resources. According to what is shown in Fig. 9, the genetic algorithm has the worst performance in providing service with average quality and no standard deviation, and in different time periods, the performance of this algorithm is very variable. Two recluse algorithms and the presented MFO algorithm have shown better performance. Although the difference between these two algorithms in the average response time of services implemented on the cloud is not so clear. As it has been determined, in some time intervals the references [3, 15] and in other time intervals the presented algorithm have shown better performance. But the number of services performed on the cloud by three algorithms can be seen by looking at the next two graphs and distinguishes three algorithms from each other.

Figure 10 shows the delay in responding to the service in the fog layer of different solutions in the time axis. As it has been determined, the more time-critical requests are sent to the cloud, which has unlimited capacity but with more delay, the greater the delay situation resulting from the execution on fog resources.

Unlike Fig. 9, in Fig. 10 the difference between the three algorithms is much more significant. The proposed algorithm in [3, 15] has led to a non-negligible increase in the delay in providing services in the fog layer due to the lack of appropriate allocation of the appropriate part of IoT services to the fog layer. On the other hand, due to the more inappropriate allocation of the proposed algorithm in [3] compared to the proposed algorithm in the fog layer, it has led to a very clear increase in the delay in providing the service in contrast to the cloud layer. The main reason for this noticeable difference is the large number of services allocated in the fog layer.

Figure 10 shows the delay in responding to the service in the fog and cloud layer together for different solutions in the time axis. Based on what is seen in Figs. 9 and 10, it can be concluded that the proposed algorithms in [3, 15] have the worst performance in terms of service delivery delay to IoT devices, which is also evident in Fig. 11. Also, although in the cloud layer, there was not so much difference in terms of delay between the three proposed algorithm, [3, 15], considering all IoT services, it can be seen that in general,



	[15]	[3]	Proposed
Static energy = 0.5	33.4	35.4	31.2
Static energy = 0.4	30.1	33.1	27.5
Static energy = 0.3	26.3	28.4	24.3
Static energy = 0.2	25.3	29.1	22.3

Fig. 13 The effect of changing the static and dynamic energy of different sources on the amount of energy consumption of different solutions

the proposed solution is significantly improved. in terms of delay.

The Number of Fog and Cloud Sources Used

Figure 12 shows the efficiency of resources in the compared algorithms. The proposed algorithm in [3] places all the Sense services of the environment and the action in the environment in different cells of the fog in the fog colony, four processing services are also placed in the controller node. The remaining 11 processes are also propagated to the side colonies.

The proposed algorithms in [3, 15] also executes the Sense services of programs A1, A3 and A5 in fog cells. It places operating services in fog cells or cloud computing. It also places processing services in the cloud computing environment. This type of placement causes one fifth of the placements to be done in cloud computing and cloud computing resources are not used well. The number of placements performed in cloud computing is very high because in this algorithm, any solution that violates the deadline must pay a large fine. On the other hand, there is no penalty for using remote resources (cloud computing) or not using cloud computing control nodes. The proposed method in this research brings a better distribution for the use of resources than other methods. In addition, exchanges with cloud computing are less than other methods. Cloud method only uses cloud computing resources, so the usage rate of other resources for this method is zero.

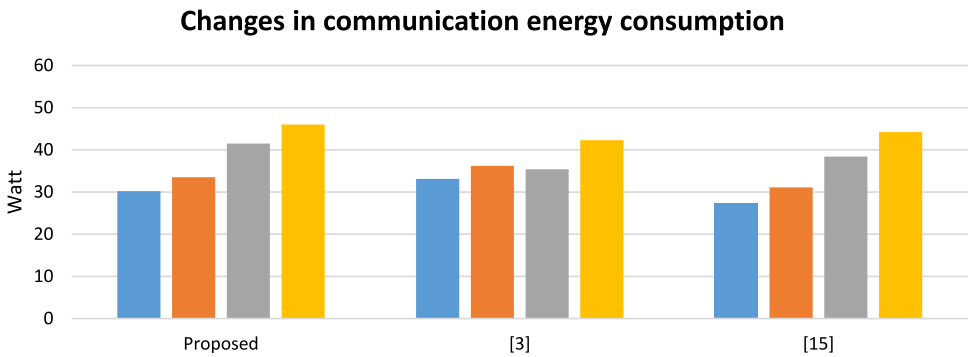


Fig. 14 The effect of communication energy change on the amount of energy consumption of different solutions

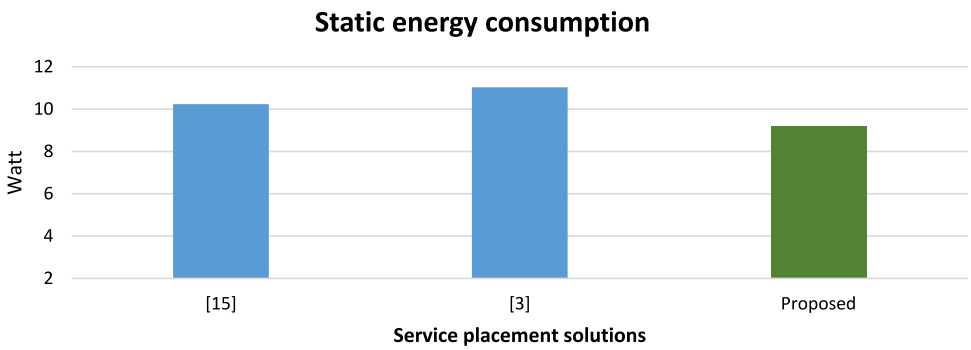


Fig. 15 Overall average static energy consumption in different solutions for all tests

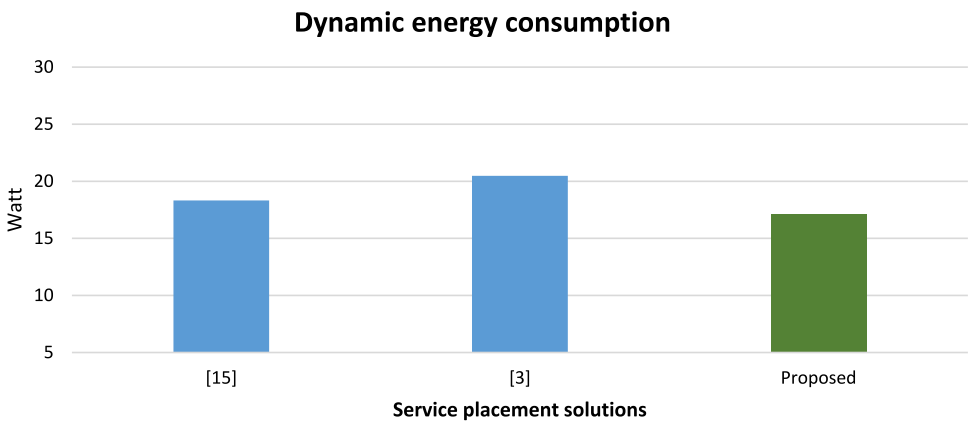


Fig. 16 Overall average dynamic energy consumption in different solutions for all tests

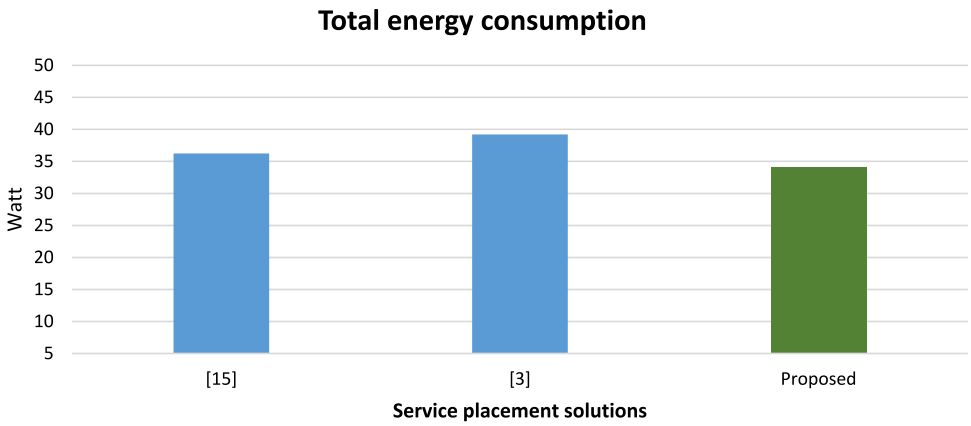


Fig. 17 Overall average total energy consumption in different solutions for all experiments

Energy Consumption

In this section, the performance analysis of the proposed solution and the basic solutions are discussed in terms of the amount of energy consumption, which includes the energy consumption of fog and cloud sources, as well as the energy consumption of communication and data transmission costs. Figure 13 shows the effect of changing the static and dynamic energy of different sources on the amount of energy consumption of different solutions. As it has been determined, 4 different tests have been performed as follows:

- Experiment 1: static energy 0.2 and dynamic energy 0.8.
- Experiment 2: static energy 0.3 and dynamic energy 0.7.
- Experiment 3: static energy 0.4 and dynamic energy 0.6.
- Experiment 4: static energy 0.5 and dynamic energy 0.5.

Based on the above tests, we were able to accurately evaluate the conditions of different service placement solutions in various conditions of static and dynamic energy consumption. Based on the evaluations, the current author obtained almost similar results in all the experiments, in a way that the proposed algorithms in [3, 15] consumed more energy than other solutions in almost all experiments, and as a result, the genetic algorithm in [3] performed the worst in terms of energy consumption compared to there have been other solutions. After the proposed algorithm in [15], the proposed algorithm in [3], which has performed worse than that [3] in two tests, ranks second, and as a result, the proposed MFO solution has achieved the best performance statistically and overall in all tests.

After evaluating the effect of static and dynamic energy change of different sources on the energy consumption of different solutions, we will investigate the effect of communication energy change on the energy consumption of different solutions. Figure 14 shows the effect of communication energy change on the amount of energy consumption of different solutions. As it has been determined, 4 different tests have been performed as follows:

- Experiment 1: very low communication cost (minimum according to the range set in the energy consumption settings table).
- Experiment 2: low communication cost (according to the table of energy consumption settings).
- Experiment 3: average communication cost (according to the table of energy consumption settings).
- Experiment 4: Communication cost is too high (maximum according to the range set in the energy consumption settings table).

Based on the above tests, we were able to accurately evaluate the conditions of the placement solutions of different services in various conditions of energy consumption in the field of communication and data transmission. Based on the evaluations, the current author obtained almost similar results in all the experiments, in a way that the genetic algorithm in [3] consumed more energy than other solutions in almost all experiments, and as a result, the proposed algorithm in [15] performed the worst in terms of energy consumption compared to there have been other solutions. After it, the proposed algorithm in [15], which has performed worse than the genetic algorithm in two tests, ranks second, and as a result, the proposed MFO solution has achieved the best performance statistically and overall in all tests.

Based on the analysis done for the consumption cost of resources and communication in Figs. 13 and 14, it can be concluded that the proposed solutions in [3, 15] have the worst performance in the overall cost of energy consumption and the presented algorithm has the best performance. Figures 15, 16 and 17 also show the average of all static, dynamic and general tests for different solutions in all tests. As it has been determined, the presented solution has the best performance in all three states of static, dynamic, and total energy consumption statistically in all scenarios and thus proves this conclusion.

Conclusion

Fog computing is assumed as a set of fog colonies in such a way that each colony includes a set of fog computing cells and IoT devices. For this purpose, it will be possible to use the existing computing cells in a cloud colony to provide computing and processing resources to services requested by cyber-physical applications. In addition, it will be possible to use the resources of neighboring fog colonies to provide services to neighboring IoT devices. In addition to affecting the quality of the service provided with a direct effect on the delay, the placement of the service directly affects the energy consumption of IoT devices and fog cells as well as cloud resources. In this research, a solution based on the population-based MFO algorithm has been presented for energy-aware placement of fog services. Meta-heuristic Moth-Flame algorithm is used to place fog services on fog computing cells. In the fourth chapter, we reviewed the configurations made for conducting experiments. We also stated the results obtained from the simulation on iFogSim. We used different criteria to evaluate the effectiveness of the proposed method in this research. Among these criteria, we can mention the response time, meeting the deadline, and the costs imposed on the infrastructure. The programs used to evaluate the system include five programs that have different services. The obtained results show that the proposed method in this research is able to obtain a shorter response time than genetics and relaxation in most programs, and also the energy consumption of the presented solution is significantly better than the two basic algorithms called improvement. (12% and 25% improvements have been obtained, respectively). Advantages of the proposed model are international repository, unlimited extent, and quickness of local repository. Data management is the limitation and disadvantage of the suggested model. Beneficial of the proposed model in decentralized systems are the obtained reduced latency and enhanced security.

According to the results obtained from this thesis and for further evaluation of the results and completion and development of this research, using learning algorithms to analyze the pattern of computing, data, and network loads are suggested.

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Wu C, Gunatilaka D, Sha M, Lu C. Real-time wireless routing for industrial internet of things. In: 2018 IEEE/ACM 3rd international conference on internet-of-things design and implementation (IoTDI). 2018. p. 261–66.
2. Selonen P, Taivalsaari A. Kiuas–iot cloud environment for enabling the programmable world. In: 2016 42th Euromicro conference on software engineering and advanced applications (SEAA). 2016. p. 250–57.
3. Zeng D, Gu L, Yao H. Towards energy efficient service composition in green energy powered cyber–physical fog systems. Future Gener Comput Syst. 2020;105:757–65.
4. Raafat HM, Hossain MS, Essa E, Elmougy S, Tolba AS, Muhammad G, et al. Fog intelligence for real-time IoT sensor data analytics. IEEE Access. 2017;5:24062–9.
5. Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: The communication perspective. IEEE Commun Surv Tutor. 2017;19:2322–58.
6. Shi C, Ren Z, He X. Research on load balancing for software defined cloud-fog network in real-time mobile face recognition. In: International conference on communications and networking in China. 2016. p. 121–31.
7. Xi S, Li C, Lu C, Gill CD, Xu M, Phan LT, et al. Rt-open stack: Cpu resource management for real-time cloud computing. In: 2015 IEEE 8th international conference on cloud computing. 2015. p. 179–86.
8. Casoni M, Grazia CA, Klapez M. An sdn and cps based opportunistic upload splitting for mobile users. In: International internet of things summit. 2015. p. 67–76.
9. Calvaresi D, Marinoni M, Sturm A, Schumacher M, Buttazzo G. The challenge of real-time multi-agent systems for enabling IoT and CPS. In: Proceedings of the international conference on web intelligence. 2017. p. 356–64.
10. Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst. 2015;89:228–49.
11. Mohanty B, Acharyulu B, Hota P. Moth-flame optimization algorithm optimized dual-mode controller for multiarea hybrid sources AGC system. Optim Control Appl Methods. 2018;39:720–34.
12. Wu Z, Shen D, Shang M, Qi S. Parameter identification of single-phase inverter based on improved moth flame optimization algorithm. Electric Power Compon Syst. 2019;47:456–69.
13. Ali S, Ghazal M. Real-time heart attack mobile detection service (RHAMDS): an IoT use case for software defined networks. In: 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE). 2017. p. 1–6.
14. Bonomi F, Milito R, Natarajan P, Zhu J. Fog computing: a platform for internet of things and analytics. In: Big data and internet of things: a roadmap for smart environments. Springer; 2014. p. 169–86.
15. Tiwary M, Puthal D, Sahoo KS, Sahoo B, Yang LT. Response time optimization for cloudlets in mobile edge computing. J Parallel Distrib Comput. 2018;119:81–91.
16. Zhu J, Chan DS, Prabhu MS, Natarajan P, Hu H, Bonomi F. Improving web sites performance using edge servers in fog computing architecture. In: 2013 IEEE 7th international symposium on service-oriented system engineering. 2013. p. 320–23.
17. Vaquero LM, Rodero-Merino L. Finding your way in the fog: towards a comprehensive definition of fog computing. ACM SIGCOMM Comput Commun Rev. 2014;44:27–32.
18. Do CT, Tran NH, Pham C, Alam MGR, Son JH, Hong CS. A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing. In: 2015

international conference on information networking (ICOIN). 2015. p. 324–29.
19. Wang S, Urgaonkar R, Zafer M, He T, Chan K, Leung KK. Dynamic service migration in mobile edge-clouds. In: 2015 IFIP networking conference (IFIP Networking). 2015. p. 1–9.
20. Deng R, Lu R, Lai C, Luan TH. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In: 2015 IEEE international conference on communications (ICC). 2015. p. 3909–14.
21. Colistra G, Pilloni V, Atzori L. The problem of task allocation in the internet of things and the consensus-based approach. Comput Netw. 2014;73:98–111.
22. Yang Y, Zhao S, Zhang W, Chen Y, Luo X, Wang J. DEBTS: delay energy balanced task scheduling in homogeneous fog networks. IEEE Internet Things J. 2018;5:2094–106.
23. Skarlat O, Nardelli M, Schulte S, Borkowski M, Leitner P. Optimized IoT service placement in the fog. SOCA. 2017;11:427–43.
24. Taneja M, Davy A. Resource aware placement of IoT application modules in fog-cloud computing paradigm. In: 2017 IFIP/IEEE symposium on integrated network and service management (IM). 2017. p. 1222–28.
25. Brogi A, Forti S, Ibrahim A. How to best deploy your fog applications, probably. In: 2017 IEEE 1st international conference on fog and edge computing (ICFEC). 2017. p. 105–14.
26. Aazam M, Huh E-N. Dynamic resource provisioning through fog micro datacenter. In: 2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops). 2015. p. 105–10.
27. Han B, Gopalakrishnan V, Ji L, Lee S. Network function virtualization: challenges and opportunities for innovations. IEEE Commun Mag. 2015;53:90–7.
28. Voegler M, Schleicher JM, Inzinger C, Dustdar S. Optimizing elastic IoT application deployments. IEEE Trans Serv Comput. 2016;11:879–92.
29. Hong K, Lillethun D, Ramachandran U, Ottenwälder B, Koldehofe B. Mobile fog: a programming model for large-scale applications on the internet of things. In: Proceedings of the second ACM SIGCOMM workshop on mobile cloud computing. 2013. p. 15–20.
30. Gu L, Cai J, Zeng D, Zhang Y, Jin H, Dai W. Energy efficient task allocation and energy scheduling in green energy powered edge computing. Future Gener Comput Syst. 2019;95:89–99.
31. Munoz D-J, Montenegro JA, Pinto M, Fuentes L. Energy-aware environments for the development of green applications for cyber–physical systems. Future Gener Comput Syst. 2019;91:536–54.
32. Mahmoud MM, Rodrigues JJ, Saleem K, Al-Muhtadi J, Kumar N, Korotaev V. Towards energy-aware fog-enabled cloud of things for healthcare. Comput Electr Eng. 2018;67:58–69.
33. Djemai T, Stolf P, Monteil T, Pierson J-M. A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures. In: 2019 18th international symposium on parallel and distributed computing (ISPDC). 2019. p. 32–40.
34. Mehran N, Kimovski D, Prodan R. MAPO: a multi-objective model for IoT application placement in a fog environment. In: Proceedings of the 9th international conference on the internet of things. 2019. p. 1–8.
35. Kumar P, Kumar R, Kumar A, Franklin AA, Garg S, Singh S. Blockchain and deep learning for secure communication in digital twin empowered industrial IoT network. IEEE Trans Netw Sci Eng. 2022.
36. Kumar P, Kumar R, Kumar A, Franklin AA, Jolfaei A. Blockchain and deep learning empowered secure data sharing framework for softwarized UAVs. In: 2022 IEEE international conference on communications workshops (ICC Workshops). 2022. p. 770–75.
37. Aljuhani A, Kumar P, Kumar R, Jolfaei A, Islam AN. Fog intelligence for secure smart villages: Architecture, and future challenges. IEEE Consum Electron Mag. 2022.
38. Gupta H, Vahid-Dastjerdi A, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw Pract Exp. 2017;47:1275–96.