



An Evolutionary Multi-objective Optimization Technique to Deploy the IoT Services in Fog-enabled Networks: An Autonomous Approach

Mahboubah Salimian, Mostafa Ghobaei-Arani & Ali Shahidinejad

To cite this article: Mahboubah Salimian, Mostafa Ghobaei-Arani & Ali Shahidinejad (2022) An Evolutionary Multi-objective Optimization Technique to Deploy the IoT Services in Fog-enabled Networks: An Autonomous Approach, Applied Artificial Intelligence, 36:1, 2008149, DOI: [10.1080/08839514.2021.2008149](https://doi.org/10.1080/08839514.2021.2008149)

To link to this article: <https://doi.org/10.1080/08839514.2021.2008149>



© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 05 Jan 2022.



Submit your article to this journal [↗](#)



Article views: 1340



View related articles [↗](#)





View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)

An Evolutionary Multi-objective Optimization Technique to Deploy the IoT Services in Fog-enabled Networks: An Autonomous Approach

Mahboubeh Salimian, Mostafa Ghobaei-Arani , and Ali Shahidinejad 

Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

ABSTRACT

The Internet of Things (IoT) generates countless amounts of data, much of which is processed in cloud data centers. When data is transferred to the cloud over longer distances, there is a long latency in IoT services. Therefore, in order to increase the speed of service provision, resources should be placed close to the user (i.e., at the edge of the network). To address this challenge, a new paradigm called Fog Computing was introduced and added as a layer in the IoT architecture. Fog computing is a decentralized computing infrastructure in which provides storage and computing in the vicinity of IoT devices instead of sending to the cloud. Hence, fog computing can provide less latency and better Quality of Service (QoS) for real-time applications than cloud computing. In general, the theoretical foundations of fog computing have already been presented, but the problem of IoT services placement to fog nodes is still challenging and has attracted much attention from researchers. In this paper, a conceptual computing framework based on fog-cloud control middleware is proposed to optimally IoT services placement. Here, this problem is formulated as an automated planning model for managing service requests due to some limitations that take into account the heterogeneity of applications and resources. To solve the problem of IoT services placement, an automated evolutionary approach based on Particle Swarm Optimization (PSO) has been proposed with the aim of making maximize the utilization of fog resources and improving QoS. Experimental studies on a synthetic environment have been evaluated based on various metrics including services performed, waiting time, failed services, services cost, services remaining, and runtime. The results of the comparisons showed that the proposed framework based on PSO performs better than the state-of-the-art methods.

ARTICLE HISTORY

Received 9 September 2021
Revised 8 November 2021
Accepted 15 November 2021

Introduction

The Internet of Things (IoT) is a network of devices that have the ability to sense the environment, process, communicate, and perform specific actions (Taneja and Davy 2016). IoT has many applications, including environmental

CONTACT Mostafa Ghobaei-Arani  m.ghobaeiarani@gmail.com  Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

monitoring, traffic management, smart homes, and smart cities (Rezaeipanah, Nazari, and Ahmadi 2019). Cloud computing is one of the main technologies for realizing IoT. Due to the large amount of data generated by IoT devices, cloud data centers are the best place to store this data. The cloud theoretically has unlimited and scalable resources and can meet the basic requirements of IoT applications (Dastjerdi and Buyya 2016). However, there are more than 50 billion networked devices by 2020, and Cisco predicts that by 2030, more than 500 billion IoT devices will be connected to the Internet (Mohan et al. 2017). Therefore, it is clear that the current cloud infrastructure will not be able to support all the data generated by these devices. In addition, the geo-centralization of the cloud leads to its inability to meet the needs of real-time IoT applications (Rezaeipanah, Mojarad, and Fakhari 2020). Despite this distance, cloud-connected IoT devices suffer from the problem of response time, latency, bandwidth, and network congestion. Therefore, the relatively centralized structure of the cloud does not conform to the decentralized nature of IoT, and a new computing model was needed to overcome these limitations (Goswami et al. 2021; Karatas and Korpeoglu 2019).

Fog or edge computing has been introduced as a new computing model for hosting IoT applications in order to overcome the limitations of using cloud data centers (Souza et al. 2018). Society has not yet converged on clear definitions of these terms (Chen et al. 2020; Forouzandeh, Rostami, and Berahmand 2021; Skarlat et al. 2017; Souza et al. 2018). Fog computing involves a large number of heterogeneous nodes that allow IoT services to execute in the vicinity of resources without involving the cloud. This technology extends services to the network edge in a distributed manner and brings storage, analysis and processing closer to where data are created and end users. Fog computing is considered as an intermediate layer between cloud servers and IoT devices, and turns the network into an edge network (Khosroabadi, Fotouhi-Ghazvini, and Fotouhi 2021). In this layer, fog and cloud resources work together to provide services. Significant advantages of fog calculations include the reduction of latency and computational costs, as well as its ability to meet the response time required in applications sensitive to latency and real-time (Taneja and Davy 2016). Currently, resources management is one of the main challenges in research based on fog computing (Puliafito et al. 2019; Xavier et al. 2020). There are many issues related to resources management in fog and cloud computing, such as resources forecasting, resources provisioning, services provisioning, scheduling, dispatching, and services migration (Khosroabadi, Fotouhi-Ghazvini, and Fotouhi 2021).

One of the main obstacles to accepting fog computing is “how to efficiently deploy services on available fog nodes” (Karatas and Korpeoglu 2019). Because, unlike cloud data centers, dynamic fog devices are resource-limited and distributed and this makes resources management challenging. Therefore, one of the most important problems in fog and cloud resources management

is the decision-making to services provision or services placement autonomously. The purpose of solving this problem is to deploy distributed components of IoT applications (i.e., services) in fog and cloud resources, which causes mapping between fog nodes and IoT applications to use resources (Yang et al. 2018). The benefits of provisioning dynamic services include reduced bandwidth utilization, optimal resources management, reduced computational costs, availability, reduced latency, increased reliability, reduced energy consumption, improved QoS, and meeting the response time required for real-time applications (Santoyo-González and Cervelló-Pastor 2018).

International Business Machines (IBM) corporation has introduced the MADE-k autonomous model to implement fog computing (Jacob et al. 2004). This model consists of four phases of monitoring, analysis, decision-making and execution that are shared with a knowledge-base (Salimian, Ghobaei-Arani, and Shahidinejad 2021). IoT services and fog resources are controlled during the monitoring phase. The analysis phase is responsible for prioritizing the execution of services based on the deadline. In the decision-making phase, planning is done for the proper deployment of services, and finally, in the execution phase, the planned placement is applied. According to this model, we present an autonomous conceptual computing framework based on fog-cloud control middleware for optimal fog and cloud resources management. This middleware is centrally responsible for applying the MADE-k model to the entire fog landscape.

There are many studies in the literature to improve the decision-making and optimal deployment of IoT services on fog nodes (Brogi et al. 2018; Murtaza et al. 2020; Salimian, Ghobaei-Arani, and Shahidinejad 2021; Yousefpour et al. 2019). The purpose of most of these studies is to optimize an objective function (e.g., energy consumption, response time, latency, cost, and utilization of fog resources); however, some studies also perform optimization based on several objectives. Multi-objective problem solving can provide a better decision for mapping fog nodes to IoT services applications (Salimian, Ghobaei-Arani, and Shahidinejad 2021). Hence, in this paper, a multi-objective algorithm for the problem of IoT services placement in the fog landscape is proposed, which aims to reconcile the factors of cost, latency and utilization of fog resources. Because of their simplicity and flexibility, evolutionary approaches focus on solving a wide range of optimization problems. Therefore, we use an evolutionary approach to solve the problem of IoT services placement. Here, the decision-making phase of the MADE-k model is performed using PSO as an evolutionary approach that aims to optimally plan the deployment of services on fog nodes. PSO is a population-based optimization algorithm that performs the evolutionary process based on the motion of bird flocks and fish (Kennedy and Eberhart 1995). This algorithm has performed well in optimization scenarios and real-world applications.

The main contribution of this paper is as follows

- Design of an autonomous framework based on fog-cloud control middleware and MADE-k model for fog and cloud resources management
- Development of PSO to solve the problem of IoT services placement as a multi-objective optimization problem
- Prioritize requests based on deadlines for better deployment

The rest of the paper is organized as follows: [Section 2](#) describes the background of fog computing architecture, formulates the problem of IoT services placement, and addresses evolutionary approaches. [Section 3](#) is devoted to literature review. An overview of the proposed framework for resources management is discussed in [Section 4](#). The proposed scheme for deploying IoT services is presented in [Section 5](#). [Section 6](#) describes the experimental results and discussion. Finally, [Section 7](#) concludes this paper.

Background

This section consists of three subsections. The first subsection describes the three-layer architecture for the cloud-fog-IoT ecosystem. In the second subsection, the problem of locating IoT services is formulated and finally, evolutionary approaches are discussed in the third subsection.

Cloud-Fog-IoT Ecosystem Architecture

As shown in [Figure 1](#), the cloud manufacturing architecture based on fog computing consists of three layers: IoT devices, fog computing layer, and cloud computing layer (Natesha and Guddeti 2021). According to this architecture, fog computing is an intermediate layer between IoT devices and cloud

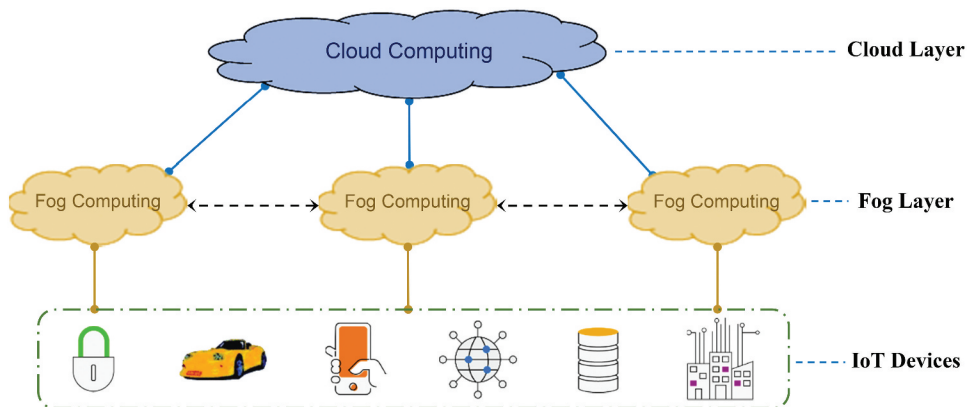


Figure 1. Cloud-Fog-IoT three-layer ecosystem architecture.

computing that provides the proper balance between these layers. The devices layer consists sensors and monitors that sense data in real-time and transmit it to the fog layer. The fog layer is used for real-time IoT data collection and analysis. This layer can provide the required service for the received requests based on the available resources. Finally, the cloud layer is responsible for storing and assigning services to requests that the fog layer is not capable of executing.

Fog computing perform communications, processing, and storage on resources that are close to end users. This is the most important advantage of fog computing, which leads to increased QoS for real-time applications (Neware and Shrawankar 2020). Figure 2 highlights some of the advantages of fog computing over other computing models.

Problem Formulation

Fog computing improves the usability of IoT-based resources, but requires an efficient strategy for services placement (Jia et al. 2018). Applications (a set of services) are sent to the fog layer by IoT devices for execution. In general, the fog landscape consists of a number of fog colonies, so that each colony consists of a number of fog nodes and a fog orchestration control node (Zhang, Wang, and Huang 2018). In addition, there is a fog-cloud control middleware in the fog layer, which plans the deployment of IoT services based on the MADE-k model (Salimian, Ghobaei-Arani, and Shahidinejad 2021). The orchestration nodes are responsible for managing and overseeing the subordinate fog

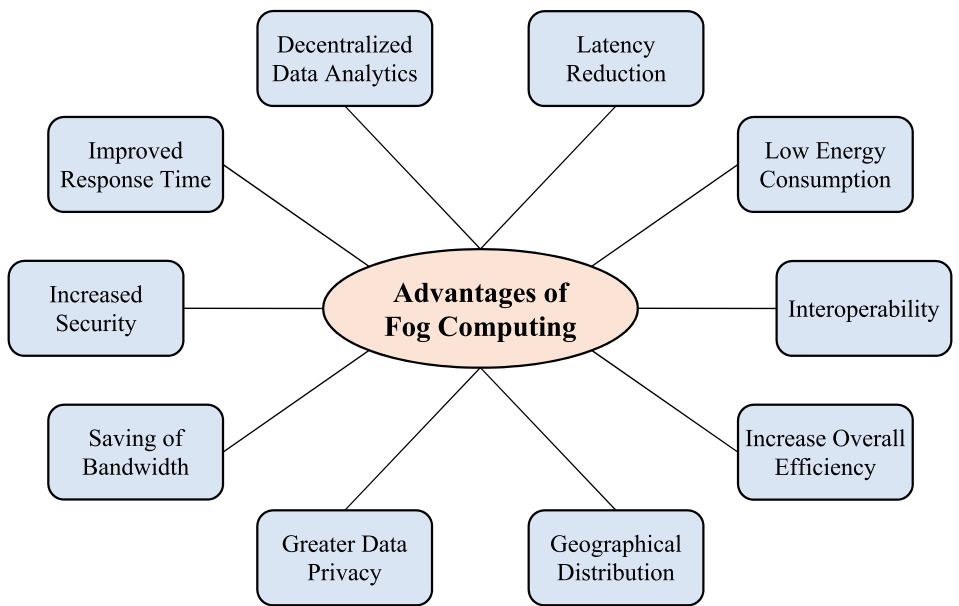


Figure 2. Some advantages of fog calculations.

colony. Therefore, each orchestration node receives the details of services placement associated with the subordinate fog colony from the middleware and apply it on the fog colony. In this regard, the orchestration nodes interact with the middleware and report the status of the colonies in each period. In addition, the fog colonies are interconnected, as shown in [Figure 1](#), where the link between the colonies is through the orchestration nodes ([Jia et al. 2018](#)). However, when the colonies do not have the resources needed to execute a request, the request is transmitted to the cloud by middleware.

In the problem of IoT services placement, the fog landscape includes n fog colonies, where F^i refers to i^{th} fog colony. O^i is the fog orchestration control node associated with F^i , and f^j refers to the j^{th} fog node in F^i . $Res(F^i)$ is the set of subordinate fog nodes in F^i , where $f^j \in Res(F^i)$, $j = 1, 2, \dots, c^i$. Links between system entities have a certain latency; l^{ij} is the link latency between f^j and O^i , l^N is the link latency of a colony with the nearest neighboring fog colony, and l^R is the link latency between the fog-cloud control middleware and cloud resources R . Let $A^k \in AP$, $k = 1, 2, \dots, m$ be the k^{th} application/request that is sent to the fog layer by IoT devices. A^k consists of r^k independent service, where $a^l \in A^k$, $l = 1, 2, \dots, r^k$ refers to the l^{th} service and must be mapped to a fog node. Therefore, before IoT applications being executed, the computational resources of all services must be specified in fog.

The three most important factors in each colony are $CT = \{U, M, S\}$, where U indicates CPU usage rate, M indicates RAM usage rate, and S indicates storage usage rate ([Skarlat et al. 2017](#)). Similarly, every service to execute has CT requirements that it should not exceed the available resources of the fog nodes. Here, U_{F^i} , M_{F^i} and S_{F^i} show the CT factors for F^i , U_{f^j} , M_{f^j} and S_{f^j} represents the CT factors for f^j , and U_{a^l} , M_{a^l} , and S_{a^l} refer to the CT factors associated with a^l . In addition to these factors, each service a^l has a deadline to execute that is denoted by D_{a^l} . Also, RT_{A^k} is the response time of A_k and should not exceed D_{A^k} (deadline A^k to execute). Therefore, there are two limitations to the resources' utilization and the response time to execute IoT applications that should not be violated in the placement process. A description of all the symbols and variables related to the problem of IoT services placement can be found in [Table 1](#).

Evolutionary Approaches

Solving the problem of optimizing the distribution of services between service providers is in the NP-Hard problem category ([Salimian, Ghobaei-Arani, and Shahidinejad 2021](#)). In general, there are two methods to solve optimization problems: exact and approximate. The approximate method consists of two types of heuristics and meta-heuristics. Examples of meta-heuristic methods

are Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Whale Optimization Algorithm (WOA), Crow Search Algorithm (CSA), Cuckoo Optimization Algorithm (COA), and Open-source Development Model Algorithm (ODMA) (Maier et al. 2019; Talatian Azad, Ahmadi, and Rezaeipناه 2021). Swarm intelligence-based meta-heuristic methods are also known as evolutionary approaches. This paper uses an evolutionary approach to problem solving (Maier et al. 2019). Here, the PSO algorithm is used as the proposed approach and algorithms ODMA, CSA, WOA and COA as the comparative approaches. Hence, these algorithms are briefly described below.

- PSO: This algorithm was introduced by Kennedy and Eberhart (1995) for optimization applications. PSO is inspired from the nature social behavior and dynamic movements with communications of fish, birds and insects. Uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Each particle in search space adjusts its “flying” according to its own flying experience as well as the flying experience of other particles. Each particle has three parameters position, velocity, and previous best position (*pbest*), particle with best fitness value is called as global best position (*gbest*). Collection of flying particles (swarm) are trying to movement toward a promising area to achieve global optimization. Each particle dynamically adjusts its velocity according to flying. In addition, each particle modifies its position according to the current position, current velocity, distance between current position and *pbest*, and distance between current position and *gbest*.
- ODMA: This algorithm was introduced by Hajipour, Khormuji, and Rostami (2016) as a new meta-heuristic algorithm to solve optimization problems. ODMA is a social-based method and includes swarm-based and population-based features. In this algorithm, each point in the solution space is considered as an open-source software and evolves over time by the open-source development mechanism. ODMA assumes two general types of software: Leading and Promising. Leading software is more optimistic, so that other software is known as promising. Over time, some softwares become leading or show to be promising, and some of them cannot developed enough and removed. Evolution phase operations in this algorithm include moving toward leading softwares, evolution of the leading softwares based on their history, and forking from leading softwares.
- CSA: This algorithm was introduced by Askarzadeh et al. (2016) as a novel meta-heuristic technique based on the intelligent behaviors of crows. The CSA works on the idea that crows store their excess food in hiding places and retrieve it when needed. The optimization process in this algorithm is done by setting only two parameters (i.e., flight length and awareness probability), which is very attractive for solving various

engineering design problems. CSA uses the flight length parameter to control convergence and the awareness probability parameter to control diversity.

- WOA: This algorithm has been introduced by Mirjalili and Lewis (2016) as a population-based optimization algorithm. WOA mimics the social behavior of humpback whales and is inspired by the bubble-net hunting strategy. This algorithm performs the evolution process in three phases: encircling prey, exploitation (bubble-net attack) and exploration (prey search). The WOA assumes that the best solution for the current candidate is to prey for the target or close to the desired state. After that, other agents try to update their position to the best search agent. WOA has been tested on 29 well-known test functions that confirm its performance in practice.
- COA: This algorithm was introduced by Yang and Deb (2009) for continuous nonlinear optimization problems. COA is based on the life of a bird family called a Cuckoo, where the population of cuckoos in different societies is divided into two types of mature cuckoos and eggs. In this algorithm, cuckoos are required to lay eggs in nest of other birds. A number of eggs that are more similar to the host eggs will have a better chance of growing and becoming cuckoos. Other eggs are also detected and destroyed by the host. Therefore, the basis of COA is derived from the effort to survive among cuckoos. The survived cuckoo societies immigrate to a better environment and start reproducing and laying eggs.

Literature Review

In this section, resource management approaches in fog and cloud computing, including single-objective and multi-objective, are examined. The purpose of reviewing these studies is to highlight the importance of using multi-objective approaches versus single-objective approaches. In the problem of services placement, the decision maker is usually interested in multi-objective problem optimization (Skarlat et al. 2017). However, many studies use single-objective models to solve this problem (Lee, Saad, and Bennis 2019; Ren et al. 2019). In general, the importance of optimal resource management and service provision in fog and cloud computing is understood by the research society. Unlike the cloud, research in fog is still immature (Ren et al. 2019). Currently, the focus of resource management researchers on fog computing has been to reduce costs or latencies (Ren et al. 2019). In addition, objectives such as improving QoS, minimizing energy consumption, maximizing the utilization of network resources, improving response time, and minimizing transfer to cloud have also been considered by researchers (Ren et al. 2019).

Minh et al. (2017) proposed a solution to the problem of services placement in fog with the aim of maximizing the utilization of resources. Here, latency is considered as a limitation and this has led to inadequate control of response time. In some cases, a possible solution to the problem is not found because the response time does not exceed one threshold. In addition, this paper does not consider the service costs for deciding whether to deploy services. However, response time and cost are important components in this problem and there is a lot of research in the literature that considers these components to provide services (Jia et al. 2018).

Ramirez et al. (2017) reviewed the benefits of using fog-to-cloud (F2C) in providing service for IoT applications. F2C can reduce energy consumption, network congestion and service time. The authors proposed two dynamic service placement methods, including Random Fit and First-Fit, but did not consider QoS and service cost. Skarlat et al. (2017) introduced an architecture for providing services in fog colonies and then proposed several algorithms for use in this architecture. The purpose of these algorithms is to optimize the utilization rate of fog resources. Mahmoud et al. (2018) developed an IoT architecture with fog enabled cloud for smart healthcare applications. In this architecture, an energy-aware module is used to services placement. The authors make all the decisions to the services placement based on energy consumption and do not consider deadlines and costs.

Yousefpour et al. (2018) introduced a framework for IoT applications in the fog and cloud and a cooperation and offloading policy aimed at reducing latencies. The main purpose of this paper is to reduce the latency in responding to requests using offloading. Souza et al. (2018) used the service atomization technique to deploy distributed services in the IoT-fog-cloud environment. In this method, services are atomized and distributed in parallel in the environment. The authors used best-fit to map fog nodes to IoT services based on service deadlines. Kim and Chung (2018) considered the problem of IoT applications deployment as a mixed-integer non-linear programming (MINLP) problem. The authors used the energy consumption of fog resources to optimize the problem. In addition, they proposed the fog portal as a cloud computing infrastructure that highlights user participation in service provision.

Santoyo-Gonzalez and Cervello-Pastor (2018) formulated the problem of IoT services placement for 5 G networks and used a simulated annealing algorithm to solve it. The authors considered the problem as a MILP model based on cost reduction. Jia et al. (2018) used a double matching strategy (DMS) to improve the services deployment in fog computing. The main purpose of this method is to allocate cost-effective resources. Yang et al. (2018) proposed a Lyapunov optimization algorithm to solve the problem of IoT service placement in fog computing. The main purpose of this algorithm

for optimization is to reduce latency and energy consumption. Here, the authors use delay energy-based task scheduling (DBTS) to minimize energy consumption.

Zhang, Wang, and Huang (2018) optimized the problem of service placement by considering several objectives (i.e., multi-objective). The objectives considered in this method include load balancing and reliability. The authors use adaptive bacterial foraging optimization to solve the problem. Brogi et al. (2018) proposed a multi-objective optimization scheme to solve the problem of services placement in a fog environment. This solution provides a balance between QoS assurance, utilization of resources and deployment costs. However, when comparing considered metrics, response times are more important in fog computing. Yousefpour et al. (2019) proposed a comprehensive model of IoT applications and resources in fog and cloud computing. The authors formulated the problem of service placement and proposed an integer nonlinear model to solve it. In addition, two heuristic algorithms are proposed to reduce the complexity of solving this problem. Although this approach focuses on reducing complexity, but its efficiency is drastically reduced by considering both latency and cost.

Hassan, Azizi, and Shojafar (2020) deployed services based on energy consumption and priority in the fog-cloud environment. The authors consider a maximum of two services in IoT applications and choose the most appropriate platform from the fog and cloud. Murtaza et al. (2020) introduced an adaptive placement approach in the fog-cloud environment that improves QoS in terms of latency and energy consumption. The authors use the reinforcement learning policy to determine the optimal mapping between fog nodes and IoT services. Xavier et al. (2020) introduced an innovative approach to providing resources in the IoT-fog-cloud environment. The authors consider the interaction between fog and cloud as well as the heterogeneity of devices in fog to allocate resources. Chen et al. (2020) proposed a Stackelberg-based resource allocation framework for fog computing. The entities of this framework include mobile fog-cloud and end-users, which breaks down the problem into subproblems. Here, for each subproblem, only the type of resource is considered and the price is estimated.

Khosroabadi, Fotouhi-Ghazvini, and Fotouhi (2021) proposed an approach for services placement in IoT networks with the real-time fog-assisted. The authors use a heuristic algorithm based on fog node clustering to solve the problem. This algorithm based on QoS has reported promising results. Natesha and Guddeti (2021) solved the problem of deploying services in fog computing using an elitism-based genetic algorithm. This problem is considered as a multi-objective optimization problem to minimize energy consumption, service time and cost. In addition, the authors developed a fog framework for two-level resources provisioning using containers. Salimian, Ghobaei-Arani, and Shahidinejad (2021) introduced an autonomous approach to the

Table 1. Description of the symbols related to the problem of services placement.

Parameter	Description	Parameter	Description
t	Current time	R_{A^k}	Request time of A_k
ρ	Time period	$x_{a^l}^{f^j}$	Decision variables related to f^j and a^l
n	Total number of colonies	$x_{a^l}^O$	Decision variables related to orchestration and a^l
F^i	i^{th} fog colony	$x_{a^l}^N$	Decision variables related to nearest colony and a^l
O^j	j^{th} fog orchestration control node	K	Total number of services
f^j	j^{th} fog node	C	Total number of resources available in the period ρ
$Res(F^i)$	Set of subordinate fog nodes in F^i	UF	utilization of fog
$Res^{a^l}(F^i)$	set of fog nodes associated with a^l in F^i	TP	Throughput
AP	Set of requested applications	RT	Response time
A^k	k^{th} application/request	SC	Service cost
m	Total number of applications	ξ_{UF}	Utilization of fog weight
r^k	Number of services in A_k	ξ_{TP}	Throughput weight
a^l	l^{th} service of an application	ξ_{RT}	Response time weight
U_{F^i}	CPU usage rate of F^i	ξ_{SC}	Service cost weight
M_{F^i}	RAM usage rate of F^i	$x_{a^l}^{f^j}$	Decision variable for node f^j
S_{F^i}	Storage usage rate of F^i	$x_{a^l}^O$	Decision variable for orchestration node
U_{f^j}	CPU usage rate of f^j	$x_{a^l}^N$	Decision variable for nearest neighbor colony
M_{f^j}	RAM usage rate of f^j	Srv_D	Number of services deployed before the deadline
S_{f^j}	Storage usage rate of f^j	Srv_{TD}	All services sent to fog colony
U_{a^l}	CPU usage rate of a^l	$Cost_{Srv_D}$	Cost of services related to Srv_{TD}
M_{a^l}	RAM usage rate of a^l	WT_{a^l}	Waiting time for service deployment a^l
S_{a^l}	Storage usage rate of a^l	EC_{a^l}	Execution time for service a^l
D_{a^l}	Deadline of a^l	SC_{comm}	Communication cost
D_{A^k}	Deadline of A^k	SC_{comp}	Computing cost
l^{f^j}	Latency between f^j and O^j	$cv_{\alpha,\beta}$	Data size between requests t_α and t_β
l^N	Link latency of a colony with the nearest colony	BW	Bandwidth between two nodes
l^R	Link latency between the middleware and cloud	CP	Communication unit price
RT_{A^k}	Response time of A_k	PP	Computing unit price

optimal deployment of MADE-k-based IoT services in the fog landscape. The authors used the Gray Wolf Optimization (GWO) algorithm to optimize service placement. This method increases system performance by taking into account execution costs and adapts to the dynamic behavior of the environment. Also, the simulation results show that this method has been converged with 93.7% for services placement in fog nodes.

The studies examined are summarized based on the objectives set out in Table 2.

Many studies in the literature have addressed the problem of services placement in fog computing, so that only some of them were examined in this section. However, there are shortcomings in this research, which can be attributed to the lack of association quantify between different objectives. The problem of service placement involves deployment planning with some limitations (Natesha and Guddeti 2021). This paper addresses five key features, as shown in Figure 3. The first feature refers to the planning scheme, where it can be done centrally or distributed. The second feature is related to the problem-

Table 2. A summary of the approaches examined.

Authors	Optimization type	Utilization of resource	Response time	Cost	Latency	QoS	Throughput	Deadline	Energy	Two-level
Minh et al. (2017)	Single objective	✓	x	x	x	x	x	x	x	x
Ramirez et al. (2017)	Single objective	x	✓	x	x	x	x	x	✓	x
Skarlat et al. (2017)	Single objective	✓	x	x	x	x	x	x	x	x
Mahmoud et al. (2018)	Single objective	x	x	x	x	x	✓	x	✓	x
Yousefpour et al. (2018)	Single objective	x	✓	x	x	x	x	x	x	x
Souza et al. (2018)	Single objective	✓	✓	x	x	x	x	x	x	x
Kim and Chung (2018)	Single objective	x	x	x	x	x	x	x	x	x
Santoyo-Gonzalez and Cervello-Pastor (2018)	Single objective	x	x	✓	x	x	x	✓	✓	x
Jia et al. (2018)	Single objective	x	x	✓	x	x	✓	x	x	x
Yang et al. (2018)	Single objective	x	✓	x	x	x	x	x	✓	x
Zhang, Wang, and Huang (2018)	Multi-objective	✓	✓	x	x	x	✓	x	x	x
Broggi et al. (2018)	Multi-objective	x	x	✓	x	✓	x	x	x	x
Yousefpour et al. (2019)	Single objective	x	x	✓	✓	x	x	x	x	x
Hassan, Azizi, and Shojafar (2020)	Single objective	x	✓	x	x	x	x	x	✓	x
Murtaza et al. (2020)	Single objective	x	✓	x	x	x	x	x	✓	x
Xavier et al. (2020)	Single objective	✓	✓	x	x	x	x	x	x	x
Chen et al. (2020)	Single objective	✓	✓	x	x	x	x	x	x	x
Khosroabadi, Fotouhi-Ghazvini, and Fotouhi (2021)	Multi-objective	✓	✓	✓	✓	x	✓	x	x	x
Natesha and Guddeti (2021)	Multi-objective	✓	✓	✓	x	x	x	✓	✓	✓
Salimian, Ghoabaei-Arani, and Shahidinejad (2021)	Multi-objective	✓	✓	✓	✓	x	✓	✓	x	x
Proposed approach	Multi-objective	✓	✓	✓	✓	x	✓	✓	x	x

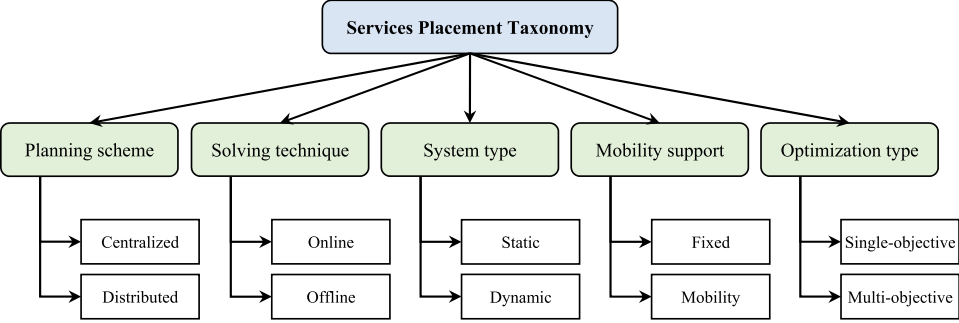


Figure 3. Taxonomy of services placement problem.

solving technique, where the problem can be solved offline or online. The third feature controls whether the system is static or dynamic, and supports the mobility of fog nodes and end users in the fourth feature. Finally, the fifth feature refers to the type of optimization, which can be single-objective or multi-objective.

Proposed Framework

The proposed fog computing framework includes the three-layer cloud-fog-IoT ecosystem, as shown in Figure 4. As depicted, the cloud is at the highest level, then the fog layer, and at the lowest level of IoT devices. At the IoT devices layer, there is a set of smart objects that collect data and transfer it to the fog layer. The fog layer processes requests received from the IoT devices layer to provide services. Depending on some characteristics (for example, real-time), each request can be executed at the same level (i.e., fog layer) or sent to the cloud layer.

The proposed conceptual computing framework is based on a fog-cloud control middleware that provides optimal resource management. This middleware is responsible for controlling the fog landscape and interacting with the cloud layer. The middleware according to MADE-k model can process applications/requests received from IoT devices and determine the appropriate platform for servicing them without engaging the cloud. Each IoT application includes several services, the resources of which must be provided by fog or cloud. Therefore, the middleware must have a mechanism for planning the services deployment, where the deployment process takes place at each ρ time period. The MADE-k model consists of four phases of monitoring, analysis, decision-making and execution, which are linked to a knowledge base.

The fog layer is composed of several fog colonies. Each fog colony includes a fog orchestration control node and a number of fog nodes. Fog node is an application software that is implemented on IoT devices. The orchestration node is responsible for managing and monitoring the

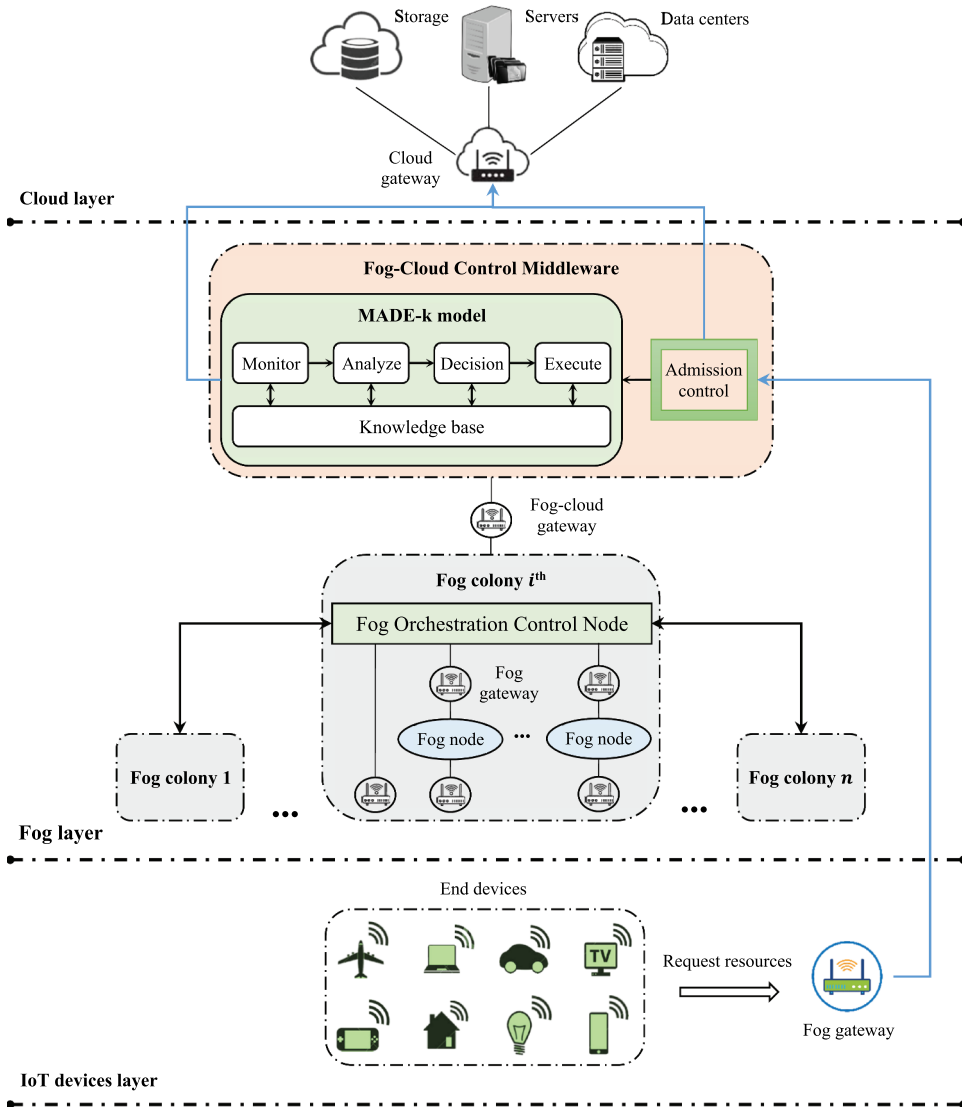


Figure 4. Proposed ecosystem framework with three layers of cloud-fog-IoT.

subordinate fog nodes and is supported by the fog-cloud control middleware. If a colony fails to provide the resources needed for a request, the request is passed by the orchestration node to the nearest neighboring fog colony. This scenario can be repeated by other colonies. Finally, if the colonies fail to provide the resources needed for the request, the request is sent to the cloud layer. Thus, fog colonies communicate with each other through orchestration nodes. The orchestration node can identify the nearest neighboring colony by examining the link latency when communicating with other colonies. Each node in the fog colony is connected to

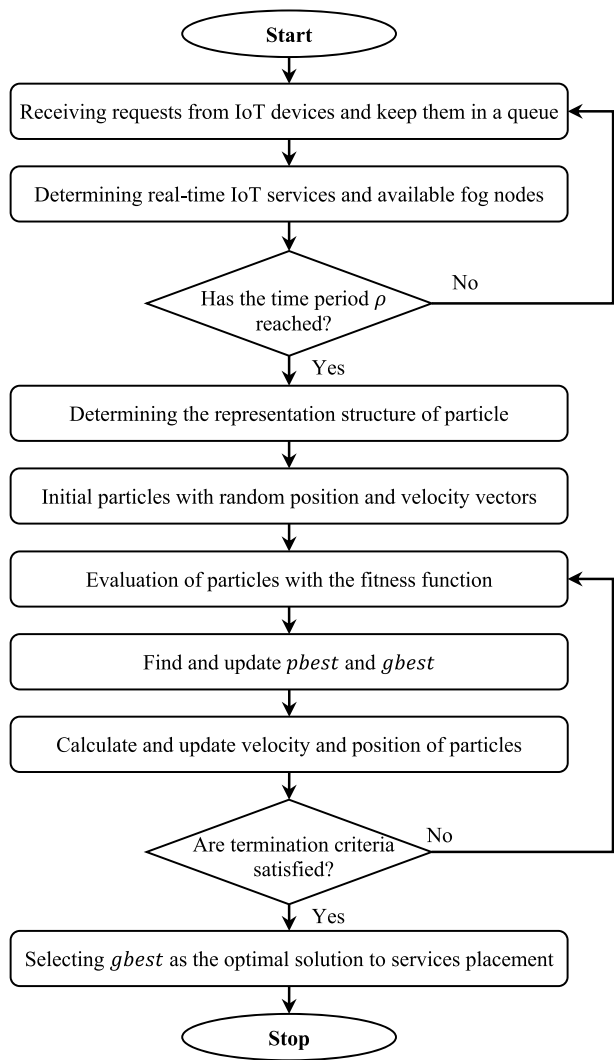


Figure 4. Continued.

the orchestration node with a separate link. The latency of these links and the links between the orchestration nodes with the middleware as well as the links between the orchestration nodes themselves are insignificant and negligible.

fog-cloud control middleware has an admission control unit that can process requests to provide resources. Requests for IoT devices are sent to the admission control unit through the fog gateway. This unit has a response time threshold for identifying real-time applications. Therefore, the admission control unit can detect real-time applications based on the applications deadline and the response time threshold. Finally, the unit sends latency-sensitive and real-time applications to the fog layer and refers other applications to the

cloud layer. In addition, the MADE-k model is one of the main components of the middleware and decides on the deployment of each IoT service. This model performs the deployment process for each fog colony in a distributed manner. Deployment planning details are sent by the middleware to the orchestration nodes in each colony, and then IoT services are executed by the fog nodes. Fog nodes have computing power (i.e., RAM, CPU, and storage), so they can execute IoT services. The pseudocode of the proposed framework for IoT services placement is shown in [Algorithm 1](#).

Algorithm 1. Pseudocode of the proposed framework

Input: Set of IoT services requested for A_k and details of fog nodes
Output: Services deployment planning

```

1:   for each time period  $\rho$  in MADE-k model do
2:       for each set of application  $A_k$  at time period  $\rho$  do
3:           Monitor (set of IoT services requested for  $A_k$ ).
4:           Monitor ( $U_{a_i}, M_{a_i}, S_{a_i}, D_{a_i}$ ).
5:       end for
6:       for each fog node  $f^j$  in  $F^i$  at time period  $\rho$  do
7:           Monitor ( $f^j$  status in  $F^i$  at time period  $\rho$ ).
8:           Monitor ( $U_{f_i}, M_{f_i}, S_{f_i}, D_{f_i}$ ).
9:       end for
10:      for each set of application  $A_k$  at time period  $\rho$  do
11:          Calculate the priority of IoT services with Eq. (1).
12:      end for
13:      for each set of application  $A_k$  at time period  $\rho$  do
14:          Using PSO to determine a deployment plan to map fog nodes to IoT services.
          // Deployment planning is based on service priority and monitoring of fog nodes.
15:      end for
16:      for each set of application  $A_k$  at time period  $\rho$  do
17:          Perform IoT services placement on fog nodes based on deployment plan for  $A_k$ .
18:      end for
19:  end
20:  Planning the deployment of IoT services created by the PSO algorithm.

```

The loop embedded in line 1 is for performing MADE-k model phases in different time periods. According to lines 2–5, the monitoring phase is performed for incoming requests. This phase applies to fog nodes in lines 6–9. In the monitoring phase, usage rate of CPU, RAM and storage are analyzed, and then the status of fog nodes, IoT services, and IoT devices are stored in the knowledge base. The analysis phase is performed in lines 10–12, where according to Eq. (1) the priority of execute application implementation is determined (Skarlat et al. 2017).

$$P(A_k) = \alpha \times \frac{1}{D_{A_k}} + (1 - \alpha) \times \frac{1}{t - R_{A_k}} \quad (1)$$

Where, $P(A_k)$ indicates the priority of the A_k application. D_{A_k} and R_{A_k} are the deadline and request time for A_k , respectively. Also, t is the current time and α is weight factor to consider the effect of the D_{A_k} and R_{A_k} parameters.

In lines 13–15 of the proposed pseudocode, the PSO algorithm is applied to deploy applications (IoT services) on fog nodes. Deployment planning is related to the decision-making phase of the MADE-k model, the details of

which are discussed in [Section 5](#). Finally, lines 18–16 are related to the execution phase that apply the created planning scheme to the fog colonies. The computational complexity of Algorithm 1 depends on the number of services (i.e., K), the number of nodes available (i.e., C) as well as the complexity of the PSO. In general, the complexity of PSO depends on its fitness function. Let FF_{PSO} be the computational complexity of PSO based on fitness function. Therefore, the computational complexity of the algorithm is $O(K + C + FF_{PSO})$.

Proposed Placement Scheme

The purpose of solving the problem of IoT services placement is to optimize the mapping between fog nodes and IoT services so that some constraints are satisfied. In this paper, the PSO algorithm is used as an evolutionary algorithm for optimization work. The cloud-fog-IoT ecosystem has many challenges in providing resources for IoT applications; for example, minimizing the service cost, service time, response time, latency, response time, energy consumption, SLA violations, and maximizing the throughput, utilization of fog and number of services performed. However, many studies have highlighted service cost, response time, throughput, and utilization of fog as the main objectives in solving this problem (Natesha and Guddeti [2021](#); Xavier et al. [2020](#)). Hence, we formulate the problem of IoT services placement as a multi-objective optimization problem to maximize throughput and utilization of fog, and minimize service cost and response time in fog computing.

Figure 4. Flowchart proposed algorithm for planning services deployment

The PSO algorithm is configured as a deployment planning scheme in the decision-making phase of the MADE-k model. Here, the services requested by IoT devices are queued and each ρ period is deployed. Like other evolutionary algorithms, the PSO algorithm consists of four main steps: (1) solution representation and generate initial population, (2) fitness function calculation, (3) population evolution strategy, and (4) determining the stop condition. These steps are described below. For a better understanding, the flowchart of the proposed algorithm is shown in [Figure 5](#). In addition, a description of the symbols and variables associated with this algorithm can be found in [Table 3](#).

The Initial Population

The first step of any evolutionary algorithm is to generate a set of possible solutions (often randomly) as the initial population. In the PSO algorithm, each solution is known as a particle. Each solution is an encoding of the problem that actually shows the structure of the solution representation. In this problem, each solution is a sequence of IoT services that are mapped to available nodes. Here, the structure of the solution representation is a vector of

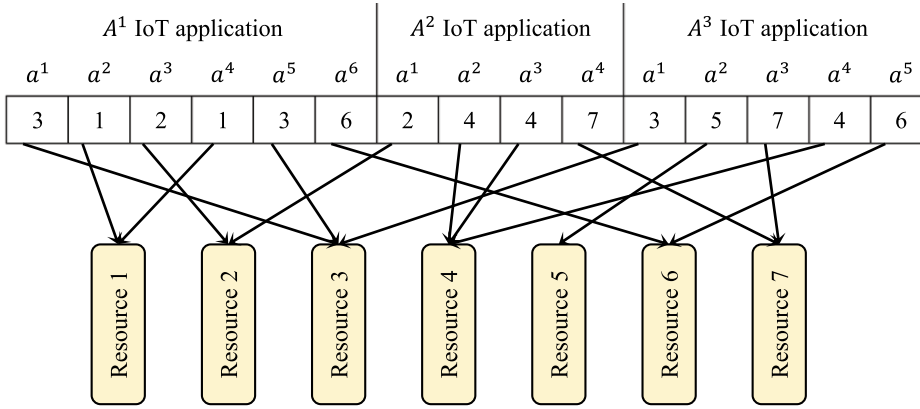


Figure 5. An example of a solution structure in the PSO algorithm.

length K , so that each element is a natural number in the range $[1 - C]$. Here, K is the total number of services from all applications and C is the total number of resources available in the ρ time period. Available resources can be fog nodes, fog orchestration control nodes or cloud. Consequently, each service of IoT applications is identifiable with a specific position in the solution. An example of a solution structure with 15 services from 3 IoT applications and 7 available resources is shown in Figure 5.

In this example, the available resources are $Res = [1, 2, 3, 4, 5, 6, 7]$ and the sequence of services based on the solution is $P = [3, 1, 2, 1, 3, 6, 2, 4, 4, 7, 3, 5, 7, 4, 6]$. Accordingly, the first service is mapped to the third resource, the second service is mapped to the first resource, and so on. Due to the fact that resources in fog nodes are considered as blocks, so one node can support more than one service. For example, both services a^2 and a^4 are developed in the first resource. The initial population in the PSO algorithm is defined based on the solution representation structure and also randomly generated according to the services in the queue. Here, the population consists of N_p solutions that are generated in each time period.

Fitness Function

In this paper, the problem of IoT services placement is formulated as a multi-objective optimization problem to maximize throughput and utilization of fog, and minimize service cost and response time in fog computing. Therefore, the proposed objective function for deployment planning consists of four objectives, as shown in Eq. (2).

$$F_{ness} = \frac{\xi_{RT} * RT + \xi_{SC} * SC}{\xi_{UF} * UF + \xi_{TP} * TP} \quad (2)$$

Table 3. Description of the symbols related to the PSO algorithm.

Description	Parameter	Description	Parameter
N_p	Population size	x_{min}	Lower bound of position
$Iter_{max}$	Maximum iteration	x_{max}	Upper bound of position
$pbest_i$	Local best position of i^{th} particle	v_{min}	Lower bound of velocity
$gbest$	Global best position of all particles	v_{max}	Upper bound of velocity
X_i	Position vector	c_1 and c_2	Acceleration constants
V_i	Velocity vector	r_1 and r_2	Random values in $[0, 1]$ range
ω	Inertia weight to control the velocity impact	τ	Iteration

Where, RT , SC , UF and TP are response time, service cost, utilization of fog and throughput, respectively. Also, ξ_{RT} , ξ_{SC} , ξ_{UF} and ξ_{TP} are the weights of these objectives, respectively. In this paper, the effect of all objectives is considered the same, so the weight coefficient of all objectives is set to 1. According to the definition of the fitness function, the purpose of optimization is to minimize it. The process of calculating RT , SC , UF and TP are discussed below.

Response Time (RT): This criterion for a^l service refers to the time interval between receiving the service by the fog orchestration control node and issuing the first response. This time is defined as RT_{a^l} and is calculated according to Eq. (3).

$$RT_{a^l} = WT_{a^l} + EC_{a^l} \quad a^l \in A^k, A^k \in AP \quad (3)$$

Where, WT_{a^l} is the waiting time for service deployment and EC_{a^l} is the service execution time on the fog node. WT_{a^l} is calculated based on the total waiting time to start the next period and the time required to find the fog node with the lowest service cost. In addition, EC_{a^l} is considered based on the cost of executing the service on a fog node.

Service Cost (SC): The cost of service related to communication and computing is based on the utilization of fog resources (Faraji Mehmandar, Jabbehdari, and Haj Seyyed 2020), as shown in Eq. (4).

$$SC = SC_{comm} + SC_{comp} \quad (4)$$

Where, SC_{comm} is the communication cost between two consecutive services (for example, t_α and t_β) on a fog node, and SC_{comp} refers to the computing cost. These costs are defined based on Eq. (5) and Eq. (6).

$$SC_{comm} = CP * \frac{cv_{\alpha,\beta}}{BW} \quad (5)$$

$$SC_{comp} = PP * (t_\beta - t_\alpha) \quad (6)$$

Where, CP and PP are the unit price of communication and computing for a fog node, respectively. t_α and t_β refer to the time of requests α and β , respectively. $cv_{\alpha,\beta}$ is the size of the data between requests t_α and t_β , and BW refers to the bandwidth between two nodes. Saeedi et al. (2020) recommend CP at \$ 0.1 and BW at 20 Mbps.

Utilization of Fog (UF): This criterion indicates the number of service placements used for fog resources that should be maximized. The UF is calculated based on the communication and deployment letancy associated with sending the service to the nearest neighboring fog colony or cloud, as this can lead to breaches of application deadlines. Therefore, to calculate the UF , the priority of the applications must be calculated based on the deadline (Skarlat et al. 2017). Accordingly, the rate of utilization of fog resources is calculated based on Eq. (7).

$$UF = \sum_{A^k \in AP} \left(P(A^k) * \sum_{a^l \in A^k} \left(\sum_{f^j \in Res^{a^l}(F^i)} x_{a^l}^{f^j} + x_{a^l}^O + x_{a^l}^N \right) \right) \quad (7)$$

Where, AP is the set of requested applications, a^l is l^{th} service of A^k , f^j is j^{th} fog node, F^i is i^{th} fog colony, and $Res^{a^l}(F^i)$ is the set of fog nodes associated with a^l in F^i . $P(A^k)$ is the priority of the application A^k and is calculated through Eq. (1). Also, for each fog node that is ready to execute service a^l , three decision variables are defined binarily. These variables include $x_{a^l}^{f^j}$, $x_{a^l}^O$ and $x_{a^l}^N$, which refer to the j^{th} fog node, the fog orchestration control node, and the nearest neighboring fog colony, respectively. In fact, decision variables indicate to which resource the service a^l is mapped.

Throughput (TP): Throughput is calculated based on various factors. This criterion can mean maximizing the number of services executed relative to the total services sent to the fog colonies. Also, SC can mean minimizing the cost of executing these services. Based on different interpretations, we use a hybrid approach to calculate throughput, as shown in Eq. (8).

$$TP = \frac{TP_A}{TP_B} \quad (8)$$

Where TP_A is the throughput of the number of services executed relative to the total of services sent to the fog colonies. Also, TP_B is the operating cost of executing these services, which is considered after complete deployment. TP_A and TP_B are calculated based on Eq. (9) and Eq. (10).

$$TP_A = \frac{Srv_D}{Srv_{TD}} \quad (9)$$

$$TP_B = Cost_{Srv_D} \quad (10)$$

Where, Srv_D refers to the number of services that have been deployed and executed on the fog nodes before the deadline, and Srv_{TD} is the total number of services sent to the fog colony. Also, $Cost_{Srv_D}$ refers to the total cost of services executed in the fog colony before the deadline.

Evolution Strategy

Each particle in the PSO has a position and is represented by $X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,K}\}$, where K is the dimensionality of the search space. In addition, each particle has a velocity, which is represented as $V_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,j}, \dots, v_{i,K}\}$. In each iteration, each particle updates its position and velocity according to $pbest$ and $gbest$, as given in Eq. (11) and (12).

$$v_{ij}^{t+1} = \omega \cdot v_{ij}^t + c_1 \cdot r_1 \cdot (pbest_i - x_{ij}^t) + c_2 \cdot r_2 \cdot (gbest - x_{ij}^t) \quad (11)$$

$$x_{ij}^{\tau+1} = x_{ij}^{\tau} + v_{ij}^{\tau+1} \quad (12)$$

Where, x_{ij}^{τ} and $x_{ij}^{\tau+1}$ are the positions of the j^{th} element of the i^{th} particle in iteration τ and $\tau + 1$, respectively. Similarly, v_{ij}^{τ} and $v_{ij}^{\tau+1}$ refer to velocity. Also, ω is inertia weight to control the velocity impact, c_1 and c_2 are acceleration constants and r_1 and r_2 are random values in $[0, 1]$ range. In addition, parameters within particles are indexed with τ , which it refers to iteration in the evolutionary process.

First, each $x_{i,j} \in X_i$ is randomly initialized in the range $[1 - C]$. In addition, the velocity vector for each $v_{i,j} \in V_i$ is randomly set in the range $[v_{min} - v_{max}]$ where v_{min} and v_{max} are the lower and upper bound velocities, respectively. Due to the solution structure, the position of IoT services in the solution cannot exceed the total number of available nodes (i.e., C). Therefore, for the position of the IoT services it is always $x_{min} = 1$ and $x_{max} = C$. Therefore, in the evolution process, all elements of the position vector must be a natural value in the range $[x_{min}, x_{max}]$. Accordingly, after each update, the position of each $x_{i,j}$ is corrected to the nearest allowed natural within the specified range.

Stop Condition

After completing an iteration of the PSO algorithm, the current population has evolved from the previous population, although this evolution may be small. This process is repeated until a stop condition is reached. In this paper, the

condition of stopping the algorithm is defined based on the fixed number of iterations, i.e., $Iter_{max}$. Once evolution has stopped, the best solution based on the fit function is used to solve the IoT services placement.

Experimental Results

This section deals with evaluating and comparing the proposed approach to solving the problem of IoT services placement in the cloud-fog-IoT ecosystem. Evaluation and comparison are performed based on various services performed, waiting time, failed services, services cost, services remaining, and runtime. Four evolutionary algorithms (i.e., ODMA, CSA, WOA and COA) have been used to compare the performance of the proposed PSO algorithm. The results of all the compared algorithms are reported based on the framework proposed in this paper. In addition, the simulation was performed by MATLAB R2019a and its source code is available from <https://github.com/mostafa13651365/PSO-FSPP>. The algorithms are implemented on Lenovo laptop IdeaPad 320 with Intel Core i7-7500 U processor at 3.5 GHz and 8GB RAM. In addition, the results of all experiments are presented on the basis of an average of 25 independent performances to be more reliable.

Experimental Setup

The proposed framework is simulated for 1 fog colony (Salimian, Ghobaei-Arani, and Shahidinejad 2021), where the cost of cloud computing is \$0.3 per billing time unit (Ibrahim et al. 2020). The simulation parameters based on Salimian, Ghobaei-Arani, and Shahidinejad (2021) and Skarlat et al. (2017) are set as follows: $N_p = 25$, $Iter_{max} = 100$, $\omega = 0.9$, $c_1 = 2$, $c_2 = 2$, $\alpha = 0.5$, $t^R = 1s$, $t^N = 0.5s$, $t^f = 0.3s$.

The results of the algorithms are compared based on a time period ($\rho = 1$) as well as 10 consecutive time periods ($\rho = 10$) for 1000 requests. In the first scenario, the fog colony structure and the requested IoT services are assumed according to Table 4. These details for the second scenario are presented in Table 5. Here, it is assumed that these parameters follow the normal distribution. In addition, resources are considered as blocks that are allocated to services according to demand.

Table 4. Fog colony structure and the IoT services requested for the first scenario.

Characteristic	Value
Number of types of services supported in the fog colony	20
Number of resource blocks for each fog node	Random between 5500 and 6000
Number of resource blocks for each service	Random between 25 and 35
Cost of each service	Random between 10 and 20
Number of fog colony nodes	15
The range of the number of services requested in a time period	{50, 100, 150, 200, 250, 300, 350, 400, 450, 500}

Table 5. Fog colony structure and the IoT services requested for the second scenario.

Characteristic	Value									
Number of types of services supported in the fog colony	20									
Number of resource blocks for each fog node	Random between 3000 and 3500									
Number of resource blocks for each service	Random between 5 and 15									
Cost of each service	Random between 5 and 15									
Number of fog colony nodes	5									
Number of services requested to fay colony (random selection)	Period ρ	1	2	3	4	5	6	7	8	9 10
	Time t	8	16	24	32	40	48	56	64	72 80
	Number of services	71	48	48	46	96	77	37	52	85 55

Table 6. Details of resources for fog nodes.

Fog node type	CPU (MIPS)	RAM (MB)	Storage (MB)
Fog node 1	100–200	256	256
Fog node 2	200–300	512	512
Fog node 3	300–1400	1024	1024
Fog node 4	1400–1600	2048	2048
Fog node 5	1600–3000	4096	4096

Table 7. Details of resources for different types of services.

Service	CPU (MIPS)	RAM (MB)	Storage (MB)
Service type 1	100–200	128	128
Service type 2	200–300	256	256
Service type 3	300–1400	512	512
Service type 4	1400–1600	1024	2048
Service type 5	1600–3000	4096	4096

Table 6 show the details of resource demand for fog nodes in the first and second scenarios. Here, for every 5 consecutive nodes, resource demand is considered similarly. Various analyzes have been performed on the number of fog nodes considered in the scenarios. In general, considering the large number of nodes in each colony increases the planning time for deploying services. Also, colonies with low number of nodes cannot show the performance of the placement approach well, because low number of nodes does not lead to the creation of a planning queue. Experimentally, the appropriate number of nodes is considered to be 5 for the first scenario and 15 for the second scenario. This number can well show the performance of the placement approach according to the considered parameters. In general, requests are sent to the fog layer as applications by IoT devices. The deadline for all applications is randomly set between 120s and 240s. In this paper, 5 different types of services are considered that the details of their resources are given in Table 7. However, each IoT device can have different types, for example Apple smartwatch, Samsung smartwatch and so on. According to Tables 4–5, the number of services types associated with different types of IoT devices that can be supported by fog colonies is 50.

Comparison and Discussion

The purpose of this section is to evaluate and compare five optimization algorithms (i.e., ODMA, CSA, WOA, COA and PSO) in solving the problem of IoT services placement on fog and cloud resources. This comparison was performed for 1000 incoming requests based on two scenarios. In the first scenario, the evaluation of all algorithms is performed in one time period, but in the second scenario, the comparison of algorithms is presented based on 10 time periods. Convergence, runtime and fitness function are the evaluation metrics in the first scenario. However, the evaluation metrics in the second scenario include services performed, waiting time, failed services, services cost, remaining services, and runtime.

Comparison Based on One Time Period

Here, the results of different algorithms are analyzed and compared based on one time period. Here, deploying and executing applications is assumed in the fog landscape, so there is no cost to executing services in the cloud. In addition, due to the availability of fog landscape, the cost of service in fog is ignored. First, the convergence of ODMA, CSA, WOA, COA and PSO algorithms in reducing the fitness function is compared. Convergence shows the speed of reaching the final answer and causes evolution to stop. Because in this case, all solutions in the population are almost the same and the evolution continuation due to the reduction of diversity will not have a significant effect on improving the fitness function. A comparison of the convergence speed of the different algorithms is shown in [Figure 6](#). This comparison is based on 100 iterations, so that in each iteration for each algorithm, the value calculated for the fitness function is reported. As depicted, the efficiency and convergence of the PSO to find service deployment planning in the fog colony is better than other algorithms. The PSO managed to reduce the fitness function to about 6830 with only 40 iterations. However, the COA converged at 35 iterations, but the efficiency of this algorithm is 0.5% lower than that of PSO. After PSO and COA, it can be seen that CSA, WOA and ODMA algorithms have provided better results, respectively.

Although the solution provided by PSO has less convergence and fitness function, due to the dynamic environment and the need for real-time decision making, the runtime metric should also be considered. In general, low runtime for planning can reduce service waiting time for deployment on fog nodes. Hence, [Figure 7](#) reports the results of the runtime of different algorithms in 100 independent iterations. The results clearly show the inefficiency of COA in solving real-time problems and dynamic environments. This algorithm has the worst performance among other algorithms with 79 s of runtime. The reason

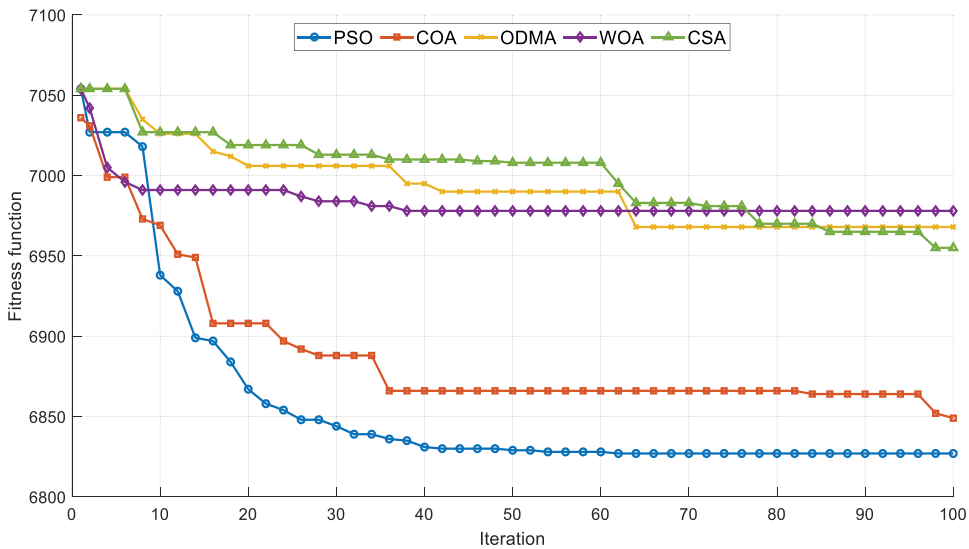


Figure 6. Comparison results of different algorithms based on convergence speed for one period.

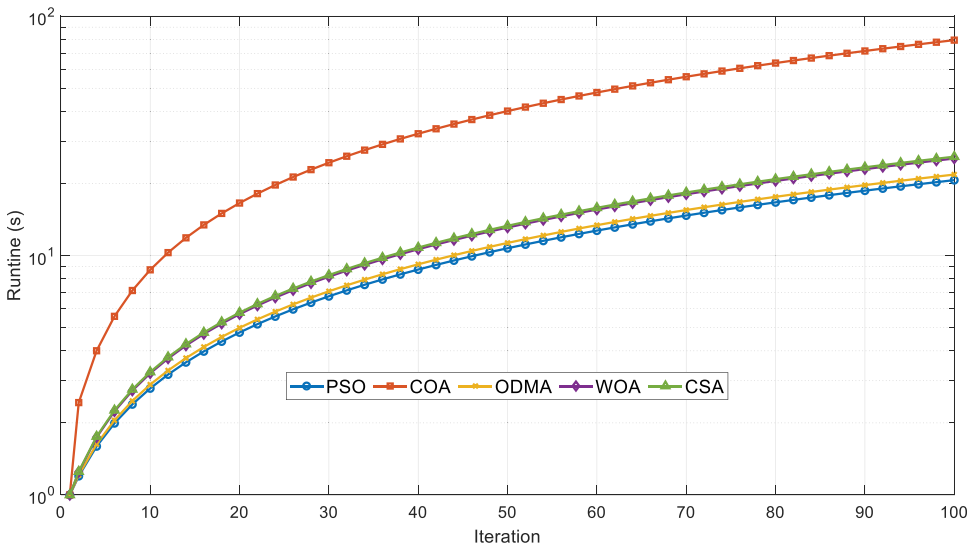


Figure 7. Comparison results of different algorithms based on runtime for one period.

for this can be the characteristic of COA in successive spawning and community building. Other algorithms are almost identical since runtime, however, the ODMA and PSO results are slightly better.

In this paper, the problem of IoT services placement is formulated as a multi-objective optimization problem to maximize utilization of fog and throughput, and minimize response time and service cost in fog computing, as shown in Eq. (2). [Figure 8](#) shows the value obtained from the fitness function for different algorithms. According to the results of

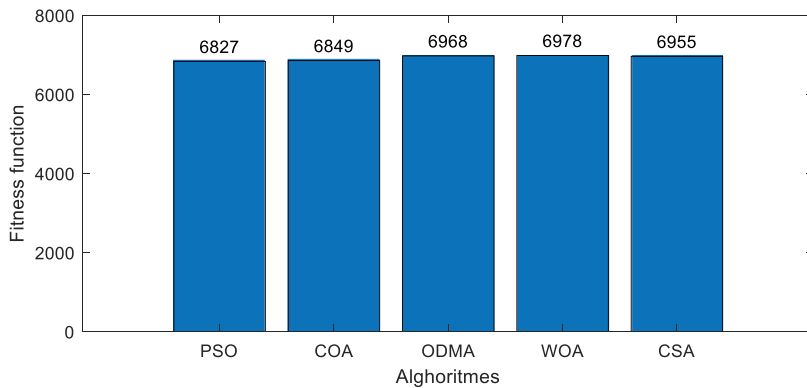


Figure 8. Comparison results of different algorithms based on the fitness function for one period.

this figure, the PSO algorithm has better performance and then the COA algorithm is in the next rank of efficiency. Therefore, deployment planning by PSO can execute the service at the lowest cost on the fog node, which in addition to shortening the response time, also frees up the fog node resources sooner.

Comparison Based on 10 Time Period

In the second scenario, the results of the algorithms are compared for 10 consecutive time periods and 1000 services. Here, different metrics are used to compare the algorithms. First, the algorithms are compared based on the number of services performed to the total number of monitored services. The results of this comparison are reported in [Figure 9](#) for 10 consecutive

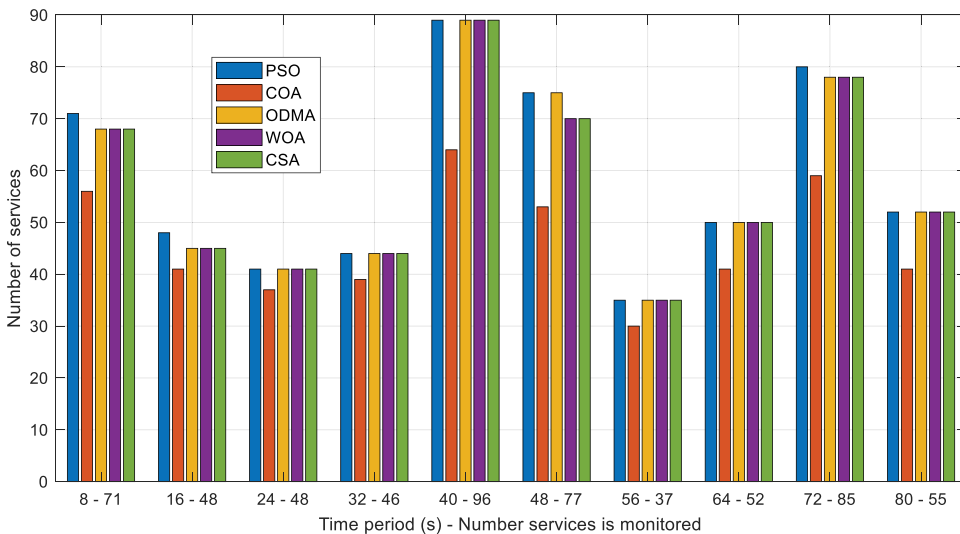


Figure 9. Comparison results of different algorithms based on the number of services performed before the deadline for 10 periods.

time periods. The purpose of the proposed framework is to execute requests transferred to the fog layer before the deadline. As illustrated, PSO has managed to deploy more services than other algorithms before the deadline in most time periods. At the end of period 10, the total number of services deployed before the deadline for PSO is 585. After PSO, ODMA performed better with 577 services. In addition, the number of services performed before the deadline for the COA, WOA and CSA is 461, 572 and 572, respectively.

Figure 10 shows the results of comparing different algorithms based on runtime in 10 time periods. The purpose of optimization is to find a suitable deployment plan with the least runtime. Runtime is a function of the MAPE-k model in the decision-making phase. As illustrated, the PSO has the shortest runtime in terms of service deployment planning. This superiority is clearly evident in all time periods for the PSO. In general, after the end of 10 periods, the total runtime of PSO is 7.51 s, which is better than COA, ODMA, WOA and CSA with 34.11 s, 8.17 s, 8.43 s and 8.91 s, respectively. Compared to other algorithms, COA has a very high runtime, which is due to the spawning policy of cuckoos and therefore more run of the fitness function.

The average waiting time for services performed on the fog colony for 10 periods is shown in Figure 11. Waiting time refers to the sum of processing latencies (queue latency) and deployment planning time. Given that the PSO had less runtime in the decision-making phase, it can be predicted that this algorithm provides less waiting time for services. As illustrated, PSO has the minimum waiting time for services at all time periods. At the end of 10 periods, the average waiting time of services performed for PSO is 5.37 s, which is the

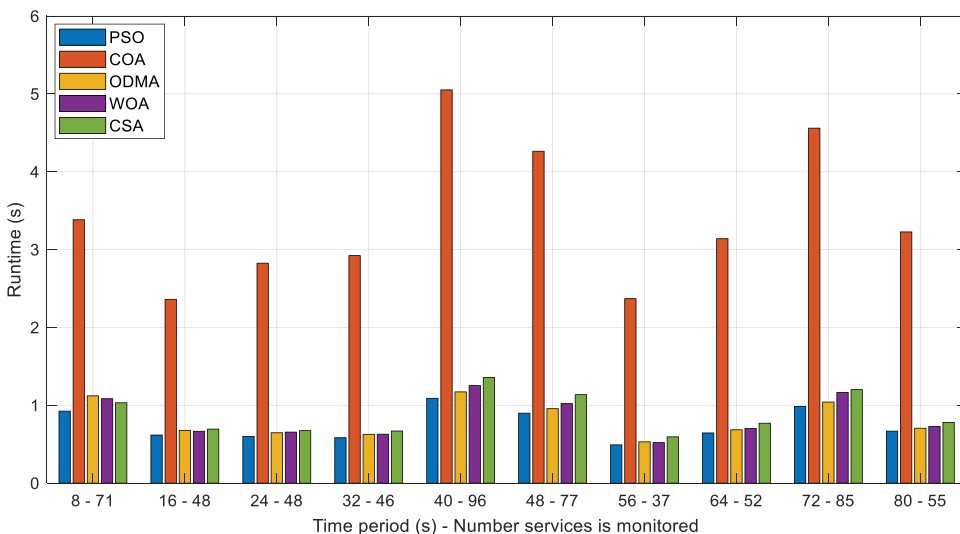


Figure 10. Comparison results of different algorithms based on the runtime for 10 periods.

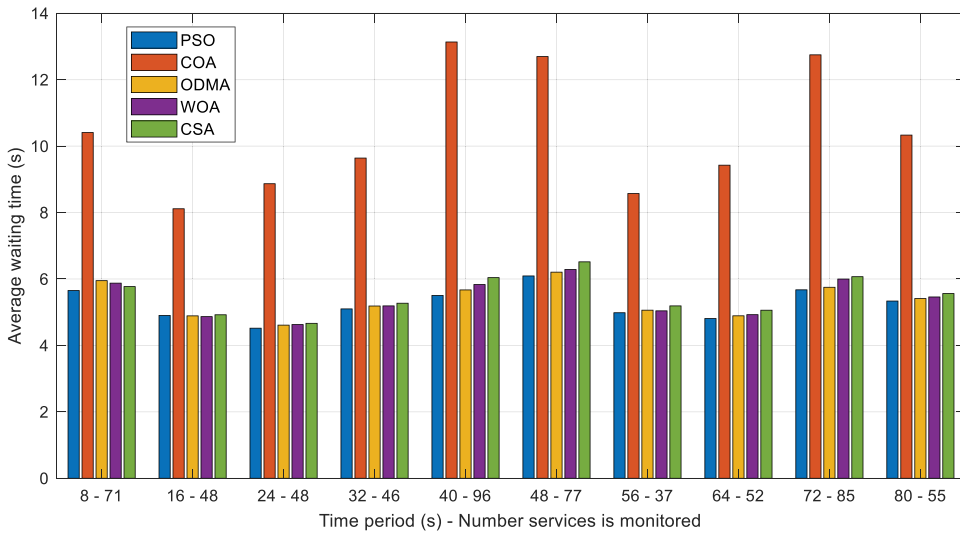


Figure 11. Comparison results of different algorithms based on the average waiting time before the deadline for 10 periods.

lowest compared to other algorithms. After PSO, ODMA has a better average waiting time of 5.48 s. Meanwhile, the results of this metric for COA, WOA and CSA are 10.74, 5.54 and 5.64, respectively.

The number of failed services relative to the total monitored services is shown in Figure 12 for the different algorithms. These results are for 10 independent time periods and 1000 requests. The failed services refer to services that could not be performed before the deadline in deployment

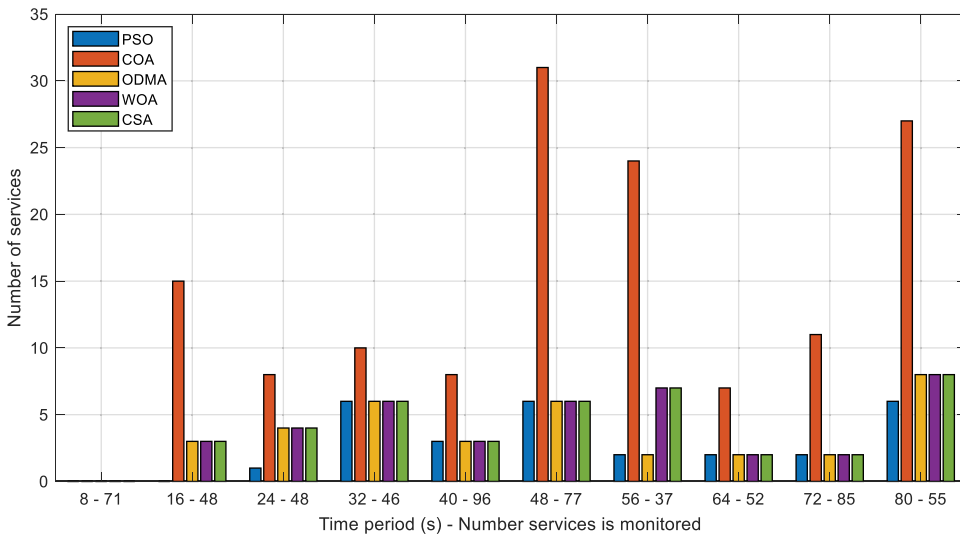


Figure 12. Comparison results of different algorithms based on the number of failed services for 10 periods.

planning. Due to the empty queue at the beginning of the planning stage, there is no failed service for all algorithms in the first time period. However, in subsequent time periods there have been failed services due to the increase in waiting time. Based on the results of different algorithms, PSO has provided fewer failed services in all time periods. Meanwhile, PSO is the only algorithm that in the second time period has managed to plan all monitored services before the deadline. In general, after the end of 10 periods, the total number of failed services for PSO is 28, which is superior to COA, ODMA, WOA and CSA with 141, 36, 41 and 41 failed services, respectively.

Figure 13 shows the results of the fitness function for different algorithms in 10 time periods. According to the definition, the purpose is to minimize the fitness function by algorithms. This metric should be considered based on the number of services performed relative to the total monitored services. The results of this comparison for all time periods show that PSO performs better than other algorithms. At the end of 10 periods, the average fitness function for PSO, CSA, ODMA, WOA, and COA is 327.4, 248.5, 322.7, 319.4, and 319.4, respectively, as shown in Figure 14. Accordingly, the PSO provides the lowest value of the fitness function compared to other algorithms.

In another experiment, the number of remaining services passed to the 11th time period was compared for different algorithms. This metric indicates that the free resources of the fog nodes are not sufficient to execute some services and must be placed for the next time period. Table 8 shows the results of this comparison for different algorithms, where some services must be placed in the 11th time period. According to this table, the first column refers to algorithms name, the second column shows the total number of services

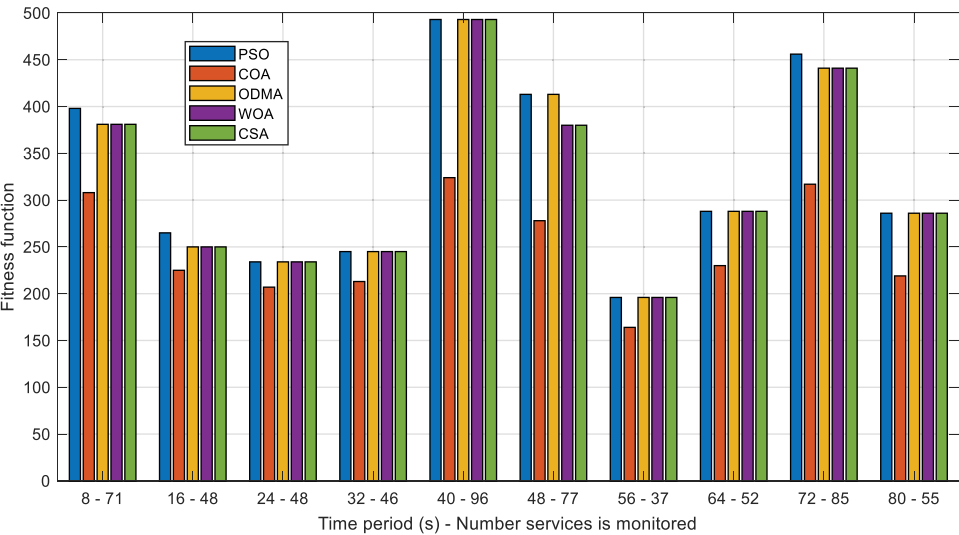


Figure 13. Comparison results of different algorithms based on the fitness function for 10 periods.

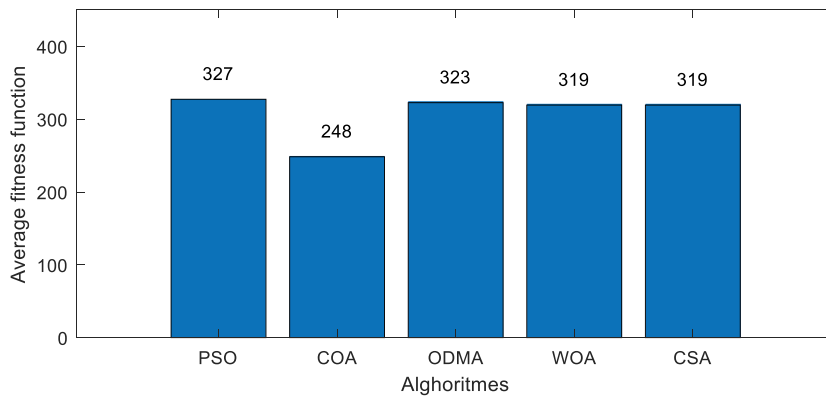


Figure 14. Comparison results of different algorithms based on the average fitness function after the end of 10 periods.

Table 8. Comparison results of different algorithms based on the number of remaining services for planning in the 11th period.

Algorithms	Total services monitored in fog	Services performed up to 10 th period	Number of services failed	Remaining services for the 11 th period
PSO	615	585	28	2
COA	615	461	141	13
ODMA	615	577	36	2
WOA	615	572	41	2
CSA	615	572	41	2

monitored in fog, the third column is related to the number of services performed after the end of 10 time periods, the fourth column refers to the number of failed services, and finally the last column shows how many services there are to be placed in time period 11. The results of this comparison show that PSO, ODMA, WOA and CSA have the best performance with only 2 remaining services. However, PSO is more successful than these algorithms because it has only 28 failed services.

Table 9 summarizes the results of the PSO, COA, ODMA, WOA and CSA algorithms. These results are presented for different metrics based on 1000 requests and 10 time periods. These metrics include number of services performed, runtime, average waiting time, number of failed services, cost services (fitness function), and number of services remaining.

Table 9. Summary of results of different algorithms.

Evaluation metric	PSO	COA	ODMA	WOA	CSA
Number of monitored services	615	615	615	615	615
Average runtime (s)	7.50	34.11	8.16	8.42	8.91
Number of services performed (%)	95.12	74.96	93.82	93.01	93.01
Average waiting time (s)	5.37	10.74	5.48	5.54	5.64
Failed services (%)	4.55	22.93	4.23	6.67	6.67
Remaining services (%)	0.33	2.11	0.33	0.33	0.33
Cost services	327	248	322	319	319

The results clearly show the superiority of PSO over other algorithms. After PSO, the results show that ODMA, WOA, CSA and COA algorithms are in the next ranks, respectively. Algorithms such as PSO and ODMA rarely violate application deadlines, but ODMA uses more cloud resources than PSO. The use of cloud resources due to the physical distance of the cloud data center leads to higher service costs, as indicated in the ODMA results. On average, PSO results are 242.8%, 1.1%, 11.2% and 12.6% superior relative to COA, ODMA, WOA and CSA, respectively. In general, the following results can be obtained from comparing PSO with other algorithms. PSO solutions are significantly more diverse than solutions of other algorithms due to their better maintenance strategy. PSO has better distribution capability which helps to converge and reduce runtime. Also, PSO seems to perform better in high-dimensional search spaces than other algorithms, especially COA.

Conclusion

Cloud computing is not able to meet all the needs of IoT applications due to the distance from the network edge. The aim of fog computing is to utilization of storage and computing resources close to the network edge to execute IoT applications. Fog computing are still in their infancy, and there is a lack of theoretical basis for IoT services placement and provisioning fog resources. In this paper, an efficient conceptual computing framework based on fog-cloud control middleware is introduced to IoT services placement in the fog layer. The problem of IoT services placement in fog was established as an optimization problem that aims to reduce latency and costs, and maximize the utilization of fog resources. The framework uses the MADE-k model (monitoring, analysis, decision-making and execution with a shared knowledge-base) to solve the problem. This model is configured in fog-cloud control middleware and manages all fog colonies, where each colony contains a number of fog nodes and one fog orchestration control node. In the monitoring phase, all IoT services and fog nodes are evaluated and their details are stored in the knowledge-base. IoT services are prioritized in the analysis phase. The placement process is performed autonomously by applying the PSO in the decision-making phase, and then the placement plan is applied to the landscape in the execution phase. The proposed framework improves system reliability by establishing fog-cloud collaboration and provides an effective placement of IoT services. The placement plan of IoT services provided by PSO based on the utilization of fog resources is more efficient and has less latency and cost compared to the plans created by ODMA, CSA, WOA and COA algorithms. Finding multi-objective policies based on the Pareto Archive is suggested as future work. Also, analyzing parallel heuristic algorithms to find more accurate placements than evolutionary approaches is another aspect of future work.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Mostafa Ghobaei-Arani  <http://orcid.org/0000-0003-2639-0900>

Ali Shahidinejad  <http://orcid.org/0000-0003-4856-9119>

References

- Askarzadeh, A. 2016. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures* 169:1–12. doi:10.1016/j.compstruc.2016.03.001.
- Brogi, A., S. Forti, and A. Ibrahim (2018, March). Optimising QoS-assurance, resource usage and cost of fog application deployments. In *International Conference on Cloud Computing and Services Science* (pp. 168–89). Springer, Cham.
- Chen, Y., Z. Li, B. Yang, K. Nai, and K. Li. 2020. A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing. *Future Generation Computer Systems* 108:273–87. doi:10.1016/j.future.2020.02.045.
- Dastjerdi, A. V., and R. Buyya. 2016. Fog computing: Helping the internet of things realize its potential. *Computer* 49 (8):112–16. doi:10.1109/MC.2016.245.
- Faraji Mehmandar, M., S. Jabbehdari, and J. H. Haj Seyyed. 2020. A dynamic fog service provisioning approach for IoT applications. *International Journal of Communication Systems* 33 (14):e4541. doi:10.1002/dac.4541.
- Forouzandeh, S., M. Rostami, and K. Berahmand. 2021. Presentation a trust walker for rating prediction in recommender system with biased random walk: Effects of h-index centrality, similarity in items and friends. *Engineering Applications of Artificial Intelligence* 104:104325. doi:10.1016/j.engappai.2021.104325.
- Goswami, P., A. Mukherjee, M. Maiti, S. K. S. Tyagi, and L. Yang. 2021. A neural network based optimal resource allocation method for secure IIoT network. *IEEE Internet of Things Journal* 1–1. doi:10.1109/JIOT.2021.3084636.
- Hajipour, H., H. B. Khormuji, and H. Rostami. 2016. ODMA: A novel swarm-evolutionary metaheuristic optimizer inspired by open-source development model and communities. *Soft Computing* 20 (2):727–47. doi:10.1007/s00500-014-1536-x.
- Hassan, H. O., S. Azizi, and M. Shojafar. 2020. Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments. *IET Communications* 14 (13):2117–29. doi:10.1049/iet-com.2020.0007.
- Ibrahim, A., M. Noshay, H. A. Ali, and M. Badawy. 2020. PAPSO: A power-aware VM placement technique based on particle swarm optimization. *IEEE Access* 8:81747–64. doi:10.1109/ACCESS.2020.2990828.
- Jacob, B., R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin. 2004. A practical guide to the IBM autonomic computing toolkit. *IBM Redbooks* 4 (10):1–268.
- Jia, B., H. Hu, Y. Zeng, T. Xu, and Y. Yang. 2018. Double-matching resource allocation strategy in fog computing networks based on cost efficiency. *Journal of Communications and Networks* 20 (3):237–46. doi:10.1109/JCN.2018.000036.
- Karatas, F., and I. Korpeoglu. 2019. Fog-based data distribution service (F-DAD) for Internet of Things (IoT) applications. *Future Generation Computer Systems* 93:156–69. doi:10.1016/j.future.2018.10.039.

- Kennedy, J., and R. Eberhart (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Perth, WA, Australia, (Vol.4, pp. 1942–48). IEEE.
- Khosroabadi, F., F. Fotouhi-Ghazvini, and H. Fotouhi. 2021. SCATTER: service placement in real-time fog-assisted IoT networks. *Journal of Sensor and Actuator Networks* 10 (2):26. doi:10.3390/jsan10020026.
- Kim, W.-S., and S.-H. Chung. 2018. User-participatory fog computing architecture and its management schemes for improving feasibility. *IEEE Access* 6:20262–78. doi:10.1109/ACCESS.2018.2815629.
- Lee, G., W. Saad, and M. Bennis. 2019. An online optimization framework for distributed fog network formation with minimal latency. *IEEE Transactions on Wireless Communications* 18 (4):2244–58. doi:10.1109/TWC.2019.2901850.
- Mahmoud, M. M., J. J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev. 2018. Towards energy-aware fog-enabled cloud of things for healthcare. *Computers & Electrical Engineering* 67:58–69. doi:10.1016/j.compeleceng.2018.02.047.
- Maier, H. R., S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk, and B. A. Tolson. 2019. Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental Modelling & Software* 114:195–213. doi:10.1016/j.envsoft.2018.11.018.
- Minh, Q. T., D. T. Nguyen, A. Van Le, H. D. Nguyen, and A. Truong (2017, November). Toward service placement on fog computing landscape. In *2017 4th NAFOSTED conference on information and computer science*, Hanoi, Vietnam, (pp. 291–96). IEEE.
- Mirjalili, S., and A. Lewis. 2016. The whale optimization algorithm. *Advances in Engineering Software* 95:51–67. doi:10.1016/j.advengsoft.2016.01.008.
- Mohan, A., K. Gauen, Y. H. Lu, W. W. Li, and X. Chen (2017, May). Internet of video things in 2030: A world with many cameras. In *2017 IEEE international symposium on circuits and systems (ISCAS)*, Baltimore, MD, USA, (pp. 1–4). IEEE.
- Murtaza, F., A. Akhunzada, S. Ul Islam, J. Boudjadar, and R. Buyya. 2020. QoS-aware service provisioning in fog computing. *Journal of Network and Computer Applications* 165:102674. doi:10.1016/j.jnca.2020.102674.
- Natesha, B. V., and R. M. R. Guddeti. 2021. Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment. *Journal of Network and Computer Applications* 178:102972. doi:10.1016/j.jnca.2020.102972.
- Neware, R., and U. Shrawankar. 2020. Fog computing architecture, applications and security issues. *International Journal of Fog Computing (IJFC)* 3 (1):75–105. doi:10.4018/IJFC.2020010105.
- Puliafito, C., E. Mingozzi, F. Longo, A. Puliafito, and O. Rana. 2019. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)* 19 (2):1–41. doi:10.1145/3301443.
- Ramírez, W., X. Masip-Bruin, E. Marin-Tordera, V. B. C. Souza, A. Jukan, G. J. Ren, and O. G. de Dios. 2017. Evaluating the benefits of combined and continuous Fog-to-Cloud architectures. *Computer Communications* 113:43–52. doi:10.1016/j.comcom.2017.09.011.
- Ren, J., G. Yu, Y. He, and G. Y. Li. 2019. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology* 68 (5):5031–44. doi:10.1109/TVT.2019.2904244.
- Rezaeipanah, A., H. Nazari, and G. Ahmadi. 2019. A Hybrid Approach for Prolonging Lifetime of Wireless Sensor Networks Using Genetic Algorithm and Online Clustering. *Journal of Computing Science and Engineering* 13 (4):163–74. doi:10.5626/JCSE.2019.13.4.163.

- Rezaeipanah, A., M. Mojarad, and A. Fakhari. 2020. Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. *International Journal of Computers and Applications* 1–9. doi:[10.1080/1206212X.2019.1709288](https://doi.org/10.1080/1206212X.2019.1709288).
- Saeedi, S., R. Khorsand, S. G. Bidgoli, and M. Ramezanpour. 2020. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. *Computers & Industrial Engineering* 147:106649. doi:[10.1016/j.cie.2020.106649](https://doi.org/10.1016/j.cie.2020.106649).
- Salimian, M., M. Ghobaei-Arani, and A. Shahidinejad. 2021. Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment. *Software: Practice and Experience* 51 (8):1745–1772.
- Santoyo-González, A., and C. Cervelló-Pastor. 2018. Latency-aware cost optimization of the service infrastructure placement in 5G networks. *Journal of Network and Computer Applications* 114:29–37. doi:[10.1016/j.jnca.2018.04.007](https://doi.org/10.1016/j.jnca.2018.04.007).
- Skarlat, O., M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. 2017. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications* 11 (4):427–43. doi:[10.1007/s11761-017-0219-8](https://doi.org/10.1007/s11761-017-0219-8).
- Souza, V. B., X. Masip-Bruin, E. Marín-Tordera, S. Sánchez-López, J. Garcia, G. J. Ren, A. J. Ferrer, and A. Juan Ferrer. 2018. Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures. *Future Generation Computer Systems* 87:1–15. doi:[10.1016/j.future.2018.04.042](https://doi.org/10.1016/j.future.2018.04.042).
- Talatian Azad, S., G. Ahmadi, and A. Rezaeipanah. 2021. An intelligent ensemble classification method based on multi-layer perceptron neural network and evolutionary algorithms for breast cancer diagnosis. *Journal of Experimental & Theoretical Artificial Intelligence* 1–21. doi:[10.1080/0952813X.2021.1938698](https://doi.org/10.1080/0952813X.2021.1938698).
- Taneja, M., and A. Davy (2016, October). Resource aware placement of data stream analytics operators on fog infrastructure for internet of things applications. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, Washington, DC, USA, (pp. 113–14). IEEE.
- Xavier, T. C. S., I. L. Santos, F. C. Delicato, P. F. Pires, M. P. Alves, T. S. Calmon, A. C. Oliveira, and C. L. Amorim. 2020. Collaborative resource allocation for Cloud of Things systems. *Journal of Network and Computer Applications* 159:102592. doi:[10.1016/j.jnca.2020.102592](https://doi.org/10.1016/j.jnca.2020.102592).
- Yang, X. S., and S. Deb (2009, December). Cuckoo search via Lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, Coimbatore, India, (pp. 210–14). Ieee.
- Yang, Y., S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang. 2018. DEBTS: Delay energy balanced task scheduling in homogeneous fog networks. *IEEE Internet of Things Journal* 5 (3):2094–106. doi:[10.1109/JIOT.2018.2823000](https://doi.org/10.1109/JIOT.2018.2823000).
- Yousefpour, A., A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, J. P. Jue, W. Xie, and J. P. Jue. 2019. FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework. *IEEE Internet of Things Journal* 6 (3):5080–96. doi:[10.1109/JIOT.2019.2896311](https://doi.org/10.1109/JIOT.2019.2896311).
- Yousefpour, A., G. Ishigaki, R. Gour, and J. P. Jue. 2018. On reducing IoT service delay via fog offloading. *IEEE Internet of Things Journal* 5 (2):998–1010. doi:[10.1109/JIOT.2017.2788802](https://doi.org/10.1109/JIOT.2017.2788802).
- Zhang, B., X. Wang, and M. Huang. 2018. Multi-objective optimization controller placement problem in internet-oriented software defined network. *Computer Communications* 123:24–35. doi:[10.1016/j.comcom.2018.04.008](https://doi.org/10.1016/j.comcom.2018.04.008).