Journal of King Saud University – Computer and Information Sciences 35 (2023) 157–175

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

An efficient and autonomous scheme for solving IoT service placement problem using the improved Archimedes optimization algorithm

Zhijun Zhang [a], Hui Sun [a,*], Hajar Abutuqayqah [b]

[a] Weifang University of Science and Technology, Shouguang, Shandong 262700, China
[b] Mathematics Department, University College of Duba, University of Tabuk, 49311, Saudi Arabia

ARTICLE INFO

Article history:
Received 16 December 2022
Revised 19 January 2023
Accepted 15 February 2023
Available online 21 February 2023

Keywords:
IoT
SPP
AOA
Deployment Distribution
Shared Parallel Architecture

ABSTRACT

The ever-increasing growth of the number of Internet of Things (IoT) devices connected to the network has led to the emergence of cloud computing shortcomings such as delay, storage and bandwidth. Fog computing has been developed as an emerging computing paradigm to overcome the challenges of cloud computing. This paradigm can support delay-critical and computationally intensive applications by providing resources at the network edge. As fog nodes appear with limited resources, IoT service placement schemes can improve the Quality of Service (QoS) and system performance. In general, the mapping between fog nodes and IoT services is known as the Service Placement Problem (SPP) in fog computing. To date, various *meta*-heuristic approaches have been introduced to solve SPP, but few are known in the research society due to computational complexity. Hence, this study proposes an efficient and autonomous scheme to solve SPP using *meta*-heuristic approaches with shared parallel architecture that can overcome the problem complexity. Specifically, we use the Archimedes Optimization Algorithm (AOA) as a new *meta*-heuristic approach inspired by the physics law of Archimedes' Principle. The proposed scheme, as SPP-AOA, formulates SPP as a multi-objective problem and performs the placement of autonomous services on distributed fog domains. Our main concerns in SPP are related to resource utilization, service cost, energy consumption, delay cost and throughput. SPP-AOA performs placement based on extracting resource distribution over time, which can save more resources to handle future requests. The effectiveness of the proposed scheme has been proven through evaluation on Barabasi-Albert network topology. Compared to the state-of-the-art methods, SPP-AOA deploys an average of 5% more service in fog, lowers costs by 2% and waiting time by 20%, and reduces placement on the cloud by 14%.

1. Introduction

Various computing resources such as grid computing, cloud computing, and fog computing have led to the emergence of Internet of Things (IoT) devices (Shakarami et al., 2022). The concept of IoT was introduced as an Internet-connected smart device in the early 2000 s, and billions of it is now used to collect and exchange data worldwide (Xavier et al., 2020). The increasing development of the Internet and computer computing technology has made online services very popular. End users can use online services in computing resources to process data produced by IoT devices. This reduces processing, cost, memory and energy consumption in the end-user devices. In this regard, the cloud computing paradigm can meet the demands of IoT devices without worrying about processing, memory and storage (Khosroabadi et al., 2021).

With the expansion of the number of IoT devices to 46 billion in 2021 and its increase to 75 billion in 2025, the speed and quality of data transmission has become even more important (Shahidinejad et al., 2021). Today, IoT applications based on real-time data processing have become very common (Li et al., 2020; Si et al., 2021). Cloud computing with a centralized architecture for processing these applications faces the problem of high delay because cloud servers are too far from the data source (Shakarami et al., 2022). In addition to delay, cloud computing faces challenges such as network congestion, bandwidth, and response time due to the enormous amount of data produced by IoT devices (Cao et al., 2023). Therefore, cloud computing alone will not be able to

* Corresponding author.
E-mail addresses: zzj@wfust.edu.cn (Z. Zhang), huisun221@126.com (H. Sun), haajjr01@gmail.com (H. Abutuqayqah).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.02.015
1319-1578/

support the future needs of IoT devices. After proving the inefficiency of the current cloud infrastructure, a new paradigm called fog computing evolved (Ibrahim et al., 2020). In fact, fog computing moves the network core from cloud computing to the edge of the network, where it can overcome the challenges of advancing cloud computing such as delay.

In general, IoT has led to the presence of various network computing equipment in large public and private spaces (Dokeroglu et al., 2019). This equipment is part of the computing and storage resources of the network that define the theoretical foundations of fog computing. This equipment, known as heterogeneous and decentralized devices, can work together to provide services to IoT devices. The idea of fog computing is the efficient use of these devices, which can bring the provision of services closer to the data source (Cheng et al., 2022; Tan et al., 2022). Therefore, fog computing provides better Quality of Service (QoS) solutions than cloud computing. This is especially important for real-time IoT applications (Farahbakhsh et al., 2021; Zhang et al., 2022a). In addition, fog computing improves network congestion and bandwidth consumption by significantly reducing requests to cloud data centers (Skarlat et al., 2017).

The location of fog computing is somewhere between the cloud and IoT devices, which acts as a bridge of communication (Li et al., 2022; Trik et al., 2023). In cloud-fog-IoT architecture, moving towards the cloud will provide high storage access and processing power. However, issues such as high delay and network congestion will also appear. Nevertheless, moving towards fog has apt response time, low delay and real-time data processing capability, but computing and storage resources are limited. The concept of fog computing has been proposed as a local and decentralized infrastructure for managing IoT requests (Skarlat et al., 2017). The main components of fog computing are IoT devices (end users) and fog nodes (heterogeneous and decentralized devices). IoT devices can be any smart device connected to the Internet (such as smartphones, smart door locks, and smart fire alarms), and fog nodes are devices with limited resources such as access points, routers, and set-top boxes. Meanwhile, due to the geographical distribution of fog nodes, fog computing management is done as non-overlapping domains.

Requests are sent from IoT devices to the fog landscape where each request is treated as an IoT application. Each IoT application consists of one or more independent IoT services, where each IoT service requires resources that can be provided by fog nodes (Liu et al., 2022). Since fog nodes have resource limitations, it is challenging to effectively allocate IoT services to fog nodes. This problem is expressed in fog computing as Service Placement Problem (SPP) (Ayoubi et al., 2021). It is proved that the complexity of solving SPP for QoS improvement due to the geographical distribution of fog nodes is an NP-hard problem (Liu et al., 2022). Resolving SPP is very important because of the decentralized fog resources and the dynamics of incoming IoT applications. In addition, there are many objectives for improving SPP, including resource utilization, delay, cost, throughput, response time, reliability, availability, scalability, and energy consumption (Skarlat et al., 2017). When there are several conflicting objectives to improve SPP, it is necessary to determine a compromise between objectives that complicates the problem.

SPP aims to deploy each IoT service on a fog node with limited resources, where QoS requirements are satisfied. The International Business Machines (IBM) corporation introduced the MADE-k (Monitoring, Analysis, Decision-making, and Execution with knowledge base) model to perform autonomous computing, which can make decisions about deploying IoT services on fog nodes (Liu et al., 2022). A wide range of optimization-based metaheuristic approaches have been presented in previous studies to solve SPP through MADE-k (Ayoubi et al., 2021). In this regard, this paper proposes an algorithm based on metaheuristic approaches to efficiently solve SPP through MADE-k.

As emphasized in the "No Free Lunch Theorems for Search" theory, there is no *meta*-heuristic algorithm that can dominate all optimization problems (Joyce and Herrmann, 2018). Hence, the use of *meta*-heuristic algorithms in solving different problems is an open problem. In recent years, extensive studies have been conducted to solve SPP more effectively with *meta*-heuristic approaches. For example, Zhao et al. (2022) used the Open-source Development Model Algorithm (ODMA), Liu et al. (2022) used the Cuckoo Search Algorithm (CSA), and Ghobaei-Arani and Shahidinejad (2022) used the Whale Optimization Algorithm (WOA) to solve SPP. Here, we use Archimedes Optimization Algorithm (AOA) as a new *meta*-heuristic approach to solve SPP. AOA is very popular in solving optimization problems and its superiority has been proven in many applications (Hashim et al., 2021). We use the abbreviation SPP-AOA for the proposed scheme. SPP-AOA performs the placement process based on the MADE-k autonomous model and considering the resource usage distribution of fog nodes. Meanwhile, SPP-AOA formulates SPP as a multi-objective problem by compromising between different objectives. In addition to the above, the performance of AOA is improved by configuring the evolution process through a shared parallel architecture.

The main contributions to this work can be summarized as follows:

- We develop a conceptual computational framework based on the MADE-k autonomous model.
- We propose a parallel version of AOA with a shared memory as a *meta*-heuristic approach to solve SPP.
- The distribution of the use of fog nodes is considered to improve resource management in future requests.
- The performance of the proposed placement scheme has been evaluated through extensive experiments in a synthetic fog environment.

The rest of the paper is organized as follows: Section 2 reviews previous work related to SPP; Section 3 deals with the background of the system model, objective function and heuristic approaches; Section 4 describes the conceptual framework for cloud-fog-IoT ecosystem; Section 5 describes the details related to the proposed SPP-AOA scheme; Section 6 explains the details of the experimental study and analysis of the results; and finally, Section 7 presents the conclusions and future works.

2. Related work

So far, extensive studies on SPP in fog computing have been presented and the importance of this issue has been understood by the research society (Liu et al., 2015; Cao et al., 2022; Azad et al., 2022; Tang et al., 2022). Unlike the cloud, research in fog is still immature and there are still challenges and shortcomings. Many works such as Zhang et al. (2019) have been done in the field of service deployment planning with different objectives. However, in most of these studies, the association quantity between the objectives is not considered. In general, compromise between objectives and different limitations in SPP can lead to improved overall system performance. In general, the presented methods for solving SPP can be examined from different aspects. For example, offline or online placement, centralized or distributed placement, static or dynamic placement, single-objective or multi-objective placement, and etc. (Berahmand et al., 2021; Ali et al., 2022; Nasiri et al., 2022). Accordingly, Fig. 1 shows a taxonomy of different aspects for solving SPP.

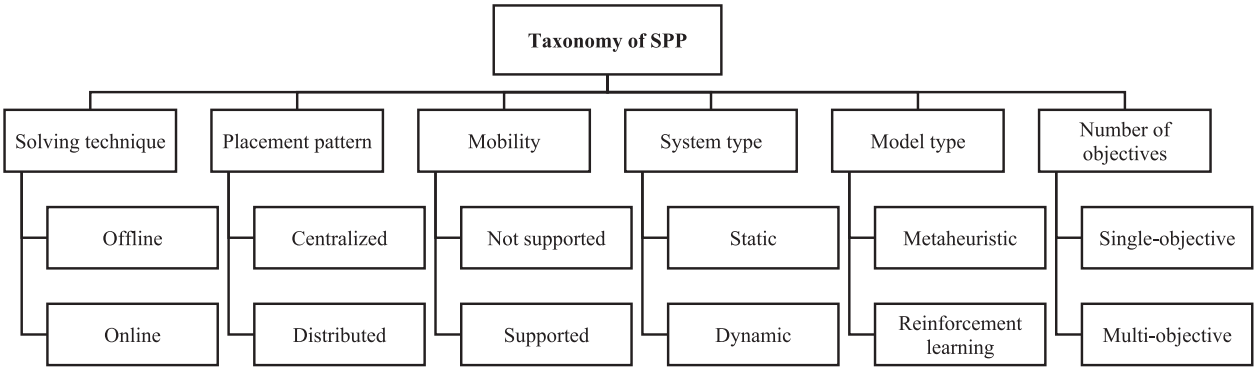

Fig. 1. Taxonomy of available methods to solve SPP.

As illustrated, there are many factors involved in analyzing SPP algorithms in fog computing. In this study, we compare SPP-related algorithms with different strategies based on single-objective and multi-objective. The purpose of this study is to highlight the importance of multi-objective algorithms compared to single-objective algorithms. In general, service providers are interested in solving SPP by considering several objectives to increase their profits and improve QoS (Ghobaei-Arani and Shahidinejad, 2022). However, many studies have attempted to solve SPP with single-objective algorithms (Skarlat et al., 2017; Liu et al., 2022; Zhang et al., 2022b). In most studies, researchers seek to reduce costs and delays in resolving SPP; however, objectives such as resource utilization, service cost, energy consumption, delay cost, throughput, response time, deadline, two-level fog, reduced data transfer to the cloud, service-level agreement violations, resource efficiency, reliability, availability and scalability have also been considered in some studies (Zhang et al., 2022b). Given the breadth of studies in this area, we will review only some of them. Table 1 summarizes the studies reviewed.

A Double Matching Strategy (DMS) is proposed by Jia et al. (2018) for resource management in the fog. DMS can provide cost-effective resources using a deferred acceptance algorithm. The lightweight QoS-aware Dynamic Fog Service Provisioning (QDFSP) algorithm was introduced by Yousefpour et al. (2019). QDFSP can solve SPP with low complexity considering delay and cost requirements. Chen et al. (2020) used Stackelberg game-based techniques to solve SPP. The purpose of this method is to balance resource allocation when deploying services.

Xavier et al. (2020) proposed a Collaboration Resource Allocation Algorithm (CRAA) for the Cloud of Things (CoT) architecture, which simultaneously supports heterogeneity of applications and nodes. This approach uses the distributed nature of fog nodes to enhance cooperation in the allocation process. Murtaza et al. (2020) introduced an approach to managing delayed sensitive IoT applications. Here, an intelligent layer is embedded between IoT devices and fog nodes to perform an efficient resource allocation plan through the Learning Repository Fog-Cloud (LRFC). Hassan et al. (2020) solved SPP with the aim of minimizing energy consumption (called MinRE) in cloud-fog based networks. The authors divided the IoT services into critical and normal categories and proposed two different algorithms to solve it. The objective of the critical service-based algorithm is to minimize response time, while the objective for normal services is to reduce energy consumption.

Natesha and Guddeti (2021) used an Elitism-based Genetic Algorithm (EGA) to solve SPP in fog computing. EGA is based on the provision of two-level resources using containers. The authors optimized IoT services to ensure QoS with several objectives such as energy consumption, cost and response time. Baranwal and Vidyarthi (2021) proposed a Fog Orchestrator Node Selection (FONS) approach to solving SPP. The authors emphasize the importance of the orchestrator node in a decentralized form and use

Table 1
A summary of the studies reviewed.

Authors	Model name	Optimization type	Resource utilization	Response time	Cost	Delay	Deadline	Energy consumption	Throughput
Jia et al. (2018)	DMS	Single objective	–	–	√	–	–	–	–
Yousefpour et al. (2019)	QDFSP	Single objective	–	–	√	√	–	–	–
Chen et al. (2020)	–	Single objective	–	√	–	–	–	–	–
Xavier et al. (2020)	CRAA	Single objective	–	–	–	√	–	–	–
Murtaza et al. (2020)	LRFC	Single objective	–	–	–	–	–	√	–
Hassan et al. (2020)	MinRE	Single objective	–	√	–	–	–	√	–
Salimian et al. (2021)	SPP-GWO	Single objective	–	–	√	–	–	–	–
Natesha and Guddeti (2021)	EGA	Multi-objective	√	√	√	–	√	√	–
Baranwal and Vidyarthi (2021)	FONS	Multi-objective	–	–	–	–	√	√	√
Khosroabadi et al. (2021)	SCATTER	Multi-objective	√	√	√	√	–	–	–
Salimian et al. (2022)	SPP-PSO	Multi-objective	√	√	√	–	–	–	√
Liu et al. (2022)	CSA-FSPP	Multi-objective	√	√	√	√	–	√	–
Zhao et al. (2022)	FSP-ODMA	Multi-objective	√	√	√	√	–	√	–
Ghobaei-Arani and Shahidinejad (2022)	WOA-FSP	Multi-objective	–	–	–	–	–	√	√
Azimzadeh et al. (2022)	FSPCN	Multi-objective	√	–	–	√	–	–	–
Proposed scheme	SPP-AOA	Multi-objective	√	–	√	√	–	√	√

FONS to improve the overall performance of the system. FONS is a lightweight model that can solve even SPP through least powerful IoT devices as an orchestrator node. Khosroabadi et al. (2021) introduced a heuristic algorithm called clustering of fog devices and requirement-sensitive service first (SCATTER) to solving SPP. SCATTER evaluation is performed by considering a smart home application and various QoS criteria.

Salimian et al. (2021) proposed the SPP-GWO algorithm for solving SPP, which uses Gray Wolf Optimization (GWO) to perform efficient deployment of services. SPP-GWO offers an autonomous resource management system that is compatible with the dynamic behavior of the fog environment by considering the service cost and blocking of fog resources. Salimian et al. (2022) proposed an extended version of Particle Swarm Optimization (PSO) to solve SPP, which takes into account some limitations and heterogeneity of application and resources in the deployment process. This algorithm, called SPP-PSO, uses the MADE-k model for resource management and prioritizes requests based on deadlines. The purpose of the SPP-PSO is to compromise between costs, delay and resource utilization. Liu et al. (2022) proposed an approach called CSA-FSPP that uses CSA to solve the Fog Service Placement Problem (FSPP). CSA-FSPP deploys services using a centralized cloud-fog control middleware that manages cloud-to-fog communications to reduce data exchange. Here, FSPP is considered as a multi-objective optimization problem based on the Pareto archive.

Zhao et al. (2022) proposed a QoS-aware placement approach to minimize response time, delay, energy consumption, service cost, and maximize the utilization of fog resources. The authors used ODMA as a metaheuristic approach to fog service placement (FSP-ODMA). This approach uses a three-tier conceptual computational framework to describe the interactions between system components. FSP-ODMA reduces the cost of deploying IoT applications by an average of 10 % over similar metaheuristic approaches. Ghobaei-Arani and Shahidinejad (2022) proposed a cost-effective approach using the WOA for fog service placement (WOA-FSP). WOA-FSP is an autonomous placement approach based on the MADE-k model that solves SPP by considering energy consumption and throughput while satisfying QoS requirements. Azimzadeh et al. (2022) introduced an approach to solving the Fog Service Placement algorithm based on Complex Networks feature (FSPCN). The FSPCN seeks to reduce the complexity of the SPP by considering the concept of community. The authors use network structure and nodes and link feature to build communities to improve resource utilization and delay. FSPCN works on the basis of a genetic algorithm (GA), which uses a new neighborhood distance metric to form communities and prioritize them.

As mentioned above, there are many *meta*-heuristic approaches to solve SPP in the past literature. Although these works are similar to the current paper, but there are potential differences. These differences have proven the superiority of our method. The proposed scheme performs the placement process based on the MADE-k autonomous model and considering the resource usage distribution of fog nodes. Distributing resources over time can save more resources to handle future requests. In addition to the above, the performance of AoA is improved by configuring the evolution process through a shared parallel architecture. In general, considering the distribution of resource usage over time as well as AoA configuration by a shared parallel architecture is the main contribution of this paper compared to past literature.

3. Background

In this section, we discuss the system model and objective functions related to SPP. We also briefly review the metaheuristic approaches that are very popular for solving SPPs.

3.1. System model

The basic architecture of fog infrastructure consists of three layers: cloud (at the highest level), fog (middle level) and IoT devices (at the lowest level). A schematic of the basic cloud-fog-IoT ecosystem architecture is shown in Fig. 2. Resource requests are sent from IoT devices (as end users) to the fog layer. Since the fog elements are located at the network edge, the fog layer has the ability to process real-time requests with low delay. Nevertheless, non-real-time requests or non-executable requests in the fog can be

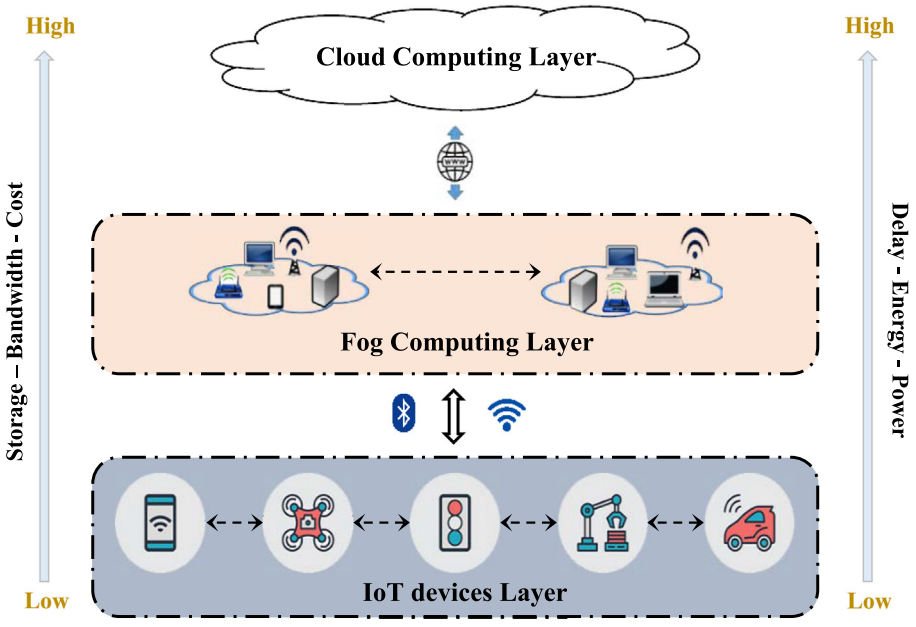

Fig. 2. Basic architecture of cloud-fog-IoT ecosystem.

Table 2
Description of the symbols used.

Symbols	Description	Symbols	Description
t	Current time	RT_{a_k}	Response time of a_k
τ	Time period	R_{a_k}	Time of sending a_k by the user
F	Set of fog colonies	λ	Impact coefficient D_{a_k} relative to R_{a_k}
A	Set of IoT applications	ε	The free resource rate of a fog node
S	Set of IoT services	$P(a_k)$	Priority of a_k to execute
f_i	Fog colony i (set of nodes)	$x_{n^j}^{s_l}$	Binary decision variable for n^j about executing s_l
a_k	Application k	$x_o^{s_l}$	Binary decision variable for FOCN about executing s_l
f	A fog colony	$x_{nn}^{s_l}$	Binary decision variable for NNFC about executing s_l
$\|F\|$	Total number of fog colonies	$x_{mw}^{s_l}$	Binary decision variable for CFCM about executing s_l
$\|A\|$	Number of applications	cp	Computation cost for a service
$\|S\|$	Total number of IoT services	P_{dv}	Computation unit price
o_i	FOCN in f_i	cm	Communication cost for a service
o	FOCN in a fog colony	C_{dv}	Communication unit price
n_i^j	Fog node j in f_i	(t_α, t_β)	Time range of service execution
s_k^l	IoT service l in a_k	$cv_{\alpha,\beta}$	Data size between t_α and t_β
n^j	Fog node j in a fog colony	BW	Output bandwidth
s_l	IoT service l in an application	$E_{PS}^{s_l}$	Energy consumption associated with the deployment of s_l
$\|f_i\|$	Number of fog nodes in f_i	$E_{ES}^{s_l}$	Energy consumption associated with the execution of s_l
$\|a_k\|$	Number of IoT services in a_k	SS_{s_l}	Number of bits in s_l
$\|f\|$	Number of fog nodes in a fog colony	e_C	Energy consumption per CPU cycle
U_{s_l}	CPU required for s_l	$\mathbb{N}_C$	Number of cycles required to process a bit
U_o	CPU usage by FOCN	$e_{s_l}(t)$	Energy consumed to execute s_l at time t
U_{n^j}	CPU usage by n^j	T_{s_l}	Number of tasks executed in s_l
M_{s_l}	RAM required for s_l	EP_{max}	Energy consumption at the peak load
M_o	RAM usage by FOCN	EP_{min}	Minimum energy consumption in active mode
M_{n^j}	RAM usage by n^j	$\mathbb{N}_{s_l}$	Number of cycles required to execute s_l
S_{s_l}	Storage required for s_l	ϑ	Assigned node frequency
S_o	Storage usage by FOCN	Srv_D	Number of services executed before the deadline
S_{n^j}	Storage usage by n^j	Srv_{TD}	All services sent to fog colony
D_{s_l}	Deadline required for s_l	RU	Resource utilization
D_{a_k}	Deadline required for a_k	ξ_{ru}	Weight associated with RU
$d_o^{n^j}$	FOCN delay in communication with n^j	SC	Service cost
d_o^{mw}	FOCN delay in communication with CFCM	ξ_{sc}	Weight associated with SC
d_o^{IoT}	FOCN delay in communication with IoT devices	EC	Energy consumption
d_o^{nn}	FOCN delay in communication with NNFC	ξ_{ec}	Weight associated with EC
$d_{n^j}^{s_l}$	Delay processing of s_l on n^j	DC	Delay cost
$d_o^{s_l}$	Delay processing of s_l on FOCN	ξ_{dc}	Weight associated with DC
$d_{mw}^{s_l}$	Delay processing of s_l on CFCM	TP	Throughput
$d_{nn}^{s_l}$	Delay processing of s_l on NNFC	ξ_{tp}	Weight associated with TP

processed by powerful cloud servers. In addition, the Cloud-Fog Control Middleware (CFCM) centralized between the cloud and fog layers is considered to better manage the cloud-fog-IoT ecosystem. Although fog nodes have less powerful computing than cloud servers, but they are still closer to the end users and can ensure QoS improvements for real-time applications.

The fog computing layer is organized by non-overlapping domains as fog colonies (Liu et al., 2022). Each fog colony contains a number of fog nodes with limited resources that can provide the resources required by IoT services. In addition, each fog colony has a dedicated node as Fog Orchestration Control Node (FOCN), which is in charge of managing and supporting the subordinate fog colony. Suppose $F = \left[f_1, f_2, \cdots, f_i, \cdots f_{|F|}\right]$ is the set of fog colonies, where f_i refers to the i-th fog colony. In this text, $|*|$ represents the number of elements in a set, for example $|F|$ is the total number of colonies. The components of f_i are defined as $f_i = \left[o_i, n_i^1, n_i^2, \cdots, n_i^j, \cdots, n_i^{|f_i|}\right]$, where o_i and n_i^j refer to the FOCN and j-th fog node in f_i, respectively. Let $f = \left[o, n^1, n^2, \cdots, n^j, \cdots, n^{|f|}\right]$ be the details of the current fog colony.

Meanwhile, let $A = \left[a_1, a_2, \cdots, a_k, \cdots a_{|A|}\right]$ be the set of IoT applications received by f, where a_k is the kth IoT application. According to the microservice architecture, a_k contains several independent IoT services, i.e., $a_k = \left[s_k^1, s_k^2, \cdots, s_k^l, \cdots, s_k^{|a_k|}\right]$. Here, s_k^l refers to the l-th IoT service in a_k. Meanwhile, let $S = \left[s_1, s_2, \cdots, s_l, \cdots s_{|S|}\right]$ be the entire set of IoT services. Each $s_l \in S$ has CPU, RAM, storage and deadline requirements for execution, which are expressed by symbols U_{s_l}, M_{s_l}, S_{s_l} and D_{s_l}, respectively. On the other hand, the usage of CPU, RAM and storage by n^j is represented by U_{n^j}, M_{n^j} and S_{n^j} respectively. Similarly, U_o, M_o and S_o are associated with FOCN.

All cloud-fog-IoT ecosystem components have two-way communication links, where each communication link has a delay. Let d_α^β be the delay between α and β devices in the ecosystem. In this regard, $d_o^{n^j}$, d_o^{mw}, d_o^{IoT} and d_o^{nn} are the delays between FOCN and n^j, CFCM, IoT devices, and Nearest Neighboring Fog Colony (NNFC), respectively. In general, the total resources required by services deployed on o or n^j should not exceed its available resources. This constraint for CPU, RAM and storage resources is defined in Eq. (1). Also, each $a_k \in A$ contains a deadline as D_{a_k} for execution that must be satisfied for all $s_k^l \in a_k$. This constraint is defined in Eq. (2).

$$\sum_{s_l \in S} RR_{s_l} \leq \varepsilon RR_{n^j}, \quad \forall n^j \in f, RR = \{U, M, S\} \tag{1}$$

Table 3
Details of abbreviations used.

Acronyms	Definition
IoT	Internet of Things
QoS	Quality of Service
SPP	Service Placement Problem
AOA	Archimedes Optimization Algorithm
SPP-AOA	Service Placement Problem - Archimedes Optimization Algorithm
MADE-k	Monitoring, Analysis, Decision-making, and Execution with knowledge base
ILP	Integer Linear Programming
FOCN	Fog Orchestration Control Node
NNFC	Nearest Neighboring Fog Colony
CFCM	Cloud-Fog Control Middleware

$$RT_{a_k} \leq D_{a_k}, \forall a_k \in A \tag{2}$$

where, ε is the free resource rate of each node and RT_{a_k} is the response time of a_k.

To better cover the text, we have summarized all variables and symbols used in Table 2. Also, the description of all abbreviations used is given in Table 3.

3.2. Objective function

Basically, SPP can be formulated as a multi-objective problem. In this paper, SPP-AOA solves SPP as a multi-objective problem by making compromises between five different objectives (i.e., resource utilization, service cost, energy consumption, delay cost and throughput). Therefore, the proposed objective function is a combination of different objectives related to SPP, which are discussed below.

Resource Utilization (*RU*): This factor refers to the number of locations utilized in fog by IoT services, which should be maximized. Each IoT service can be execute by a fog node or FOCN in the current colony, a fog node in NNFC, or in the cloud. Therefore, the deadline for executing applications in the *RU* calculation must be considered (Zhang et al., 2022c). The resource utilization factor is defined in Eq. (3).

$$RU = \sum_{a_k \in A}\left[P(a_k) * \sum_{s_l \in a_k}\left(\sum_{n^j \in f} x_{n^j}^{s_l}\right) + x_o^{s_l} + x_{nn}^{s_l} + x_{mw}^{s_l}\right] \tag{3}$$

where, $x_{n^j}^{s_l}$, $x_o^{s_l}$, $x_{nn}^{s_l}$ and $x_{mw}^{s_l}$ are binary decision variables for n^j, FOCN, NNFC, and CFCM, respectively. For example, $x_{n^j}^{s_l} = 1$ indicates that the service s_l is placed on n^j, otherwise it is $x_{n^j}^{s_l} = 0$. Also, $P(a_k)$ is the priority of a_k, which is calculated through Eq. (14).

Service Cost (*SC*): This factor refers to the service execution time (i.e., communication cost) as well as the monetary cost of service execution (i.e., computing cost). Meanwhile, communication and computing costs for a service must be calculated based on where it is executed. Let cm_{n^j} and cp_{n^j} be the communication and computing costs of a service to execute on n^j, respectively. Also, cm_o, cm_{nn} and cm_{mw} are the communication cost for executing a service on FOCN, NNFC or CFCM respectively. Similarly, cp_o, cp_{nn} and cp_{mw} related to the communication cost are placed by device type. Ayoubi et al. (2021) calculates *cm* and *cp* based on Eq. (4) and Eq. (5), respectively.

$$cm = C_{dv}.\frac{cv_{\alpha,\beta}}{BW} \tag{4}$$

$$cp = P_{dv}(t_\beta - t_\alpha) \tag{5}$$

where, C_{dv} and P_{dv} refer to the unit costs of communication and computing on the device dv, respectively, t_α and t_β are the time range of service execution, $cv_{\alpha,\beta}$ is the data size between t_α and t_β, and BW is the output bandwidth. Liu et al. (2022) use $BW = 20Mbps$ and $C_{dv} = 0.1$ in their simulations.

According to the definition of *cm* and *cp*, the service cost factor can be calculated by Eq. (6) for all IoT applications.

$$SC = \sum_{a_k \in A}\sum_{s_l \in a_k}\left[x_{n^j}^{s_l}(cp_{n^j} + cm_{n^j}) + x_o^{s_l}(cp_o + cm_o) + x_{nn}^{s_l}(cp_{nn} + cm_{nn}) + x_{mw}^{s_l}(cp_{mw} + cm_{mw})\right] \tag{6}$$

Energy Consumption (*EC*): This factor is calculated based on the energy consumed for placement in the FOCN as well as the energy consumed to service executing in the fog node, which should be minimized (Ayoubi et al., 2021; Liu et al., 2022). The energy consumption factor is defined in Eq. (7).

$$EC = \sum_{a_k \in A}\sum_{s_l \in a_k} E_{PS}^{s_l} + E_{ES}^{s_l} \tag{7}$$

where, $E_{PS}^{s_l}$ and $E_{ES}^{s_l}$ are energy consumption related to placement and service execution of s_l, respectively. Liu et al. (2022) defines these parameters based on Eq. (8) and Eq. (9).

$$E_{PS}^{s_l} = SS_{s_l}.e_C.\mathbb{N}_C \tag{8}$$

$$E_{ES}^{s_l} = \int_{t_\alpha}^{t_\beta} e_{s_l}(t)dt \tag{9}$$

where, SS_{s_l} is the number of bits in s_l, e_C is the energy consumed per CPU cycle, $\mathbb{N}_C$ is the number of cycles required to process one bit, and $e_{s_l}(t)$ is the energy consumed to s_l executing at time t. Here, e_{s_l} is calculated based on the total number of tasks in s_l (Ibrahim et al., 2020; Azimirad et al., 2022), as shown in Eq. (10).

$$e_{s_l} = (EP_{max} - EP_{min})T_{s_l} + EP_{min} \tag{10}$$

where, T_{s_l} is the number of tasks executed in s_l, EP_{max} is the energy consumption at the peak load (100 % CPU usage) and EP_{min} is the minimum energy consumption in active mode (1 % CPU usage).

Delay Cost (*DC*): This factor includes processing delay and communication link delay that must be considered for each service. According to the system model, $d_o^{n^j}$, d_o^{mw}, d_o^{IoT} and d_o^{nn} are the communication link delays. Meanwhile, the processing delay is calculated based on the workloads of the service and the device allocated to the placement. In general, the processing delay for s_l is calculated by Eq. (11).

$$d^{s_l} = \frac{SS_{s_l}.\mathbb{N}_{s_l}}{\vartheta} \tag{11}$$

where, $\mathbb{N}_{s_l}$ is the number of cycles required to execute s_l and ϑ is the frequency of the device selected for placement.

Liu et al. (2022) and Zhang et al. (2022c) calculate the delay cost according to Eq. (12). Since the placement is done by the FOCN, the processing delay associated with the FOCN must always be taken into account.

$$DC = \sum_{a_k \in A}\sum_{s_l \in a_k} d_o^{s_l} + \left[x_{n^j}^{s_l}.d_{n^j}^{s_l} + x_o^{s_l}.d_o^{s_l} + x_{nn}^{s_l}.d_{nn}^{s_l} + x_{mw}^{s_l}.d_{mw}^{s_l}\right] \tag{12}$$

where, $d_{n^j}^{s_l}$, $d_o^{s_l}$, $d_{nn}^{s_l}$, and $d_{mw}^{s_l}$ are the processing delay of s_l on n^j, FOCN, NNFC, and CFCM (as cloud), respectively.

Throughput (*TP*): This factor is related to the number of services executed relative to the total services sent to the fog colony, which should be maximized (Ghobaei-Arani and Shahidinejad, 2022). The throughput factor is defined in Eq. (13).

$$TP = \frac{Srv_D}{Srv_{TD}} \tag{13}$$

where, Srv_D is the number of services that have been deployed and executed on the fog nodes before the deadline, and Srv_{TD} refers to all the services sent to the fog colony.

3.3. Metaheuristic approach

In general, SPP is an NP-Hard problem and has no definitive solution (Sami and Mourad, 2020). So far, various classical approaches to this problem have been proposed, which usually do not work well for big data problems. Metaheuristic approaches are a type of stochastic algorithms that are used to find optimal solutions to large-scale problems (Dokeroglu et al., 2019). These approaches offer far better solutions to placement problems (Sami and Mourad, 2020). Various classes of metaheuristic approaches have been developed in recent decades, such as GA, PSO, Harmony Search Algorithm (HSA), WOA, CSA, GWO, ODMA, Imperialist Competitive Algorithm (ICA), AOA (Dokeroglu et al., 2019). Most metaheuristic algorithms imitate natural phenomena for optimization work. For example, GA is inspired by Charles Darwin's of evolution theory, PSO is inspired by the concept of swarm intelligence in bird and fish groups, HAS is inspired by musical performance process, WOA is inspired by the behavior of humpback whales in hunting, CSA is inspired by the spawning of cuckoo birds in bird nests hosting other species, GWO is inspired by grey wolves during hunting, ODMA is inspired by the open source development model and user communities, ICA is inspired by the human's sociopolitical evolution process, and AOA is inspired by Archimedes' Principle.

Among metaheuristic algorithms, AOA has high convergence speed and avoid local minima, which provides better performance compared to similar algorithms such as GA, PSO, CSA and WOA (Hashim et al., 2021). In addition, AOA has been successfully used to solve optimization problems in various fields such as data mining, social networking, image processing, IoT, and cloud computing (Hashim et al., 2021). AOA is presented by Hashim et al. (2021) as a global optimization algorithm because it includes both exploitation and exploration techniques. The optimization process in AOA is based on the law of physics Archimedes' Principle. In general, AOA simulates the buoyancy force when objects of different weights and emulates are immersed in a fluid. AOA is a population-based *meta*-heuristic approach where each individual is considered as an immersed object. Like other *meta*-heuristic approaches, AOA creates the initial population from random solutions with different volumes, densities, and accelerations. Moreover, each solution has an initial random position in the fluid. After evaluating the fitness of the initial population, AOA tries to evolve the population in successive iterations until the termination condition is satisfied. The evolution process in AOA is done by updating volume and density as well as updating acceleration based on the conditions of collision with neighboring objects/solutions. By updating the factors of volumes, densities, and accelerations, the new position of each solution can be calculated. In this study, AOA is used to efficiently deploy IoT services on fog nodes.

4. Proposed framework

This section is related to the analysis of the interactions between different components in the cloud-fog-IoT ecosystem through a conceptual computing framework. This conceptual framework is developed based on the SPP solving strategy in Fig. 3. In fog computing, resources are organized in several fog colonies in a distributed manner. Colonies can be formed and managed based on geographic distribution. Accordingly, the resource needs of IoT devices can be met by the nearest fog colony, which leads to a reduction in delay. Since each colony may receive IoT services independently, it is necessary to configure the autonomous computing model (i.e., MADE-k) to manage resources in each colony (Salimian et al., 2022).

As illustrated, the proposed framework consists of three layers (i.e., IoT devices, fog and cloud). The IoT devices layer can include any smart device connected to the Internet. Smartwatch, smart bicycle, smart fire alarm, smartphone and medical sensor are examples of IoT devices. Each device can have some processing requests such as real-time applications. Requests are sent to the fog layer for processing by IoT devices (i.e., end users) and through fog gateways. Basically, the fog computing layer is managed locally by fog colonies. It can be organized by geographic regions, where requested IoT applications are processed by the nearest fog colony. Fog colonies consist of one or more fog nodes with limited resources. Switch, gateway, router, server, base station, set-top box, and access point are examples of fog nodes. As mentioned earlier, user requests are considered as IoT applications, and each application can contain several independent IoT services. In this regard, each fog node includes a set of virtual machines that can host multiple IoT services depending on the resources available.

In addition, a FOCN is embedded in each fog colony that can manage and support the functional elements. The most important task of FOCN is to plan the deployment of IoT services in the current time period. FOCN uses an admission control unit for placement. This unit receives the services from the fog gateways and then decides on the deployment location of each service. This is done by considering the deadline of each service and subordinate fog colony resources. Since we formulate the SPP as an optimization problem, the placement must be done in each time period τ. Based on this, the requests are placed in a queue and wait until the next time period.

4.1. Fog colony

The fog layer can be considered as a set of non-overlapping colonies/domains for efficient management. Each colony contains a set of fog nodes that are managed by a dedicated node called FOCN. Each colony contains a set of fog nodes that are managed by a dedicated node called FOCN. Fog colonies can communicate with each other through FOCN to deploy IoT services. Also, each fog colony is supported by a head element in the cloud called CFCM. This middleware acts as a bridge of communication between fog and cloud through the cloud gateways. Therefore, non-real-time requests can be transmitted to CFCM via FOCN to be processed in the cloud. Each colony has a knowledge base managed by FOCN and is used to store details of IoT services and available resources of active fog nodes. According to the knowledge base, the fog colony can plan the deployment of IoT services by FOCN.

Fog nodes are the main entities of any fog colony. These nodes have limited computing and storage resources and can host IoT services. Communication between fog nodes is provided through protocols such as NETwork CONFiguration (NETCONF), Constrained Application Protocol (CoAP) and Simple Network Management Protocol (SNMP) (Slabicki and Grochla, 2016). Each fog node is a software development technology and consists of five components: listener, monitor, database, control manager, and compute. The listener receives the details of the services from FOCN. The monitor can supervise the execution of services in the compute component. Details of services for processing and available resources are stored in the database. The control manager can perform some actions and the compute component provides the resources required for executing services. In general, fog nodes are one of the thin and fat types that only the fat type can host IoT services and thin nodes are often used as sensors (Skarlat et al., 2017).

4.2. Fog orchestration control node

Each colony contains a dedicated node called FOCN, which manages the subordinate fog colony. Other tasks of the FOCN include

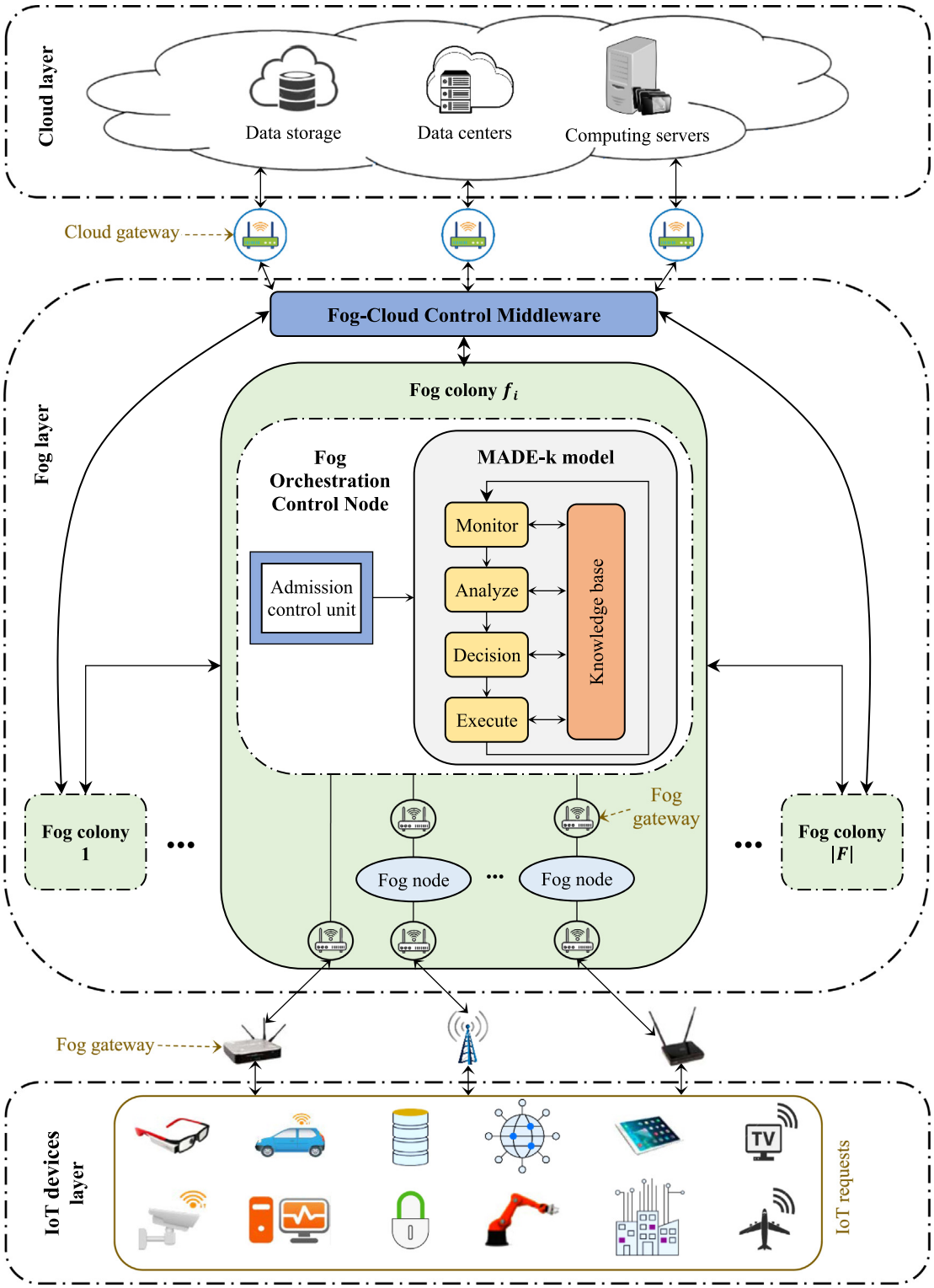

Fig. 3. Proposed conceptual framework for the cloud-fog-IoT ecosystem.

subordinate fog colony reconstruction and resource efficiency analysis. The admission control unit and the MADE-k autonomous model are the main components of FOCN. The admission control unit is responsible for processing requests received from the IoT device layer, and MADE-k solves the SPP in each time period.

FOCN consists of six components: listener, reasoner, propagation, watchdog, registry, and storage (Vashani et al., 2016). The listener receives details of IoT applications from IoT devices. The reasoner component analyzes the placement details of IoT services. Requests whose resources are not satisfied by the current fog col-

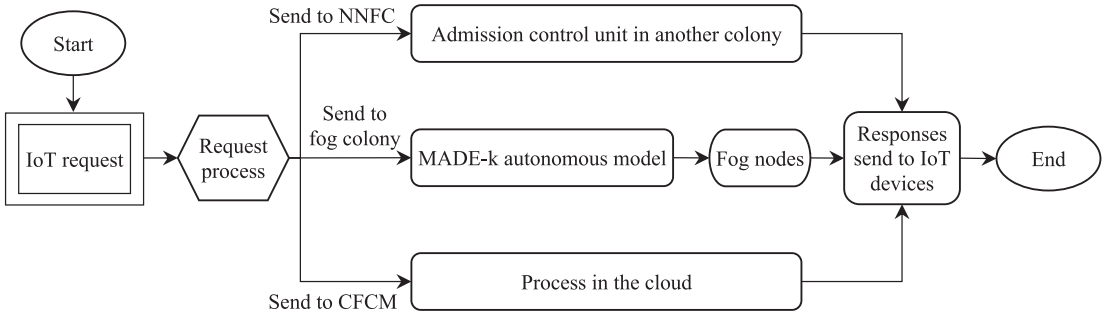

Fig. 4. Architecture of admission control unit.

ony are transmitted by the propagation component to the NNFC (for processing by fog nodes in another colony) or CFCM (for processing by the cloud). In general, NNFCs are detected by examining communication link delays between FOCNs. The watchdog analyzes QoS requirements and resources occupied in the fog colony. The placement of services in the compute component of the nodes is provided by the registry component and the placement pattern is maintained by the storage component.

4.2.1. Admission control unit

All requests sent by IoT devices are processed in the admission control unit. The architecture of this unit is shown in Fig. 4. The unit identifies real-time applications to be executed by subordinate fog nodes based on the application deadline and available resources of the colony. Some real-time applications may not be able to deploy on subordinate fog nodes due to lack of resources. These applications are sent to NNFC by the admission control unit. This can be repeated by other colonies, where it increases delay and makespan. Meanwhile, the admission control unit sends non-real-time applications to the CFCM to execute on the cloud.

4.2.2. MADE-k autonomous model

Autonomous computing has been introduced by IBM to describe system activities without user intervention (Ayoubi et al., 2021). The purpose of this paradigm is to adapt to the environment through self-managing theory. Self-managing can ensure the privacy, availability and reliability of computer systems. The MADE-k model as a control loop model has the ability to manage and schedule resource requests, as shown in Fig. 5 (Zhao et al., 2022). MADE-k consists of four main phases: monitoring, analysis, decision-making, and execution. Data related to all these phases are shared by a knowledge base. The pseudocode of the MADE-k model for managing requests received by the current fog colony is shown in Algorithm 1.

Monitoring phase: This phase is responsible for monitoring the status of subordinate nodes, available resources of nodes, and resources required by IoT applications. This phase also monitors resource usage (i.e., CPU, memory and storage) and requests transmitted to NNFC and CFCM. Monitoring is performed in each time period τ and related reports are stored in the knowledge base.

Analysis phase: In this phase, the priority of executing requests for each IoT application is calculated. Numerous studies have shown that the time interval before the deadline of applications can well express the execution priority of applications (Liu et al., 2022; Ghobaei-Arani and Shahidinejad, 2022; Zhang et al., 2022c). Let $P(a_k)$ be the execution priority of application a_k, as shown in Eq. (14).

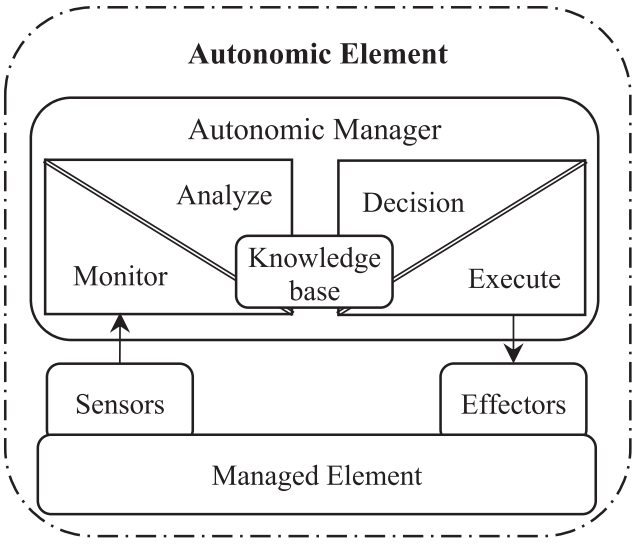

Fig. 5. Architecture of MADE-k autonomous model.

$$P(a_k) = \lambda . \frac{1}{D_{a_k}} + (1 - \lambda) . \frac{1}{t - R_{a_k}} \tag{14}$$

where, D_{a_k} is the deadline for a_k, R_{a_k} is the sending time of a_k by the user, t is the current time, and λ is a coefficient for considering the effect of D_{a_k} on R_{a_k}.

Decision-making phase: This phase is responsible for deploying IoT services on the subordinate fog nodes. In each time period τ, the received IoT services are processed for placement. This can ensure system dynamics in the presence of real-time applications. SPP is a placement problem for deciding whether to deploy IoT services efficiently on subordinate fog nodes. Here, the SPP is dissolved in a distributed form for each fog colony. In this paper, the decision-making process is performed by AOA as a metaheuristic approach, which is discussed in Section 5. The AOA makes decisions about the placement of each IoT service based on the available resources of the nodes, the resources required by the services, and the application prioritization. After placement, the details of deploying the services are stored in the knowledge base.

Execution phase: IoT services placement planning is executed in this phase. Here, each IoT service is deployed on a fog node according to the decision-making phase. The node then provides the

resources required for the service and stores the execution results in the knowledge base.

Algorithm 1. Pseudocode of the MADE-k autonomous model for request management

Input: IoT services/applications and the available resources of the fog nodes.
Output: Details of the deployment of each IoT service.

for each τ time period **do**
  for each application $a_k \in A$ **do** // Monitoring phase
    Monitor (set of the IoT services requested for a_k).
    Monitor (U_{s_l}, M_{s_l}, S_{s_l}, D_{s_l}).
    Storing details of the IoT services in the knowledge base.
  end
  for each fog node n^j in the subordinate fog colony **do** // Monitoring phase
    Monitor (status of fog node n^j).
    Monitor (U_{n^j}, M_{n^j}, S_{n^j}).
    Storing details of the fog nodes status in the knowledge base.
  end
  for each set of application $a_k \in A$ **do** // Analysis phase
    Calculate the priority of a_k through Eq. (14).
    Storing details of the application prioritization in the knowledge base.
  end
  for each IoT service $s_l \in S$ **do** // Decision-making phase
    Placement of the IoT service s_l is done by AOA taking into account the priority of the applications.
    Storing the placement details of the IoT service s_l in the Knowledge base.
  end
  for each IoT service $s_l \in S$ **do** // Execution phase
    Deploy and execute the IoT service s_l on the fog node according to the placement decision.
    Storing the execute result of the IoT service s_l in the knowledge base.
  end
end
Return the placement plan discovered by AOA as output.

5. Proposed placement scheme

Given the effectiveness of metaheuristic techniques in solving placement problems, we model placement based on AOA as a multi-objective optimization problem. AOA is a new metaheuristic approach that has proven effective in solving optimization problems over most equivalence methods. We refer to the proposed scheme as SPP-AOA, which solves SPP through AOA. SPP-AOA seeks to improve overall system performance and QoS satisfaction by compromising between different objectives. The decision for the placement of each service is made through the MADE-k autonomous model in each time period τ. Here, the MADE-k model within each fog colony is configured to perform placement in a distributed manner. This means that IoT requests received will have to wait until the next time period. Based on the proposed conceptual framework, MADE-k performs placement only on available fog nodes from the current colony.

To improve placement performance, SPP-AOA extracts the distribution of resources consumed by sensor nodes over time to handle more future requests. In addition to the above, AOA is configured based on a shared parallel architecture to improve overall system performance by reducing placement time. Fig. 6 shows the AOA framework based on shared parallel architecture. In this architecture, Z AOA algorithm is configured and executed simultaneously by parallel computing toolbox with Simulink in MATLAB. This toolbox in MATLAB provides multicore processors and computer clusters to solve computationally intensive problems. Simulink allows the use of full multi-core processing power. Here, Simulink and parallel for-loops are used to implement the AOA architecture in parallel. Each ICA_i implemented in parallel is shared by one memory. This memory contains elite solutions discovered by all AOA algorithms. Each ICA_i exchanges its solutions with the shared memory based on a predefined exchange threshold, where the final solution is the best solution from the shared memory.

Moreover, SPP-AOA is equipped with a retracing mechanism that can backtrack to the state before service processing during unsuccessful deployment. If the resources required for even one service of the IoT application are not successfully satisfied, the placement for that application will fail. In this case, the retracing mechanism is responsible for resetting the fog colony to the pre-placement conditions of the application's services (i.e., freeing up occupied resources). The proposed scheme sends the IoT service to the NNFC when encountering failed deployment, and this can be repeated for other colonies as well.

In general, like other evolutionary approaches, AOA includes steps 1) solution encoding scheme, 2) generate the initial population, 3) fitness function, 4) population evolution, and 5) stop conditions. The details of the AOA steps in the proposed SPP-AOA scheme are described in detail below.

5.1. Solution encoding structure

The encoding structure represents SPP mapping based on Integer Linear Programming (ILP), which provides the necessary condi-

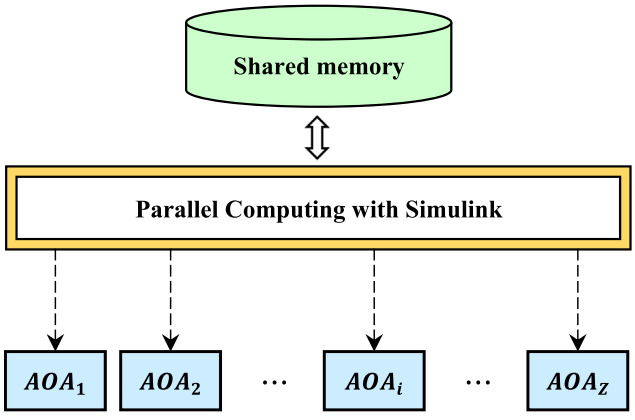

Fig. 6. AOA framework based on shared parallel architecture.

tions for analyzing and searching for the optimal solution. In general, different IoT services can be assigned to the same fog nodes, where the nodes must have the resources required by the services. In this regard, SPP-AOA proposes the reuse of fog nodes in various IoT services to achieve computation acceleration. MADE-k performs placement only on available fog nodes from the current colony based on the proposed conceptual framework. Therefore, the purpose of deploying $|S|$ IoT service from $|A|$ application received on $|f|$ nodes is the current available colony in time τ. In addition, FOCN is considered as a fog node and can host IoT services. Fig. 7 shows the solution encoding structure in SPP-AOA, where the solution vector is highlighted in blue. Moreover, each solution is recognized in the AOA as a "country".

As depicted, every IoT application such as a_k contains $|a_k|$ IoT service, and in total there are $|S|$ IoT services in time period τ for placement. Every IoT service such as s_l must be deployed on a node such as $n^\alpha \in f$. n^α can be a fog node with sufficient resources among the $|f|$ fog nodes in the current colony. Therefore, multiple IoT services can be deployed on n^α. According to the encoding structure, each solution consists of a vector of length $|S|$, where each element of this vector refers to an IoT service and the value of each element refers to the assigned fog node index.

5.2. Generate the initial population

Generating an initial population is the first step in the process of optimizing evolutionary algorithms. The population is a subset of the solutions in the current iteration. In general, the initial population is generated randomly or through innovative methods based on a defined encoding structure (Rezaeipanah et al., 2021). Usually, the random method due to various limitations and requirements in SPP leads to impossible solutions or low quality. In addition, innovative methods can reduce population diversity and lead to a condition called Premature Convergence (Ali et al., 2020; Mojarad et al., 2021).

The random-innovative technique is modeled based on the distribution of resources. Because fog nodes with sufficient resources can host multiple IoT services, considering the distribution of resources in the initial population generation could lead to an increase in the number of successful placements in the future. In this technique, the first solution is created randomly. Then, the average computing resources occupied by each fog node associated with the current request is stored in memory. This memory can show the distribution of resource consumption in fog nodes according to the deployment of IoT services. In this regard, the second solution is greedy generated. In the greedy method, the services are sorted in ascending order based on the average computing resources, and each service is assigned to the fog node with the most resources, respectively. This greedy method ultimately generates an innovative placement solution that considers efficient distribution of services over fog nodes. Other solutions are then generated, where after each random solution a solution is greedy method generated. In general, the efficient distribution of resources consumed during initial population generation can lead to an increase in the number of successful placements. The architecture of generating the initial population with random-innovative technique is shown in Fig. 8.

According to this technique, the initial population consists of N_P solutions (as immersed objects) that are generated randomly-innovative. According to the AOA, each solution is presented as position of an immersed object. Let $P_k, \forall k = 1, 2, \cdots, N_P$ be the position of k-th solution of the population and $OF(P_k)$ denote the fitness value for P_k. Accordingly, SPP-AOA performs the initial population generation using a random-innovative technique, as shown in Algorithm 2. Here, the random solution is generated according to the encoding structure defined in Fig. 7. Eq. (15) is used to generate a random solution.

$$P_k^i = lb_k + rand.(ub_k - lb_k), \forall i = 1, 2, \cdots, |S|; k = 1, 2, \cdots, N_P \quad (15)$$

where, P_k^i is the i-th element of the k-th solution, lb_k is the lower bound of the search space, ub_k is the upper bound of the search space, and *rand* generates a random number between [0, 1]. Here, $lb_k = 1$ and $ub_k = |f|$.

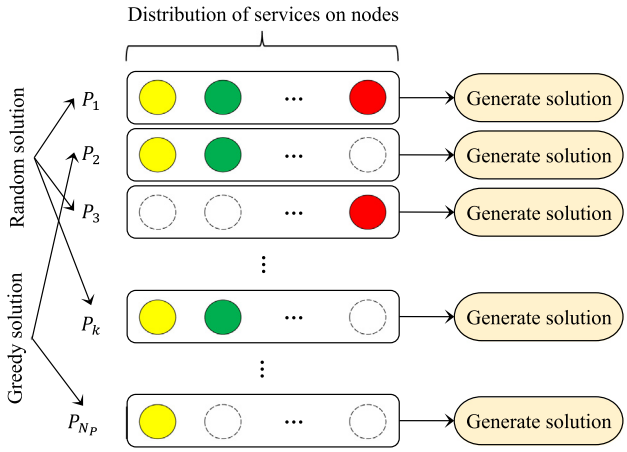

Fig. 8. Architecture of initial population generation with random-innovative technique.

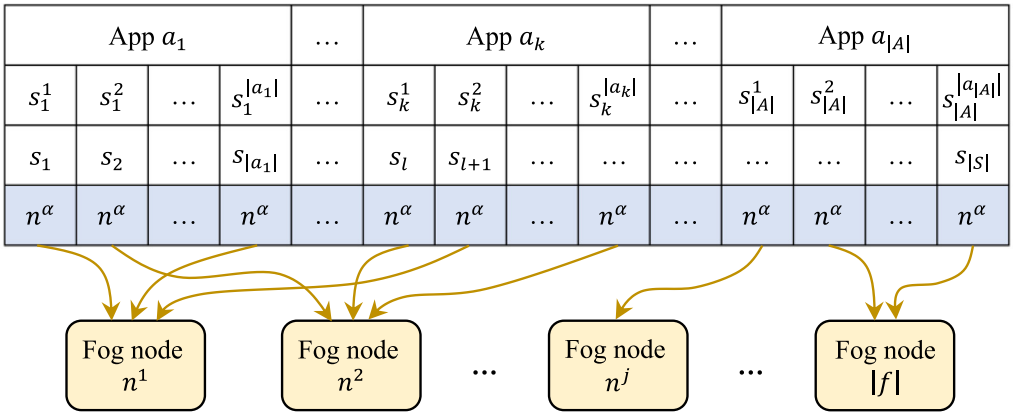

Fig. 7. Solution encoding structure in SPP-AOA.

In addition to position, AOA considers volume, density and acceleration parameters for each solution. Let vol_k, den_k and acc_k be the volume, density, and acceleration for P_k, respectively, which are initialized according to Eq. (16).

$$vol_k^i = rand \tag{16}$$

$$den_k^i = rand$$

$$acc_k^i = lb_k + rand.(ub_k - lb_k)$$

Algorithm 2. Pseudocode of the generating the initial population with the random-heuristic technique

Input: Details of IoT services, details of fog resources and AOA parameter values.

Output: The initial population and values of fitness.

1: Ascending sorting of IoT services based on the average resources required.
2: **for** k = 1 to $N_P/2$ **do**
3: $P_k \leftarrow$ Generate a random solution based on the encoding structure defined in Fig. 7.
4: $OF(P_k) \leftarrow$ Calculate the fitness value of P_k based on the Eq. (17).
5: Update the fog node resources based on the solution generated.
6: Save the average resources occupied by each fog node in memory.
7: Allocate IoT services in order of sort to fog nodes with the freest resources.
8: $P_k \leftarrow$ Save the solution generated based on the distribution of resource consumption on the fog nodes.
9: $OF(P_k) \leftarrow$ Calculate the fitness value of P_k based on the Eq. (17).
10: **end**
11: Return initial population and values of fitness.

Line 1 sorts IoT services based on occupied resources. Line 2 specifies the number of solutions generated with $N_P/2$, because two solutions (one random and one greedy) are created in each iteration. Line 3 generates the solution at random and Line 4 calculates its fitness value. In line 5, the resources of fog nodes are updated based on the placement provided by the solution generated. Line 6 stores the resources of fog nodes in memory. Lines 7 and 8 refer to greedy method generating the solution and storing it in memory, and lines 9 calculate the fitness value of the solution generated. Eventually, line 11 returns the initial population and fitness values as output.

5.3. Fitness function

Each solution from the population represents a placement plan that can have a fitness value given the limitations of the fog environment and QoS requirements. SPP-AOA calculates the fitness value of solutions by considering multiple objectives. The purpose of the SPP-AOA is to compromise the costs (i.e., resource utilization, service cost and energy consumption) and QoS (i.e., delay cost and throughput) through an efficient placement plan. The parameters needed to calculate these objectives are available through the fog colony information and details of the IoT services. Therefore, the proposed objective function consists of five different factors that are calculated according to Eq. (17), which should be minimized.

$$OF = \frac{\xi_{sc} * SC + \xi_{dc} * DC + \xi_{ec} * EC}{\xi_{ru} * RU + \xi_{tp} * TP} \tag{17}$$

where, RU, SC, EC, DC and TP are resource utilization, service cost, energy consumption, delay cost and throughput, respectively, as described in subsection 3.2. Also, ξ_{ru}, ξ_{sc}, ξ_{ec}, ξ_{dc} and ξ_{tp} are the weights of these objectives, respectively. Due to the lack of priority between the defined objectives, all weights are considered the same.

5.4. Population evolution

The population evolution process in AOA includes updating volume, density and acceleration and finally updating the position of each solution based on these parameters. The update of volume and density for iteration $t+1$ is done by Eq. (18) and Eq. (19), respectively.

$$vol_k^i(t+1) = vol_k^i(t) + rand.\left(vol_{best} - vol_k^i(t)\right) \tag{18}$$

$$den_k^i(t+1) = den_k^i(t) + rand.\left(den_{best} - den_k^i(t)\right) \tag{19}$$

where, vol_{best} and den_{best} refer to the volume and density of the best solution found so far, respectively.

After updating the volume and density, AOA updates the acceleration parameter for each object/solution. The acceleration update is simulated based on the collision between objects in the fluid. AOA uses Transfer Factor (TF) operator for this. The TF operator states that first collisions occur between objects and after some time the objects in the fluid reach equilibrium. AOA uses this idea to change the search mode from exploration to exploitation. Eq. (20) models the TF operator. This operator shows how the transmission increases gradually with time. In addition to TF, Density Reduction Factor (DRF) is also effective in updating acceleration, as shown in Eq. (21). DRF decreases with time, which provides a balance between exploration and exploitation.

$$TF = exp\left(\frac{t - t_{max}}{t_{max}}\right) \tag{20}$$

$$DRF(t+1) = exp\left(\frac{t_{max} - t}{t_{max}}\right) - \left(\frac{t}{t_{max}}\right) \tag{21}$$

where, t and t_{max} refer to the current iteration and the maximum iteration in AOA, respectively.

According to TF and DRF, acceleration can be updated in two modes including exploration and exploitation. Basically, the exploration phase occurs when $TF \leq 0.5$. In this case, it is assumed that the current object collided with a neighboring object. Let mr be the index of a randomly selected neighbor object. Based on this, the acceleration update for iteration $t+1$ is done by Eq. (22).

$$acc_k^i(t+1) = \frac{den_{mr} + vol_{mr}.acc_{mr}}{vol_k^i(t+1).den_k^i(t+1)} \tag{22}$$

Meanwhile, the exploitation phase occurs when $TF > 0.5$. In this case, it is assumed that the object is reaching equilibrium in the fluid after the collision. Based on this, the acceleration update for iteration $t+1$ is done by Eq. (23).

$$acc_k^i(t+1) = \frac{den_{best} + vol_{best}.acc_{best}}{vol_k^i(t+1).den_k^i(t+1)} \tag{23}$$

where, acc_{best} acceleration is the best solution discovered so far.

After updating the volume, density and acceleration, the position of each solution is updated in the search space. The position is updated based on TF and DRF value in two modes of exploration

and exploitation. Eq. (24) updates the position of P_k for iteration $t+1$.

$$P_k^i(t+1) = \begin{cases} P_k^i(t) + C_1.rand.acc_k^i(t+1).DRF.\left(P_{rand} - P_k^i(t)\right) & TF \le 0.5 \\ P_k^{best}(t) + F.C_2.rand.acc_k^i(t+1).DRF.\left(T.P_{best} - P_k^i(t)\right) & TF > 0.5 \end{cases} \quad (24)$$

where, C_1 and C_2 are number constants, which are set to 2 and 6, respectively (Hashim et al., 2021). T controls the balance between exploration and exploitation and increases with time Hashim et al. (2021) increase T with time in the range of $0.3TF$ to 1. In addition, F is a flag to control the direction of motion, which is initialized by Eq. (25).

$$F = \begin{cases} +1 & rand \le 0.5 \\ -1 & Otherwise \end{cases} \quad (25)$$

5.5. Stop conditions

Each iteration of the AOA leads to population evolution, where the process of optimization eventually converges when it reaches only one empire. The convergence points in evolutionary algorithms are determined by stopping criteria because execution may continue indefinitely. We provide the stopping conditions in SPP-AOA by 1) reaching the convergence point and 2) conventional iterative maximum. The first method is satisfied when there is only one empire and the second method defines the stop condition with a maximum iteration (e.g., t_{max}). After satisfying the stop conditions, the best solution based on the objective function is considered as the output of the algorithm and then the placement of IoT services is performed based on it.

6. Performance evaluation

This section examines the performance of the proposed scheme compared to state-of-the-art methods on a synthetic fog environment. The proposed SPP-AOA scheme for placement of IoT applications uses AOA as a metaheuristic algorithm. Hence, we compare SPP-AOA with equivalence algorithms such as SPP-PSO (Salimian et al., 2022), CSA-FSPP (Liu et al., 2022) and FSP-ODMA (Zhao et al., 2022). All of these algorithms are simulated on the same fog environment using MATLAB R2019a. We analyze experiments and comparisons based on various performance metrics such as convergence speed, runtime, location of deployed services, number of services executed, average waiting time, number of services failed, number of remaining services. All experiments were performed on the ASUS VivoBook S533 Laptop, Intel Core i7-1165G7 Processor at 4.70 GHz, 16 GB DDR4 RAM and Windows 10 Home. In addition, all simulations are performed based on 15 independent run due to the existence of some stochastic parameters to ensure reliable results. Details on simulation setup, performance metrics, comparison results, and performance analysis of algorithms are provided later in this section. Besides, the source code of the simulation sections is available at https://github.com/ServicePlacement/SPP-AOA.

6.1. Simulation setup

We randomly generated the network topology as the Albert–Barabasi topology, which has also been used in other fog resource management studies (Salimian et al., 2021; Salimian et al., 2022; Ghobaei-Arani and Shahidinejad, 2022). In this topology, network traffic is considered as an IoT application for each request received, which includes details of application services and the resources they require. Although we model the Barabási-Albert network topology, we consider intermediate devices as fog nodes that are installed in a hierarchical style. In fact, we consider a combination of realistic network topology and synthetic network topology for the simulation environment. In each time period τ, a certain number of requests are randomly generated and received by the fog colony. The total number of available fog colonies is considered to be 5 and the distance between them is available. Each request is randomly assigned to a fog colony, and each fog colony plans the placement of the request on the subordinate fog nodes. FOCN in each colony can deploy services on subordinate fog nodes or transfer it to NNFC (to deploy on fog nodes of other fog colonies) or CFCM (to executing by cloud servers). Also, FOCN can host some services depending on the resources available. Here, the cost of cloud processing for each Billing Time Unit (BTU) is $0.3 (Ghobaei-Arani and Shahidinejad, 2022).

Each node contains a certain number of computing and storage resources, which are considered as blocks. On the other hand, every IoT service requires a certain number of resource blocks. When each service is deployed on a fog node, part of its resource block is occupied. In general, occupied resource blocks are not released until the end of the simulation (Salimian et al., 2021). In addition, the number of types of services provided by the nodes is limited and is set to 5. In fact, the types of services can vary depending on the type of IoT device. For example, smartwatches, smartphones, smart TVs, smart refrigerators, and smart bicycles are considered as types of IoT devices (i.e., IoT applications). In addition, each IoT device can have different types of services for processing. For example, Samsung smartphone and Apple smartphone are two different types of services from the IoT device of smartphone. Here, the number of types of services associated with different types of IoT devices is 20.

In this study, experiments are performed based on 10 consecutive time periods (i.e., $\tau = 1, 2, \cdots, 10$). Table 4 shows how many IoT services and at what time are received by the fog perspective (Salimian et al., 2022). Here, 615 IoT services are processed in all 10 time periods.

Fog nodes and IoT services have different parameters. Table 5 shows the settings of these parameters in the simulation (Salimian et al., 2022; Liu et al., 2022; Zhang et al., 2et al., 2022c). Let the values of the parameters with a certain range be randomly selected with a uniform distribution.

Each node contains resources of storage, RAM and CPU. Similarly, every IoT service has a demand for resources including storage, RAM, and CPU. Table 6 shows the details of the available resources of the nodes and the resources required by the services (Liu et al., 2022; Mohaidat et al., 2022; Zhang et al., 2022c). Here, the resource details are the same for every 5 nodes in the fog colony. Also, the number of types of IoT applications is 5, so each node

Table 4
Number of requested IoT services in 10 time periods.

Time period τ	1	2	3	4	5	6	7	8	9	10
Time t	8	16	24	32	40	48	56	64	72	90
Number of services	71	48	48	46	96	77	37	52	85	55

Table 5
Parameter settings for fog nodes and IoT services in the simulation.

Parameters	Values
Number of service types for each IoT application	20
Number of resource blocks for each fog node	5500–6000
Number of resource blocks for each IoT service	25–35
Cost of each IoT service ($)	10–20
Number of fog nodes in each colony	15

Table 6
Resource details for nodes and services.

Items	Type	Storage (MB)	RAM (MB)	CPU (MIPS)
Fog node	1	256	256	100–200
	2	512	512	200–300
	3	1024	1024	300–1400
	4	2048	2048	1400–1600
	5	4096	4096	1600–3000
IoT service	1	128	128	100–200
	2	256	256	200–300
	3	512	512	300–1400
	4	2048	1024	1400–1600
	5	4096	4096	1600–3000

can support up to 5 different types of services. In addition, each IoT application has a deadline that is randomly set between 120 s and 240 s.

The proposed scheme has several parameters that must be adjusted before simulating the appropriate values. We adjust the parameter values to search for the most appropriate scenarios based on the Taguchi approach. In addition, some parameters are adjusted based on similar tasks (Skarlat et al., 2017; Liu et al., 2022; Ghobaei-Arani and Shahidinejad, 2022). Finally, the values of the parameters in the simulation are set as follows: $e_C = 1W$, $\mathbb{N}_C = 1bit$, $\mathbb{N}_{s_l} = 1$, $\vartheta = 2.0\,GHz$, $\lambda = 0.5$, $N_P = 35$, $t_{max} = 100$.

6.2. Performance metrics

We evaluate comparable algorithms for solving SPP with different performance metrics. These metrics include convergence speed, runtime, location of deployed services, number of services executed, average waiting time, number of services failed, number of remaining services.

Convergence speed: This metric is related to the optimization process in evolutionary algorithms that occurs with decreasing population variability.

Runtime: This metric can indicate the complexity of placement algorithms for solving SPP, which is reported based on the total runtime (s) of the simulation.

Location of deployed services: Each IoT service can be deployed on a subordinate fog node or transferred to NNFC or CFCM depending on the FOCN decision. In addition, the service can be executed on FOCN. Therefore, this metric shows the location of a service for execute.

Service cost: This metric for each service represents the monetary cost of that IoT service, which is calculated based on Eq. (4).

Number of services executed: This metric shows the number of services whose resources have been allocated before the deadline.

Average waiting time: This metric shows the average waiting time for services executed before the deadline is violated.

Number of services failed: This metric refers to the number of services that could not be placed and executed before the deadline was violated.

Number of remaining services: Each service may be executed on a node over a time period depending on the plan performed. However, due to the maximum number of time periods provided, some services may not be successful placement. Among these services, services with non-violation of the deadline are considered as the remaining services for placement in the next time period.

6.3. Results and discussion

This section includes simulation results, comparisons, and discussion. The proposed SPP-AOA scheme is based on a metaheuristic algorithm that uses AOA to decide on the placement of services. For this reason, SPP-AOA is compared to other similar metaheuristic algorithms. Here, PSO, CSA and ODMA are among the *meta*-heuristic algorithms that are evaluated in comparison with AOA. We report the results of these algorithms based on SPP-PSO (Salimian et al., 2022), CSA-FSPP (Liu et al., 2022) and FSP-ODMA (Zhao et al., 2022), respectively.

In the first experiment, the convergence of different algorithms to increase the fitness value and reach the convergence point is compared. At the point of convergence, the evolutionary process can be stopped due to the reduction of population diversity. However, we report the comparison in each algorithm for 100 iterations after the end of the time period $\tau = 1$, as shown in Fig. 9. As illustrated, the convergence speed of SPP-PSO and CSA-FSPP is appropriate and they reach the convergence point in 35 iterations. The worst results are for FSP-ODMA, because the algorithm is developed for continuous environments and does not perform well for discrete problems. In general, SPP-PSO, CSA-FSPP and FSP-ODMA achieved fitness values of 684, 685 and 696 after 100 iterations, respectively. Nevertheless, SPP-AOA performed better with a fitness value of 682, although this was achieved at a slower convergence speed than other algorithms. The reason for the slow convergence speed of the AOA is the absorption policy, because it must be applied to all solutions.

Although SPP-AOA achieves lower fitness value and SPP-PSO converges better, runtime must also be analyzed due to the decision on the dynamic environment. This metric can indicate the complexity of placement algorithms for solving SPP, which is reported based on the total runtime of the simulation. Execution time is very important in reducing the service waiting time, because the placement process is done in each time period and the recently received services have to wait until the next time period. Hence, in the second experiment, the runtime of different algorithms is analyzed after the end of the time period $\tau = 1$. Fig. 10 shows the results of this comparison for different algorithms at 100 iterations. As illustrated, the CSA-FSPP results are clearly the worst with 97 s runtime. In general, the evolutionary process in CSA involves several steps such as laying eggs and community formation, which complicates the algorithm. Other algorithms have almost the same runtime, although the SPP-PSO results are slightly better with 23 s.

Each IoT service can be deployed on subordinate fog nodes, NNFC, CFCM or FOCN itself depending on the FOCN decision. An efficient placement algorithm is applied to deploy services on FOCN or fog nodes in the current fog colony. Because transferring services to NNFC or CFCM will lead to more delays and costs. Therefore, the location of deployed services is important in the performance of placement algorithms. Hence, in the third experiment, we compare the placement rates of services for different algorithms. The results of this comparison are presented in Table 7 after the end of time period $\tau = 1$ and $\tau = 10$. Here, the placement rate of the services on the fog nodes, FOCN, NNFC, and CFCM for each algorithm are reported. Also, the last column of this table refers to the service placement rate on the current fog colony

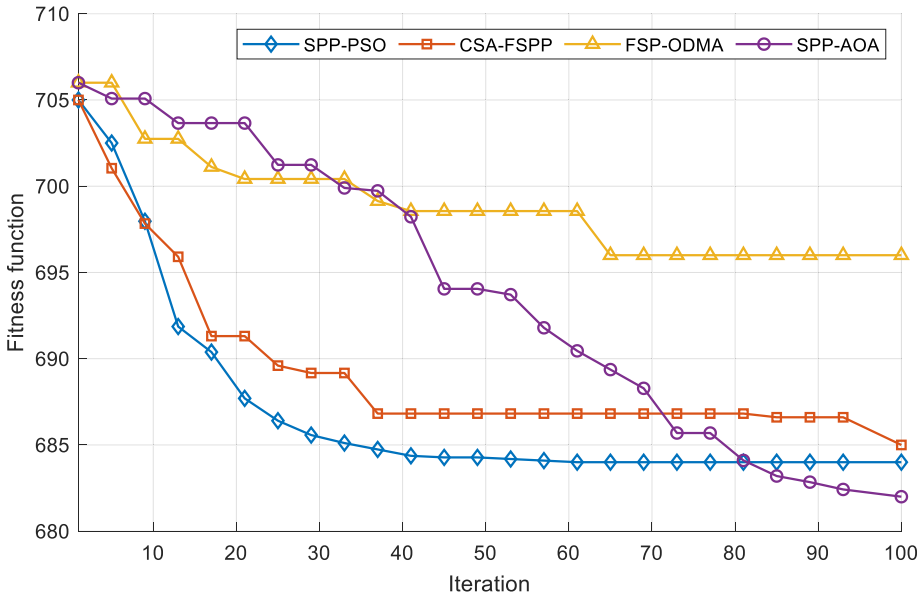

Fig. 9. Results for the convergence speed at the end of the time period. $\tau = 1$.

(i.e., fog nodes and FOCN). The results show that after the end of the time period $\tau = 1$, most of the services are places on the current fog colony (fog nodes or FOCN). Because at the beginning of placement, the available resources of the nodes are abundant and fewer services are forced to move to NNFC or CFCM. However, comparative results after the end of the time period $\tau = 10$ indicate that more requests have been transferred to the NNFC or CFCM due to the occupation of current colony resources. Overall, SPP-AOA performs best with 98 % of services deployed on the current fog colony based on $\tau = 1$. This advantage also exists for $\tau = 10$, and SPP-AOA is the best with 80 % compared to other algorithms.

The fourth experiment compares the service cost in different algorithms. Service cost is one of the most important metrics in QoS improvement, where it refers to the monetary cost paid by the user to executing services. Service cost can be reduced by deploying user requests to fog nodes. Therefore, more efficient placement algorithms can provide lower service costs for the user. Fig. 11 shows the service cost results for $\tau = 1$ and $\tau = 10$. As illustrated, SPP-AOA has provided better results with lower service costs in both scenarios. On average, SPP-AOA reduced service costs by 2.05 %, 1.52 % and 1.51 % compared to SPP-PSO, CSA-FSPP and FSP-ODMA algorithms, respectively. The reason for this superiority is the number of services placed more on the current fog colony by SPP-AOA.

The number of successfully performed services is analyzed in the fifth experiment for different algorithms. Fig. 12 shows the

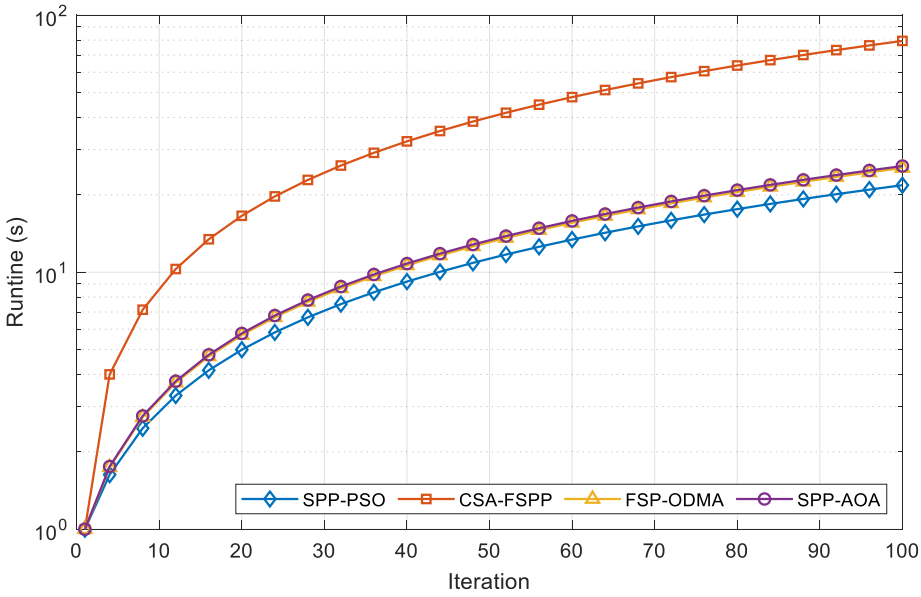

Fig. 10. Results for runtime at the end of the time period. $\tau = 1$.

Table 7
Results of location of deployed services in different algorithms.

Time period	Algorithms	Fog nodes	FOCN	NNFC	CFCM	Current colony
$\tau = 1$	SPP-PSO	0.77	0.16	0.03	0.04	0.93
	CSA-FSPP	0.82	0.15	0	0.03	0.97
	FSP-ODMA	0.84	0.12	0.01	0.03	0.96
	SPP-AOA	0.84	0.14	0	0.02	0.98
$\tau = 10$	SPP-PSO	0.41	0.23	0.23	0.13	0.64
	CSA-FSPP	0.46	0.25	0.21	0.08	0.71
	FSP-ODMA	0.42	0.24	0.20	0.14	0.66
	SPP-AOA	0.49	0.31	0.16	0.04	0.80

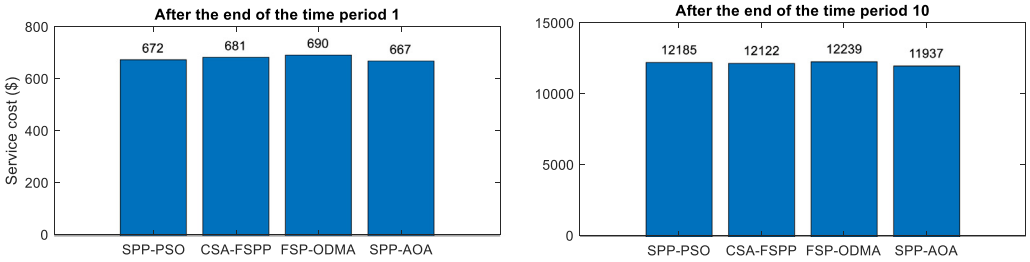

Fig. 11. Comparison of different algorithms based on service cost.

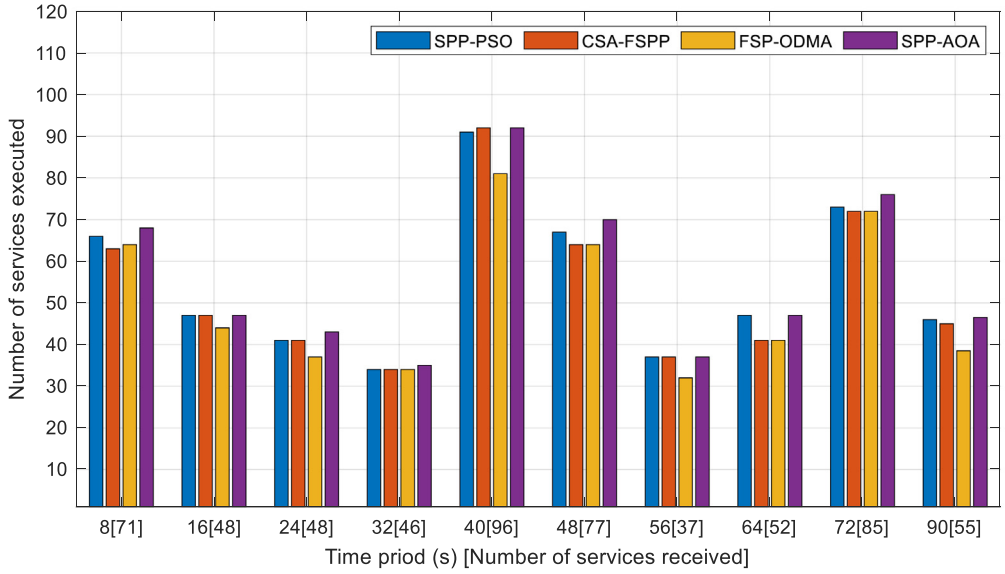

Fig. 12. Results for the number of services performed in each time period.

results of this experiment for each time period. In each time period, the number of services received by the fog landscape is predefined. Each algorithm for placement should assign the most suitable fog nodes to IoT services, taking into account resource constraints and deadlines. If these constraints are satisfied for a service, this service is considered as executed service. As shown, SPP-AOA succeeds in performing a greater number of services compared to other algorithms in most periods. In all 10 time periods, out of 615 available services, SPP-AOA has succeeded in placing 561 services. These results for SPP-PSO, CSA-FSPP and FSP-ODMA are 549, 536 and 507, respectively.

Each received service must wait until the next time period for placement. In addition, there are times related to placement planning as well as various delays. In the sixth experiment, we deal with the average waiting time in each time period. Because SPP-AOA has achieved promising results in the placement process, it is expected to provide a lower average waiting time for services. Fig. 13 shows the results of this comparison for different algorithms at the end of each time period. Here, for each service the total waiting time is calculated and then for each algorithm the average for all executed services is reported. As illustrated, SPP-AOA has been able to reduce service waiting times over all time

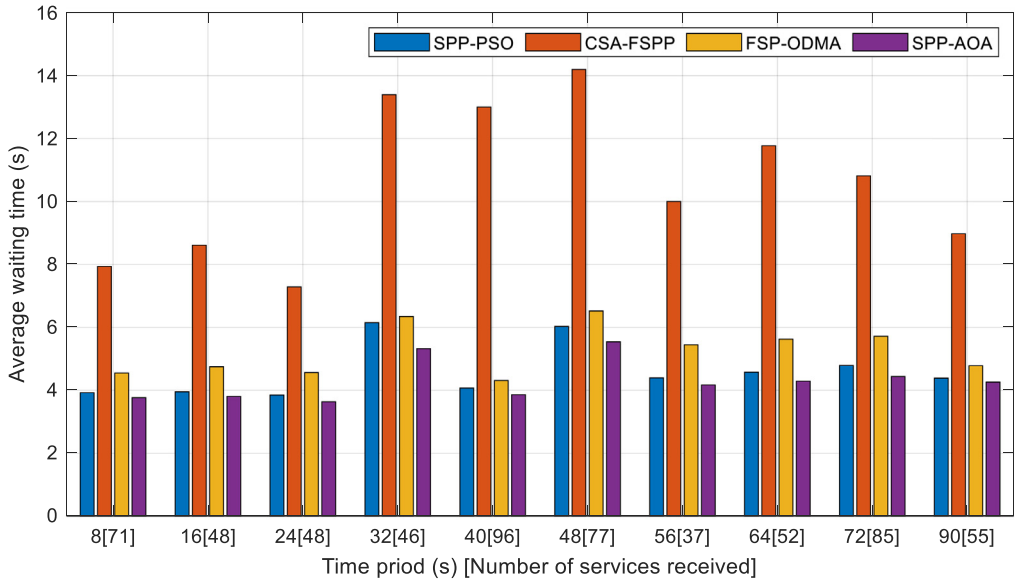

Fig. 13. Results for the average waiting time for services executed in each time period.

Table 8
Results of the number of remaining services and the number of failed services in different algorithms.

Time period	Algorithms	Number of services received	Number of services executed	Number of services failed	Number of remaining services
$\tau = 1$	SPP-PSO	71	66	0	5
	CSA-FSPP	71	63	0	8
	FSP-ODMA	71	64	0	7
	SPP-AOA	71	68	0	3
$\tau = 10$	SPP-PSO	615	549	57	9
	CSA-FSPP	615	536	62	17
	FSP-ODMA	615	507	89	19
	SPP-AOA	615	561	47	7

periods. The average waiting time based on 561 services executed by SPP-AOA after the end of 10 time periods was 43.1 s. These results are 46.1 s, 105.9 s and 52.6 s for SPP-PSO, CSA-FSPP and FSP-ODMA algorithms, respectively. The poor performance of CSA-FSPP is clearly evident in the average waiting time metric. The reason for this is the temporal complexity of CSA in the evolution process. Overall, the superiority of SPP-AOA compared to SPP-PSO, CSA-FSPP and FSP-ODMA was reported to be 7.01 %, 145.8 % and 22.06 %, respectively.

In the last experiment, the number of remaining services and the number of services failed were compared to the total number of services received for each algorithm. The proposed SPP-AOA scheme has managed to execute 561 services out of 615 services received. Also, the SPP-PSO, CSA-FSPP and FSP-ODMA algorithms report 549, 536 and 507 services executed, respectively. Of the 54 services not executed in SPP-AOA, 7 services were not placed due to lack of resources, while still having deadlines. Therefore, these services can be executed in the next time period. However, the deployment of 47 services in SPP-AOA has failed. In fact, the resources required for these services were not provided before the deadline. The results related to the number of remaining services and the number of services failed for other algorithms are shown in Table 8. Here, the results are reported after the end of time periods $\tau = 1$ and also $\tau = 10$. The results show the superiority of the proposed scheme, because it has succeeded in reducing the number of remaining services and the number of services failed. The results of the algorithms for $\tau = 1$ show that there is no service failed. In fact, the resources required by the services are abundant at the beginning of the simulation, and there is no deadline violation. However, in later time periods due to lack of resources some services may not be able to execute before their deadline.

Considering all performance metrics, SPP-AOA is superior to other compared algorithms. After that, SPP-PSO, CSA-FSPP and FSP-ODMA are in the next ranks of the best, respectively. The SPP-AOA and SPP-PSO algorithms have the least deadline violations, but SPP-PSO consumes more cloud resources than SPP-AOA. The utilization of cloud resources leads to increased delay and service waiting time, which is due to the long distance from the cloud to the data source. In addition, SPP-PSO reports higher service costs compared to SPP-AOA, which is due to the higher cost of consuming cloud resources. On average, SPP-AOA has the best results and reports 9.7 %, 21.6 % and 17.4 % superiority over SPP-PSO, CSA-FSPP and FSP-ODMA, respectively. Meanwhile, we model the fog landscape as multiple non-overlapping domains and perform the deployment process on each domain independently. However, we consider communication between domains for better deployment management. Therefore, this architecture has the ability to support large-scale networks.

7. Conclusion

Most IoT-based applications have distributed components called service, and how they are deployed on fog nodes is one of the most important resources management challenges in fog computing. In general, SPP in fog computing is defined as the efficient deployment of IoT services on fog nodes with considering some QoS requirements such as cost, delay, and throughput. This paper proposes an efficient solution for SPP using metaheuristic-based approaches and considering the concept of autonomous planning model. The proposed autonomous planning model is developed through a MADE-k-based conceptual framework, where decision-making on service placement is done using AOA as a metaheuristic approach. The proposed AOA models the SPP as a multi-objective problem by reconciling different objectives (i.e., resource utilization, service cost, energy consumption, delay cost and throughput), where configured with a shared parallel architecture. Due to the various factors used in the objective function, the proposed scheme improves the overall performance of the system through cloud-fog cooperation and increases QoS satisfaction. In addition, the proposed scheme has been able to save resources and accept more requests by deploying distributed resources. Extensive simulations on a synthetic fog environment based on various criteria have proven the superiority of the proposed scheme compared to other metaheuristic approaches. On average, the superiority of the proposed scheme compared to SPP-PSO, CSA-FSPP and FSP-ODMA algorithms is reported to be 9.7 %, 21.6 % and 17.4 %, respectively. Because SPP solving through metaheuristic approaches leads to postponing requests to the next time gap, the problem-solving ability can be flexed with Deep Reinforcement Learning (DRL) approaches. As a future work, we intend to consider DRL with a long-term cumulative reward policy to improve the solution of SPP.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author Contribution

All authors contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Natural Science Foundation of Jiangsu Province under Grant BK20180209. Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 18KJD520004. Research Funds of Suzhou Vocational Institute of Industrial Technology 2019kyqd018.

References

Ali, S., Pandey, M., Tyagi, N., 2020. Wireless-Fog Mesh: A framework for in-network computing of microservices in semipermanent smart environments. Int. J. Netw. Manag. 30 (6), e2125.

Ali, S., Pandey, M., Tyagi, N., 2022. SDFog-Mesh: A software-defined fog computing architecture over wireless mesh networks for semi-permanent smart environments. Comput. Netw. 211, 108985.

Ayoubi, M., Ramezanpour, M., Khorsand, R., 2021. An autonomous IoT service placement methodology in fog computing. Software: Practice Exp. 51 (5), 1097–1120.

Azad, F.A., Rad, S.A., Arashpour, M., 2022. Back-stepping control of delta parallel robots with smart dynamic model selection for construction applications. Autom. Constr. 137, 104211.

Azimirad, V., Ramezanlou, M.T., Sotubadi, S.V., Janabi-Sharifi, F., 2022. A consecutive hybrid spiking-convolutional (CHSC) neural controller for sequential decision making in robots. Neurocomputing 490, 319–336.

Azimzadeh, M., Rezaee, A., Jassbi, S.J., Esnaashari, M., 2022. Placement of IoT services in fog environment based on complex network features: a genetic-based approach. Clust. Comput. 25 (5), 1–23.

Baranwal, G., Vidyarthi, D.P., 2021. FONS: a fog orchestrator node selection model to improve application placement in fog computing. J. Supercomput. 77 (9), 10562–10589.

Berahmand, K., Nasiri, E., Li, Y., 2021. Spectral clustering on protein-protein interaction networks via constructing affinity matrix using attributed graph embedding. Comput. Biol. Med. 138, 104933.

Cao, C., Wang, J., Kwok, D., Cui, F., Zhang, Z., Zhao, D., Zou, Q., 2022. webTWAS: a resource for disease candidate susceptibility genes identified by transcriptome-wide association study. Nucleic Acids Res. 50 (D1), D1123–D1130.

Cao, Z., Niu, B., Zong, G., Xu, N., 2023. Small-gain technique-based adaptive output constrained control design of switched networked nonlinear systems via event-triggered communications. Nonlinear Anal. Hybrid Syst 47, 101299.

Chen, Y., Li, Z., Yang, B., Nai, K., Li, K., 2020. A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing. Futur. Gener. Comput. Syst. 108, 273–287.

Cheng, F., Wang, H., Zhang, L., Ahmad, A.M., Xu, N., 2022. Decentralized adaptive neural two-bit-triggered control for nonstrict-feedback nonlinear systems with actuator failures. Neurocomputing 500, 856–867.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A., 2019. A survey on new generation metaheuristic algorithms. Comput. Ind. Eng. 137, 106040.

Farahbakhsh, F., Shahidinejad, A., Ghobaei-Arani, M., 2021. Multiuser context-aware computation offloading in mobile edge computing based on Bayesian learning automata. Trans. Emerg. Telecommun. Technol. 32 (1), e4127.

Ghobaei-Arani, M., Shahidinejad, A., 2022. A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment. Expert Syst. Appl. 200, 117012.

Hashim, F.A., Hussain, K., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W., 2021. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl. Intell. 51 (3), 1531–1551.

Hassan, H.O., Azizi, S., Shojafar, M., 2020. Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments. IET Commun. 14 (13), 2117–2129.

Ibrahim, A., Noshy, M., Ali, H.A., Badawy, M., 2020. PAPSO: A power-aware VM placement technique based on particle swarm optimization. IEEE Access 8, 81747–81764.

Jia, B., Hu, H., Zeng, Y., Xu, T., Yang, Y., 2018. Double-matching resource allocation strategy in fog computing networks based on cost efficiency. J. Commun. Networks 20 (3), 237–246.

Joyce, T., Herrmann, J.M., 2018. A review of no free lunch theorems, and their implications for metaheuristic optimisation. Nature-inspired Algorithms Appl. Optim. 744, 27–51.

Khosroabadi, F., Fotouhi-Ghazvini, F., Fotouhi, H., 2021. SCATTER: Service Placement in Real-Time Fog-Assisted IoT Networks. J. Sens. Actuator Netw. 10 (2), 26.

Li, P., Yang, M., Wu, Q., 2020. Confidence interval based distributionally robust real-time economic dispatch approach considering wind power accommodation risk. IEEE Trans. Sustainable Energy 12 (1), 58–69.

Li, Y., Wang, H., Zhao, X., Xu, N., 2022. Event-triggered adaptive tracking control for uncertain fractional-order nonstrict-feedback nonlinear systems via command filtering. Int. J. Robust Nonlinear Control 32 (14), 7987–8011.

Liu, C., Wang, J., Zhou, L., Rezaeipanah, A., 2022. Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. Neural Process. Lett. 54 (3), 1823–1854.

Liu, Z., Zheng, Z., Sudhoff, S.D., Gu, C., Li, Y., 2015. Reduction of common-mode voltage in multiphase two-level inverters using SPWM with phase-shifted carriers. IEEE Trans. Power Electron. 31 (9), 6631–6645.

Mohaidat, M., Grantner, J.L., Shebrain, S.A., Abdel-Qader, I., 2022. In: May). Instrument detection for the intracorporeal suturing task in the laparoscopic box trainer using single-stage object detectors. IEEE, pp. 455–460.

Mojarad, M., Sarhangnia, F., Rezaeipanah, A., Parvin, H., Nejatian, S., 2021. Modeling hereditary disease behavior using an innovative similarity criterion and ensemble clustering. Curr. Bioinform. 16 (5), 749–764.

Murtaza, F., Akhunzada, A., ul Islam, S., Boudjadar, J., Buyya, R., 2020. QoS-aware service provisioning in fog computing. J. Netw. Comput. Appl. 165, 102674.

Nasiri, E., Berahmand, K., Samei, Z., Li, Y., 2022. Impact of centrality measures on the common neighbors in link prediction for multiplex networks. Big Data 10 (2), 138–150.

Natesha, B.V., Guddeti, R.M.R., 2021. Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment. J. Netw. Comput. Appl. 178, 102972.

Rezaeipanah, A., Matoori, S.S., Ahmadi, G., 2021. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Appl. Intell. 51 (1), 467–492.
Salimian, M., Ghobaei-Arani, M., Shahidinejad, A., 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.
Salimian, M., Ghobaei-Arani, M., Shahidinejad, A., 2022. An Evolutionary Multi-objective Optimization Technique to Deploy the IoT Services in Fog-enabled Networks: An Autonomous Approach. Appl. Artif. Intell. 36 (1), e2008149.
Sami, H., Mourad, A., 2020. Dynamic on-demand fog formation offering on-the-fly IoT service deployment. IEEE Trans. Netw. Serv. Manag. 17 (2), 1026–1039.
Shahidinejad, A., Ghobaei-Arani, M., Souri, A., Shojafar, M., Kumari, S., 2021. Light-edge: a lightweight authentication protocol for IoT devices in an edge-cloud environment. IEEE Consum. Electron. Mag. 11 (2), 57–63.
Shakarami, A., Shakarami, H., Ghobaei-Arani, M., Nikougoftar, E., Faraji-Mehmandar, M., 2022. Resource provisioning in edge/fog computing: A Comprehensive and Systematic Review. J. Syst. Archit. 122, 102362.
Si, Z., Yang, M., Yu, Y., Ding, T., 2021. Photovoltaic power forecast based on satellite images considering effects of solar position. Appl. Energy 302, 117514.
Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., Leitner, P., 2017. Optimized IoT service placement in the fog. SOCA 11 (4), 427–443.
Slabicki, M., Grochla, K., 2016. Performance evaluation of CoAP, SNMP and NETCONF protocols in fog computing architecture. In: NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1315–1319.
Tan, J., Liu, L., Li, F., Chen, Z., Chen, G.Y., Fang, F., Zhou, X., 2022. Screening of endocrine disrupting potential of surface waters via an affinity-based biosensor in a rural community in the Yellow River Basin, China. *Environ. Sci. Technol.* 56 (20), 14350–14360.
Tang, F., Niu, B., Zong, G., Zhao, X., Xu, N., 2022. Periodic event-triggered adaptive tracking control design for nonlinear discrete-time systems via reinforcement learning. Neural Netw. 154, 43–55.
Trik, M., Akhavan, H., Bidgoli, A.M., Molk, A.M.N.G., Vashani, H., Mozaffari, S.P., 2023. A new adaptive selection strategy for reducing latency in networks on chip. Integration 89, 9–24.
Vashani, H., Sullivan, J., El Asmar, M., 2016. DB 2020: Analyzing and forecasting design-build market trends. J. Constr. Eng. Manag. 142 (6), 04016008.
Xavier, T.C., Santos, I.L., Delicato, F.C., Pires, P.F., Alves, M.P., Calmon, T.S., Amorim, C. L., 2020. Collaborative resource allocation for Cloud of Things systems. J. Netw. Comput. Appl. 159, 102592.
Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H.C., Jue, J.P., 2019. FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework. IEEE Internet Things J. 6 (3), 5080–5096.
Zhang, D., Haider, F., St-Hilaire, M., Makaya, C., 2019. Model and algorithms for the planning of fog computing networks. IEEE Internet Things J. 6 (2), 3873–3884.
Zhang, H., Zhao, X., Zhang, L., Niu, B., Zong, G., Xu, N., 2022a. Observer-based adaptive fuzzy hierarchical sliding mode control of uncertain under-actuated switched nonlinear systems with input quantization. Int. J. Robust Nonlinear Control 32 (14), 8163–8185.
Zhang, H., Zou, Q., Ju, Y., Song, C., Chen, D., 2022b. Distance-based support vector machine to predict DNA N6-methyladenine modification. Curr. Bioinform. 17 (5), 473–482.
Zhang, Y., Zhang, F., Tong, S., Rezaeipanah, A., 2022c. A dynamic planning model for deploying service functions chain in fog-cloud computing. J. King Saud Univ.-Computer Information Sci. 34 (10), 7948–7960.
Zhao, D., Zou, Q., Boshkani Zadeh, M., 2022. A QoS-Aware IoT Service Placement Mechanism in Fog Computing Based on Open-Source Development Model. J. Grid Comput. 20 (2), 1–29.