Peer-to-Peer Networking and Applications (2022) 15:2326–2346
https://doi.org/10.1007/s12083-022-01349-w

A Mixed-integer programming model using particle swarm optimization algorithm for resource discovery in the cloudiot

Parisa Goudarzi[1] · Amir Masoud Rahmani[1,2] · Mohammad Mosleh[1]

Received: 14 December 2021 / Accepted: 22 June 2022 / Published online: 15 July 2022

Abstract
Cloud computing and the Internet of Things (IoT) are new platforms in the information and communication technology revolution. Selecting Cloud of Things (CloudIoT) in applications with fixed and mobile resources can provide many opportunities in different technologies, such as healthcare and transportation. Discovering fixed and mobile resources are one of the main concerns of the CloudIoT paradigm that requires a proper discovery mechanism. This paper proposes a mathematical optimization model to minimize response time, cost, and bandwidth of CloudIoT platforms by considering fixed and mobile resources in resource discovery. Moreover, a heuristic Single Resource Discovery algorithm is presented based on a Mathematical optimization model (SRDM). Furthermore, a heuristic Multi Resource Discovery algorithm is introduced based on a Mathematical optimization model (MRDM). In addition, this paper employs Particle Swarm Optimization (PSO) and Multi-Objective Particle Swarm Optimization (MOPSO) to solve the optimization problem. Finally, according to the simulation results, the proposed MOPSO-based algorithm significantly reduces the latency and improves the success ratio and availability compared to other algorithms.

Keywords CloudIoT · Resource discovery · Mobility · PSO algorithm · Goal programming · Linear programming

1 Introduction

The development of practical solutions has created new Cloud of Things (CloudIoT) patterns in real environments. This model uses cloud computing operating systems with infinite storing and processing capabilities to introduce a new and promising solution for Internet of Things (IoT) systems. These systems consist of devices with limited resources, which are not powerful enough to process and store complex tasks on data-generating devices. Therefore, the combination of cloud computing and IoT provides a framework for users to use the offered services by different providers in each location [1–3]. A trading market provides computing resources for a set of computing resource buyers with a group of computational resource sellers. CloudIoT can be used in various fields, such as smart healthcare systems/hospitals and smart transportation systems [4–6].

The decentralized processing of data in IoT devices with cloud technology has led to a new computing method to reduce communication overhead and data transfer time, which, in turn, has led to a promising trend in fog computing. In fog computing, services, data, computing power, and decision-making are distributed, so unbearable delays and long response times can be prevented. Furthermore, fog computing prevents unpredictable connections to the cloud by blindly sending IoT data for processing and storing in the cloud and resending it to the users. Integrating cloud, fog, and IoT resources into a single architecture has created a reliable platform called IFCIoT. Some of the benefits of this platform for future IoT applications are better performance, faster response time, scalability, and higher accuracy [7, 8].

Fog computing faces challenges such as node mobility and keeping resources permanently available. The status of fog nodes changes considerably because of various

✉ Amir Masoud Rahmani
rahmania@yuntech.edu.tw

Parisa Goudarzi
goudarzi_p@yahoo.com

Mohammad Mosleh
mosleh@iaud.ac.ir

[1] Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran

[2] Future Technology Research Center, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan

parameters, such as broken wireless access links and battery life limitations. For this reason, resource management in fog computing must be able to manage mobility [9]. Another important issue is different owners' ownership of fog nodes [10]. Additionally, a sudden increase in workload reduces resources for the fog, and this causes unbearable additional delays in the execution of the tasks [11]. Applications or data may require a high processing speed. In addition, the lack of bandwidth to process information and the expensive bandwidth needed for sending information to a data center or cloud is the issues that fog computing needs to manage [12, 13].

The efficient use of resources to minimize response time, cost, and energy consumption is an important optimization problem in cloud computing and IoT environments. The mobile devices that are connected to edge servers have a common communication network. Guo et al. [14] presented a high-resistance offloading algorithm in the cloud computing environment, which minimized the average response time for offloading strategy and determining the common bandwidth. Abdel-Basset et al. [15] proposed a bandwidth-based Virtual Machine (VM) allocation algorithm using the whale optimization algorithm and an algorithm for merging VMs. The latter algorithm was based on energy improvement and cost-awareness by using genetic algorithms to minimize the number of active physical servers. Finally, an algorithm for dynamic and safe loading was introduced by Alli and Alam [16] to reduce the latency and energy consumption in the fog computing environment using machine learning methods. Their proposed algorithm used the Particle Swarm Optimization (PSO) algorithm to select the optimal node for dynamic loading at the IoT level and the Reinforcement Learning (RL) to choose the appropriate cloud at the fog level.

This study aims to discover IoT and cloud resources with three objectives. Due to many requests in the cloudIoT environment, using three dimensions is very important. The first dimension deals with responding to requests that must be handled within a reasonable response time. The second dimension considers the reasonable cost of using resources. The third dimension necessitates the rational distribution of bandwidth among requests. We use Mixed-Integer Linear Programming (MILP) for the three dimensions and consider the proposed method in single-objective and multi-objective modes.

The use of evolutionary algorithms, such as PSO, to present, Single Resource Discovery algorithm based on a Mathematical optimization model (SRDM) is discussed as follows. The PSO algorithm has fewer parameters, easier implementation, and higher flexibility than other evolutionary algorithms, so we use PSO to implement the SRDM algorithm. In addition, we use Multi-Objective PSO (MOPSO) to present Multi Resource Discovery algorithm based on a Mathematical optimization model (MRDM). The main reason for using the MOPSO algorithm instead of other multi-objective algorithms is that MOPSO uses the actual values of the variables as members of the population and reduces the computational burden. The most important innovations of this study are as follows:

- Combining PSO and MILP to present a new mathematical model to optimize response time, cost, and bandwidth so that a PSO-based meta-heuristic algorithm is introduced for resource discovery
- Providing a multi-objective approach to the proposed method using the MOPSO algorithm, which uses the actual values of variables as the members of the population and reduces the computational burden

We simulate the proposed algorithm using the IFogSim simulator. Then, the SRDM algorithm and MRDM algorithm are compared with the algorithms presented in [29, 30, 35], and [36].

The remaining of this paper includes the following sections. Section 2 reviews the related works. The proposed method is analyzed in Sect. 3. In Sect. 4, the results are discussed, and simulations are examined. The conclusion is presented in the last section.

2 Related work

The related work section is divided into 1) PSO algorithm, 2) MOPSO algorithm, and 3) resource discovery and allocation algorithms.

2.1 PSO Algorithm

PSO is one of the most important evolutionary optimization methods to solve complex optimization problems. The main part of the PSO is the particles' initialization. In PSO, the system is initialized with many random solutions. The solutions are called particles with an assigned random velocity and position. Each particle calculates the value of an objective function according to its position in a multi-dimensional space. Moreover, each particle adjusts its velocity and position according to each generation's best position and the best global population [17]. The velocity and position of particles can be obtained using Eqs. (1) and (2).

$$v_p(k+1) = w \times v_p(k) + c_1 \times rand_1 \times \left(pbest - x_p(k)\right) + c_2 \times rand_2 \times \left(gbest - x_p(k)\right) \tag{1}$$

$$x_p(k+1) = x_p(k) + v_p(k+1) \tag{2}$$

In these relationships, $v_p(k+1)$ and $v_p(k)$ denote the current and previous velocities of particle p; $x_p(k+1)$ and $x_p(k)$

are the current and previous positions of particle p. The two acceleration coefficients c_1 and c_2 and the two random numbers $rand_1$ and $rand_2$ (which are between 0 and 1) are used to calculate the velocity. The best position of particle p and the best particle position in the population are represented by $pbest$ and $gbest$, respectively; w represents the inertia weight; the other two variables, $N - pop$ and $Maxiter$, denote the population size and the maximum number of iterations [18].

2.2 MOPSO Algorithm

The MOPSO algorithm is a generalization of the PSO algorithm used to solve multi-objective problems. In MOPSO, a concept called archive or repository, also known as the hall of fame, has been added to the PSO algorithm. Before moving, particles select a member of the repository as a leader. This leader must be a member of the repository and dominant. The members of the repository represent the Pareto front and contain dominant particles. Therefore, instead of $gbest$, one of the repository members is selected. There is no repository in PSO because there is only one objective in it, and there is a particle that is the best.

Particles have two parts: position and velocity. Particles are updated via velocity vectors differently from the genetic algorithm. There are two leaders: choosing the global best solution ($gbest$) and the personal best memory ($pbest$) [19]. The velocity and position of particle i are updated as follows:

$$v_{i,k+1} = w \times v_{i,k} + c_1 \times r_1 \times \left(x_{ib} - x_{i,k}\right) + c_2 \times r_2 \times (x_{gb} - x_{i,k}) \quad (3)$$

$$x_{i,k+1} = x_{i,k} + v_{i,k+1} \quad (4)$$

where $i = 1, 2, \ldots, N$ is the population size, k is the particle iteration index, and w is the inertia weight with a linear decrease from 0.9 to 0.4 as the particles are updated. Thus, local search c_1 and c_2 are two acceleration coefficients; r_1 and r_2 are two uniformly distributed random numbers in the range $[0, 1]$. Moreover, xib is the best position for the ith particle, and xgb is the best position in the whole swarm [20].

2.3 Resource discovery and allocation algorithms

The main purpose of this study is to provide a meta-heuristic algorithm for resource discovery by reducing response time, cost and bandwidth. For this reason, it was necessary to examine further the meta-heuristic algorithms for discovering and allocating resources in the related work section. Consequently, the meta-heuristic algorithm was investigated by reducing several objectives and presenting various mathematical models for resource discovery. In addition, to present the algorithm, we studied some papers that used different mathematical theories.

Resource management is an essential element in fog computing environments. Fog computing provides short response times through a virtual intermediate layer to procure latency-sensitive real-time programs in an IoT infrastructure. Furthermore, it enables data calculation, storage, and network services between data centers of the cloud and end-users. Therefore, resource management is an essential element in fog computing environments. Gill et al. [21] presented a PSO-based resource management approach to optimize network bandwidth, response time, synchronized delays, and energy consumption simultaneously. This approach was suggested for managing fog computing resources in smart homes. The results showed that this approach reduces bandwidth, latency, and energy consumption.

Bharti et al. [22] presented a method for discovering resources called the Iterative K-Means Clustering Algorithm (IKM-CA), which grouped the textual information of clusters for efficient search using similarity coefficients of a vector space model. By using metadata to identify an object, this method makes it possible to calculate and access resources. The IKM-CA consists of three stages of cluster formation, repetitive K-means clustering, and discovering matching conditions. In general, this algorithm replicates the formation of clusters repetitively to search for resources using the matching criteria. These criteria emphasize the relationship between the two points considering the threshold value.

Resource discovery is a complex and challenging problem requiring an effective algorithm for optimal performance. Ezugwu and Adewumi [23] introduced an optimization algorithm called Soft-Set SymbIoTic Organism's Search (SSSOS) to simulate selecting resources for effective planning in a cloud computing environment. This algorithm searches and selects the best resources using the techniques available in both symbIoTic organisms search optimization and soft set attribute reduction theory. The SSSOS algorithm provides users with high-quality services using tracking, matching, and easy selection of information configuration. The algorithm consists of three steps: 1) reducing resource-dependent properties using a software feature reduction algorithm, 2) searching and matching the candidate resources from the first stage output, and 3) selecting the best source from the first and second stages.

Scheduling user tasks in VMs and data centers is a challenging issue due to many users. Accordingly, Panwar et al. [24] introduced a hybrid algorithm to solve task scheduling problems using the PSO algorithm with a Technique of Order Preference by Similarity to Ideal Solution (TOPSIS). The proposed algorithm uses TOPSIS to calculate the optimized Fitness Value (FV). Then, the FV evaluated for each task is introduced to PSO to be optimized further. The main purpose of this algorithm is to connect the user task collection set to the distributed resources set for achieving some goals, such as minimizing transfer time, minimizing Makespan execution time, and maximizing resource utilization in the cloud computing environment.

The integration of Wireless Sensor Networks (WSNs) and IoT makes the cluster-head selection very complicated because it is necessary to consider the features of both IoT and WSN networks. To overcome fundamental limitations, such as low accuracy and slow convergence, Reddy and Babu [25] presented an algorithm called the Self-Adaptive Whale Optimization Algorithm (SAWOA) to achieve cluster head selection WSN-IoT networks. This algorithm considers the distance, energy, and delay of sensor nodes in WSN, temperature, and IoT devices load. The performance of SAWOA was compared with other cluster-head algorithms, and the results showed that SAWOA performs better than different algorithms.

Alzubi et al. [26] introduced a Location-assisted Delayless Service Discovery (LDSD) for processing IoT user requests. Depending on the location of the resources, LDSD classifies resources based on service delivery delay, availability for rapid resource mapping, and service responsiveness. However, on-time response in IoT is challenging because of access to heterogeneous self-adaptable resources. Therefore, replication and location errors were considered during resource discovery and mapping in this research. The primary goal of this algorithm is to improve resource access and minimize resource access costs.

A method of discovering and allocating resources was introduced by Kalaiselvi and Selvi [27]. Resource discovery and allocation algorithm are the two main parts of the proposed solution. First, the tasks are executed by the resource discovery approach. Then, the Multiple Kernel Fuzzy C-Means (MKFCM) clustering algorithm delivers the available resources. The Cloud provider selects the most cost-effective VM from the resources available to run the tasks. If the desired resource is unavailable, the provider will request it from the owner. Reducing the total cost over a period is the main objective of this approach.

Skarlat et al. [28] implemented a fog frame framework with the necessary communication mechanisms to run services in fog environments. Additionally, two heuristic algorithms were considered to place the service in fog (the first fit algorithm and the genetic algorithm). Considering the capacities of the available resources, the first fit algorithm discovers the right resource to deploy the service. Genetic algorithm chromosomes are defined using a vector in this algorithm. If the desired service can be applied to each device, it is expressed as one. Otherwise, it is expressed as zero. As a result, the genetic algorithm performs better than other algorithms for distributed requests and services.

An Elimination-Selection (E-S) algorithm was suggested by Nunes et al. [29] for searching and discovering resources in IoT environments. Their algorithm used the TOPSIS algorithm and quick sorting. However, quick sorting approaches have much complexity in time and storage, making them difficult to run. Consequently, these authors tried to take advantage of both approaches in their proposed approach; namely, they benefited from the speed of TOPSIS and the ability to select the best option by the sorting methods. One of the contrasting points of the E-S algorithm compared to the algorithm proposed in the present study is that the E-S algorithm is designed for IoT environments. Still, it does not address mobility, resource availability, and time constraints of tasks.

Md et al. [30] proposed a new approach for cloud services with ease of resource identification, dissemination, and discovery based on dynamic Quality of Service (QoS) features through the web Graphical User Interface (GUI) interface backed by a set of validation tests. The proposed approach consisted of three algorithms. First, they presented an efficient algorithm based on the QoS criterion given by cloud consumers using the decision tree classification algorithm to identify cloud services. Second, they introduced an algorithm for registering cloud service resources to enable Cloud Service Providers (CSPs) to register their services with their QoS features. Finally, they proposed an algorithm to find the appropriate cloud service and its features by Cloud Consumers (CCs).

One of the best-structured programs to run in a federated cloud is a Bag-Of-Tasks (BOT) application since it uses independent tasks. Additionally, the program's total cost can be affected by the policies of running programs in the federated cloud. Consequently, a mathematical planning model was proposed by Abadi et al. [31] to allocate resources in a federated hybrid cloud. The proposed model is binary linear programming that includes time constraints of tasks and resource limitations in federated clouds. The main purpose of this algorithm is to minimize the total cost of programs.

Kalantary et al. [32] used the hidden Markov chain learning method to address the challenges of searching and selecting resources for the combined IoT and fog computing. This method reduces latency and increases scalability. These authors implemented the proposed solution using Cloudsim simulator and compared the results with decision algorithms such as TOPSIS. They showed the superiority of their proposed method over other methods in terms of latency and scalability.

Bharti and Jindal [33] provided a framework for discovering optimal clustering-based resources in IoT, called Optimal Clustering-based Discovery Framework on IoT (OCDF-IoT). The proposed architecture can automatically discover resources and related services using an ontology, form/display knowledge about resources, and list resources based on maximum similarity and optimal selection of resources among candidates. The results from the real environment indicate that this architecture minimizes CPU power for query processing and increases CPU performance with less load on the server.

Xu et al. [34] examined confidential performance predictions for mobile IoT health care networks. The proposed solution is an improved Convolutional Neural Network (CNN) model that combines four convolution layers and a four-prong primary block. The four-prong primary block reduces CNN width by extracting parameters, extracting different sizes of health care data, and adapting to nonlinear health care data. According to the comparisons made, this algorithm performs 20% better than other methods in forecasting.

Human Resource Management (HRM) in federal cloud edge computing and selecting the optimal hardware and software resources to respond to the requests based on QoS factors in IoT environments are major challenges. As a result, Liu et al. [35] introduced an optimization model for the HRM problem in cloud edge computing using the WOA. The results showed that this model reduces response time and allocation costs and increases the number of allocated human resources in two different scenarios compared to other meta-heuristic algorithms.

Murturi and Dustdar [36] proposed a decentralized resource discovery mechanism with the ability to detect resources automatically in edge networks. The presented solution exchanges resource information in each domain with other domains by repeating resource descriptions peer-to-peer. In addition, this mechanism was proposed as a flat model that can better address the complexities of resource discovery and allow the organization of edge devices in clusters. These authors evaluated the prototype in a testbed consisting of low-power-based edge devices to validate the approach's feasibility.

The mentioned studies focus on reducing bandwidth, response time, mobility, and time constraints of tasks in cloud or IoT environments. However, at the same time, they do not consider resource discovery to cope with these challenges in cloud-based IoT. In this study, we try to address this shortcoming.

3 Proposed method

The proposed method is divided into three sub-sections, i.e., the proposed mathematical model, the proposed architecture, and a proposed algorithm. First, the proposed mathematical model is analyzed thoroughly. Then, the proposed architecture and its components are introduced in full detail. Finally, a proposed algorithm is presented for resource discovery.

3.1 Mathematical model

In this section, a MILP model is presented to pursue three goals of minimizing the response time, cost, and bandwidth to discover resources in CloudIoT platforms. Table 1 shows the symbols used in the mathematical model. We assume that the domains of fog nodes are independent. Therefore, there are several resources in each domain which process the received requests from IoT resources. Furthermore, some requests will be sent to the cloud without processing in the fog domains (Table 2). The main assumptions of the proposed model are as follows:

- The number of fog node domains is already specified.
- The cost of using each resource is determined in advance.
- Each task is assigned only to one resource.

The following is the proposed mathematical model.

Parameters z_1, z_2, z_3, are the optimization models of response time, cost, and bandwidth, which are presented in Eqs. (5), (6) and (7), respectively. In the present system, we define the time between sending a request and responding to the request as the response time, the duration of using resources as the cost, and the amount of transfer of requests by a network (wireless) connection or an interface the bandwidth. The total time of using the resources by tasks is considered the total cost. The goal is to maximize the fog bandwidth and minimize the system's cloud bandwidth. The main goal of the introduced system is to reduce response time, cost, and bandwidth for all requests.

A series of constraints presented in Eqs. (8) to (15) produces a possible domain. Equation (8) ensures that the sum of the time constraints of requests will be greater than or equal to the response time of the requests. Equation (9) states that the total number of requests is greater than or equal to the number of performed requests as the number of requests is unknown. Equation (10) states that the sum of the bandwidth of all domains fog nodes is larger than or equal to the sum of the bandwidth allocated to the requests in each domain. Equation (11) checks that the cloud bandwidth is greater than or equal to the sum of the bandwidth allocated to the requests to be performed in the cloud. Equation (12) ensures that each request is allocated to a maximum of one bandwidth from the *Rth* node of the fog due to the mobility of some resources. In addition, each request is allocated through a maximum of one fog node to the cloud bandwidth, and this is expressed by Eq. (13). The last two equations, (14) and (15), specify the decision variablesV^f_{rki}, V^c_{ji}, H^f_{ri} and H^c_i with the values that can only be zero or one.

Decision-making problems and the management of conflicting criteria are the solutions to the MILP model.

Table 1 Symbol definition for the MILP model

Symbols	Description
Indices	
i	Indicate of request $(i = 1, 2, \ldots, n)$
j	Indicate of cloud resource $(j = 1, 2, \ldots, m)$
k	Indicate of fog resource $(k = 1, 2, \ldots, g)$
r	Indicate the fog domain $(r = 1, 2, \ldots, q)$
l	Indicate of IoT resource $(l = 1, 2, \ldots, w)$
Parameters	
S_i^c	Number of cloud resource
S_k^f	Number of fog resource
A_r^f	Number of fog domain
S_i^l	Number of IoT resources
N	Indicate the number of requests
$Time_{rki}^f$	Time to execute request i on resource k of domain r of fog
$Time_{ji}^c$	Time to execute request i on resource j of cloud
$Time_{rki}^{If}$	Transfer time from IoT resource to fog
$Time_{rji}^{fc}$	Transfer time from node r fog to cloud
$Time_{rki}^{waitf}$	Waiting time request i on resource k of domain r of fog
$Time_{ji}^{waitc}$	Waiting time request i on resource j of cloud
C_{rki}^f	Execution price request i on resource k of domain r of fog
C_{ji}^c	Execution price request i on resource j of cloud
C_{rki}^{If}	Transfer price from IoT resource to fog
C_{ji}^{fc}	Transfer price from fog to cloud
M_{ri}^f	A measure of request i from domain r of fog
M_i^c	A measure of request i from the cloud
$Deadline_i$	Deadline of request i
$Bandwidth_i^c$	The bandwidth size for request i of cloud bandwidth
$Bandwidth_{ri}^f$	The bandwidth size for request i of node r fog bandwidth
$Bandwidth^c$	Cloud bandwidth
$Bandwidth_r^f$	Fog bandwidth of node r
Variables	
$V_{rki}^f = \begin{cases} 1, & \text{if request i excute on resource k of domain r fog} \\ 0, & \text{otherwise} \end{cases}$	
$V_{ji}^c = \begin{cases} 1, & \text{if request i excute on resource j of cloud} \\ 0, & \text{otherwise} \end{cases}$	
$H_{ji}^f = \begin{cases} 1, & \text{if fog bandwidth of node r allocates to request i} \\ 0, & \text{otherwise} \end{cases}$	
$H_i^c = \begin{cases} 1, & \text{if cloud bandwidth allocates to request i} \\ 0, & \text{otherwise} \end{cases}$	

The main challenge of solving multi-objective problems is determining the solution that simultaneously optimizes all objective functions. Given the contradiction between objective functions, it is usually challenging to find such a solution; therefore, the efficient multi-objective method is a method that balances functions well. One of the best ways to solve multi-objective problems is to turn them into single-objective problems, which Goal Programming (GP) is a good way to do. Still, the main limitation of GP is that it can only achieve levels of aspiration with numerical values. We use the Multi-Choice GP (MCGP) method [37]. The main advantage of the MCGP method is that it can be employed as a measurement tool for helping decision-makers make the best policies or use the most appropriate policies under their goals with the highest level of benefit.

Table 2 The MILP model

$Min\ z_1 = \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} Time_{rki}^{lf} + \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} Time_{rki}^{f} \times V_{rki}^{f} + \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} Time_{rki}^{waitf} + \sum_{i=1}^{n}\sum_{j=1}^{m} Time_{ji}^{waitc} + \sum_{i=1}^{n} M_i^c \div Bandwidth_i^c + \sum_{i=1}^{n}\sum_{j=1}^{m} Time_{ji}^{c} \times V_{ji}^{c}$		(5)
$Min\ z_2 = \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} C_{rki}^{lf} + \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} C_{rki}^{f} \times V_{rki}^{f} \times (\sum_{i=1}^{n}\sum_{r=1}^{q} M_{ri}^{f} \div Bandwidth_{ri}^{f}) + \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ji}^{fc} + \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ji}^{c} \times V_{ji}^{c} \times (\sum_{i=1}^{n} (M_i^c \div Bandwidth_i^c))$		(6)
$Min\ z_3 = (1 \div \sum_{i=1}^{n}\sum_{r=1}^{q} H_{ri}^{f} \times (\sum_{i=1}^{n}\sum_{r=1}^{q} M_{ri}^{f} \div \sum_{i=1}^{n}\sum_{k=1}^{g}\sum_{r=1}^{q} Time_{rki}^{lf}) + \sum_{i=1}^{n} H_i^c \times (\sum_{i=1}^{n} M_i^c \div \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{r=1}^{q} Time_{rji}^{fc}))$		(7)
$\sum_{k=1}^{g}\sum_{r=1}^{q} Time_{rki}^{f} \times V_{rki}^{f} + \sum_{j=1}^{m} Time_{ji}^{c} \times V_{ji}^{c} \le Deadline_i$	for $i \in \{1,2,\ldots,n\}$	(8)
$\sum_{k=1}^{g}\sum_{r=1}^{q} V_{rki}^{f} + \sum_{j=1}^{m} V_{ji}^{c} \le N$	for $i \in \{1,2,\ldots,n\}$	(9)
$\sum_{i=1}^{n} H_{ri}^{f} \times Bandwidth_{ri}^{f} \le Bandwidth_r^f$	for $r \in (1,2,\ldots,q\}$	(10)
$H_i^c \times Bandwidth_i^c \le Bandwidth^c$	for $i \in \{1,2,\ldots,n\}$	(11)
$\sum_{r=1}^{q} H_{ri}^{f} \le 1$	for $i \in \{1,2,\ldots,n\}$	(12)
$H_i^c \le 1$	for $i \in \{1,2,\ldots,n\}$	(13)
$V_{rki}^{f}, V_{ji}^{c} \in \{0,1\}$	for $i \in \{1,2,\ldots,n\}, k \in \{1,2,\ldots,g\}$, $r \in \{1,2,\ldots,q\}$, $j \in \{1,2,\ldots,m\}$	(14)
$H_{ri}^{f}, H_i^c \in \{0,1\}$	for $i \in \{1,2,\ldots,n\}$, $r \in \{1,2,\ldots,q\}$	(15)

Moreover, these features improve the practical application of MCGP for solving decision/management problems in the real world. The MCGP solution with a utility function is used to formulate resource discovery in CloudIoT to minimize response time, cost, and bandwidth in Table 3. Consequently, a multi-objective MCGP with the utility function approach is used to solve the problem. The fitness function determines the problem by considering three objectives (response time, cost, and bandwidth). Simultaneous focus on these three objectives leads to a reasonable solution, so we used Eqs. (16)–(23) to define the fitness function in the proposed algorithm. These equations transform the multi-objective problem into a single-objective one.

In these relationships,$U_{1,min}$, $U_{1,min}$ are the *kth* level of the goal of y_k is a continuous variable. d_k^+, d_k^- are positive and negative deviations of$f_x(k)$. δ_k^- is the normal deviation ofy_k, w_d^k is the weight of(d_k^+, d_k^-), w_k is the weight of f_k^-. In Eq. (16), $\beta_1^d\left(d_1^+ + d_1^-\right)$ represents the response time, $\beta_2^d\left(d_2^+ + d_2^-\right)$ represents the cost, and $\beta_3^d\left(d_3^+ + d_3^- + e_1^+ + e_1^-\right)$ represents the bandwidth. Due to the known bandwidth of fog and cloud, it is needed to define $e_1^+ + e_1^-$ in Eq. (16). Equations (16) to (23) express the limits, and the reasonable solution of the fitness function is determined considering the results of these equations. The total value of β is equal to 1. For obtaining equal values, 1 is divided by 3, so the value of each β is equal to 0.33.

Table 3 MCGP model

$Min\beta_1^d\left(d_1^+ + d_1^-\right) + \beta_2^d\left(d_2^+ + d_2^-\right) + \beta_3^d\left(d_3^+ + d_3^- + e_1^+ + e_1^-\right) + \beta_1^d\delta_1^- + \beta_2^-\delta_2^-,$		(16)
$\lambda_i \le \frac{U_{i,max}-y_1}{U_{1,max}-U_{1,min}}$	fori = 1, 2	(17)
$f_i + d_1^- - d_1^+ = y_1,$	fori = 1, 2	(18)
$f_3 - e_1^+ + e_1^- = y_3,$		(19)
$\lambda_i + \delta_1^- = 1,$	fori = 1, 2	(20)
$U_{i,min} \le y_1 \le U_{i,max},$	fori = 1, 2	(21)
$Bandwidth\ fog \le y_3 \le Bandwidth\ cloud,$		(22)
$Systems\ constraints(3-13),$		(23)

3.2 The proposed architecture

The proposed three-layer architecture is shown in Fig. 1. In this architecture, the layers are examined separately. In addition, it shows the components needed to implement the proposed algorithm. Our primary purpose in presenting this section is to offer an overview of the proposed system and the components required to implement the proposed algorithm in the IaaS service model. Each layer is described in the following.

- User devices/IoT, such as smartphones and tablets, are the lowest layer. Requests of these devices are sent to the top layer to run.

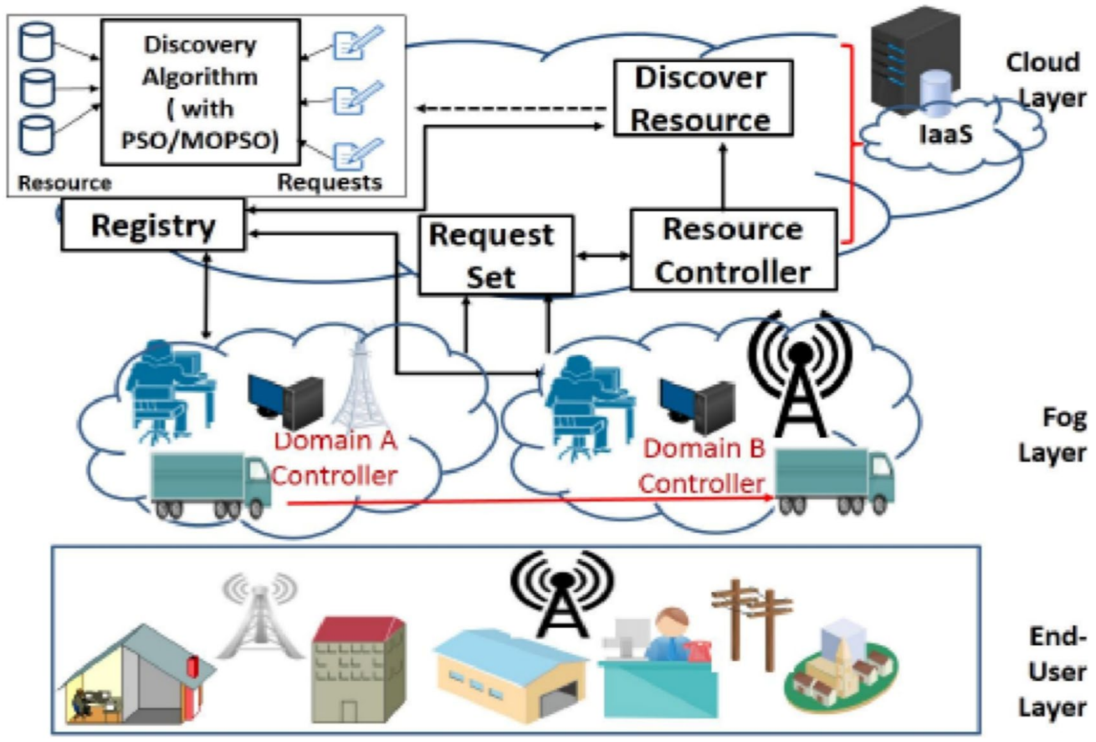


Fig. 1 Proposed architecture

- Several independent domains based on geographical areas with limited network connection make the middle layer fog computing [38]. A fog controller is specified for each domain. Fog nodes, which include smart devices, such as edge routers and switches, are capable of computing, data storing, and networking. Lightweight containers such as LXC [39], and Docker [40] are promising approaches to performing tasks in fog nodes of devices with limited resources. As a result, each fog node in the bed container uses lightweight containers to manage its containers locally [41].
- Long requests and storage are completed in the top layer (cloud computing layer).

The requests are sent by the lower layer devices and processed in the middle layer to be placed in a specific template format. The template format includes the optimization criteria (such as cost and response time) and a set of constraints (such as deadlines for each task execution time and required storage space). Requests are divided into two groups. The first group can be executed in the fog nodes, while the second group should be sent to the top layer. The controller in each node creates a suitable resource discovery strategy to execute user requests. In addition, the controller selects the best resource for requests based on the end-users, fog node locations, and resources in the nodes. Requests for the transfer to another fog domain will be sent to the node container of that domain. Immigration is done by the container platforms embedded in each fog node.

The cloud controller must optimize resources based on user requirements by placing requests in a queue. The resource controller uses an optimization model to minimize response time, cost, and bandwidth. The output of the resource controller is delivered in the form of a plan to identify the best resource for the resource management component. The resource management component contains fog and cloud resources.

Resource discovery can be implemented by a software platform, such as OpenStack [42], OpenNebula [43], and Eucalyptus [44]. The discovery component uses the PSO/MOPSO algorithm to discover the most suitable resource, considering response time, cost, and bandwidth criteria.

Resources in the fog nodes send their information to the registry via messages. New resource information can be used in the next step of decision-making. Moreover, unavailable resource information will be sent to the registry via the corresponding nodes to update information about the resources. The registry needs to be periodically updated with information from different providers and resources to define a lifetime. Resources and providers that are out of reach do not participate in the decision-making process in the next step.

Fig. 2 The structure of a particle with a sample allocation

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Fog(F)	2	10	4	7	4	8	16	2	9	2
Cloud(C)	1	5	12	2	6	11	5	4	19	16

3.3 The proposed algorithm

Our goal was to use the meta-heuristic algorithm in a new dimension. The combination of cloud technology and IoT is a new technology, so we used a meta-heuristic algorithm to discover resources in cloudIoT. There are several methods to solve resource discovery problems in cloudIoT, one of which is meta-heuristic algorithms. Meta-heuristic algorithms are approximate optimization algorithms with solutions that can exit local optimal points and are used in many problems. For this reason, we used meta-heuristic algorithms. This subsection examines the proposed method based on the two algorithms, PSO and MOPSO. In order to use PSO, as mentioned earlier, the proposed method was transformed into a single-objective problem using MCGP.

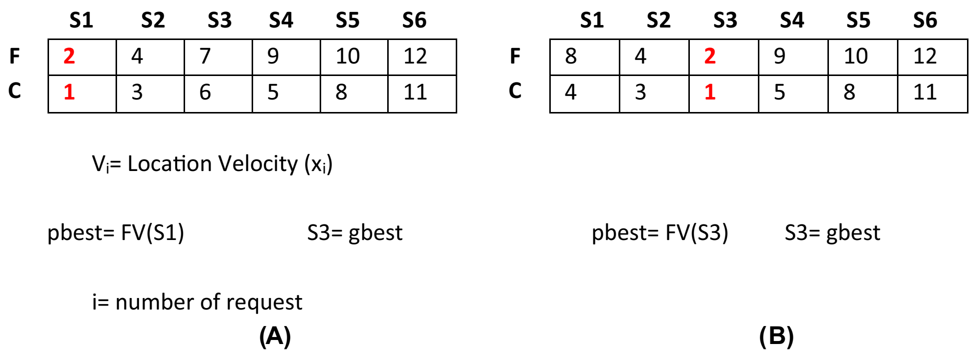
3.3.1 SRDM algorithm

The PSO algorithm makes it possible to select resources based on the defined criteria while the amount of search in resource discovery is reduced. This is done by increasing the number of resource requests and evaluating them. We implemented the SRDM, a fitness function defined in the algorithm using Eq. (16) and the constraints applied in Eqs. (17)–(23). The function defined according to these relations discovers the most reasonable resource for a request. Using the PSO algorithm includes features such as rapid convergence of PSO compared to other meta-heuristic methods, the proper weighting of quantitative and qualitative elements, and presenting a comprehensive analysis. We define a $2 * m$ matrix to construct the proposed algorithm so that the rows and columns represent the cloud/fog computing and resources, respectively. Each element specifies the number of requests. Accepted requests in fog nodes and cloud are placed in a separate queue with a different assigned number. As Fig. 2 shows, some requests will not be executed due to their expiration. Moreover, some of them require multiple resources to run.

Each particle (which represents a resource in the proposed method) at any given time is affected by the best position and *pbest* position in the search space. FV is used to evaluate each particle. Particles are randomly selected. *pbest* is the best particle result (FV) ever obtained by a particle, and *gbest* is the best particle in the search space. The obtained *gbest* is compared with the FV for each particle. If it is smaller, *gbest* will be updated. Otherwise, the best resource will be marked with the largest *gbest*, and the request will be assigned to that resource. This process significantly reduces response time, cost, and bandwidth for each request. The performance of PSO is shown in Algorithm 1. Additionally, Fig. 3 shows the algorithm.

Algorithm1: resource discovery with PSO

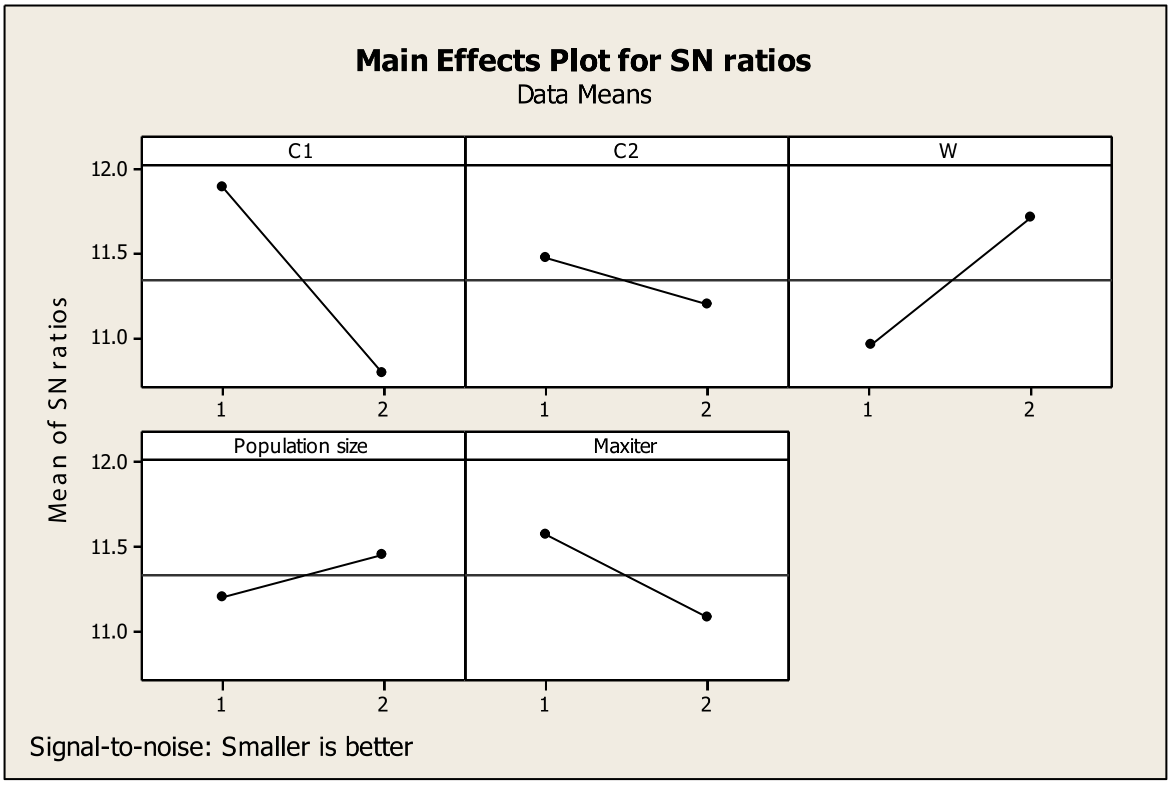
```
Input:   User Request List= { $n_1, n_2,…,n_n$} and List of Resource= { {$R_{I1},…,R_{Im}$}, {$R_{F1},…,R_{Fh}$}, {$R_{C1},…,R_{Cq}$}}
Output:  mapping between user request and list of resource
Begin
 Set particle dimension ← number of tasks in the list
 Initialize randomly (velocity $V_i$ location $X_i$) with pbest and gbest
 For i ←1 to n do
    Xpbest$_i$ ← $x_i$
    $V_i$ ← $x_i$
    Fitness value ← f($x_i$) /* Calculate the fitness using mathematical equations (16- 23) */
    Pbest$_i$ ← fitness
    If fitness$_i$ < gbest  then
        Xgbest ← $x_i$
        gbest ← fitness$_i$
        end if
  end for
 While ( t < periods )
  For i ←1 to n do
    $V_i \leftarrow wv_i + c_1 r_1(x_{pbesti} - x_i) + c_2 r_2 (x_{gbest} - x_i)$ and update $x_i$
    Check if current fitness value < pbest
        $X_{pbesti}$ ← $x_i$ and pbest$_i$ ← fitness$_i$
      else
          Fitness$_i$ < gbest  then
          $X_{gbest}$ ← $x_i$ and gbest ← fitness$_i$
     End if
    Assign the optimal resource to the particle with the highest gbest
    Assign t to resource
    t++
  end for
 end while
end
```



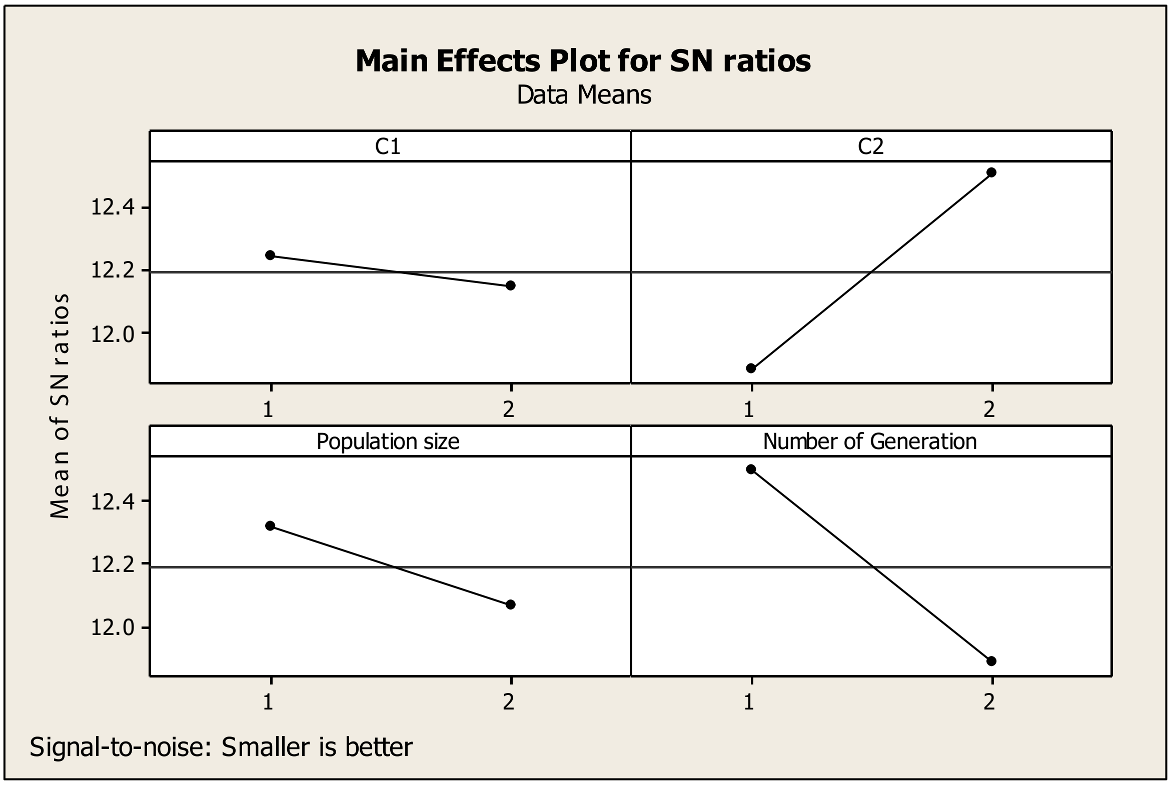
	S1	S2	S3	S4	S5	S6
F	2	4	7	9	10	12
C	1	3	6	5	8	11

	S1	S2	S3	S4	S5	S6
F	8	4	2	9	10	12
C	4	3	1	5	8	11

Fig. 3 The PSO algorithm, according to the image



(A) SRDM



(B) MRDM

Fig. 4 Taguchi ratios for SRDM algorithm and MRDM algorithm. Figure 4**A** SRDM algorithm, Fig. 4**B** MRDM algorithm

3.3.2 MRDM algorithm

The transformation steps of the objective function using the MOPSO algorithm are as follows:

Step 1: Entering the information required by the program, including the numbers of fog nodes, resources per node, resources in the cloud layer, and requests.
Step 2: Selecting the parameters of the MOPSO algorithm (i.e., the parameters specified in Subsect. 2.2).
Step 3: The initial solutions are generated randomly based on the constraints imposed by the objective function. According to the objective function, which optimizes the resource discovery, the initial solutions include a variable x matrix, which is assigned $''0''$ or $''1''$ to each of its cells. $''1''$ indicates the presence of the resource in different nodes of fog or cloud, and "0" is the absence of the resource.
Step 4: Calculating the objective function using Eqs. (5) to (7).
Step 5: Producing a new generation using the production function.
Step 6: Upgrading generations (previously-dominant members are added to the archive, and non-dominant ones are removed from the archive).
Step 7: Repeating until reaching the specified number of iterations.

3.3.3 Complexity of proposed algorithm

(A) Complexity of SRDM algorithm

In order to analyze the complexity of the SRDM, the total number of complex additions and complex multiplications per iteration were counted. To calculate the complexity, the following parameters were considered: r the domain of fog, R the number of resources, d the dimension, and n the total resources available in each fog domain.

1. Updating the velocity and position of each resource at each node requires five complex multiplication operations and seven complex addition operations in each dimension. The values of c_1 and c_2 were considered equal to 1. Moreover, velocity is easily calculated. For resource R in any fog domain with d dimensions, $5dR$ complex multiplication operations and $7dR$ complex addition operations are required to update the position and velocity of each resource.
2. The values of $v_p(k+1)$ and $v_p(k)$, These correspond to the current and previous state of velocity for each resource and require a complex addition operation stored in a table for faster access.
3. Updating each resource requires four complex multiplication operations and five complex addition operations due to the limitations set for the intended purposes.
4. Because there are r domains for fog nodes, each resource in the domain requires $(d-2)rR$ multiplication operations and drR addition operations to update all available resources.
5. nr complex addition operations are required to calculate the most suitable resource among the available resources.

The SRDM requires $(d-2)rR + 5dR + 9$ complex multiplication operations and $nr + drR + 7dR + 13$ complex addition operations. In general, the complexity of the proposed algorithm is equal to drR. As a result, the complexity of the SRDM is equivalent to $\mathrm{O}\left(N^3\right)$.

(B) Complexity of MRDM algorithm

The complexity analysis of the MRDM is as follows:

1. Updating the velocity and position of each resource at each node requires ten complex multiplication operations and eight complex addition operations in each dimension. For resource R in any fog domain with d dimensions, $10dR$ complex multiplication operations and $8dR$ complex addition operations are required to update the position and velocity of each resource.
2. The values of $v_{i,k+1}$ and $x_{i,k}$, which correspond to the velocity modes for each resource, require n complex addition operations, which are stored in a table for faster access.
3. Updating each resource requires four complex multiplication operations and two complex addition operations due to the constraints set for the intended purposes.
4. Because there are n domains for fog nodes, each resource in the domain requires $(d-2)rR$ addition operations and drR multiplication operations to update all available resources.
5. nr complex addition operations are required to calculate the most appropriate resource among the available resources.

The complexity of the MRDM requires $nr + (d-2)rR + 8dR + 11$ complex addition operations and $drR + 10dR + 14$ complex multiplication operations. According to the existing relationship, the highest degree is related to drR. As a result, the complexity of the MRDM is equal to $\mathrm{O}\left(N^3\right)$.

4 Results and experiments

We simulate the proposed algorithm to evaluate its performance. In the following, the simulation environment and its configuration are briefly introduced. Finally, the evaluation criteria and the results, along with their analyses, are presented.

4.1 Simulation and configuration environment

We use the IFogSim platform to simulate the proposed algorithm. This platform provides the opportunity to use the capabilities of the PSO algorithm, which is available as an optimization tool to facilitate simulations.

To analyze the algorithm, the number of clouds, fog nodes, and requests are 1, 4, and from 50 to 500, respectively. Moreover, the number of IoT devices has increased from 500 to 3,000. Table 4 depicts the range of SRDM and MRDM parameters.

For performance analysis, the SRDM algorithm and MRDM algorithm are compared with the algorithms presented in [29, 30, 35], and [36].

4.1.1 Adjusting parameters

There are several statistical methods for designing experiments for meta-heuristic algorithms, and in this study, the Taguchi method [45] is used. In this method, the orthogonal arrays set contains complete information about the factors affecting the performance of the algorithms. These factors fall under (1) control label or signal factors and (2) noise factors. This method uses a signal-to-noise ratio ($\frac{S}{N}$) to calculate the number of response variables. All objective functions related to resource discovery in the cloudIoT platform are of the minimization type. In experiments, the aim is to find the value of the parameters of the algorithms as input variables to obtain the optimal solution. As a result, the goal is to minimize$\frac{S}{N}$. The corresponding $\frac{S}{N}$ is calculated using Eq. (24), in which n denotes the number of iterations of the experiment and y_i is the solution of the problem.

$$\frac{S}{Ns} = -10log(\frac{1}{n}\sum_{i=1}^{n} {y_i}^2) \tag{24}$$

Table 4 PSO & MOPSO parameter ranges and levels

Algorithms	Parameter	Level 1	Level 2
SRDM	C_1	1	2
	C_2	1	2
	w	0.5	1
	Population Size	150	300
	Maxiter	250	500
MRDM	C_1	1	2
	C_2	1	2
	Population Size	50	100
	Number of Generation	150	300

4.1.2 Popular metrics for evaluating the MOPSO algorithm

We use the following popular performance metrics to analyze the performance of the MRDM algorithm.

Set coverage (C metric) This measurement is suggested in [46]. For P and Q as two PFs, $C(P,Q)$ is the percentage of Q solutions dominated by at least one solution in P:

$$C(P,Q) = \frac{|\{q \in Q | \exists p \in P : pdominatesq\}|}{|Q|} \tag{25}$$

where $|X|$ specifies the size of PFX. Given the correct value of PFP^* and the approximation of PFP, the smallest value of $C(P^*,P)$ is better than P solutions.

Spacing (SP) This parameter measures the distribution of non-dominant solutions on the approximation front. This parameter is defined as follows [47]:

$$SP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - {}^{-}d)^2} \tag{26}$$

where n specifies the number of dominant solutions on the approximation front, $d_i = min_j \sum_{k=1}^{m} |f_k^i - f_k^j|$, $i,j = 1,2,\ldots,n, m$ specifies the number of objectives and $d = \sum_{i=1}^{n} \frac{d_i}{n}$. Solutions are evenly distributed for a close-to-zero value.

The number of non-dominant solutions (NS) This criterion represents many dominant solutions in the set. When we have a large NS, the problem is better solved [20].

Inverted generational distance (IGD) This criterion reflects the convergence and diversity of solutions. For a lower value of IGD, the quality of the P solution is higher. To obtain a lower IGD, the set P must be close to the true PF in each part. Considering the uniform distribution of PFP^* and the PFP approximation, for IGD, we have [48]:

$$IDG(P,P^*) = \frac{\sum_{v \in P} d(v,P^*)}{|P^*|} \tag{27}$$

where $d = (v,p^*)$ denotes the minimum Euclidean distance from v to all points P^*.

Hypervolume (HV) This criterion is determined by approximating the volume of PF using a reference point. This is a metric measure of both proximity and diversity [49], which is calculated as follows:

$$HV(P) = \left\{ \cup_i a(x_i) \middle| \forall x_i \in P \right\} \tag{28}$$

where x_i is an individual in PFP, and (x_i) is a rectangular area bounded by a reference point and $f(x_i)$. Here, the adopted reference point is $\left[maxf_1(x), maxf_2(x)\right]$. The set of solutions P with a large HV value performs better.

Combinatorial ratio (CR) criterion The Taguchi method deals with only one response function. Consequently, the combination of performance metrics must be defined. This criterion is the combination ratio introduced in [20], which has the role of a response variable of the Taguchi method. This variable is calculated as follows:

$$C.R. = \frac{IGD}{HV} \tag{29}$$

4.2 Mathematical results

Mathematical results consist of three parts. First, the introduced parameters of the multi-objective algorithm for the MRDM algorithm are calculated and investigated. Then, according to the Taguchi method, the optimal parameters of PSO and MOPSO algorithms are calculated for the SRDM algorithm and MRDM algorithm. Finally, the mathematical results of the proposed algorithm are compared with SAC and E-S algorithms using a t-test and sign test.

4.2.1 Mathematical results of the MRDM algorithm

For evaluating the mathematical results of the MRDM algorithm, we define three scenarios. First, in the defined scenarios, we consider the numbers of clouds, fog nodes, and IoT devices per fog node as 4,1, and 500, respectively. Next, we change the number of requests from 1000 to 10,000. Then, according to the defined scenarios, we calculate the parameters introduced for the MRDM algorithm and present them in Table 5.

4.2.2 Evaluation of the results of the SRDM algorithm and MRDM algorithm

To evaluate the mathematical results of the SRDM algorithm and MRDM algorithm, we consider the numbers of clouds, fog nodes, IoT devices per fog node, and requests as 4, 1, 500, and 5000, respectively. In each algorithm, two levels are considered for each factor. The results of $\frac{S}{N}$ for both algorithms are shown in Fig. 2A, B. According to this figure, we designed Table 6, which shows the optimization for the parameters defined for both algorithms.

Table 6 Optimal value of SRDM algorithm & MRDM algorithm

Algorithms	Parameters	Optimal Value
SRDM	C_1	2
	C_2	2
	W	0.5
	Population Size	150
	Maxiter	500
MRDM	C_1	2
	C_2	1
	Population Size	100
	Number of Generation	300

4.2.3 Evaluating and comparing mathematical results

We use two tests to evaluate the proposed algorithm performance and compare it with other algorithms. The T-test is a parametric test of the results, and the sign test is a non-parametric test of results.

Tables 7 and 8 show comparisons of statistical analyses of the proposed algorithm and other algorithms. We use a t-test for the statistical analysis of the algorithms. In Table 7, there are 10 numbers, and the option value is equal to 300. This table shows the Mean *Std.Dviation* and *std.Error Mean*. According to the obtained numbers, the MRDM algorithm performs better than other algorithms. Table 9 shows the mean difference and the difference interval between the MRDM algorithm and different algorithms. $Test\ value = 300$ is a specific value used to compare the mean of the population. The analyst guesses its value. We used *SPSS* in our study, and according to the statistical sample and population, *SPSS* suggested the best test value as 300. We use this value to compare the mean of the population. The results show a smaller difference between the mean and the value related to the MRDM, which indicates the MRDM algorithm's better performance than other algorithms. MRDM algorithm has a smaller difference than other algorithms, and the difference

Table 5 Process MRDM algorithm with a different scenario

Scenarios				MRDM				
Cloud	Fog	IoT device	order	C-metric	SP	NS	IGD	HV
1	4	500	1000	Ave. 0.93 [4] St.dev. 0.082	Ave. 323.53 [3] St.dev. 42.80	Ave. 33.11 [4] St.dev. 4.97	Ave. 97.06 [3] St.dev. 51.13	Ave. 0.67 [1] St.dev.0.049
1	4	500	5000	Ave.0.81 [5] St.dev. 0.092	Ave. 356.27 [4] St.dev. 51.12	Ave. 37.28 [3] St.dev. 5.3	Ave. 101.02 [3] St.dev. 56.15	Ave. 0.75 [2] St.dev. 0.052
1	4	500	10000	Ave. 0.95 [3] St.dev. 0.097	Ave. 371.12 [5] St.dev. 55.14	Ave. 42.34 [4] St.dev. 5.7	Ave. 106.01 [4] St.dev. 59.25	Ave. 0.82 [1] St.dev. 0.062

Table 7 One-sample statistics of the different algorithms. MRDM algorithm, E-S algorithm in [29], SAC algorithm in [30], algorithm in [36], algorithm in [35], SRDM algorithm

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
MRDM algorithm	10	3.3640E2	12.49178	3.95025
E-S algorithm in [29]	10	4.6720E2	40.87325	12.92526
SAC algorithm in [30]	10	3.8350E2	33.20057	10.49894
Algorithm in [35]	10	3.978E2	31.2567	9.6742
Algorithm in [36]	10	3.732E2	26.9625	8.9354
SRDM algorithm	10	3.5340E2	22.63754	7.46735

interval of the MRDM algorithm has the lowest value in both the lower and upper parts. Consequently, MRDM algorithm performs better than other algorithms.

Table 9 shows the sign test results to compare the MRDM algorithm with the other three algorithms. Part (A) compares the MRDM and the E-S algorithms. In the table of frequencies, the first row shows that the value of the MRDM is larger than the E-S algorithm, the second row shows that the values of the E-S algorithm are larger than the MRDM, the third row shows that the values of both algorithms are equal. The last row shows the total values. The test statistics table in section (A) shows the sign test results to compare the MRDM with the E-S algorithm; the sign value is 0.008, far less than 0.5. Therefore, there is a significant difference between the MRDM and the E-S algorithm. Section (B) compares the MRDM with the SAC algorithm. The table of frequencies in part (B) shows that the MRDM value is larger than the SAC algorithm in one case in the first row. Finally, the statistics table in section (B) shows a value of 0.07 (less than 0.5), which indicates a large difference between the MRDM and the SAC algorithm. Next, section (C) shows a comparison between the MRDM and the algorithm in [36]. The table of frequencies in part (C) shows that in the first row, the value of the MRDM algorithm is larger than the algorithm in [36], however, according to the value of 0.098 (less than 0.5) in the test statistics table. Then, section (D) shows a comparison between the MRDM and the algorithm in [35]. The table of frequencies in part (D) shows that in the first row, the value of the MRDM algorithm is larger than the algorithm in [35]. However, according to the value of 0.215 (less than0.5) in the test statistics table. Finally, we can see a significant difference between the SRDM algorithm and the MRDM algorithm. However, according to the sign test results, there is a significant difference between the MRDM and the E-S algorithm. A slight difference can be observed between the MRDM algorithm and the SRDM algorithm.

4.2.4 Comparison of the complexity of algorithms

One of the best ways to compare algorithms is to consider their complexity. Table 10 shows the complexity of the proposed algorithm and other algorithms. Algorithms (MRDM, SRDM, algorithm in [35], and E-S algorithm) are equally complex due to their rapid resource discovery techniques. However, the other two algorithms (SAC algorithm and algorithm in [36]) are more complex than these algorithms.

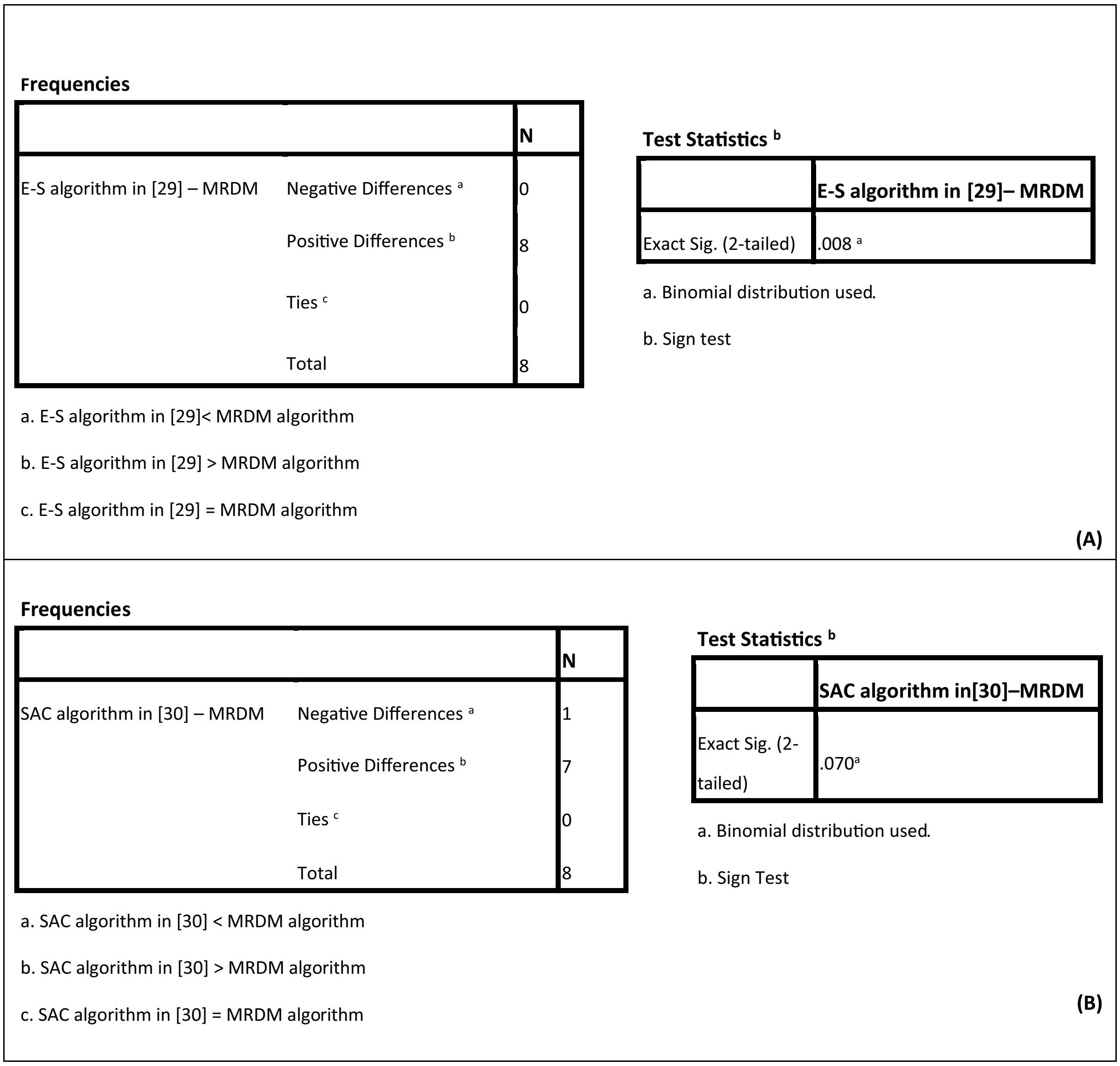
4.3 Performance comparison

We measure the success ratio and availability for evaluating the performance of the proposed algorithm. These parameters can be measured in the simulator using decision variables. The success ratio parameter shows the number of successful responses to requests. The latency equals the response time to a request from when the request has been sent. The maximum availability is an interval in which a device is available for the requests. Resource efficiency measures how resources respond to requests in the shortest

Table 8 One-sample test of different algorithm. MRDM algorithm, E-S algorithm in [29], SAC algorithm in [30], Algorithm in [36], Algorithm in [35], SRDM algorithm

	Test Value = 300					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
MRDM algorithm	9.215	9	0.000	36.40000	27.4639	45.3361
E-S algorithm in [29]	12.936	9	0.000	167.20000	137.9610	196.4390
SAC algorithm in [30]	7.953	9	0.000	83.50000	59.7497	107.2503
Algorithm in [35]	10.325	9	0.000	67.90000	48.9632	85.9312
Algorithm in [36]	6.927	9	0.000	63.50000	42.6874	78.9362
SRDM algorithm	8.105	9	0.000	54.40000	38.4574	68.3656

Table 9 Sign test statistics of the different algorithms. MRDM algorithm, E-S algorithm in [29], SAC algorithm in [30], algorithm in [36], algorithm in [35], SRDM algorithm



Frequencies

		N
E-S algorithm in [29] – MRDM	Negative Differences [a]	0
	Positive Differences [b]	8
	Ties [c]	0
	Total	8

a. E-S algorithm in [29]< MRDM algorithm

b. E-S algorithm in [29] > MRDM algorithm

c. E-S algorithm in [29] = MRDM algorithm

Test Statistics [b]

	E-S algorithm in [29]– MRDM
Exact Sig. (2-tailed)	.008 [a]

a. Binomial distribution used.

b. Sign test

(A)

Frequencies

		N
SAC algorithm in [30] – MRDM	Negative Differences [a]	1
	Positive Differences [b]	7
	Ties [c]	0
	Total	8

a. SAC algorithm in [30] < MRDM algorithm

b. SAC algorithm in [30] > MRDM algorithm

c. SAC algorithm in [30] = MRDM algorithm

Test Statistics [b]

	SAC algorithm in[30]–MRDM
Exact Sig. (2-tailed)	.070[a]

a. Binomial distribution used.

b. Sign Test

(B)

possible time. The energy consumption demonstrates the energy consumption rate of IoT devices when the number of requests changes. Several simulations with different resources, from 500 to 3000, and a different number of requests from 2000 to 20,000 are performed to evaluate these parameters.

Figure 5 shows the results of the success ratio for four algorithms. The X- and Y-axis are the number of requests and success ratio. The MRDM has a higher success ratio than the other algorithms. The main reason for the higher success ratio is that most requests are performed in the fog nodes besides dividing IoT devices into different domains. This allows the proposed algorithm to respond to more requests in the cloudIoT platform successfully. As the number of requests increases, a more significant number of IoT devices and cloud resources are engaged, and the success rate in resource discovery decreases. However, the MRDM is more stable and successful in resource discovery and can respond to more requests positively. For 2000 requests, the MRDM, SRDM, algorithm in [35], and algorithm in [36] can successfully respond to all requests, while the SAC algorithm and E-S respond to, respectively, 97% and 92% of requests. The most considerable difference in success ratio between the MRDM and other algorithms occurs for 20,000 requests; the difference is equal to 18% for the SRDM, to 24% for the algorithm in [35], to 26% for algorithm in [36], and 35% for both SAC and E-S algorithms.

In Fig. 6, the horizontal axis shows the number of requests ranging from 2000 to 20,000. The vertical axis represents the latency of algorithms. According to this

Table 9 (continued)

Frequencies

		N
Algorithm in [36] – MRDM	Negative Differences [a]	2
	Positive Differences [b]	6
	Ties [c]	0
	Total	8

a. Algorithm in [36] < MRDM algorithm

b. Algorithm in [36] > MRDM algorithm

c. Algorithm in [36] = MRDM algorithm

Test Statistics [b]

	Algorithm in [36] – MRDM
Exact Sig. (2-tailed)	.098 [a]

a. Binomial distribution used.

b. Sign Test

(C)

Frequencies

		N
Algorithm in [35] – MRDM	Negative Differences [a]	2
	Positive Differences [b]	6
	Ties [c]	0
	Total	8

a. Algorithm in [35]< MRDM algorithm

b. Algorithm in [35] > MRDM algorithm

c. Algorithm in [35] = MRDM algorithm

Test Statistics [b]

	Algorithm in [35] –MRDM
Exact Sig. (2-tailed)	.215 [a]

a. Binomial distribution used.

b. Sign test

(D)

Frequencies

		N
SRDM – MRDM	Negative Differences [a]	2
	Positive Differences [b]	6
	Ties [c]	0
	Total	8

a. SRDM algorithm < MRDM algorithm

b. SRDM algorithm > MRDM algorithm

c. SRDM algorithm = MRDM algorithm

Test Statistics [b]

	SRDM –MRDM
Exact Sig. (2-tailed)	.289 [a]

a. Binomial distribution used.

b. Sign Test

(E)

Table 10 Comparison of the proposed algorithm complexity with other algorithms

Algorithms	Complexity
MRDM algorithm	O(N^3)
SRDM algorithm	O(N^3)
The algorithm in [35]	O(N^3)
The algorithm in [36]	O(N^4)
SAC algorithm in [30]	O($N^3 \log N$)
E-S algorithm in [29]	O(N^3)

figure, for 4000 requests, the minimum latency difference occurs between the MRDM and the SRDM, which is 20(ms). The minimum differences between the MRDM and each of the other algorithms occur for 2000 requests. However, for 20,000 requests, the largest latency occurred between the MRDM and the SRDM, the algorithm in [35], the algorithm in [36], E-S, and SAC with the values of 26, 40, 59, 67, and 145(ms), respectively. The latency difference between the MRDM and the SRDM is 10%. The latency increases with the number of requests since more time is needed to finish the requests. Because the MRDM can respond to more search requests, the latency of the MRDM is reduced.

Figure 7 shows the experimental results of the availability of the proposed algorithm and the other algorithms. The horizontal and vertical axes are the number of IoT devices from 500 to 3000 and the availability in percent. Some IoT devices that may have expired are moved from one domain to another and become unavailable in order to reduce power consumption. This is a reason for reducing the availability rate of the proposed algorithm in the CloudIoT platform. Increasing the number of IoT devices and their mobility increases the complexity of the network structure and reduces the availability of devices. However, the MRDM is more successful in discovering available resources. When the number of IoT devices is 500, the availability rate for the MRDM, SRDM, the algorithm in [35], the algorithm in [36], SAC, and E-S are 45%, 40%,35%,34%, 33%, and 30%, respectively. Availability of all four algorithms increases when the number of devices increases from 1000 to 25,000. A minor difference in the availability of algorithms occurs for 2500 devices, where the difference between the MRDM and SRDM is 4%, while the difference between the MRDM and other algorithms is approximately equal to 12%.

Figure 8 demonstrates the results of the resource efficiency for the examined algorithms. The X- and Y-axis are, respectively, the number of requests and resource

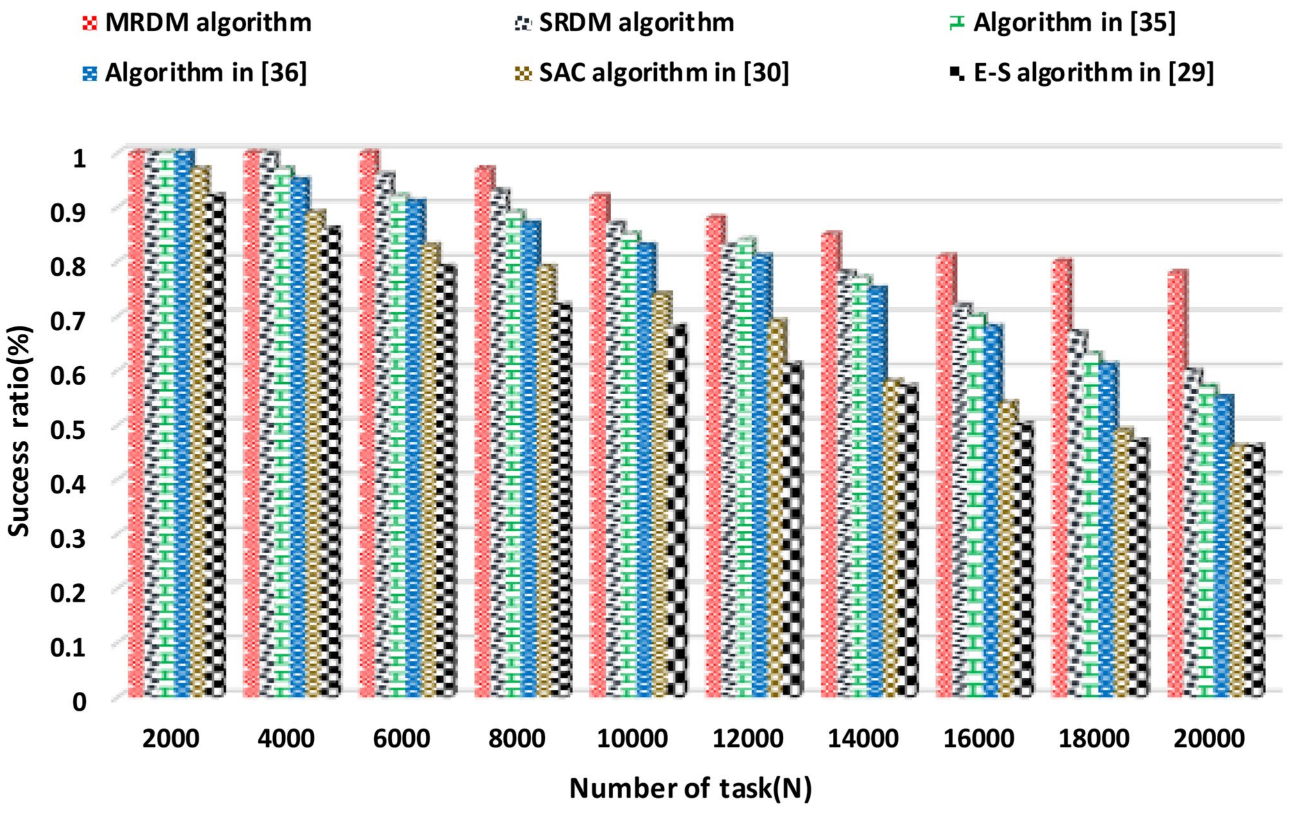


Fig. 5 The average success ratio in response to requests. The MRDM, SRDM, Algorithm in [35], Algorithm in [36], SAC algorithm [30], E-S algorithm in [29]

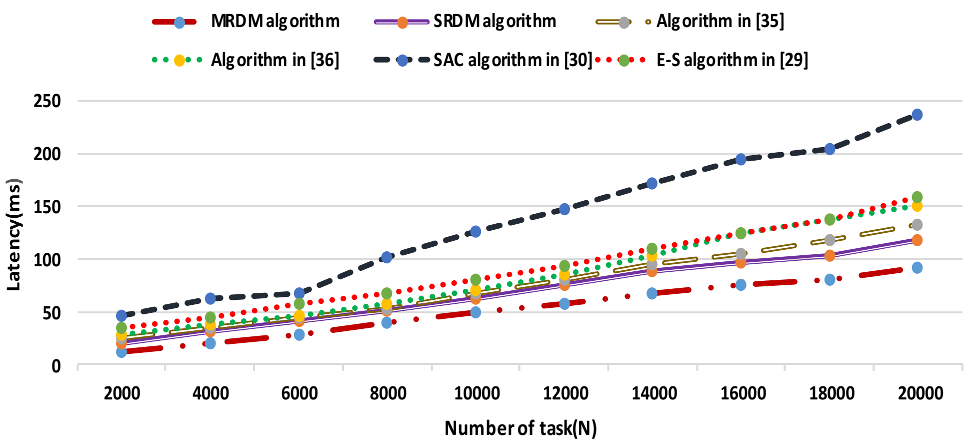


Fig. 6 The average latency in response to requests. The MRDM, SRDM, Algorithm in [35], Algorithm in [36], SAC algorithm in [30], E-S algorithm in [29]

efficiency. The MRDM has a lower resource efficiency than other algorithms. The main reason for the lower resource efficiency is that most requests are handled in the fog nodes besides dividing IoT devices into different domains. This allows the proposed algorithm to respond to more requests in the cloudIoT platform successfully. As the number of requests increases, many IoT devices and cloud resources are engaged, and the resource efficiency in resource discovery decreases. However, for 20,000 requests, the largest occurs between the MRDM and other algorithms. The energy efficiency difference between the MRDM and the SRDM is 8%, while the differences between the MRDM and the algorithm in [35], the algorithm in [36], SAC, and E-S are 21%, 24%, 28%, and 31%, respectively.

Figure 9 illustrates the experimental results of the energy consumption of the proposed algorithm and other algorithms. The number of IoT devices is considered 500. The horizontal and vertical axes are the number of requests from 2000 to 20,000 and the energy consumption of resources. As the number of requests increases, the energy consumption in the MRDM algorithm decreases. The main reason for the lower value is that most requests are handled in the fog nodes. When the number of requests is 2000, the difference between the energy consumption of the MRDM and the SRDM, the algorithm in [35], the algorithm in [36], SAC, and E-S are 13%, 17%, 19%, 22%, and 25%, respectively. Energy consumption of all algorithms increases sharply when the number of requests increases from 12,000 to 20,000.

4.4 Analysis of results

The main factors affecting the performance of the MRDM algorithm include success rate, latency, availability, resource efficiency, and energy consumption. According to the experiments, the results show that the proposed solution has a high success rate in responding to requests, low latency, high availability, and low energy consumption. Resources in different domains of fog provide faster access to the right resource, which makes it possible to better respond to requests with low latency and low power consumption. This has significantly improved the performance of the proposed algorithm.

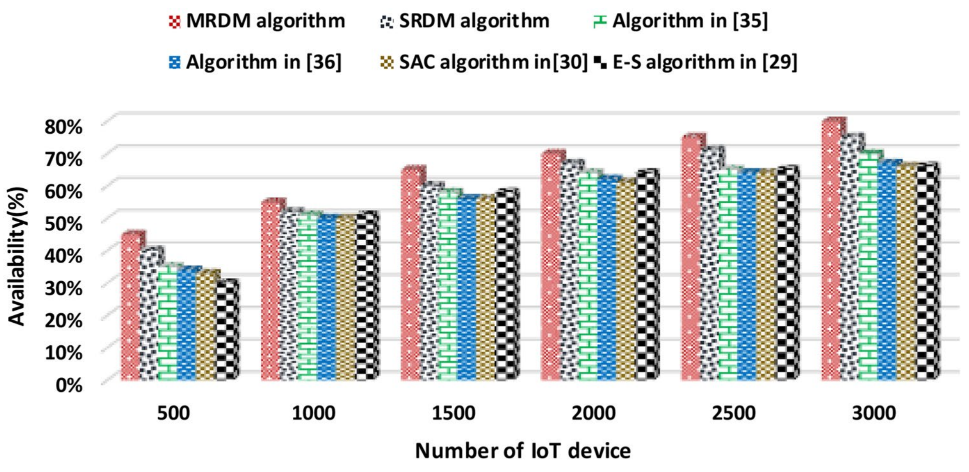


Fig. 7 The average availability of IoT devices. The MRDM, SRDM, Algorithm in [35], Algorithm in [36], SAC algorithm in [30], E-S algorithm in [29]

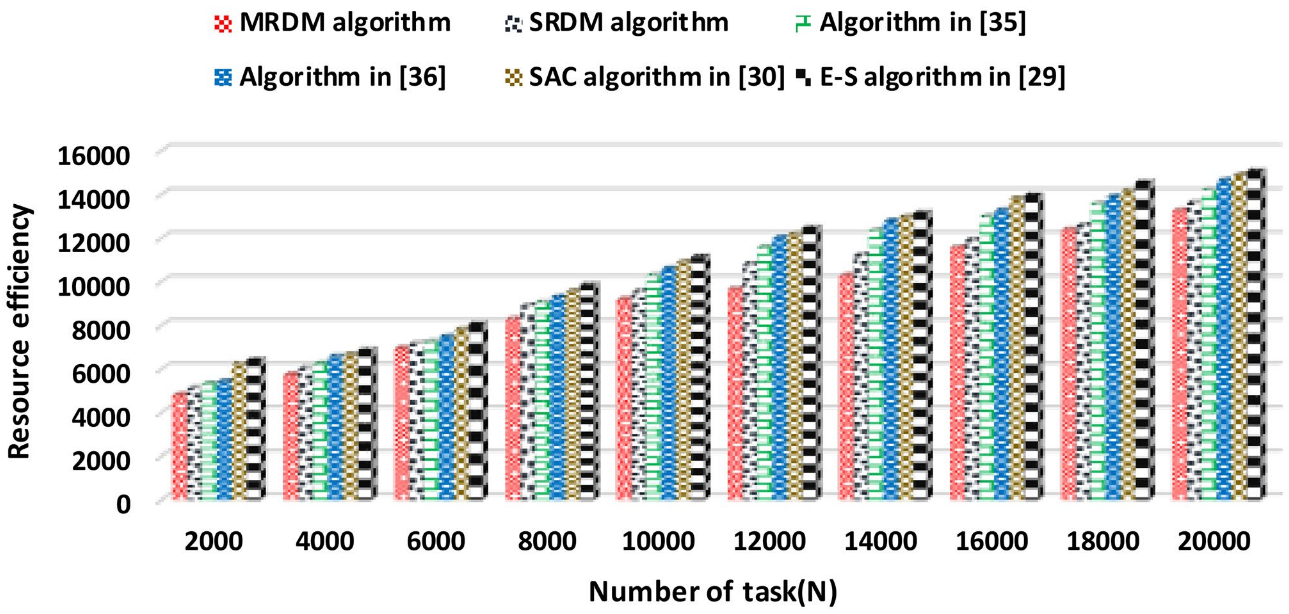


Fig. 8 The average resource efficiency in response to requests. The MRDM, SRDM, Algorithm in [35], Algorithm in [36], SAC algorithm in [30], E-S algorithm in [29]

We implemented the proposed method using both single-objective and multi-objective approaches. However, according to the evaluation, the multi-objective approach works better than the single-objective approach. In contrast, using these two approaches, the proposed method provides better results than the previous algorithms for discovering resources, so the proposed algorithm has a higher performance than the previously presented solutions.

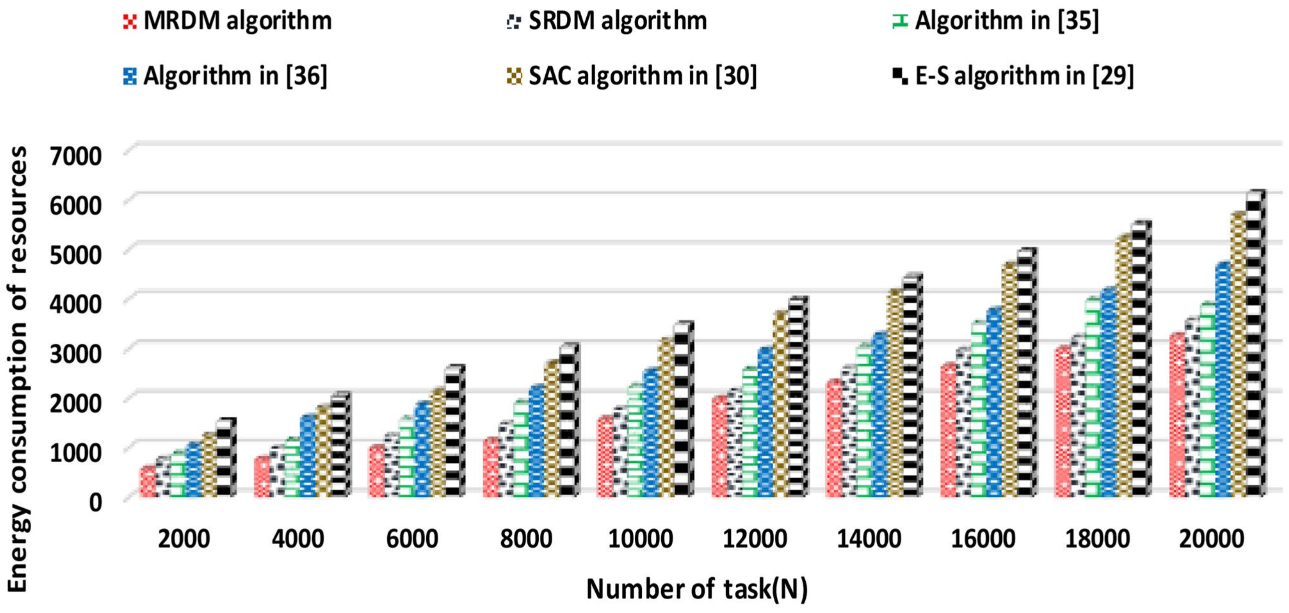


Fig. 9 The average energy consumption in response to requests. The MRDM, SRDM, Algorithm in [35], Algorithm in [36], SAC algorithm in [30], E-S algorithm in [29]

5 Conclusions

In this paper, the main purpose of the proposed resource discovery algorithm was to minimize response time, cost, and bandwidth in CloudIoT platforms. First, a new mathematical model was developed using a MILP model. Since the proposed MILP has three purposes, a multi-objective MCGP with a utility function was used to solve it. Then, PSO was used for solving the MCGP. In addition, we implemented the proposed algorithm based on MOPSO. Finally, to compare the performances of the proposed algorithm and other algorithms, we defined a scenario in which the weights of all functions were the same. The parameters of comparison were success ratio, latency, and availability. The results indicated a considerable performance improvement in MRDM algorithm compared to other algorithms.

Authors' contributions All authors contributed equally to this manuscript.

Availability of data and material Not applicable.

Declarations

Competing interests There is no conflict of interest.

References

1. Nawaz F, Hussain O, Hussain FK, Janjua NK, Saberi M, Chang E (2019) Proactive management of SLA violations by capturing relevant external events in a Cloud of Things environment. Futur Gener Comput Syst 95:26–44
2. Xavier TC, Santos IL, Delicato FC, Pires PF, Alves MP, Calmon TS, Amorim CL (2020) Collaborative resource allocation for Cloud of Things systems. J Netw Comput Appl 159:102592
3. Tian Y, Kaleemullah MM, Rodhaan MA, Song B, Al-Dhelaan A, Ma T (2019) A privacy preserving location service for cloud-of-things system. J Parallel Distrib Comput 123:215–222
4. Li Z, Yang Z, Xie S (2019) Computing resource trading for edge-cloud-assisted Internet of Things. IEEE Trans Industr Inf 15(6):3661–3669
5. Elazhary H (2019) Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. J Netw Comput Appl 128:105–140
6. Darwish A, Hassanien AE, Elhoseny M, Sangaiah AK, Muhammad K (2019) The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: opportunities, challenges, and open problems. J Ambient Intell Humaniz Comput 10(10):4151–4166
7. Munir A, Kansakar P, Khan SU (2017) IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. IEEE Consum Electron Mag 6(3):74–82
8. Wang SC, Tseng SC, Yan KQ, Tsai YT (2018) Reaching agreement in an integrated fog cloud IoT. IEEE Access 6:64515–64524
9. Jiang Y, Huang Z, Tsang DH (2017) Challenges and solutions in fog computing orchestration. IEEE Network 32(3):122–129
10. Kochar V, Sarkar A (2016) Real time resource allocation on a dynamic two level symbiotic fog architecture. In 2016 Sixth International Symposium on Embedded Computing and System Design (ISED) (pp. 49–55). IEEE
11. Mseddi A, Jaafar W, Elbiaze H, Ajib W (2019) Joint container placement and task provisioning in dynamic fog computing. IEEE Internet Things J 6(6):10028–10040
12. Zhang F, Ge J, Li Z, Li C, Huang Z, Kong L, Luo B (2017) Task Offloading for Scientific Workflow Application in Mobile Cloud. In IoTBDS (pp. 136–148)
13. Kozyrev D, Ometov A, Moltchanov D, Rykov V, Efrosinin D, Milovanova T, Koucheryavy Y (2018) Mobility-centric analysis of communication offloading for heterogeneous Internet of Things devices. Wirel Commun Mob Comput
14. Guo K, Yang M, Zhang Y, Cao J (2019) Joint computation offloading and bandwidth assignment in cloud-assisted edge computing. IEEE Trans Cloud Comput
15. Abdel-Basset M, Abdle-Fatah L, Sangaiah AK (2019) An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Clust Comput 22(4):8319–8334
16. Alli AA, Alam MM (2019) SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications. Internet Things 7:100070
17. Zhang Q, Liang H, Xing Y (2014) A parallel task scheduling algorithm based on fuzzy clustering in cloud computing environment. Int J Mach Learn Comput 4(5):437
18. Lin W, Liang C, Wang JZ, Buyya R (2014) Bandwidth-aware divisible task scheduling for cloud computing. Softw Pract Exp 44(2):163–174
19. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
20. Ding S, Chen C, Xin B, Pardalos PM (2018) A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. Appl Soft Comput 63:249–267
21. Gill SS, Garraghan P, Buyya R (2019) ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices. J Syst Softw 154:125–138
22. Bharti M, Kumar R, Saxena S (2018) Clustering-based resource discovery on Internet-of-Things. Int J Commun Syst 31(5):e3501
23. Ezugwu AE, Adewumi AO (2017) Soft sets based symbiotic organisms search algorithm for resource discovery in cloud computing environment. Futur Gener Comput Syst 76:33–50
24. Panwar N, Negi S, Rauthan MMS, Vaisla KS (2019) TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment. Clust Comput 22(4):1379–1396
25. Reddy MPK, Babu MR (2019) Implementing self adaptiveness in whale optimization for cluster head section in Internet of Things. Clust Comput 22(1):1361–1372
26. AlZubi A, Alarifi A, Al-Maitah M, Albasheer OA (2020) Location assisted delay-less service discovery method for IoT environments. Comput Commun 150:405–412
27. Kalaiselvi S, Selvi CK (2020) Hybrid cloud resource provisioning (HCRP) algorithm for optimal resource allocation using MKFCM and bat algorithm. Wirel Pers Commun 111(2):1171–1185
28. Skarlat O, Karagiannis V, Rausch T, Bachmann K, Schulte S (2018) A framework for optimization, service placement, and runtime operation in the fog. In 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC) (pp. 164–173). IEEE
29. Nunes LH, Estrella JC, Perera C, Reiff-Marganiec S, Delbem AC (2018) The elimination-selection based algorithm for efficient resource discovery in Internet of Things environments. In 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 1–7). IEEE

30. Md AQ, Varadarajan V, Mandal K (2019) Efficient algorithm for identification and cache based discovery of cloud services. Mob Netw Appl 24(4):1181–1197
31. Abdi S, PourKarimi L, Ahmadi M, Zargari F (2017) Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds. Futur Gener Comput Syst 71:113–128
32. Kalantary S, Akbari Torkestani J, Shahidinejad A (2021) Resource discovery in the Internet of Things integrated with fog computing using Markov learning model. J Supercomput 1–22
33. Bharti M, Jindal H (2021) Optimized clustering-based discovery framework on Internet of Things. J Supercomput 77(2):1739–1778
34. Xu L, Zhou X, Tao Ye, Lei Liu XuYu, Kumar N (2021) Intelligent Security Performance Prediction for IoT-Enabled Healthcare Networks Using an Improved CNN. IEEE Trans Industr Inf 18(3):2063–2074
35. Liu Y, Zhang W, Zhang Q, Norouzi M (2021) An optimized human resource management model for cloud-edge computing in the internet of things. Cluster Comput 1–13
36. Murturi I, Dustdar S (2021) A decentralized approach for resource discovery using metadata replication in edge networks. IEEE Trans Serv Comput
37. Chang CT (2011) Multi-choice goal programming with utility functions. Eur J Oper Res 215(2):439–445
38. Varshney P, Simmhan Y (2017) Demystifying fog computing: Characterizing architectures, applications and abstractions. In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC) (pp. 115–124). IEEE
39. Helsley M (2009) LXC: Linux container tools. IBM DevloperWorks Technical Library 11
40. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux J 2014(239):2
41. Yannuzzi M, Milito R, Serral-Gracià R, Montero D, Nemirovsky M (2014) Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing. In 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) (pp. 325–329). IEEE
42. Openstack. http://www.openstack.org. Accessed 13 Oct 2018
43. OpenNebula.org. http://www.opennebula.org. Accessed 22 Jan 2021
44. Eucalyptus. https://www.eucalyptus.cloud/. Accessed 17 Apr 2020
45. Taguchi G, Chowdhury S, Wu Y (2005) Taguchi's quality engineering handbook. Wiley Publishing
46. Zitzler E, Thiele L (1998) Multi-objective optimization using evolutionary algorithms—a comparative case study. In International conference on parallel problem solving from nature (pp. 292–301). Springer, Berlin, Heidelberg
47. Schott JR (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization (Doctoral dissertation, Massachusetts Institute of Technology)
48. Zhang Q, Li H (2007) MOEA/D: A multi-objective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
49. Yen GG, He Z (2013) Performance metric ensemble for multiobjective evolutionary algorithms. IEEE Trans Evol Comput 18(1):131–144

Parisa Goudarzi received her BS in Computer Engineering from Islamic Azad University, Borujerd Branch, in 2009, the MS in Information Technology Engineering from IAU University, Qazvin, in 2015. She is a full-time PhD Candidate in Computer Engineering at Islamic Azad University, Dezful Branch, Dezful, Iran. Her research interests include IIoT, Distributed Systems, Resource Discovery, and Cloud Computing.

Amir Masoud Rahmani received his BS in Computer Engineering from AmirKabir University, Tehran, in 1996, the MS in Computer Engineering from Sharif University of Technology,Tehran, in 1998 and the PhD degree in Computer Engineering from IAU University, Tehran, in 2005.Currently, he is a Professor in the Department of Computer Engineering at the IAU University. He is the author/co-author of more than 150publications in technical journals and conferences. His research interests are in the areas of distributed systems, adhoc and wireless sensor networks and evolutionary computing.

Mohammad Mosleh received the BS degree in Computer Hardware Engineering from Islamic Azad University, Dezful Branch, Dezful, Iran in 2003, the MS degree in Architecture of Computer Systems from Islamic Azad University, Science and Research Branch, Tehran, Iran, in 2006 as well as the Ph.D degree in Computer Engineering from the Islamic Azad University, Science and Research Branch, Tehran, Iran in 2010. He is an assistant professor in the Department of Computer Engineering at the Islamic Azad University of Dezful. His research interests are in audio security (watermarking and steganagraphy), machine learning and nano computing including quantum-dot cellular automata (QCA) and reversible circuits.