



An efficient data replica placement mechanism using biogeography-based optimization technique in the fog computing environment

Jaber Taghizadeh¹ · Mostafa Ghobaei-Arani¹ · Ali Shahidinejad¹

Received: 20 October 2020 / Accepted: 6 September 2021 / Published online: 26 September 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

In recent years, the rapid growth of IoT devices has led to an increase significantly the amount of data generated. Transferring a huge amount of datasets from IoT devices to remote cloud servers will result in high latency and bandwidth usage. Fog computing has emerged as an Internet-based distributed computing model to store datasets generated by IoT devices near the user. Since IoT devices generate continuously massive amounts of datasets, placing them on the storage fog nodes with various capabilities to reduce latency and costs of data access and increase reliability and availability of data datasets while satisfying the QoS requirements as one of the challenging tasks to be considered. This paper proposes a metaheuristic-based data replica placement mechanism using biogeography-based optimization (BBO) for data-intensive IoT applications on the fog ecosystem. Besides, we design an autonomous framework to illustrate transferring data replicas between IoT devices and storage fog nodes for data replica placement problem in the fog ecosystem. The obtained simulation results by varying the number of data replicas and fog nodes demonstrate that the proposed mechanism is a cost-effective solution and it increases the average reliability and availability by up 13% and 15% and reduces the total cost and the latency 25% and 3%, respectively, compared with the other baseline mechanisms.

Keywords Fog computing · Data replica placement · Data-intensive IoT applications · Biogeography-based optimization

1 Introduction

During the last years, there is a rapid growth in the number of IoT devices connected through the Internet to control or monitor physical sensors. Such a large number of IoT devices will generate a considerable volume of data that needs to be collect and process for sharing and using by IoT applications (Devadas et al. 2020). A simple solution for storing and processing a large amount of data is using the cloud servers and returning the results to IoT devices. This solution provided IoT applications' development to access unlimited processing and storage capacities (Sengupta and Bhunia 2020). However, transferring a massive amount of data between IoT devices and cloud servers will lead to high latency and bandwidth usage. To deal with these constraints mentioned, fog computing has been explored as an extension of cloud computing to provide the storing and

processing capabilities in the network intermediate devices (e.g., routers, switches, gateways) nearby to the data generators, rather than remote cloud servers (Dadashi Gavaber and Rajabzadeh 2021; Mukherjee et al. 2018; Shahidinejad and Ghobaei-Arani 2020). Therefore, data management can play an essential role in storing, transferring, placing, and processing the big volume of data generated by IoT devices to enhance system performance in an effective fashion in the fog landscape. To this end, the data management issue is one of the challenging tasks to be considered. Data management includes several issues such as data acquisition, data storage, data pre-processing, data cleaning, data exchange, data processing, data placement, and data analytics (Kumari et al. 2019; Nikoui et al. 2019). In this work, we focus on the data replica placement issue to appropriately place the huge amount of data generated by IoT devices on the fog nodes with various storage capabilities under QoS constraints to reduce latency and costs of data access for fog-enabled data-intensive IoT applications. Since the data replica placement is an NP-hard problem, the heuristic or meta-heuristic algorithms are used to find near-optimal solutions in a reasonable time. Some data replica placement strategies have already

✉ Mostafa Ghobaei-Arani
m.ghobaei@qom-iau.ac.ir

¹ Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

been proposed to place data replicas on the cloud servers in the cloud computing scope (Martin et al. 2020; Costa Filho et al. 2020). However, compared with cloud servers, fog resources are more limited in storage capability, more widely geographically distributed in a very large ecosystem, and larger in number. Therefore, the cloud-based data replica placement strategies cannot be used directly in the fog computing environment.

In this paper, we develop an extended framework according to three-tier fog architecture to illustrate transferring data replicas between IoT devices and storage fog nodes for serving data-intensive IoT applications in the fog ecosystem. The proposed framework consists of several fog domains so that each fog domain acts based on a data replica manager to place data replicas with various QoS requirements on the fog nodes with different storage capabilities. Then, we utilized biogeography-based optimization (BBO) (Simon 2008) as a population-based meta-heuristic technique to find an appropriate data replica placement plan. The BBO as a population-based meta-heuristic technique is suitable for solving large-scale optimization problems to find cost-efficient solutions into search space. It uses migration and mutation operators to generate new solutions and share features between candidate solutions so that it provides high convergence speed and local optima avoidance during the search process compared with existing meta-heuristic techniques. The BBO technique has been successfully applied to handle practical problems in various scopes including image processing (Pal and Saraswat 2019; Zhang et al. 2019), social networks (Reihanian et al. 2017; Zhou et al. 2015), cloud computing (Sangaiah et al. 2019; Zheng et al. 2016; Khorsand et al. 2018), data mining (Alweshah 2019), and IoT (Goudarzi et al. 2019; Paraskevopoulos et al. 2017). Finally, the data replica manager is regularly performed at certain time intervals to reduce latency and costs of data access and increase the reliability and availability of data replicas while the QoS requirements of data-intensive IoT applications are satisfied.

The main contributions of this study can be summarized as follows:

- Proposing a metaheuristic-based data replica placement mechanism using the BBO technique for data-intensive IoT applications in the fog ecosystem.
- Developing an autonomous framework for data replica placement to illustrate transferring data replicas between IoT devices and storage fog nodes.
- Describing a problem formulation for data replica placement in fog ecosystem.

The remainder of this study is organized as follows: we discuss works related to the data replica placement mechanisms in Sect. 2. We describe the proposed data replica

placement approach in more detail in Sect. 3. The simulation results are provided in Sect. 4, and we finally provide future research directions and conclusions in Sect. 5.

2 Related works

In this section, we will review the data replica placement approaches in the fog/edge computing environment. Then, we summarize different works to solve the data replica placement problem.

Guerrero et al. (2019) have developed a replica placement mechanism for storing the data obtained by IoT devices on the fog infrastructure. Their proposed mechanism used graph partition techniques to choose the fog nodes to replicate data for increasing data reliability and availability. Besides, they model the topology of fog infrastructure using complex networks to specify the fog nodes close to the IoT devices. Their simulation results obtained by the YAFS simulator indicated that their proposed mechanism outperforms in terms of network usage, latency, and data availability compared with the FogStore mechanism. Shao et al. (2019) have studied the data replica placement issue for serving data-centric IoT applications in the edge/cloud ecosystem. They described the replica placement problem in the form of linear programming and proposed a population-based optimization technique to make a suitable data placement solution for data-centric IoT applications. Further, they developed an extended framework, including the replica broker component to dispatch the data generated by IoT devices among the edge servers and replica manager components to control the data replication process in the edge/cloud ecosystem. Their numerical results illustrated that their proposed solution could reduce the number of data movement, the amount of data transmission, and data access cost metrics compared with existing population-based optimization mechanisms.

Huang et al. (2019) have proposed a multiple replica placement strategy for minimizing the total latency for data-intensive IoT services in the fog-enabled environment. They used a greedy heuristic-based algorithm to achieve an approximately good solution in large-scale systems. Their proposed strategy utilized a pruning technique to refine the poor solutions and obtains the solution with the minimum latency for data-intensive IoT applications. Their simulation results obtained by iFogsim toolkit and CPLEX optimizer indicated that their solution better performance in terms of solving time and the total data latency compared with existing iFogStor mechanisms. Chen et al. (2019) have introduced a data-centric service deployment approach for improving the time-consuming efficiency in the mobile edge computing environment. They model the IoT applications as a set of data and services using graph theory to find the relationship between data-centric services and edge nodes and

used a genetic algorithm to achieve a near-optimal deployment solution. Besides, they conducted a series of simulations to illustrate the effectiveness of the proposed approach in terms of convergence speed and response time compared with existing meta-heuristic techniques. Aral and Ovatman (2018) have presented a fully distributed strategy to place data replica among edge servers according to the location of the user and the cost of storing data replicas to reduce the access latency. Further, they provided a lightweight messaging mechanism to announce between edge servers about the closest data replica requested by a data object. Their proposed strategy relies on adding or removing the data replica based on monitoring incoming data requests to edge servers for minimizing both cost and latency. Besides, they validated their distributed proposed strategy under synthetic and real-world usage patterns and indicated that it to obtain notable advantages in terms of cost storage and the access latency compared with other caching strategies.

Naas et al. (2018a) have proposed an approximate heuristic-based approach for reducing the solving time of data placement issue in fog landscape. Their proposed approach used to divide and conquer methodology with the graph partitioning and modeling techniques to divide the initial data placement problem into different sub-problems. They validate their proposed approach by extending the iFogSim tool under mixed, distributed, and zoning workloads on the smart city scenarios and demonstrated that it obtains good performance in terms of total latency and solving time compared with iFogStor and iFogStorZ approaches. In another study (Naas et al. 2017), they provided a runtime policy to place IoT data for minimizing service latency in the fog-enabled scenario. They presented an exact method using linear programming and CPLEX solver and also, a heuristic-based solution according to geographical partitioning of the underlying fog ecosystem for reducing solution time. Li et al. (2019a) have introduced a dynamic replica selection method to improve the data access time in the edge/cloud ecosystem. They considered the access heat of data replicas and the edge server capabilities and utilized the grey Markov technique to adjust the number of data replica, dynamically. Moreover, they take into account the distance between the data replica and the user and the load capacity of the edge server to find the best data replica for serving the user data access request. Their analysis results demonstrated that their proposed method achieves notable improvements in terms of the prediction accuracy of data replica access, response time, and resource utilization. In another study (Li et al. 2019b), a fault-tolerant replica placement approach with taking into account the budget and deadline requirements in the edge/cloud ecosystem has been proposed. They have provided a failure recovery mechanism according to the availability metric using backup techniques to avoid the data replica loss and the data access errors to restore the corresponding data replica.

Monga et al. (2019) have proposed a distributed reliable data replica service policy for fog and edge servers to offer transparent access and discovery stream of data replicas, persistently. Their proposed policy was inspired by peer-to-peer networks and big data storage services like HDFS. It utilized federated index strategy and Bloom filters for probabilistic and scalable searching of data replicas according to their metadata attributes. Besides, they provided adjustable reliability using global statistics about the edge storage servers for data replicas to achieve a trade-off the storage capacity across edge storage servers and minimum stability. Mayer et al. (2017) have developed a fog-enabled replica placement policy and context-aware derived consistency named FogStore for distributed data storage systems in the fog computing environment. The FogStore solution used a consistency level to extract user and data locality in the fog infrastructure domain. They validate their solution using the MaxiNet emulator on the Yahoo cloud benchmark for verifying the impact of different consistency levels in terms of latency read and write operations on the Cassandra system. Breitbach et al. (2019) have proposed a context-aware data management strategy in the edge computing domain. Their proposed strategy is consist of three-level including the data placement to locate data replica on the edge servers according to various context attributes, the task scheduling to allocate them on the most suitable edge servers, and the runtime adaptation mechanism to collect the QoS requirements for adapting data placement during the execution of tasks. They provided a prototype of their proposed strategy to investigate the trade-off between the data overhead and access latency according to the context properties.

Silva et al. (2019) have evaluated tree data placement mechanisms, namely, cloud-only, mapping, and edge-ward using iFogSim simulator on the smart healthcare scenarios. The cloud-only mechanism is based on a delay priority policy, the mapping mechanism used a concurrent method for assigning the IoT applications to fog nodes, and the edge-ward algorithm utilized the First-In-First-Out approach to place data on the fog nodes. Their results obtained indicated that the mapping mechanism provides good performance in terms of execution time and energy consumption under different topologies compared with other mechanisms. Karatas and Korpeoglu (2019) have considered a classification of data generated by IoT devices in the forms of the various data type to use by several IoT applications. They map the data-centric replica placement issue into a linear programming model as an optimization problem for minimizing the access latency. Then, they proposed two greedy heuristic-based methods according to the cost function for achieving an appropriate place for different data types. Their simulation results of the proposed solution provided good storage performance in terms of bandwidth usage and access latency. Confais et al. (2018) have introduced a tree-based

mechanism corresponding to the topology of fog infrastructure to store the position of data objects in the fog landscape. Their proposed mechanism recorded the new positions to identify the data objects close to replicas for improving bandwidth usage and the access time. Further, they proposed an extended Dijkstra's technique to calculate the lowest latency paths according to cost function in the underlying fog infrastructure. Finally, they validated their solution on the Grid'5000 benchmark under a simple topology and indicated that it better access time compared with the existing distributed hash table mechanism.

Trakadas et al. (2020) have designed an extended architectural approach using reference architecture for industry 4.0 to adopt and integrate artificial intelligence-based systems in the IoT manufacturing scope. Their proposed architecture supports data collection and processing while handling security issues with AI-enabled across the entities into manufacturing systems. In Habibi et al. (2020) and Punithallayarani and Dominic (2019), the computing paradigms, including edge, fog, and cloud computing, are reviewed in terms of algorithmic, technologic architectural aspects. Besides, they describe various fog computing model dimensions in terms of software, system, application, networking, and resource management. Finally, they provide a comprehensive review of different reference architectures for fog computing from the architectural perspective. Alvarez et al. (2019) have designed a new service platform, which utilizes software-defined networking concepts to provide multimedia services on 5G networks. Their proposed platform leverages serverless computing and cognitive management to provision network services and multimedia applications.

According to the reviewed and summarized data replica placement mechanisms, a side-by-side comparison of them in terms of the utilized technique, evaluation tool, performance metrics, and strategy (e.g., centralized, decentralized) as well as a case study of each mechanism is shown in Table 1.

3 Proposed approach

In this section, we describe the proposed solution in more detail, and then a formulation for the data replica placement problem is provided. Finally, an algorithm for data replica files onto the IoT infrastructure is presented.

3.1 Proposed framework

In this section, we describe an extended framework for the data replica placement problem in a fog computing environment, as shown in Fig. 1. The proposed framework inspired according to fog computing architecture and it consists of three layers, namely: the IoT device, the fog, and the cloud

layers. The IoT device layer (i.e., the bottom layer) includes sensors, mobile devices, and tablets and it is responsible for collecting and transferring massive amounts of datasets generated through the data replica broker to the fog layer. The fog layer (i.e., the middle layer) includes intermediate equipment's (i.e., fog nodes) such as routers, gateways, switches, access points, and base stations which can be fixed or mobile in restaurants, shopping centers, bus stations, streets, parks, and so on. These fog nodes are able to processing, transmitting, and storing datasets received from IoT devices. Besides, they are able to connect the cloud data center via the cloud gateway to obtain more computational and storage capabilities. The cloud layer (i.e., the upper layer) includes a series of cloud servers with high resource capabilities to execute the data-intensive IoT applications. We assume the fog layer is consists of several fog domains so that each fog domain includes one regional fog node (RFOG) and a series of local fog nodes (LFOG), as shown in Fig. 1.

The RFOG nodes are base stations which deployed in each fog domain to interact with other fog domains, IoT device, and cloud layers. Each RFOG node acts as the master node of each fog domain with high computation and storage capabilities and it manages and controls a geographical area in the fog landscape. Each RFOG node covers a series of LFOG nodes in a hierarchical manner into each fog domain and data replicas can be deployed on any of the RFOG or LFOG nodes. Indeed, the IoT applications on IoT devices will generate the datasets, which will be placed on the fog nodes.

In the following, we explain the main components of the fog layer to implement our proposed solution in more detail.

3.1.1 Data Replica Broker

This component is responsible for receiving data access requests from IoT devices and transferring them to storage data replicas on the nodes deployed at the fog or cloud layers. We assume the data access requests are delay-sensitive or compute-sensitive. If a data access request is delay-sensitive, it will be placed through a data replica manager deployed on the LFOG nodes into each fog domain at the fog layer. Otherwise, it will be placed on the cloud layer. Note that privacy and security concerns as one of the main challenging issues should be considered in the fog ecosystem. To this end, we focus on the data replica broker as a gateway to ensure access control and prevent vulnerabilities to different attacks such as denial of service, man-in-the-middle, and IP spoofing. Therefore, authentication protocols are needed to identify incoming data access requests. To this end, firstly, the data replica broker refines the data access requests to remove abnormal requests from incoming IoT workloads. The abnormal requests include the requests with null fields, the fake requests as robots to publicize, and the

Table 1 A comparison of the data replica placement mechanisms in fog computing

References	Utilized technique	Evaluation tool	Performance metric	Strategy	Case study
Guerrero et al. (2019)	Graph partitioning algorithms	Simulation (YAFS)	Latency, network usage, availability, solution time	Decentralized	IoT application
Shao et al. (2019)	Swarm optimization	Simulation (Java)	Data access cost, amounts of data transmission, number of data movement	Centralized	IoT-based Manufacturing application
Huang et al. (2019)	Greedy heuristic-based	Simulation (iFog-Sim + CPLEX solver)	Overall latency, solving time	Centralized	IoT application
Chen et al. (2019)	GA	Simulation (Python)	Response time, convergence speed	Centralized	Data-intensive application
Aral et al. (2018)	Heuristic-based	Simulation (CloudSim)	Latency, cost, throughput	Decentralized	IoT application
Naas et al. (2018a)	Heuristic-based + Graph partitioning algorithms	Simulation (iFog-Sim + CPLEX solver)	Overall latency, solving time	Centralized	Smart city application
Li et al. (2019a)	Grey Markov	Simulation (Java)	Response time, Network latency, Resource utilization	Centralized	IoT application
Li et al. (2019b)	Heuristic-based	Simulation (Java)	Response time, resource utilization, reliability	Centralized	IoT application
Monga et al. (2019)	Heuristic-based + Bloom filters	Simulation (Java + Apache Thrift)	Latency, reliability, storage capacity	Decentralized	IoT streaming application
Mayer et al. (2017)	Heuristic-based	Simulation (MaxiNet)	Network latency	Decentralized	IoT application
Breitbach et al. (2019)	Heuristic-based	Prototype (Java)	Turnaround times, data transfer overhead	Decentralized	Face recognition application
Silva et al. (2019)	Heuristic-based	Simulation (iFogSim)	Latency, energy consumption, execution time	Centralized	Smart healthcare application
Karatas and Korpeoglu (2019)	Heuristic-based + Classification	Simulation (MATLAB + Gurobi solver)	Access latency, network usage, storage capacity	Hierarchical	IoT application
Confais et al. (2018)	Tree-based technique	Prototype (Grid'5000 testbed)	Search time	Hierarchical	IoT application
Our approach	BBO	Simulation (iFogSim)	Latency, cost, reliability, availability	Centralized	IoT application

unauthorized requests for utilizing the data replicas, and so on. Afterwards, an identifier is assigned to each the data access request for tracking the further calls. Finally, the data access requests are recognized through the authentication protocols to use the data replicas deployed on the data hosts.

3.1.2 Data replica manager

After receiving data access requests from the data replica broker, the data replica manager will be launched as the main component for implementing our proposed solution. This component manages four sub-components, as follows:

3.1.2.1 Monitor This component gathers information about the data access requests and data replicas status in a geo-

graphical area. It includes two sub-components namely, user monitoring and data replica monitoring. The user monitoring is responsible for collecting information about the number of requests received from IoT devices to access the data replicas, and the data replica monitoring is responsible for collecting information about status data replicas (e.g., the number of data replica, time delay access, etc.) stored on the LFOG nodes into each fog domains at the fog landscape. This information is stored in a shared database to use by other components.

3.1.2.2 Data replica analyzer This component process the raw data collected from the monitoring component to obtain useful information and use it to estimate the number of data replicas needed for placing on the LFOG nodes while QoS

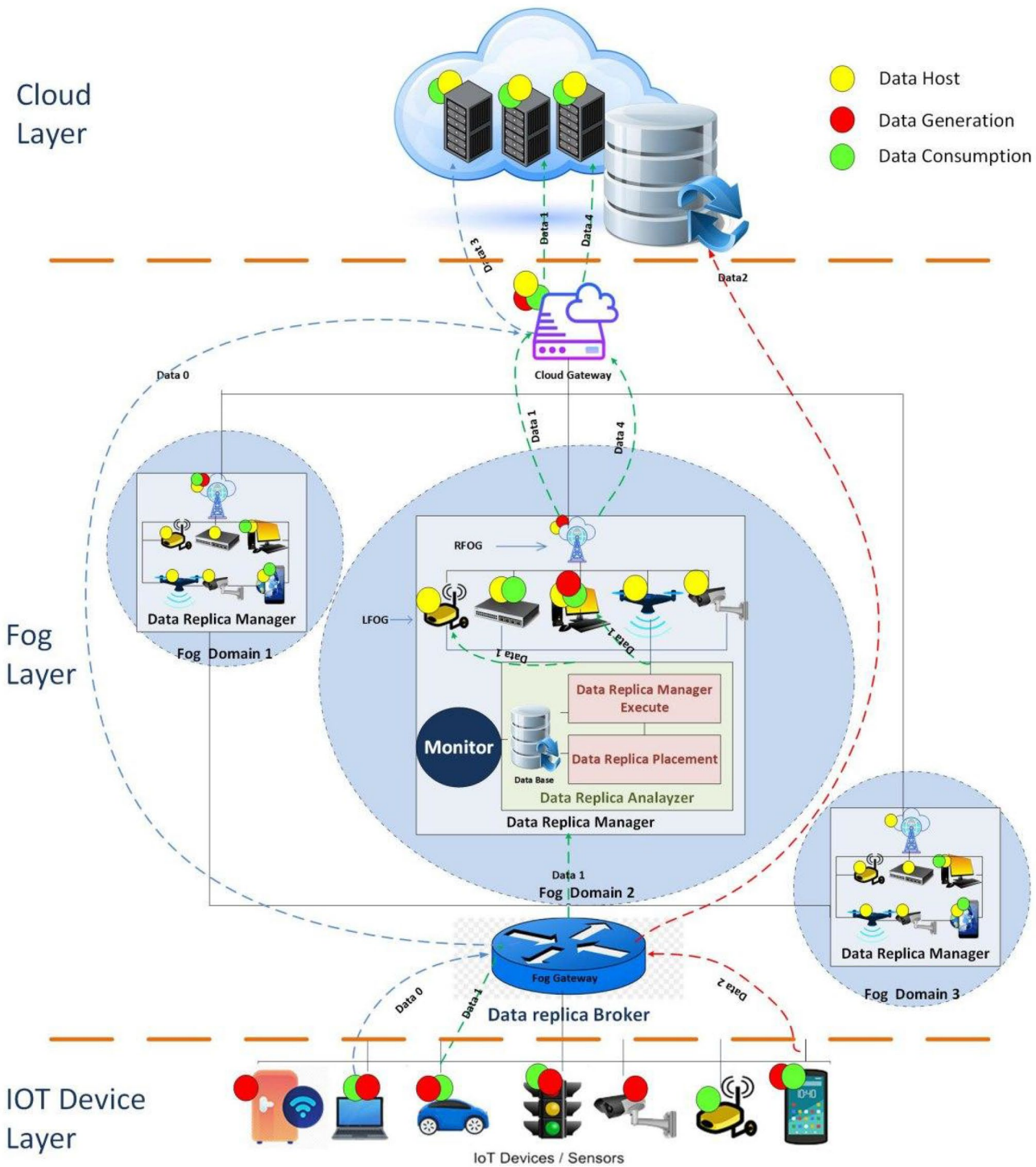


Fig. 1 The proposed framework

requirements of data-intensive IoT applications are satisfied. It dynamically determines the number of data replicas based on the history of data access requests received and their number of visits. Finally, the results obtained are stored in a shared database to use by other components.

3.1.2.3 Data replica placement After determining the number of data replicas needed, the data replica placement component appropriately places the data replicas on the LFOG nodes with various storage capabilities to reduce latency and costs of data access and increase reliability and avail-

ability of data replicas for IoT applications. In this work, we use BBO as a metaheuristic-based technique to find an appropriate data replica placement plan.

3.1.2.4 Data replica executer This component is responsible for performing the placement plan specified by the data replica placement component. It checks case-by-case the data replicas to determine whether or not to have been placed by the data replica placement component on the available LFOG nodes according to the number of data replicas specified by the data replica analyzer.

In addition to the physical entities mentioned, we consider three logical entities namely, data generator, data consumer, and data host. An IoT application can be both generator and consumer of the dataset, simultaneously. A data generator can send data generated by IoT devices to store on the LFOG nodes (i.e., data hosts). A data consumer is an entity that subscribes to access any one of the data replicas stored in the data host. Every LFOG node which has storage capability can be the data host. If the storage capacity of the data host is greater than the size of the data replica, it can host the data. Finally, all the data replicas must be stored on at least one of the LFOG nodes.

In the following, we provide a simple example for a better understanding of the data replica placement issue in the fog ecosystem. According to Fig. 1, three data replicas (i.e., $data_0$, $data_1$, $data_2$) generated by the data generators are transferred to the data replica broker. Depending on the type of generated data, the required resource capabilities for serving IoT request, the QoS requirements (e.g., deadline), and the data replica broker decides to transfer the data to the fog or cloud layer, as follows:

About scenario data replica $data_0$ (i.e., blue dotted line), it first arrives at the cloud gateway directly without entering the fog layer. Since the cloud gateway is the data generator, data consumer, and the data host at the same time, after processing $data_0$, the cloud gateway generates $data_3$ and sends it to the store on the cloud servers. Each cloud server in the cloud layer can act as both data consumer and data host.

About scenario data replica $data_1$ (i.e., green dotted line), it transfers to the data replica manager in fog domain 2. Afterward, the monitoring component extracts the required information about the data access request and available fog resources. Then, the data replica analyzer determines how many data replicas should be stored on the data hosts (i.e., LFOG nodes). Finally, the data replica placement identifies proper LFOG nodes using the BBO technique according to the size of data and the amount of available capacity data hosts to store two data replicas $data_1$ on the LFOG node2 and LFOG node3 into fog domain 2. Since one of the selected LFOG nodes is the data consumer and data host, $data_1$ is directly transfer through the RFOG node to the cloud

gateway, and then it is sent to the cloud layer to store on the appropriate cloud servers. On the other hand, another LFOG node acts as the data consumer, data generator, and data host. It processes the data replica $data_1$ and generates $data_4$ to transfer and store at the cloud layer. Finally, about scenario data replica $data_2$ (red dotted line), considering the data replica $data_2$ is a compute-intensive request, it directly forwards to the cloud servers within the cloud layer.

3.2 Problem statement

In this section, we provide the notations and equations used to solve data replica placement, as shown in Table 2. Let DG , DC , and DH be a set of data generators and data consumers, and data hosts, respectively, as follows:

$$DG = \{dg_1, dg_2, \dots, dg_I\} \tag{1}$$

$$DC = \{dc_1, dc_2, \dots, dc_J\} \tag{2}$$

$$DH = \{dh_1, dh_2, \dots, dh_K\} \tag{3}$$

where $dg_i \in DG$ denote the i th data generator, $dc_j \in DC$ denote the j th data consumer, and $dh_k \in DH$ denote the k th data host. Let $data_i$ denotes the dataset generated by dg_i and $size(data_i)$ denotes its size and cap_k be the storage capacity dh_k . Therefore, the storage capacity cap_k must be larger than the summation of storage usage of all data replicas, as follows:

$$\sum_{i=1}^I size(data_i) \times X_{ik} \leq cap_k; \quad \forall k, \quad k = 1, 2, \dots, K \tag{4}$$

where $X_{ik} = 1$, if there is at least one data replica $data_i$ on the data host dh_k , otherwise $X_{ik} = 0$, as follows:

$$X_{ik} = \begin{cases} 1 & \text{data replica is placed on the data host} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In this work, we consider four kinds of objective functions namely, total latency, total cost, reliability, and availability of data replicas, as follows:

3.2.1 Total latency

In this work, we describe the total latency as the summation of the uploading time to store data replicas on the data hosts and downloading time to retrieve data replicas from data hosts to the consumers. Let BW_{up} and BW_{down} denotes the network bandwidth capacity required to upload data replica $data_i$ on the data host and download data replica $data_i$ from the data host, respectively. Hence, the uploading time (i.e.,

Table 2 Notations and descriptions

Notation	Definition
dg_i	The i th data generator
dc_j	The j th data consumer
dh_k	The k th data host
$data_i$	The dataset generated by i th data generator
$size(data_i)$	The size of data replica $data_i$
cap_k	The storage capacity of k th data host
X_{ik}	Binary variable to indicate the data replica $data_i$ is placed on the k th data host
BW_{up}	The network bandwidth capacity required to upload data replica
BW_{down}	The network bandwidth capacity required to download data replica
T_{upload}	The uploading time required to store data replica on the data host
$T_{download}$	The downloading time required to obtain data replica from the data host
BW_{k_1,k_2}	The network bandwidth capacity between data host dh_{k_1} and dh_{k_2}
CC_{k_1,k_2}	The unit communication cost between data host dh_{k_1} and dh_{k_2}
uc_L	The unit cost for the data replica is stored in the LFOG node
uc_R	The unit cost for the data replica is stored in the RFOG node
uc_C	The unit cost for the data replica is stored in the cloud server
TL	The total latency
TC	The total cost
DSC	The data storage cost
DTC	The data transfer cost
$RE(dh_k)$	Reliability of data replica stored on the data host dh_k
$AV(dh_k)$	Availability of data replica stored on the data host dh_k

T_{upload}) and downloading time (i.e., $T_{download}$) are calculated as follows:

$$T_{upload} = \frac{size(data_i)}{BW_{up}} \tag{6}$$

$$T_{download} = \frac{size(data_i)}{BW_{down}} \tag{7}$$

Therefore, the total latency for accessing data replicas is calculated by Eq. (8):

$$TL(data_i, dh_k) = \sum_{i=1}^I \sum_{k=1}^K [(T_{download} \times X_{ik}) + (T_{upload} \times X_{ik})];$$

$$\forall j, \quad j = 1, 2, \dots, J \tag{8}$$

3.2.2 Total cost

The total cost for accessing data replicas placed on the data host include the data storage cost (i.e., DSC) and the data transfer cost (i.e., DTC), as is expressed by Eq. (9):

$$TC = DSC + DTC \tag{9}$$

3.2.2.1 Data storage cost We consider a unit cost uc for each data access request based on the storage location to access a data replica. Let uc_L , uc_R , and uc_C denote the unit cost for the data replica is stored in the LFOG node, the RFOG node, and the cloud server, as follows:

$$uc = \begin{cases} uc_L & \text{data replica is placed on the LFOG node} \\ uc_R & \text{data replica is placed on the RFOG node} \\ uc_C & \text{data replica is placed on the Cloud server} \end{cases} \tag{10}$$

Therefore, the unit cost for the data replica depends on the storage location and it is obvious that if the distance data transmission is longer, the unit cost to it will be higher, as follows:

$$uc_L \leq uc_R \leq uc_C \tag{11}$$

Finally, the cost of data storage for data replica data $data_i$ on the data host dh_k is determined by Eq. (12):

$$DSC(data_i, dh_k) = \sum_{i=1}^I \sum_{k=1}^K \left(\sum X_{ik} \times uc \times size(data_i) \right) \tag{12}$$

where uc is a unit cost for the data replica according to its location and X_{ik} denotes whether the data replica $data_i$ on the data host dh_k is stored.

3.2.2.2 Data transfer cost Let CC_{k_1,k_2} and BW_{k_1,k_2} denote the unit communication cost and the network bandwidth capacity between data host dh_{k_1} and dh_{k_2} , therefore, the data transfer cost is calculated by Eq. (13):

$$DTC(dh_{k_1}, dh_{k_2}) = \frac{size(data_i)}{BW_{k_1,k_2}} \times CC_{k_1,k_2} \tag{13}$$

3.2.3 Reliability

The reliability of the system or a subsystem is running the operations (e.g., data replication) under stable conditions for a certain time interval. The reliability of a data host is the success rate for replicating a data replica on the data host dh_k , and it is determined according to the percent of data replicas which is successfully stored on the data host. Let A_k

$$Maximize \sum_{i=1}^I \sum_{k=1}^K \left[\frac{1}{TL(data_i, dh_k)} + \frac{1}{TC(data_i, dh_k)} + RE(data_i, dh_k) + AV(data_i, dh_k) \right] \times X_{ik} \tag{18}$$

be the number of referred data replicas by a data host dh_k , and S_k be the number of data replicas accepted on the data host dh_k , the reliability of data host is obtained according to Eq. (14):

$$RE(dh_k) = \frac{S_k}{A_k} \tag{14}$$

Therefore, the total reliability for accessing data replicas on the data hosts is calculated by Eq. (15):

$$RE(data_i, dh_k) = \sum_{i=1}^I \sum_{k=1}^K [(RE(dh_k) \times X_{ik})]; \forall j, j = 1, 2, \dots, J \tag{15}$$

3.2.4 Availability

The availability is the degree to which a system or its component is able to use when it is needed or called. It is the probability of accessing the data replica requested by a data consumer from a data host. Suppose $dh_1, dh_2, \dots, dh_k, \dots, dh_K$ are data hosts within a fog domain; for any, $k = 1, 2, 3, \dots, K$ the percentage availability for a data host (i.e., LFOG node) within a fog domain is calculated using Eq. (16), as follows:

$$AV(dh_k) = \frac{MTTF_k}{MTBF_k} = \frac{MTTF_k}{MTTF_k + MTTR_k} \tag{16}$$

where $MTTF$ is the mean time in which the data host works correctly without failure, $MTTR$ is the mean time to repair the data hosts after failure, and $MTBF$ is the mean time between two failures in a data host.

Therefore, the total availability for accessing data replicas is calculated by Eq. (17):

$$AV(data_i, dh_k) = \sum_{i=1}^I \sum_{k=1}^K [(AV(dh_k) \times X_{ik})]; \forall j, j = 1, 2, \dots, J \tag{17}$$

3.2.5 Optimization problem

According to the objective functions mentioned, we use a multi-objective function as the total fitness function. Note that for any fog domain, the average latency, the average cost, the average availability, and the average reliability as objective functions will be calculated, as follows:

Subject to:

$$\sum_{i=1}^I size(data_i) \times X_{ik} \leq cap_k; \quad \forall k, \quad k = 1, 2, \dots, K \tag{19}$$

$$X_{ik} \in \{0, 1\}; \quad \forall i, \forall k \tag{20}$$

$$1 \leq i \leq I \tag{21}$$

$$1 \leq k \leq K \tag{22}$$

where $size(data_i)$ and cap_k denote the size of the data replica $data_i$ and the storage capacity of the data host dh_k . Besides, $X_{ik} = 1$ as a decision variable indicates whether at least one data replica $data_i$ has been deployed on the data host dh_k . Indeed, constraint (19) guarantees that the summation of storage space for the data replicas deployed on the data hosts should be less than storage capacity.

Besides, we should be considered preferences for storing the data replicas on the data hosts. To do this, we introduce scaling factors $\lambda_C, \lambda_L, \lambda_R$, and λ_A as the weighted parameters to reflect the preferences of the data access requests. Therefore, the objective function for the data replica placement problem is characterized by Eq. (23):

$$\text{Maximize} : Z = [\lambda_C \cdot \frac{1}{TC} + \lambda_L \cdot \frac{1}{TL} + \lambda_R \cdot RE + \lambda_A \cdot AV] \quad (23)$$

$$\text{s.t. } \lambda_C + \lambda_L + \lambda_R + \lambda_A = 1; \lambda_C, \lambda_L, \lambda_R, \lambda_A \in [0, 1] \quad (24)$$

Also, since the objective function values have various measurement units, we need to normalize the fitness values into the interval [0, 1]. In this study, we calculate the normalized fitness values of negative (i.e., latency and cost) and positive (i.e., availability and reliability) objective functions according to Eq. (25):

$$X_N = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (25)$$

where X_N , X_{\max} , and X_{\min} are the normalized, the maximum, and the minimum values of objective functions. Therefore, the normalized values of cost (i.e., TC_N), latency (i.e., TL_N), reliability (i.e., RE_N), and availability (i.e., AV_N) are calculated by Eqs. (26)–(29), respectively:

$$TC_N = \frac{TC - TC_{\min}}{TC_{\max} - TC_{\min}} \quad (26)$$

$$TL_N = \frac{TL - TL_{\min}}{TL_{\max} - TL_{\min}} \quad (27)$$

$$RE_N = \frac{RE - RE_{\min}}{RE_{\max} - RE_{\min}} \quad (28)$$

$$AV_N = \frac{AV - AV_{\min}}{AV_{\max} - AV_{\min}} \quad (29)$$

Finally, the weighted normalized objective function is expressed by Eq. (30):

$$\text{Maximize} : Z = \left[\lambda_C \times \frac{1}{TC_N} + \lambda_L \times \frac{1}{TL_N} + \lambda_R \times RE_N + \lambda_A \times AV_N \right] \quad (30)$$

3.3 Proposed algorithm

In this section, we explain the general structure of the proposed data replica manager (DRM), as shown in Algorithm 1. The DRM is regularly executed at specified time intervals to find a suitable data replica placement solution (lines 2–7). First, the monitoring phase gathers the needed information about the status available data hosts and data replicas through data access requests to store into a shared database (line 3). Then, the data replica analysis phase refines the data collected from the monitoring phase to determine the number of data replicas needed based on the history of data access requests (line 4). Afterward, the data

replica placement phase used the BBO technique to discover a proper data replica placement plan for storing the data replicas on the data hosts so that QoS requirements of data access requests are satisfied (line 5). Finally, the placement plan obtained from the previous phase performs by the data replica execution phase (line 6).

Algorithm 1: DRM

- 1: *Begin*
 - 2: **for each** (Time interval Δt until the system is running) **do**
 - 3: [Inputs]=Monitoring()
 - 4: Analyze=Data_Replica_Analaysis(Inputs)
 - 5: Plans=Data_Replica_Placement(Inputs, Analyze)
 - 6: Data_Replica_Execution(Plans)
 - 7: **end for each**
 - 8: *End*
-

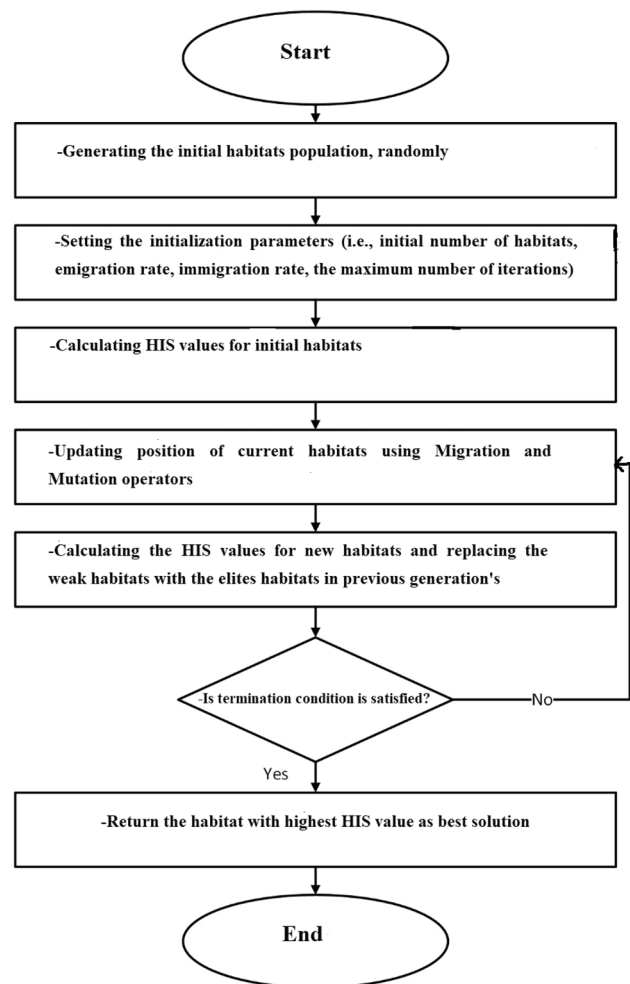


Fig. 2 The flowchart of the BBO-DRP algorithm

3.3.1 Monitoring

In this section, the monitoring phase extracts the required information from data access requests received from the data replica broker component at given time intervals, as shown in Algorithm 2. First, it collects the required information about the data access requests including the type of request (i.e., delay-sensitive or compute-sensitive), the number of required replicas, the required storage space, and the deadline time of request (line 4). Then, the latest status of resource capabilities for data hosts within the fog domains including the available storage space, the number of stored data replica, and the access delay are collected (line 7). Finally, this information is stored in a shared database to utilize by other phases (line 9).

Algorithm 2: Monitoring

Input: *data hosts, data access requests*

Output: *monitored_Inputs*

1: **Begin**

2: **for each** (Time interval Δt until the system is running) **do**

3: **for each** (data access request) **do**

4: **Monitor** ((Type_Req), (Num_Rep), (Space), (Deadline))

5: **end for each**

6: **for each** (data host dh_i into each fog domain) **do**

7: **Monitor** ((Num_stored - Rep), (Storagecapability), (accessdelay))

8: **end for each**

9: *DRM_DB* = Store(*monitored_Inputs*)

10: **return** *monitored_Inputs*

11: **End**

3.3.2 Data replica analysis

This phase used the information obtained from the previous phase to estimate the number of data replicas needed for storing on the data hosts, as shown in Algorithm 3. Since the access rate of IoT users to the data replicas varies over the time, determining the number of data replicas required to serve the data access requests is a challenging task and it depends on the amount of download demand of each data replica. Therefore, we used a linear regression model (Shahidinejad et al. 2021) as a prediction technique to specify the number of data replicas based on the history of data access requests (i.e., $Num_Rep(t)$). The general form of the linear regression model for data replica $data_i$ described by Eq. (31):

$$Pr_Num_Rep_i(t + 1) = \alpha + \beta \times Num_Rep_i(t) \tag{31}$$

where t indicates the t th instance observation, $Pr_Num_Rep_i(t + 1)$ as target variable (i.e., dependent variable) denotes the predicted number of data replica $data_i$, $Num_Rep_i(t)$ as an independent variable is actually the value of the number of data replica $data_i$ at the t th instance, and n is the total number of collected samples. In this prediction model, we use α and β as coefficients values to forecast future $Pr_Num_Rep_i(t + 1)$ values. These coefficients (i.e., α, β) calculated by solving a linear regression equation which is described by Eqs. (24) and (25):

$$\alpha = \frac{\sum (Num_Rep_i(t))^2 \times \sum Pr_Num_Rep_i(t) - \sum Num_Rep_i(t) \times \sum [Num_Rep_i(t) \times Pr_Num_Rep_i(t)]}{n \sum (Num_Rep_i(t))^2 - [\sum Num_Rep_i(t)]^2} \tag{32}$$

$$\beta = \frac{n \sum [Num_Rep_i(t) \times Pr_Num_Rep_i(t)] - \sum [Num_Rep_i(t)] \times \sum [Pr_Num_Rep_i(t)]}{n \sum [Num_Rep_i(t)]^2 - [\sum Num_Rep_i(t)]^2} \tag{33}$$

According to Algorithm 3, the prediction process consists of three phases: setting up windowing, training the model with linear regression, and generating the forecasts. In the windowing phase, we convert the time series data into a universal data set by setting windowing parameters such as window size, step size, and horizon (lines 4–5). Then, in the training phase, we utilized the linear regression technique to fit the dependent variable (i.e., $Pr_Num_Rep_i(t+1)$) and generate the predictions (lines 6–7). Finally, we need to collect and store the last prediction results in a separate data structure (lines 8).

with different resources capabilities. Indeed, this component specifies that a data replica requested to be placed on which LFOG node within that fog domain.

In the BBO technique (Simon 2008), we consider each solution as a habitat that is measured according to the habitat suitability index (HSI) as a fitness function. The habitat with a high HIS value is considered as a good solution and a habitat with a low HIS value is known as a weak solution. To improve the quality of solutions, the solutions with high HIS values share features (i.e., suitability index variables (SIVs) to other solutions with low HIS values through migration operator.

Algorithm 3: Pseudo-code for data replica Analysis Phase

```

1: Begin
2: Input: Monitored_Inputs, Number of data replica data; /* Num_Re pi(t) */
3: Output: Pr_Num_Re pi(t+1)
4: History Set ( datai ) ← History Set. Get( datai ) /*History of access rate for data replica datai */
5: History Set ( datai ) ← Windowing ( History Set ( datai ),window size, step size)
6: Calculate the coefficients  $\alpha$  and  $\beta$  using Eq. (32), (33)
7: Prediction results ←Predict the number of data replicas using Eq. (31)
8: Pr_Num_Re pi(t+1) ←Store (Prediction results)
9: Return Pr_Num_Re pi(t+1)
10: End

```

3.3.3 Data replica placement

After determining the number of data replicas required, the data replica placement component appropriately places the data replicas on the LFOG nodes with various storage capabilities to reduce latency and costs of data access and increase reliability and availability of data replicas for data-intensive IoT applications. The data replica placement as the main component of the data replica manager within each fog domain is responsible for finding an efficient data replica placement solution using the BBO as a population-based meta-heuristic technique named BBO-DRP for placing data replicas with various QoS requirements on the data hosts

In migration operator, a habitat H_x through immigration rate (λ) and habitat H_y through emigration rate (μ) is selected to migrate SIVs (i.e., genes) from habitat H_x to the habitat H_y for enhancing exploitation of the BBO-based algorithms. Besides, the mutation operator is performed to avoid local optimum on



Fig. 3 Habitat representation

some SIVs of some habitats which are chosen randomly with the probability of P_k . Finally, the overall structure of the BBO-DRP algorithm is provided step by step, as shown in Fig. 2.

In the following, we explain the BBO-DRP algorithm to find the most suitable placement plan for storing the data replicas in more detail, as shown in Algorithm 4. The input of the algorithm is a list of data replicas and available data hosts and also, the output is mapping data replicas on the fog nodes as optimal placement plan in the fog ecosystem.

First, we randomly generate the initial habitat population (i.e., the initial candidate solution) according to the list of data replicas (i.e., data access requests) and available data hosts (i.e., fog nodes). Each data replica $data_i$ is randomly assigned to a data host dh_k and a set of habitats are generated (lines 2–8), as shown in Fig. 3. Each habitat (i.e., a candidate solution) has k dimensions (i.e., number of data hosts), and each dimension has a value between 1 and i (i.e., the number of available data replicas for each data host) and it is evaluated according to HIS value. Since the data replica placement is a discrete problem, we utilized an integer version BBO technique to solve it.

To initialize the habitats, we used an initial function, as expressed by Eq. (34):

$$H(i, k) = [ub(i) - lb(i)] * rand() + lb(i) \tag{34}$$

where lb_i and ub_i represent the lower and the upper bounds for assigning i -th data replica to k -th data host, as follows:

$$lb = [lb_1, lb_2, lb_3, \dots, lb_{n-1}, lb_n] \tag{35}$$

$$H_x(SIV_k) = \begin{cases} H_y(SIV_k) & \text{If}(r_1 < I_x) / * \text{pick solution } H_y \text{ for emmigration} * / \\ H_x(SIV_k) & \text{Otherwise} / * \text{No - migration} * / \end{cases} \tag{40}$$

$$\forall x, y \in \{1, 2, \dots, x, \dots, X\}; SIV \forall k \in \{1, 2, \dots, k, \dots, K\}$$

$$ub = [ub_1, ub_2, ub_3, \dots, ub_{n-1}, ub_n] \tag{36}$$

Then, the HIS values of the initial habitats generated will be calculated according to the fitness function expressed by Eq. (30) (lines 9–10). Note that the purpose of the fitness function (i.e., HIS) is to find the best habitat to increase the positive criteria (i.e., reliability and availability) and the reduction of negative criteria (i.e., latency and cost). Afterwards, the main body of the BBO-DRP algorithm will be repeated until the termination condition is satisfied (lines 11–31). In this work, we take into account the maximum

number of iterations as the termination condition to find a suitable solution (line 11).

If the termination condition is not satisfied, the migration and mutation processes will be started for updating the parameters and generating the new candidate solutions (i.e., habitats) into the current iteration. First, all candidate solutions will be arranged based on their HIS values in descending order. Then, for each solution H_x , the immigration rate (λ_x) and the emigration rate (μ_x) will be calculated (lines 13–14). Note that the best solution has high μ_x and low λ_x , which are calculated by Eqs. (37) and (38) (Simon 2008):

$$\lambda_x = I_x(1 - \frac{Rank(H_x)}{X}) \tag{37}$$

$$\mu_x = E_x(1 - \frac{Rank(H_x)}{X}) \tag{38}$$

where $Rank(H_x)$ is the HIS rank of a solution H_x , X is the number of solutions, I_x and E_x are the maximum immigration rate and the maximum emigration rate for the solution H_x , respectively. It is clear that a candidate solution H_x with higher μ_x has more chance to be chosen for emigration. To perform migration process, we will select two habitats H_x and H_y according to I_x and E_y to share features (i.e., SIVs) with each other (line 15), as follows:

$$H_x(SIV) \leftarrow H_y(SIV) \tag{39}$$

Then, a random number r_1 in $[0, 1]$ is generated. If r_1 is less than I_x , the migration is launched, as follows:

In the migration process, one location is randomly selected between 1 and the size of the habitat (i.e., k). Then, from the selected location (i.e., SIV) to the last location, all SIVs (i.e., genes) belongs to the habitat H_y are shifted in habitat H_x . Similarly, all habitats are modified until the best solution is found (lines 16–20). The main aim of the migration process is that the worst solutions will accept some features from desirable solutions to improve exploitations.

After the migration process, we perform the mutation process on the solution H_x . First, a random number r_2 in $[0, 1]$ is generated. If r_2 is less than mutation probability P_k , the mutation process is performed and a random location (i.e.,

SIV) is selected from one solution H_x and its value modifies randomly in $[1, X]$ (lines 21–25), as follows:

$$H'_x(SIV_k) = \begin{cases} Rand[1, X] & \text{if } (r_2 < P_k) \\ H_x(SIV_k) & \text{Otherwise} \end{cases}$$

$$\forall x \in \{1, 2, \dots, x, \dots, X\}; SIV \quad \forall k \in \{1, 2, \dots, k, \dots, K\} \quad (41)$$

Therefore, a number of new candidate solutions using migration and mutation process are generated at each iteration, and then their HIS values are calculated by Eq. (30).

Then, an appropriate number of best solutions as elite habitats of each generation are saved for utilizing in the next iterations. In the next generation, the candidate solutions with low HIS values are replaced with the elite habitats obtained from the previous iteration's (lines 27–29). This ensures that HIS values of habitats (i.e., candidate solutions) are not degraded in the next iterations. Finally, all the mentioned above steps are repeated until the termination criterion is satisfied and the best found habitat with the highest HIS value would be selected as the best solution for the data replica placement problem.

Algorithm 4: BBO-DRP

Input: List of available data hosts, List of all data access requests (i.e., data replicas)

Output: An optimal data replica placement plan with highest HIS /*A mapping of data replicas to data hosts */

1: **Initialization:** The initial population size X (i.e., number of habitats), emigration (external migration) rate μ , immigration (internal migration) rate λ , the maximum number of iterations (Max_i)

/*Generate the initial habitats (i.e., initial candidate solutions), randomly $H_x, x = 1, 2, \dots, X$ */

2: **for each** habitat $x (H_x)$ **do**

3: **for each** ($data_i$ into DG) **do** /* $i=1$ to I */

4: **for each** (dh_k into DH) **do** /* $k=1$ to K */

5: $H_x(i, k) = [ub(i) - lb(i)] * rand() + lb(i)$

6: **end for**

7: **end for**

8: **end for**

/*Calculate the HSI for each habitat into H according to Eq.(30)*/

9: **for each** habitat $x (H_x)$ **do**

10:

$$Fitness(H_x(i, k)) = \sum_{i=1}^I \sum_{k=1}^K [\lambda_C \times \frac{1}{TC_N(data_i, dh_k)} + \lambda_L \times \frac{1}{TL_N(data_i, dh_k)} + \lambda_R \times RE_N(dh_k) + \lambda_A \times AV_N(dh_k)] \times X_{ik}$$

/*Main loop*/

11: **while** ($t < Max_i$) **do**

```

12: for each habitat  $x$  do /*Generating the new habitats using Migration and Mutation process*/
13:   Sorting all habitats based on their HIS values in descending order
      /*Migration process according to Eq.(40)*/
14:   for each habitat  $H_x$ , calculate immigration rate  $\lambda_x$  and emigration rate  $\mu_x$  according to Eqs. (37-38)*/
15:   Select two habitats  $H_x$  and  $H_y$  according to  $I_x$  and  $E_y$  to share SIVs
16:   Generate a random number  $r_1$  in [0, 1]
17:   if ( $r_1 < I_x$ ) then
18:     Choose a random location  $m$ , where  $1 \leq m \leq k$ 
19:     Shift SIVs belongs to habitat  $H_y$  from location  $m$  toward location  $k$  into the habitat  $H_x$ .
20:   end if
      /*Mutation process according to Eq.(41)*/
21:   Select a candidate solution  $H_x$  with the highest mutation probability  $P_k$ 
22:   Generate a random number  $r_2$  in [1, X] /*X is the number of solutions*/
23:   if ( $r_2 < P_k$ ) then
24:     Choose a random location (i.e., SIV) from a candidate solution  $H_x$  and modifies its value
25:   end if
26: end for
27: Calculate the HIS for new habitats into H according to Eq.(30)
      /*Elitism process*/
28: Replace the worst habitats with the elites habitats in the previous generation's
29: Keep proper elite habitats for the next generation.
30:  $t=t+1$ 
31: end while
32: return  $H_x^*$  /*the habitat with highest HIS value is selected as best placement solution*/

```

3.3.4 Data replica execution

The data replica execution phase is in charge of placing the data replicas on the data hosts according to the data replica placement plan specified by the BBO-DRP algorithm and the number of replicas determined by data replica analysis (i.e., Algorithm 3), as shown in Algorithm 5. First, the data replicas have been examined case-by-case to specify whether or not has been found in the previous phase any appropriate place on the available host nodes within the fog domains. If a suitable data host dh_k is found, the data replica $data_i$ is replicated on that LFOG node according to the number of data replicas estimated by the data replica analysis phase.

Algorithm 5: Data Replica Execution Phase

Input: Placement plan (i.e., H_x^*), Prediction ($data_i$)

1: *Begin*

2: *Read(DB)*

2: **for each** (*data replicas* $data_i$) **do**

3: *Replication*($data_i$, *Placement Plan*, *Prediction*)

4: **end for each**

5: *End*

4 Performance evaluation

To validate the proposed approach, we will describe the simulation setting and criteria for an evaluation in more detail. Then, the experimental results will be analyzed.

4.1 Experimental setup

In this section, we provide the simulation setting to develop experiments for validating the effectiveness of the proposed mechanism in more detail. To model and simulate the fog ecosystem and IoT services, we utilize the iFogSim toolkit (Naas et al. 2018b) with some modifications of the classes to characterize the fog infrastructure, IoT applications, and data replica placement algorithms. It provided a series of classes for modeling fog infrastructure (e.g., *Sensor*, *Actuator*, and *Fog Device* classes) and IoT applications (e.g., *AppEdge*, *Tuple*, and *AppModule* classes). With regard to simulation setting, there are two simulation types, namely: transient and steady-state. Since we will study the long-term behaviors of baseline mechanisms, we consider steady-state simulation policy. Besides, there are various solutions for the estimation of steady-state such as the batch means strategy, the regenerative strategy, the autoregressive strategy, the spectral estimation strategy, and the standard time-series strategy.

Table 3 Data host's configuration

Data host	Number	Storage capacity
Gateway	500–1000	1 GB
RPOP	10–30	500 GB
LPOP	50–70	100 GB
Cloud servers	5–10	1 TB

Table 4 Network latency configuration

Communication link	Link delay (ms)
Sensor-Broker	10
Broker-LFOG	50
LFOG-RFOG	5
RFOG-Cloud	100
Cloud-Cloud	100

To validate the effectiveness of the proposed mechanism, we utilize the batch means strategy to analyze the simulation results. Also, each simulation periods for 5 min and values in the figures are the average value of obtained results by performing the baseline mechanisms 20 times. Also, we consider RFOGs, LFOGs, gateways, and cloud servers as data hosts with resource capabilities and network latency, as shown in Tables 3 and 4, respectively. Besides, we used the random uniform distribution to generate the synthetic dataset in a simulated fog network, as shown in Table 5. Also, we performed the simulation experiments on a personal computer with Intel Core i7 2.4 GHz CPU, 1 TB disk, and 8 GB of RAM running Windows 10-64bit.

4.2 Performance metrics

We used the following metrics to evaluate the effectiveness of the proposed approach compared with other mechanisms.

Total latency: This metric is described as the summation of the uploading time and downloading time to store and retrieve the data replicas on the data hosts, as is calculated by Eq. (8).

Total cost: This metric is defined as the summation of the data transfer cost and the data storage cost, as is expressed by Eq. (9).

Reliability: This metric is specified as the success rate for storing the data replicas on a data host in a given time interval, and it is determined by Eq. (15).

Availability: This metric is the probability of a successful accessing the data replicas requested by a data consumer from a data host, as is calculated by Eq. (17).

Table 5 Simulation parameter setting

Parameter	Value
Data replica size	The random uniform distribution between [1, 20] (GB)
Data access request rate	The random uniform distribution between [10, 100] (req/s)
Network bandwidth capacity	The random uniform distribution between [5, 100] (bps)

4.3 Experimental results

To evaluate the effectiveness of the proposed data replica placement mechanism, we carried out a series of experiments under different scenarios, as shown in Table 6. Also, we compare the proposed solution (i.e., BBO-DRP) with the following data replica placement mechanisms in terms of different performance metrics:

- LA-DRP (Huang et al. 2019): The latency-aware (LA) mechanism utilized a greedy-based strategy to solve data replica placement (DRP) with aim of minimization the total latency. It reduces the search space by pruning the inappropriate solutions using different heuristic rules to provide a reasonable solution in polynomial time.
- GP-DRP (Shao et al. 2019): This strategy developed for replicating the data files for IoT workflows in an edge-cloud ecosystem. It used a meta-heuristic technique to find an efficient data replica placement plan for minimizing the data access cost while satisfying QoS requirements such as user cooperation and data reliability.

The reason for choosing these mechanisms is that these follow the centralized policy to find an efficient data replica placement plan, i.e., all aspects of the placement process including monitoring, analysis, and planning are controlled by a central component. Another reason is that these mechanisms are dynamic, i.e., the number of data replicas are automatically and dynamically added and removed to handle the changes in storage capacity and the varying-time data access request patterns.

4.3.1 First scenario

Figure 4 illustrates the effectiveness of our proposed solution (i.e., BBO-DRP) compared with LA-DRP and GP-DRP

Table 6 Specification of simulation scenarios

Scenario	Number of data hosts	Number of data replicas
First scenario	10, 20, 30, 40, 50	3
Second scenario	100	3, 5, 7, 9
Third scenario	Random distribution in [0–3000]	5

mechanisms with varying the number of data hosts. According to obtained results in terms of total latency (Fig. 4a), we realize that the BBO-DRP mechanism significantly reduces the average latency compared with LA-DRP and GP-DRP mechanisms. Indeed, if the data replicas are located near to the data consumer, the data access latency time will be reduced and subsequently, the overall latency will also be reduced. As the number of data replicas increases, the data will be closer to the data consumers and the data access delay will be reduced. According to results in the case where the number of fog nodes is equal to 50, the BBO-DRP mechanism reduces the average latency by up to 0.2% and 0.93% compared with the LA-DRP and GP-DRP mechanisms, respectively.

According to the results in terms of the total cost (Fig. 4b), we realize that the BBO-DRP mechanism significantly reduces the average cost by 31% and 53% compared with LA-DRP and GP-DRP mechanisms, respectively. This is because that our proposed solution utilized a sub-component (i.e., monitoring component) to collect information about the data access requests and the status data hosts. Besides, our mechanism used BBO as a multi-objective optimization technique to make appropriate constraints for achieving high accuracy to find an efficient data replica placement solution. As a result, we noticed that despite the increase in fog nodes, which creates higher computations and consequently the cost will be higher than, our proposed mechanism reduces the average cost compared with other methods. For example, in the case where the number of fog nodes is equal to 50, the proposed mechanism reduces the average total cost by up to 0.4% and 0.21% compared with the LA-DRP and GP-DRP mechanisms, respectively.

Finally, According to the obtained results, we find out that the proposed solution outperforms in terms of availability and reliability metrics compared with baseline mechanisms, as shown in Fig. 4c, d. It improves reliability by 12% and 20% and also the availability by 17% and 9% compared with LA-DRP and GP-DRP mechanisms, respectively. By increasing the number of data hosts, the percentage of reliability and availability in all mechanisms have increased. This is due to the increase in the number of data hosts than data consumers, and as a result, this enhances the reliability and availability of data replicas. Besides, since our proposed mechanism collects the required information about the status data storage location through a monitoring component, it causes a higher availability rate compared with other

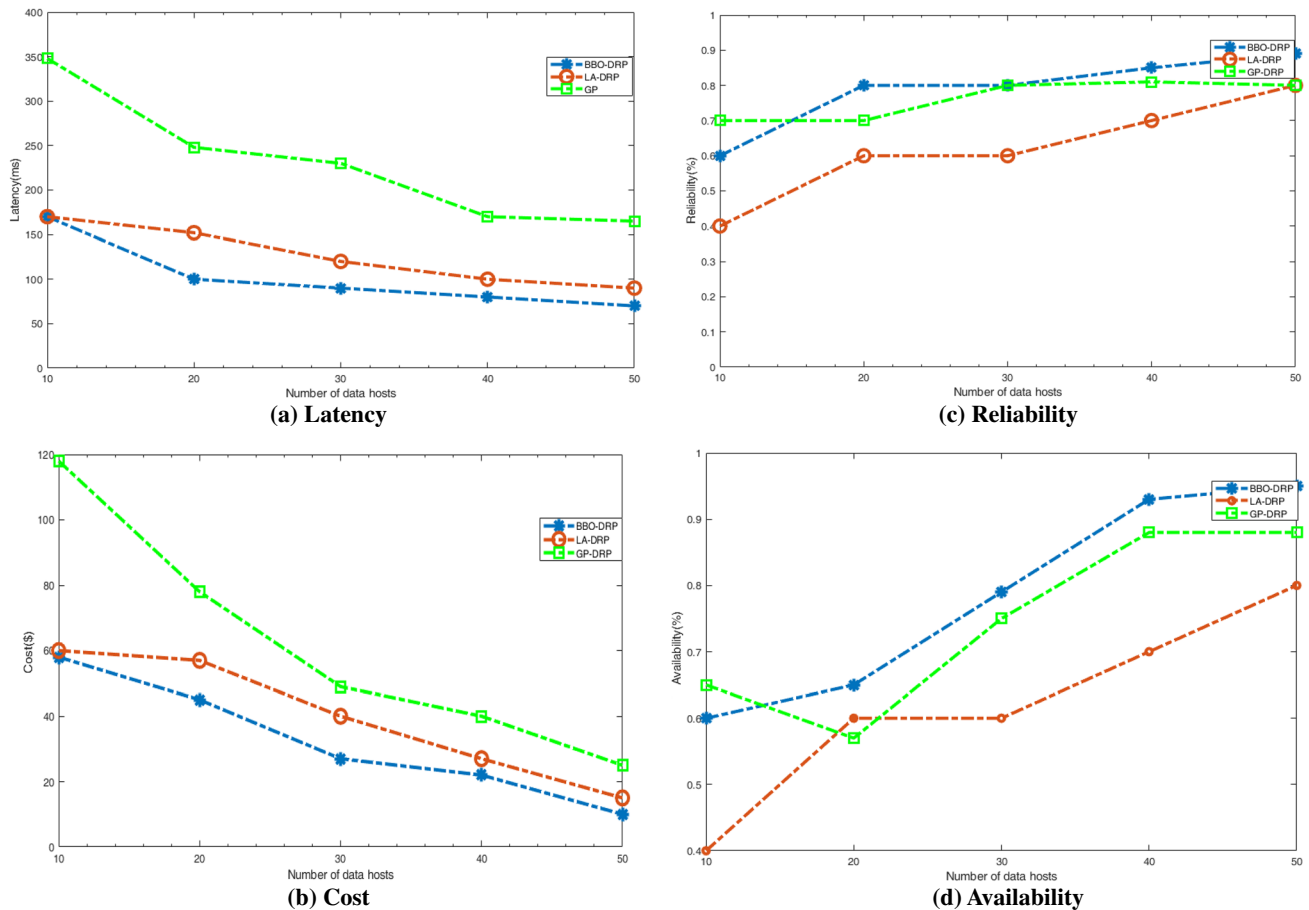


Fig.4 Comparison performance metrics in the first scenario **a** latency, **b** cost, **c** reliability, **d** availability

mechanisms. Note that the availability is an important QoS metric in the field of data replica placement so that when IoT devices communicate with data hosts, they can access the data replicas as quickly as possible.

4.3.2 Second scenario

Figure 5 depicts the average latency, cost, reliability, and availability of the baseline and proposed mechanism with 100 data hosts and a different number of data replicas. In general, according to results in the second scenario, we noticed that the BBO-DRP is able to work better in terms of specified performance criteria than all baseline mechanisms. It reduces the total cost by up to 5% and 15% and it increases the reliability by 13% and 16% and also the availability by 27% and 9% compared with LA-DRP and GP-DRP mechanisms, respectively. Since the LA-DRP is a greedy-based solution and due to its simple nature than other algorithms, it achieves a near-optimal solution very fast using some heuristic-based rules as soon as possible. Thus, the latency of the LA-DRP algorithm is lower than the GP-DRP algorithm, as shown in Fig. 5a. On the other

hand, since the GP-DRP utilized the shortest path between fog nodes to find the appropriate location for storing data replicas, the discovered paths may not always be the best path with the least amount of latency, it will perform worse than the proposed mechanism. Also, the GP-DRP algorithm follows the divide and conquer policy by partitioning the fog ecosystem into several geographical regions to solve the data replica placement problem, separately. Transferring results between regions can lead to a loss of optimality, subsequently, resulting in increased latency and cost. According to obtained results in terms of cost (Fig. 5b), we find out that the BBO-DRP mechanism outperforms compared with other mechanisms due to replacing the candidate solutions with low HIS values using saved elite habitats to select the proper number of data replicas for managing the storage capacity of data hosts, efficiently. Finally, we find out that the proposed solution has better performance in terms of availability and reliability metrics compared with two baseline mechanisms, as shown in Fig. 5c, d. According to the results of Fig. 5c, we find out that as the number of data hosts are grown; consequently, the reliability in both BBO-DRP and LA-DRP methods have increased. Besides, the BBO-DRP mechanism

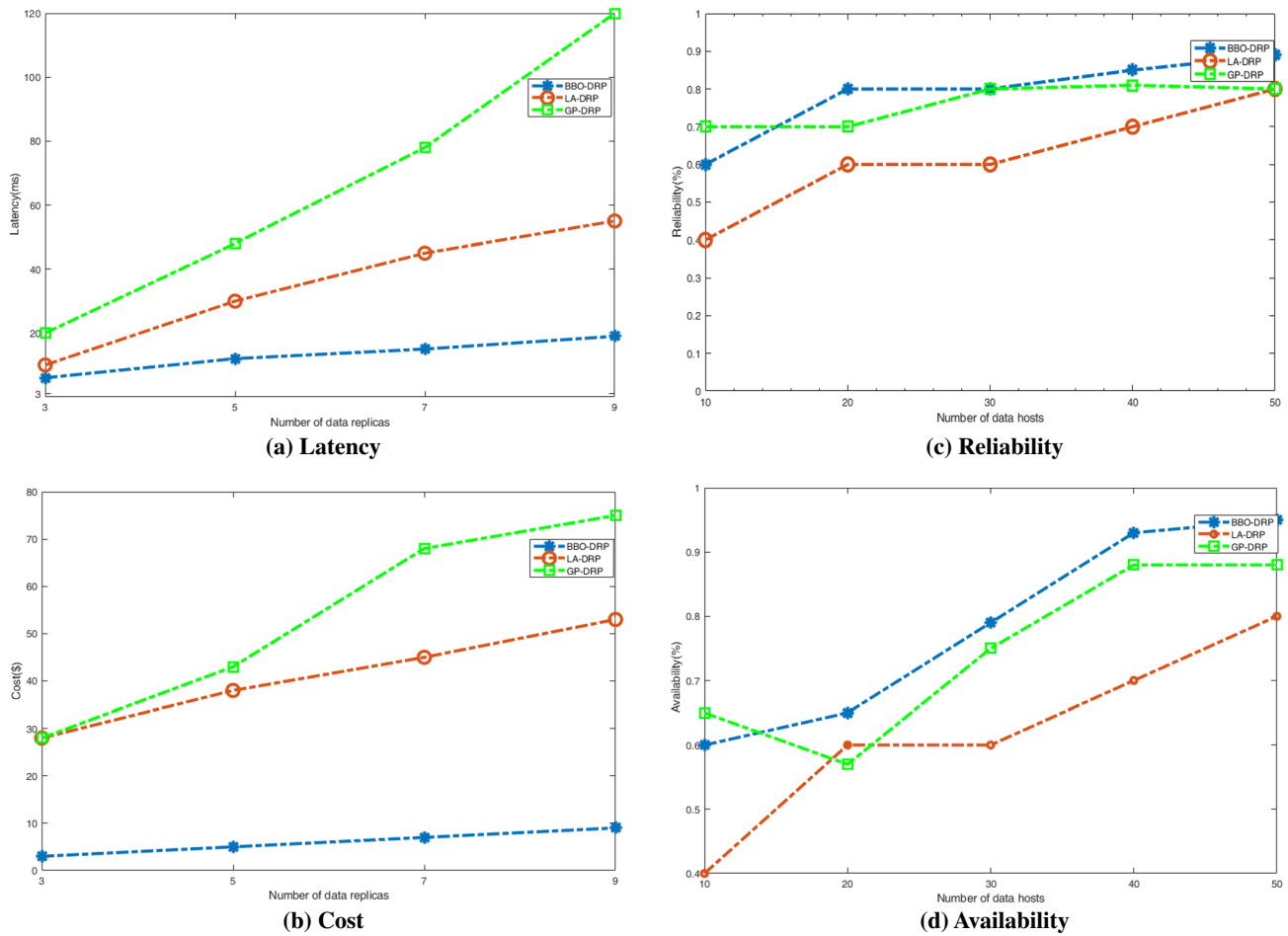


Fig.5 Comparison performance metrics in the second scenario **a** latency, **b** cost, **c** reliability, **d** availability

successfully stores more data replicas on the data hosts than the LA-DRP method. Therefore, it enhances the reliability compared with LA-DRP method. Also, since the proposed solution provides a more appropriate distribution of data replica between data hosts, it achieves high availability than LA-DRP and GP-DRP algorithms, as shown in Fig. 5d.

4.3.3 Third scenario

In this scenario, we will discuss the solving time of the proposed solution with the LA-DRP and GP-DRP mechanisms, as shown in Fig. 6. Let the number of data replica is 5 and the number of data hosts is varying between 500 and 3000 nodes. According to the obtained results, the proposed mechanism reduces the average solving time by up to 0.07% and 0.75% compared with the LA-DRP and GP-DRP mechanisms, respectively. This is because the LA-DRP mechanism used a greedy-based policy with some heuristic-based rules to find a data replica placement solution as soon as possible. Therefore, the LA-DRP mechanism DRP is able to work better in terms of solving time than the GP-DRP

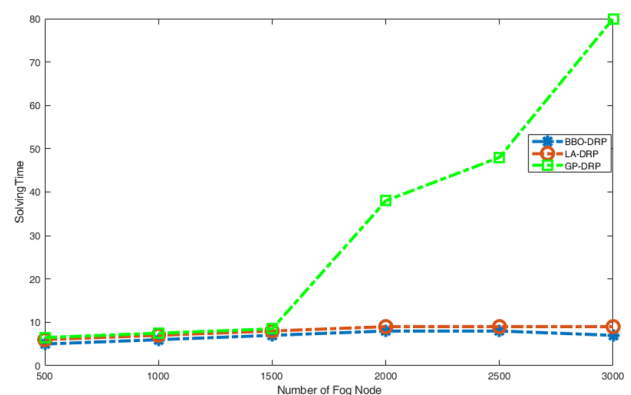


Fig.6 Comparison of solving time for different mechanisms

mechanism. Besides, the GP-DRP mechanism partitions the fog infrastructure into several geographical regions to reduce the total solution time. However, the time required to transfer the results between regions will increase the total solution

time. Thus, it will perform worse than the LA-DRP and the proposed mechanisms.

5 Conclusion

In this study, we addressed the data replica placement problem for processing data-intensive IoT applications in the fog ecosystem. Due to the generation of a large amount of data by IoT devices and the variety of the fog nodes capabilities, the data replica placement strategies can play an important role for enhancing the system performance and they should be considered as one of the challenging issues. We used BBO meta-heuristic technique to place data replicas on the storage fog nodes for minimizing costs of data access and latency and increase reliability and availability of data replicas while meeting the QoS requirements of IoT applications. Further, we develop an autonomous framework for placing data replicas to illustrate transferring them between IoT devices and storage fog nodes in the fog ecosystem. We validate the proposed mechanism under a different number of data replicas and fog nodes, and the obtained experimental results indicated that it can work better in terms of cost, latency, reliability, and availability than other baseline mechanisms. For future work, we have planned to extend our proposed mechanism with blockchain-based systems to ensure privacy-preserving and data integrity in the fog computing environment. Further, we will utilize the deep Q-learning technique to identify appropriate fog nodes for storing the datasets generated by the IoT devices, dynamically. Finally, we will design a hybrid solution using the whale optimization algorithm (WOA) and simulated annealing (SA) techniques to present an efficient data replica placement mechanism and also, using the formal verification-based techniques to confirm its soundness.

Author contributions MT, AS, MG-A conducted this research. MT: methodology, software, validation, writing original draft. AS: conceptualization, supervision, writing review and editing, formal analysis, project administration. MG-A: investigation, resources, data curation, visualization.

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

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest We certify that there is no actual or potential conflict of interest in relation to this article.

References

- Alvarez F, Breitgand D, Griffin D, Andriani P, Rizou S, Zioulis N, Moscatelli F, Serrano J, Keltsch M, Trakadas P, Phan TK (2019) An edge-to-cloud virtualized multimedia service platform for 5G networks. *IEEE Trans Broadcast* 65(2):369–380
- Alweshah M (2019) Construction biogeography-based optimization algorithm for solving classification problems. *Neural Comput Appl* 31(10):5679–5688
- Aral A, Ovatman T (2018) A decentralized replica placement algorithm for edge computing. *IEEE Trans Netw Serv Manage* 15(2):516–529
- Breitbach M, Schäfer D, Edinger J, Becker C (2019) Context-aware data and task placement in edge computing environments. In: 2019 IEEE international conference on pervasive computing and communications (PerCom). IEEE, pp 1–10
- Chen Y, Deng S, Ma H, Yin J (2019) Deploying data-intensive applications with multiple services components on edge. *Mobile Netw Appl* 25:1–16
- Confais B, Parrein B, Lebre A (2018) A tree-based approach to locate object replicas in a fog storage infrastructure. In: 2018 IEEE global communications conference (GLOBECOM). IEEE, pp 1–6
- Costa Filho JS, Cavalcante DM, Moreira LO, Machado JC (2020) An adaptive replica placement approach for distributed key-value stores. *Concurr Comput Pract Exp* 32(11):e5675
- Dadashi Gavaber M, Rajabzadeh A (2021) MFP: an approach to delay and energy-efficient module placement in IoT applications based on multi-fog. *J Ambient Intell Human Comput* 12:7965–7981. <https://doi.org/10.1007/s12652-020-02525-7>
- Devadas TJ, Thayammal S, Ramprakash A (2020) IoT data management, data aggregation and dissemination. *Principles of internet of things (IoT) ecosystem: insight paradigm*. Springer, Cham, pp 385–411
- Goudarzi S, Anisi MH, Abdullah AH, Lloret J, Soleymani SA, Hassan WH (2019) A hybrid intelligent model for network selection in the industrial Internet of Things. *Appl Soft Comput* 74:529–546
- Guerrero C, Lera I, Juiz C (2019) Optimization policy for file replica placement in fog domains. *Concurr Comput Pract Exp* 32:e5343
- Habibi P, Farhoudi M, Kazemian S, Khorsandi S, Leon-Garcia A (2020) Fog computing: a comprehensive architectural survey. *IEEE Access* 8:69105–69133
- Huang T, Lin W, Li Y, He L, Peng S (2019) A latency-aware multiple data replicas placement strategy for fog computing. *J Signal Process Syst* 91(10):1191–1204
- Karatas F, Korpeoglu I (2019) Fog-based data distribution service (F-DAD) for internet of things (IoT) applications. *Futur Gener Comput Syst* 93:156–169
- Khorsand R, Ghobaei-Arani M, Ramezanpour M (2018) FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments. *Softw Pract Exp* 48(12):2147–2173
- Kumari A, Tanwar S, Tyagi S, Kumar N, Parizi RM, Choo KKR (2019) Fog data analytics: a taxonomy and process model. *J Netw Comput Appl* 128:90–104
- Li C, Tang J, Luo Y (2019a) Scalable replica selection based on node service capability for improving data access performance in edge computing environment. *J Supercomput* 75(11):7209–7243
- Li C, Wang Y, Chen Y, Luo Y (2019b) Energy-efficient fault-tolerant replica management policy with deadline and budget constraints in edge-cloud environment. *J Netw Comput Appl* 143:152–166
- Martin JP, Kandasamy A, Chandrasekaran K (2020) Mobility aware autonomic approach for the migration of application modules in fog computing environment. *J Ambient Intell Humaniz Comput* 11:1–20

- Mayer R, Gupta H, Saurez E, Ramachandran U (2017) Fogstore: toward a distributed data store for fog computing. In: 2017 IEEE Fog World Congress (FWC). IEEE, pp 1–6
- Monga SK, Ramachandra SK, Simmhan Y (2019) ElfStore: a resilient data storage service for federated edge and fog resources. In: 2019 IEEE international conference on web services (icws). IEEE, pp 336–345
- Mukherjee M, Shu L, Wang D (2018) Survey of fog computing: fundamental, network applications, and research challenges. *IEEE Commun Surv Tutor* 20(3):1826–1857
- Naas MI, Parvedy PR, Boukhobza J, Lemarchand L (2017) iFogStor: an IoT data placement strategy for fog infrastructure. In: 2017 IEEE 1st international conference on fog and edge computing (ICFEC). IEEE, pp 97–104
- Naas MI, Lemarchand L, Boukhobza J, Raipin P (2018a) A graph partitioning-based heuristic for runtime IoT data placement strategies in a fog infrastructure. In: Proceedings of the 33rd annual ACM symposium on applied computing, pp 767–774
- Naas MI, Boukhobza J, Parvedy PR, Lemarchand L (2018b) An extension to ifogsim to enable the design of data placement strategies. In: 2018 IEEE 2nd international conference on fog and edge computing (ICFEC). IEEE, pp 1–8
- Nikoui TS, Rahmani AM, Tabarsaied H (2019) Data management in fog computing. In: Fog and edge computing: principles and paradigms, pp 171–190
- Pal R, Saraswat M (2019) Histopathological image classification using enhanced bag-of-feature with spiral biogeography-based optimization. *Appl Intell* 49(9):3406–3424
- Paraskevopoulos A, Dallas PI, Siakavara K, Goudos SK (2017) Cognitive radio engine design for IoT using real-coded biogeography-based optimization and fuzzy decision making. *Wirel Pers Commun* 97(2):1813–1833
- Punithallayarani P, Dominic MM (2019) Anatomization of fog computing and edge computing. In: 2019 IEEE international conference on electrical, computer and communication technologies (ICECCT). IEEE, pp 1–6
- Reihanian A, Feizi-Derakhshi MR, Aghdasi HS (2017) Community detection in social networks with node attributes based on multi-objective biogeography based optimization. *Eng Appl Artif Intell* 62:51–67
- Sangaiah AK, Bian GB, Bozorgi SM, Suraki MY, Hosseinabadi AAR, Shareh MB (2019) A novel quality-of-service-aware web services composition using biogeography-based optimization algorithm. *Soft Comput* 24:1–13
- Sengupta S, Bhunia SS (2020) Secure data management in cloudlet assisted IoT enabled e-health framework in Smart City. *IEEE Sens J* 20:9581–9588
- Shahidinejad A, Ghobaei-Arani M (2020) Joint computation offloading and resource provisioning for edge-cloud computing environment: a machine learning-based approach. *Softw Pract Exp* 50(12):2212–2230
- Shahidinejad A, Ghobaei-Arani M, Masdari M (2021) Resource provisioning using workload clustering in cloud computing environment: a hybrid approach. *Clust Comput* 24(1):319–342
- Shao Y, Li C, Tang H (2019) A data replica placement strategy for IoT workflows in collaborative edge and cloud environments. *Comput Netw* 148:46–59
- Silva DMAD, Asaamoning G, Orrillo H, Sofia RC, Mendes PM (2019) An analysis of fog computing data placement algorithms. In: Proceedings of the 16th EAI international conference on mobile and ubiquitous systems: computing, networking and services, pp 527–534
- Simon D (2008) Biogeography-based optimization. *IEEE Trans Evol Comput* 12(6):702–713
- Trakadas P, Simoens P, Gkonis P, Sarakis L, Angelopoulos A, Ramallogonzález AP, Skarmeta A, Trochoutsos C, Calvo D, Pariente T, Chintamani K (2020) An artificial intelligence-based collaboration approach in industrial IoT manufacturing: key concepts. *Archit Ext Potential Appl Sens* 20(19):5480
- Zhang M, Jiang W, Zhou X, Xue Y, Chen S (2019) A hybrid biogeography-based optimization and fuzzy C-means algorithm for image segmentation. *Soft Comput* 23(6):2033–2046
- Zheng Q, Li R, Li X, Shah N, Zhang J, Tian F, Chao KM, Li J (2016) Virtual machine consolidated placement based on multi-objective biogeography-based optimization. *Futur Gener Comput Syst* 54:95–122
- Zhou X, Liu Y, Li B, Sun G (2015) Multiobjective biogeography based optimization algorithm with decomposition for community detection in dynamic networks. *Phys A* 436:430–442

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.