Neural Computing and Applications (2023) 35:19581–19626
https://doi.org/10.1007/s00521-023-08759-8

ORIGINAL ARTICLE

Multi-objective QoS-aware optimization for deployment of IoT applications on cloud and fog computing infrastructure

Mirsaeid Hosseini Shirvani[1] · Yaser Ramzanpoor[2]

Received: 29 December 2021 / Accepted: 12 June 2023 / Published online: 30 June 2023

Abstract
Internet of Things (IoT) technology serves many industries to improve their performance. As such, utilizing far distant cloud datacenters to run time-sensitive IoT applications has become a great challenge for the sake of real-time interaction and accurate service delivery time requests. Therefore, the fog computing as a deployment approach of IoT applications has been presented in the edge network. However, inefficient deployment of applications' modules on the fog infrastructure leads to physical resource and bandwidth dissipations, and debilitation of quality of service (QoS), and also increases the power consumption. When all application's modules are highly utilized on a single fog node owing to the reduction in the power consumption, the level of service reliability is decreased. To obviate the problem, this paper takes the concept of fault tolerance threshold into account as a criterion to guarantee applications' running reliability. This paper formulates deployment of IoT applications' modules on fog infrastructure as a multi-objective optimization problem with minimizing both bandwidth wastage and power consumption approach. To solve this combinatorial problem, a multi-objective optimization genetic algorithm (MOGA) is proposed which considers physical resource utilization and bandwidth wastage rate in their objective functions along with reliability and application's QoS in their constraints. To validate the proposed method, extensive scenarios have been conducted. The result of simulations proves that the proposed MOGA model has 18, 38, 9, and 43 percent of improvement against MODCS, MOGWO-I, MOGWO-II, and MOPSO in terms of total power consumption (*TPC*) and it has 6.4, 15.99, 28.15, and 15.43 dominance percent against them in term of link wastage rate (*LWR*), respectively.

Keyword Industrial Internet of Things (IIoT) · Fog computing · Application module deployment · Reliable deployment · Traffic-aware deployment

1 Introduction

The modern computing and networking approaches are rapidly extending IoT applications in miscellaneous domains [1]. Real-time interaction and accurate service delivery time request associated with IoT applications make hesitation to adopt far distant cloud datacenters for these application deployment. Many researches in recent efforts are focusing on how to exploit edge network capabilities for supporting IoT applications and their requirements efficiently [1–4]. Computing nodes at the proximity of edge network act in both filtering the gathered data owing to sending low and condensed amount of data volume to the cloud datacenters and for analysis and processing of data in the vicinity where data have been perceived. Therefore, the fog computing has been developed cloud-based applications on the edge networks for executing IoT applications. Fog computing is as a complementary facility of IoT + Cloud scenarios comprising new layer of collaborative devices which are capable of delivering the services and doing business missions completely and independently. In the fog, network components like smart gateways, micro-datacenters, routers, and switches are considered as fog nodes utilized for processing

✉ Mirsaeid Hosseini Shirvani
mirsaeid_hosseini@iausari.ac.ir;
mirsaeid_hosseini@yahoo.com

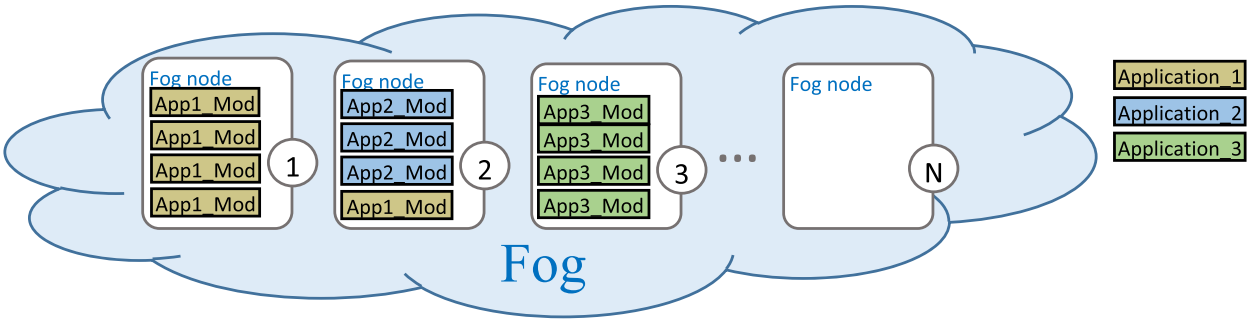
[1] Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

[2] Department of Computer Engineering, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran

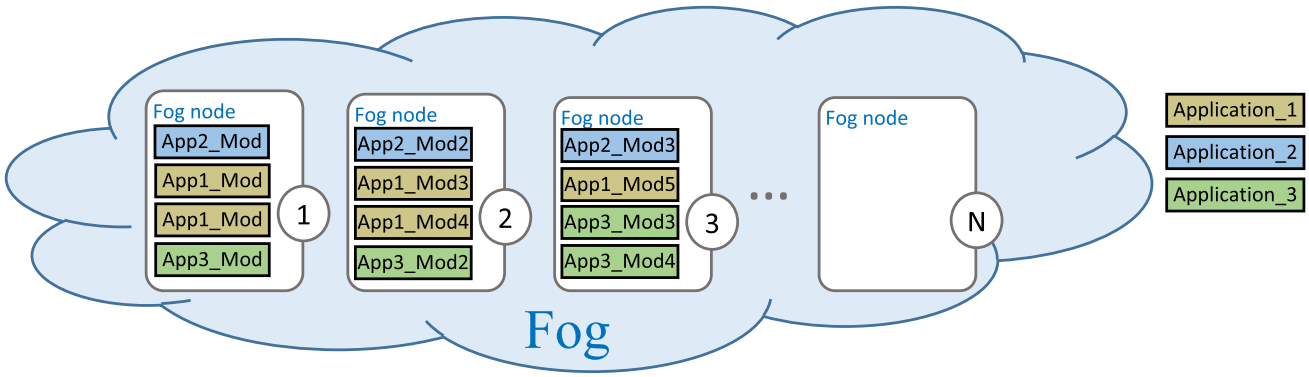
objectives [2]. Inasmuch as a fog server can autonomously process the data gathered from IoT devices without relying on cloud datacenters, it effectively saves network bandwidth usage, cloud storage space, and resource storage for data and critical processes [3, 4]. In addition, fog computing can support usage of unified cloud and edge resources. So, it facilitates IoT application deployment near to data sources. Therefore, it reduces network traffic load and guarantees on-time service delivery. Although the fog computing is the extension and development of cloud computing, the deployment, management, and updating of IoT applications in such layered environment make new challenges. Firstly, fog computing works in large-scale environment comprising a lot of number of nodes with heterogeneous and separate processing, memory, and storage capabilities. Secondly, the workload on each processing node is dynamic and variable. Finally, each IoT application has its own requirements and limitations such as the degree of sensitivity on delay, computing requirement, and preserving privacy constraints. Thus, the deployment of application modules must be properly performed in the fog infrastructure; at the same time, the application requirements along with the features of software, hardware, bandwidth, and delay between nodes in deployment of application modules on fog infrastructure must be taken into consideration [5]. The smart and intelligent deployment of a user's application modules on fog infrastructure can lead to gain the maximum amount in power consumption reduction and efficient usage of physical resources and bandwidth. For illustration, Fig. 1a demonstrates when a fog node which hosts all modules of associated an application breaks down, this user's IoT application does not work in which it has drastic effect on QoS of delivery services. For this reason, the applicable policy must be taken in an appropriate deployment of application modules for reliable service delivery. For instance as Fig. 1b depicts, the customer determines one point of fault tolerance for a requested application; then, regarding the determined amount of fault tolerance, the minimum number of fog nodes along with meeting the QoS for deployment of application modules must be taken into consideration till the effect of failure of some fog nodes be in acceptable level.

In distribution of application modules, there are several possible mappings in which one suitable and optimal mapping must be selected in regard to objective functions. According to Fig. 1, in a problem with little number of modules, there are many possible states to place modules on available fog nodes; so, with the increase of both application modules and the number of fog nodes and also with regard to heterogeneous nature of fog environment, finding an optimal solution for application module deployment on fog infrastructure is a combinatorial problem. This deployment issue is an NP-hard problem in which there is not any exact solution in polynomial time [6, 7]. Recently, several researches have been done to solve distribution of application modules on cloud and fog infrastructure [8–13]. A distribution algorithm of IoT application modules on fog nodes has been done with regard to delay sensitivity and efficient network resource utilization [8]. Moreover, Venticinque and Amato [9] have presented a methodology for solving fog service placement problem which contains an optimal mappings between IoT applications and computing resources for meeting the QoS requirements, but the sensor requirements were neglected in this article. An integrated fog computing platform has been suggested in the literature to figure out dynamic deployment of modules on fog devices [10]. In [10], to avoid from one point of failure the modules are distributed on more than one fog node, but the distribution mechanism has not been elaborated. A decentralized collaborative scheme was proposed for forwarding and placement of modules so that it controls on centralize surveillance limitations such as application management overhead, one point of failure, residual communications, and delay in decision-making process [14]. A general and developable model has been published in the literature for description of QoS-aware deployment of IoT applications over fog computing infrastructure [15]. Although many researches have been disseminated to solve service and module distribution, and deployment in the literature, there is a clear lack of multi-objective QoS-aware and reliable deployment model in the literature. To this end, this paper presents a multi-objective QoS-aware and traffic-aware algorithm for reliable module deployment on fog infrastructure. To harness this huge search space of fog colonies, the proposed algorithm exploits graph theory; then, it models search space as a connected graph. It works in two phases; at first in the preprocessing phase, it finds all full connected graphs; it abstracts them to well-known clique problem owing to have subgraphs with one-hop connected nodes provided satisfying determined QoS and constraints. Afterward, the second phase finds optimal distributed deployment inside one of the extracted cliques in regard to objective functions. Therefore, the main contributions of the current paper are as follows:

- This paper defines the new parameter of "fault tolerance against failure" to guarantee the customer desired execution reliability.
- This paper defines the "link resource wastage" concept to improve the efficiency of utilizing fog node's bandwidth.
- This paper formulates deployment of IoT applications' modules on fog infrastructure into a multi-objective optimization problem with minimizing both bandwidth



(a). Customer's Application Modules Deployed on a Single Fog Node



(b). Customer's Application Modules Deployed on Distributed Fog Nodes

Fig. 1 Deployment of customer's IoT application modules

wastage and power consumption approach which is an NP-hard problem.

- This paper presents an advanced multi-objective genetic-based optimization algorithm (MOGA) to solve the aforementioned NP-hard combinatorial problem with regard to minimization of both total power consumption and bandwidth wastage perspectives along with guarantee of user application reliability.

The proposed method has been validated in different extensive scenarios. The gained results from simulation show the significant dominance of suggested method in terms of stated objectives against other state-of-the-art approaches. The rest of the paper is organized as follows. Section 2 is dedicated to related works. The proposed system models and framework are placed in Sect. 3. Problem statement is expressed in Sect. 4. The proposed MOGA model which solves the problem of IoT application deployment on fog platforms is presented in Sect. 5. The proposed work is simulated and evaluated in Sect. 6. A brief discussion is placed in Sect. 7. Section 8 concludes this paper and indicates future direction.

2 Related works

In this section, published studies associated with service deployment or module distribution of IoT applications in fog computing environment are investigated regarding different objectives and perspectives. This approach reveals the shortcomings for paving the way for further improvement. A mapping algorithm for module deployment of distribution applications in fog environment has been proposed [8]. It was a network-aware approach in which it sorts fog nodes and application modules based on their capacity and current requests; then, the modules are distributed providing the constraints are preserved. In other words, proposed algorithm assigns preference on modules based on their waiting on resources. This policy leads a way to maximizing resource utilization for distributed IoT applications. Furthermore, this algorithm proves that interaction between cloud and fog yields better performance in terms of end-to-end delay in comparison with only cloud-based approaches. A platform as a service (PaaS) architecture was presented for IoT application management based on requirements during development process so that the deployment of applications on mixing cloud2fog scenario is facilitated [16]. A PaaS architecture

that they proposed is as centralized coordinator in which it can develop applications according to their objective domains; it can also discover, initiate, configure, and scale the resources exploitation and execution of application modules, management of execution streaming between modules, monitoring on service-level agreement (SLA), module migration, and presenting interfaces for resource management and components for evaluation. Development and deployment of IoT applications were inspired from DevOps concept propounded in [17]. The cornerstone of this approach includes remote resource management of IoT applications and an integrated tool for development, deployment, and exploitation. The concentration of the proposed approach was to guarantee the accuracy of applications functionality once the old version is substituted by the new one. To this end, the blue–green development patterns is used for a device and the Canary development method is used for reliable substitution on set of devices. With the increase in the (machine-to-machine) M2M communication traffic, bandwidth limitation of edge network, congestion prevention, and service delay became a critical issue in M2M platforms. Since IoT applications are made based on M2M platform, it is an approach presented to decrease the traffic from network to cloud and deployment of IoT application modules on M2M platform so that data are preprocessed before sending on network; therefore, sent traffic is reduced. As existing M2M platforms do not support automatic and dynamic module deployment, a dynamic deployment framework has been suggested in [18]. The main concentration was on automatic and dynamic management and optimal deployment of modules according to the user service requirements. A methodology was presented to support developers in solution of service placement problem in fog environment [9]. The proposed methodology contains finding optimal mapping between IoT applications and computing resources in regard to QoS meeting of application. In addition, an optimal deployment configuration is presented for smart energy ambit and response to application's QoS, deadline times, and throughput which must be preserved. Hong et al. [10] have proposed an integrated fog computing platform for dynamic module deployment on fog devices. To solve the module deployment problem, a heuristic approach has been suggested. In their work, users' requests are directed to a server and then are registered. Each request may be split to several modules each of which can be encapsulated by a docker or a container. After gathering a set of requests, the server runs the algorithm of distribution modules for generating a deployment plan of modules associated with one request. In the following, distribution plan sends modules of each request for deployment on to the fog platform. The main objective was to increase the number of satisfied requests in regard to their requirements. In addition, for running away from one point of failure, the modules may be distributed over more than one computing node. An automated cloud-based IoT application deployment was presented in [19]. This paper exploits Topology and Orchestration Specification for Cloud Applications (TOSCA) standards, a standard for cloud service management, for determining of systematic components and configuration of IoT applications. The automatic deployment process is done in heterogeneous IoT environment by utilizing TOSCA standards. TOSCA is a new standard of Organization for the Advancement of Structured Information Standards (OASIS) organization so that it improves cloud application portability in confronting with heterogeneous cloud environment. This standard is a model for service structure specification and IT service management. In addition, the conceptual of application modules' internal topology and IoT application deployment process are descripted by exploiting TOSCA. Regarding aforementioned issues, the main objective of proposed model was a clear description and interpretation of application modules and their adaptation executing on fog nodes. Saurez et al. published a distributed infrastructure programming interface, so-called foglets, for computing chain of fog and cloud nodes which are geographically distributed [20]. The proposed method provides application programming interface (APIs) for abstraction of data dependent on time and place for storing and retrieval of data produced by applications in local nodes and initiation of communication between resources and computing chain. Foglets manage the application components over fog nodes. This method provides different algorithms for execution of application components and management of component migrations between fog nodes based on sensors mobility patterns and applications' dynamic computing requirements. Four features are supported in Foglets. In the first stage, the automatic fog computing resources are discovered in different levels of networks hierarchy and application components are deployed over fog computing nodes commensurate with tolerable delay for each component. At the second stage, it supports co-hosted multi-programming policy on each fog node. In the third stage, it provides communication APIs for application components which are placed on different physical layers of network hierarchy so that they can negotiate and communicate based on their situation. Finally, it supports adoption of delay-sensitive resources of workload and migration dependent on time and situation for getting over in dynamic situation awareness. Moreover, Foglets supports QoS-aware migration in which quality parameter is relevant to the delay between a component and its parents in an application. To make the flexibility of applications which their deployment topology is completed during the elapsed time, the separation between application components and their executions are necessary. Topology

changes in application deployment are done not only in the time which applications are created, but also it is done in user request pattern changes, physical infrastructure changes, and in edge network changes such as drop/add sensors and gateways. It can also be done for the sake of evolutionary changes of application business during its life cycle. Therefore, Vogler et al. proposed a framework entitled DIANE for generating optimal dynamic deployment topology for IoT applications commensurate with existing physical infrastructure [21]. In this process parameters such as the time needed for deployment, edge device exploitation, bandwidth and execution time needed for running the IoT applications are investigated. Mahmud et al. propounded an approach for management of delay-aware application modules over fog computing environment [14]. In this proposed work, different facets of delay in distributed applications such as delay for service availability, service delivery time, and internal delay communications were considered. The aim of this policy was to manage the time-sensitive and delay-tolerant IoT applications via different approaches in which deadline-based QoS for all type of applications are met besides optimal resource utilization in fog environment. For the sake of optimization in utilizing the number of active fog nodes, the forward and re-allocation application module strategies are used. Furthermore, decentralized organizing is suggested for placement and forwarding the application modules so that it gets over on limitation of centralized surveillance such as application management overhead, one point of failure, residual communications, and delay in decision-making process. Availability of suitable infrastructure and fog application models are very important for success of automated QoS-aware deployment in chaining of clouds to things. Unfortunately, the most advanced tools for automated deployment of distributed software do not deal with non-functional attributes for gaining favorite deployment. Therefore, a simple and general model has been propounded by Brogi et al. for supporting of QoS-aware IoT multi-component applications on fog computing infrastructures [15]. This model descripts the quality of operational systems associated with existing infrastructures in terms of delay and bandwidth; interaction between software components and things; and the business policies. In addition, a number of algorithms have been proposed for favorite component deployment on fog infrastructure. A multi-objective fault-tolerant optimization algorithm based on multi-objective cuckoo search algorithm was extended to solve resource allocation for IoT applications deployment on fog platforms [11]. The resource allocation is done in such a way that to decrease additional power consumption and to minimize the overall latency of all applications [11]. To meet the reliability and to avoid one point of failure, some conditions were added in the problem's constraints. A multi-objective algorithm based on multi-objective optimization PSO algorithm was developed to solve micro-service QoS-aware placement of IoT applications on fog computing infrastructure [12]. The objective functions were *makespan*, budget satisfaction, and network resource utilization that the proposed algorithm generated solutions which make a compromise between conflicting objectives. A reinforcement learning heuristic algorithm was presented to solve micro-service deployment of IoT applications on edge–cloud hybrid environment [13]. The proposed algorithm is aware of challenges such as heterogeneity of underlying resources, dynamic geographical information of IoT devices to decrease the total average of waiting time for all devices in this hybrid complex environment.

Table 1 reviews the literature of researches and achievement along with existing limitations in deployment of IoT application modules on fog and cloud infrastructures by a contrast viewpoint.

The review study reveals that there is a clear lack in studies for considering the degree of fault tolerance in module distribution and also for taking the coefficients of user requested QoS during distribution and deployment of application modules into account. Note that taking aforementioned parameters into consideration has drastic impact on system's overall performance. Therefore, this paper presents solutions to make a trade-off between user's requested requirements and system's utilization objectives to meet both objectives of two prominent stakeholders in the system which are users and providers.

3 System models

This section presents system models and proposed framework for the suggested IoT application deployment scheme. For simplicity and to ease following the models, Table 2 is used for introduction of nomenclature, notation, and utilized parameters and symbols in this paper.

3.1 System framework

The proposed system framework is depicted in Fig. 2. Regarding Fig. 2, a fog orchestrator component is placed on the top of fog layer. One of the most important responsibilities of a fog orchestrator is to select appropriate fog nodes and deployment the IoT application modules. An orchestrator makes decision for module deployment on cloud or fog platforms according to application module attributes. This orchestrator is logically centralized, but it can be implemented with distributed fashion to prevent single point of failure phenomenon. In the proposed framework, the priority is to apply fog computing nodes for

Table 1 Summary of the literature review

Author(s)/ Ref	Distributed application	User-oriented		System-oriented			Merit	Limitations
		QoS Base Deployment	Reliable deployment	Optimize resource	Traffic-aware	Energy efficient		
Taneja et al. [8]	✔	✔	✗	✔	✗	✔	It supports different network topologies	It suffers from: – One point of failure – How to distribute dependent modules over fog nodes
Yangui et al. [16]	✔	✔	✗	✗	✗	✗	It provides a PaaS for automated IoT application module distribution	It suffers from: – Not supporting optimal module deployment – One point of failure
Chen et al. [18]	✔	✔	✗	✗	✗	✗	It takes benefit of distribution with the minimum delay. To this end, it considers a user's distance to the place of deployed module	With the increase in the number of modules, the QoS delivery is declined for the sake of proposed method limitation in scalability. In addition, there is no detailed description between module communications. Also, it suffers from one point of failure
Venticinque et al. [9]	✔	✔	✗	✔	✗	✗	It investigates different situations for module distribution on cloud, fog, and hybrid infrastructure models	It only considers general module communications, but it does not investigate interdependency challenges between modules during distribution
Hong et al. [10]	✔	✔	✗	✔	✗	✔	It distributes user requested modules on minimum number of fog nodes on network	To avoid one point of failure, it suggests to distribute application modules on more than one computing node, but it does not clarify how to do so
Saurez et al. [20]	✔	✔	✗	✗	✗	✗	Presenting a programming infrastructure for IoT application's development and deployment. It also supports module migration	Generally, it only considers module communications, but it does not investigate interdependency challenges between modules during distribution
Vögler et al. [21]	✔	✔	✗	✔	✗	✔	Presenting a framework for producing optimal deployment topology and supporting different IoT application's topologies	Although it pays attention to module interdependencies, it does not state how to prevent delay of dependent modules' deployment

Table 1 (continued)

Author(s)/ Ref	Distributed application	User-oriented		System-oriented			Merit	Limitations
		QoS Base Deployment	Reliable deployment	Optimize resource	Traffic-aware	Energy efficient		
Mahmud et al. [14]	✔	✔	✗	✔	✗	✔	It prioritizes the time-sensitive applications for assigning on the proximity resources	It does not investigate chain of module interdependencies
Ramzanpoor et al. [11]	✔	✔	✔	✗	✔	✔	It considers the minimum number of processing nodes to meet reliable execution	It does not consider bandwidth wastage rate which indirectly impacts on the overall delivered QoS
Pallewatta et al. [12]	✔	✔	✗	✗	✔	✗	It models satisfaction budget, total execution time, and network resource utilization in the fitness function which may lead to sustainable decision	The fitness model ignores power consumption optimization which has drastic impact on provider's cash flow
Chen et al. [13]	✔	✔	✗	✗	✗	✗	It considers resource heterogeneity model which can have efficient resource allocation tailored with diverse resource requests of IoT applications	Although the proposed model is aware of underlying heterogeneity, it does not consider all QoS relevant to all system's stakeholders
Proposed paper	✔	✔	✔	✔	✔	✔	– Utilizing the degree of QoS function for decision making in distribution strategy – Distribution of application modules regarding its degree of tolerance against faults – Attention to communication link wastage rate in module distribution process	Although it is not a drawback for proposed approach, the deployment approach is fully dependent on full-mesh subnetworks derived from whole fog infrastructure network

distribution of application modules, but the cloud data-centers are utilized only for deployment of modules which are not time-sensitive and are engaged for periodically information processing. To deploy application modules, the fog nodes are defined as fog mesh network for the sake of communication between fog nodes. The fog mesh can be defined as a computing pattern which differs from traditional mesh networks, but it uses mesh networks of fog nodes such as switches and fog servers for distribution process inside the network. Figure 2 also illustrates a high perspective of fog mesh computing pattern. The architecture of fog mesh pattern is similar to wireless mesh network (WMN) proposed in [22], but it has its own characteristics.

To manage the appropriate deployment of application modules on fog nodes, a module management framework is placed in the orchestrator component regarding system's performance. As Fig. 3 demonstrates, this framework has different components. One of them is the planner component which includes application module manager and other components. Besides the planner, there are other components to store and retrieve of network information and other fog resources. The gathered information is used for application module management and presenting a deployment

Table 2 Introduction of nomenclature and used parameters

Notation	Description	Notation	Description
N	Number of fog nodes	fn_i	ith fog node where $i = 1,\ldots,N$
Id	Fog node identifier	H	Hardware specification of fog node
S	Software specification of fog node	fn_{type}	Fog node type, where type = (high, medium, low)
R_{fn}	Resource capacity in terms of processing, memory, and storage associated with fog node fn	$sensorlist$	Sensors of fog node fn
B	Bandwidth capacity of link	L	Delay of link
B_{ij}	Bandwidth between nodes i and j	L_{ij}	Delay communication link between nodes i and j
d_{ij}	distance between nodes i and j	$Cost_{ij}$	Communication cost between nodes i and j
FP_F	Failure probability of fog node F	n_A	Fog nodes serving modules of application A
$UApp$	User application	$UApp_i$	User's ith application
M_i	Number of modules in ith application	FTT_i	Failure tolerance of ith application
$Appmodlist_i$	Module list of ith application	$Appmod_{k,i}$	kth module of ith application
h_k	Requested hardware for kth module	s_k	Requested software for kth module
$m_{i,L}$	Modules of ith application deployed on fog node L	b_{ij}	The favorite bandwidth between modules i and j
l_{ij}	The favorite delay between modules i and j	$thr_{QoSscore}$	The threshold of requested QoS for a module
r_{Appmod}	Resource capacity for requested module in terms of processor, memory, and storage	t_{mn}	traffic between modules m and n
$Score_{QoS}$	QoS ranking calculated for a module	q	qth QoS parameter
α_q	The weight of qth QoS parameter	$Score^q_{QoS}$	Ranking value for qth QoS parameter in [0..1]
r_q	The qth (in QoS) value for requested module	o_q	The qth (in QoS) value delivered for module
$diff(r_q, o_q)$	The difference of qth (in QoS) value between requested and delivered for a module	β	The level of flexibility of a module in regard to qth (in QoS) which is in [0..1] interval
$x_{\text{Appmod},fn}$	A decision binary variable which shows whether module $Appmod$ is deployed on fog node fn or not	y_{fn}	A decision binary variable which indicates whether fog node fn is active or not
$r^{CPU}_{Appmod_j}$	CPU requirement for jth module associated with an application App	$r^{RAM}_{Appmod_j}$	RAM requirement for jth module associated with an application App
$R^{CPU}_{fn_i}$	CPU capacity of fog node fn_i	$R^{RAM}_{fn_i}$	RAM capacity of fog node fn_i

scheme by planner component. In the following, the functionality of components are elaborated.

The components are enlisted and clarified as below:

Application module manager The main component which exploits other existing components of the proposed framework in decision-making process determining how to deploy application modules on cloud or fog nodes. In multi-module applications, the decision of module deployment strongly depends on several determinants such as accessible to the resources, network structure, QoS requested for the application, and load balancing. The deployment process can be done with regard to the reduction in the power consumption and network traffic minimization viewpoints.

Resource management This component decides based on processing and memory capacity requirements and deploys modules according to existing resources and objective functions. One of the main objectives is to balance loads over host nodes. Recall that load balancing is not a goal, but is a technique for improving other QoS objectives such as response time, *makespan*, and system throughput.

Communication management Communication significantly incorporates in usage of fog resources that are consuming for IoT applications. Management of IoT application modules includes optimization in utilization of computing resources, memory, and communications simultaneously. Therefore, communication management component plays vital role in this ambit. This component determines which of the fog nodes can communicate with each other and what is the optimal way for sharing data between different nodes.

Resource discovery Resource discovery means to find fog nodes which can provide information and services requested for each module. Trustable resource discovery in determined time frame is a challenge because the network is large scale and susceptible to topology changes. The gained information via this component is used for

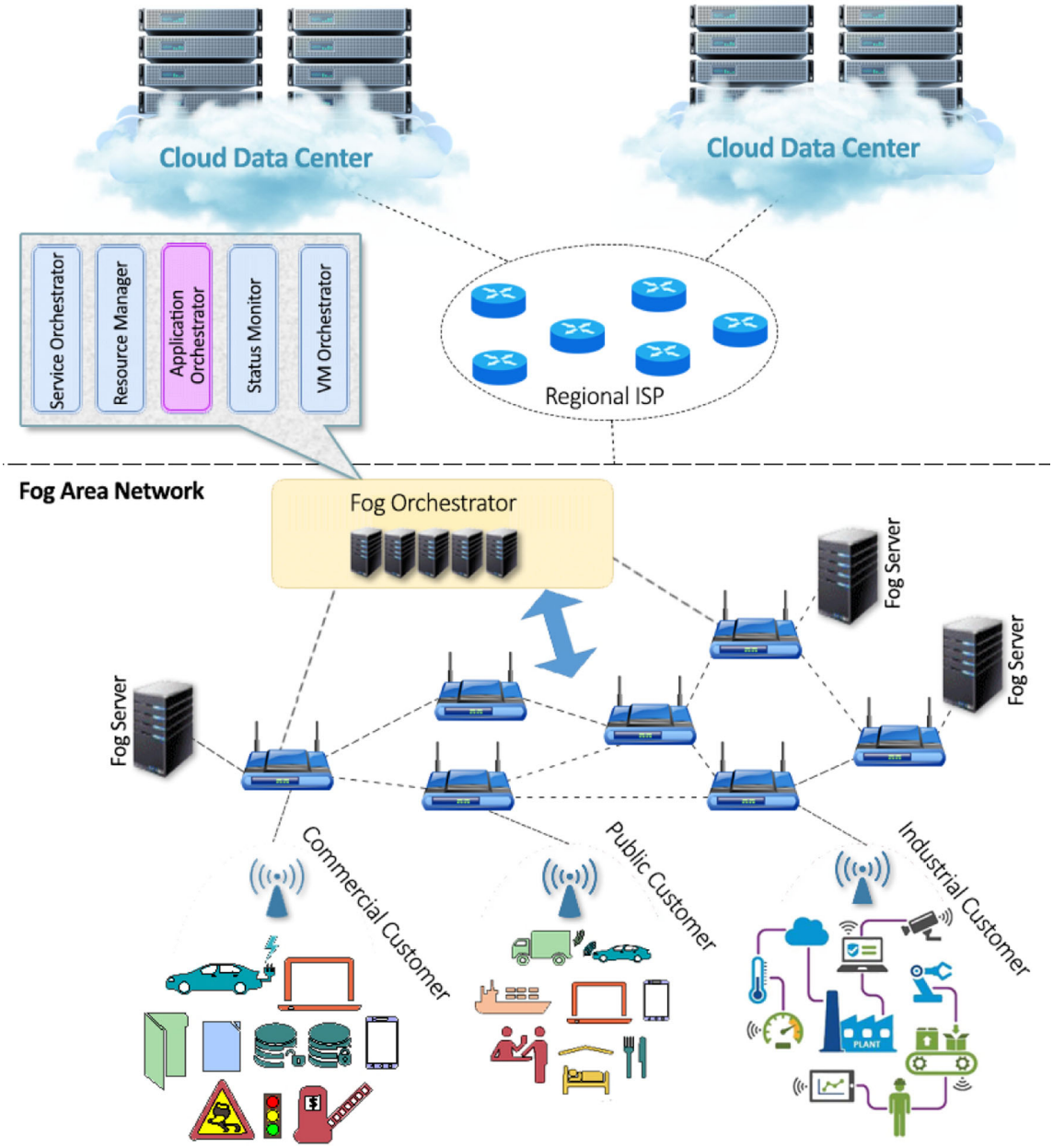


Fig. 2 System framework based on fog mesh network

determining which nodes for deployment of modules to be interacted. The decision, which planner component makes, depends on discovered information by resource discovery component.

FN Information This component saves all information about a fog node such as available sensor types, information about modules that are running on the fog node, using of existing resources like energy, memory, and CPU. The deployment of modules on each fog node depends on information presented by FN information component. This information is applied in planner component by resource management component.

Network information This component registers all information about the routers, neighbor fog nodes, connected end devices, different kinds of sensors in the network, etc. The information obtained via resource discovery component is delivered to network information component and is stored in its memory. The decision for module deployment which communication management component makes in the planner is based on the information saved in the network information component.

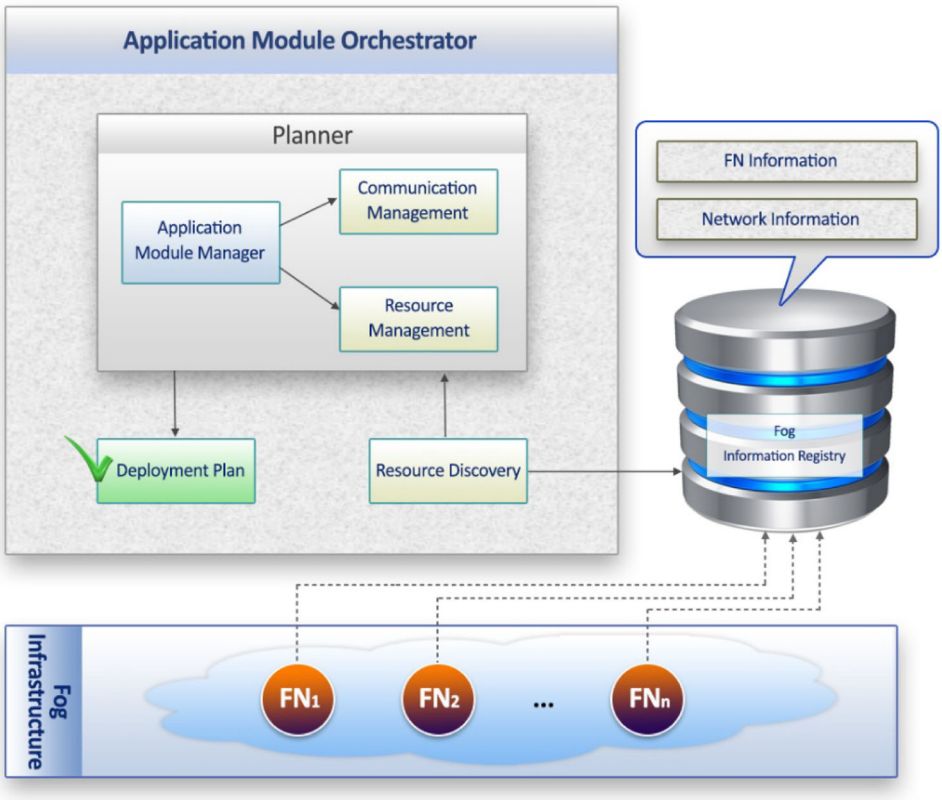


Fig. 3 Application module management framework

3.2 Fog and communication network models

Fog Model: In this paper, it is assumed that there are N fog nodes heterogeneous in terms of processing and energy capacity in the network which are capable of storing and executions of application modules. Figure 4 depicts a sample fog model specification. Each fog node accesses to different kinds of sensor nodes directly or indirectly via wired or wireless communications. Each fog node $fn \in F$ is presented with a tuple $\{\text{id}, H, S, \text{fn}_{\text{type}}, sensorlist\}$ where parameters id is fog node identifier, H is the hardware, S is the software, fn_{type} is the fog node type in terms of performance, and $sensorlist$ is the list of sensors available for the fog node. Since the fog node's energy consumption

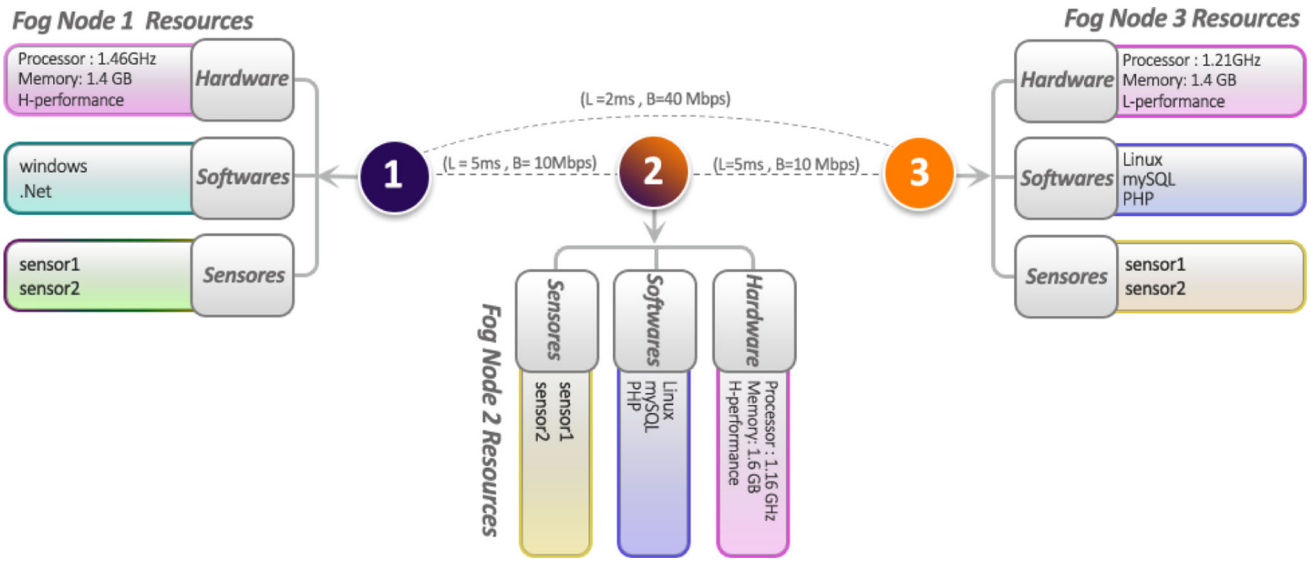


Fig. 4 Example of fog model specification

greatly depends on CPU power usage, the concentration on improvement of CPU usage associated with fog nodes can enhance both resource utilization and power consumption [23]. Therefore, the resource utilization and performance of fog nodes are taken into account during the process of application module deployment. In this regard, high-performance fog nodes are preferred in comparison with low-performance nodes because each high-performance fog node consumes relatively lower energy in comparison with the low-performance ones although they host more workload to process. Note that the communication links between two different nodes are modeled by vector (L,B) where parameters L and B are the delay of communication link and communication bandwidth, respectively.

Communication network model In this paper, communication network model comprises a mesh network with connected nodes. The communication network is modeled by a graph $G = (FN, D)$. In this graph, $\mathrm{FN} = \{fn_1, fn_2, \ldots, fn_N\}$ is a set of fog nodes and $D = \{d_{ij} =$ distance between two fog nodes fn_i and $fn_\mathrm{j}\}$ is a set of edges. In this heterogeneous model, each fog node fn_i is shown by a vector $fn_\mathrm{i}=(\mathrm{id}_\mathrm{i}, \mathrm{h}_i, s_i, sensolrlist_i)$ where its elements are identifier, hardware, software, and equipped and available sensors for fog node fn_i, respectively. In this regard, Fig. 5 illustrates communication network graph and matrix in Eq. (1) is dedicated for distance between each pair of fog nodes.

$$\begin{array}{c} \\ fn_1 \\ fn_2 \\ fn_3 \\ \vdots \\ fn_n \end{array} \begin{array}{c} \begin{array}{ccccc} fn_1 & fn_2 & fn_3 & \ldots & fn_n \end{array} \\ \begin{bmatrix} d_{11} & d_{12} & d_{13} & \ldots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \ldots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \ldots & d_{3n} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \ldots & d_{nn} \end{bmatrix} \end{array} \tag{1}$$

Owing to the reduction in the search space and for the sake of dependent module deployment, the full-mesh network is initially extracted from the existing primary fog network. In other words, all full connected subgraphs which are abstracted to a clique problem are extracted because the one-hop full connected networks can shorten delay for time-sensitive applications. Then, the distance matrix value is set via Eq. (2).

$$D_{ij} = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{else } i \neq j \end{cases} \tag{2}$$

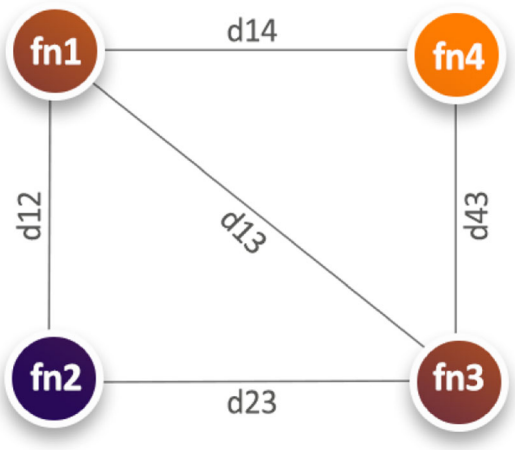


Fig. 5 Communication network graph

3.3 Application model

The current applications that process big data explosion specifically in the large scale are no longer monolithic, but they have multi-module structure [24]. Therefore, the applications which run on fog computing infrastructure are a set of dependent modules incorporating to meet customer requests. For instance, take a simple IoT application associated with a theft alarm system for smart home which are presented by a safety company to its customers. This application has three different modules. As Fig. 6 depicts, the modules are M1 (threat manager) that monitors the environment for inception of control and intrusion alarm once it detects; M2 (control center) for gathered data interpretation and manual system control; and M3 (machine learning) for storing of data history and updating the intrusion detection model to be deployed on strong fog nodes or cloud ones.

In this regard, Fig. 6 illustrates list of hardware resources and software capabilities requested for each module. The relationship between modules are depicted by links which must meet QoS constraints associated with delay and link's bandwidth. In addition, for on-time management of urgent threat circumstances, a module M1 must access to necessary sensors (acoustic, motion, virtual sensor, etc.) and an actuator which triggers safety mechanism. Note that this process must be done in 10 ms from where the M1 module is deployed to installed sensors and actuator. Furthermore, it is expected fog and cloud nodes can remotely access to things existing in their neighbor nodes in the network via API proposed by fog middleware layer [2]. The problem that must be solved for modules is how to implement three modules so that all determined non-functional constraints on software, hardware resources, software interactions, and remote access to IoT are to be met; even for a simple example where the number of application modules are 3 and the number of nodes in a full connection graph is 5 (3 fog nodes and 2 cloud nodes), there are up to 50 possible deployment options to find appropriate mapping of software modules on fog or cloud nodes. This is because that more than one module can be deployed on the nodes based on existing

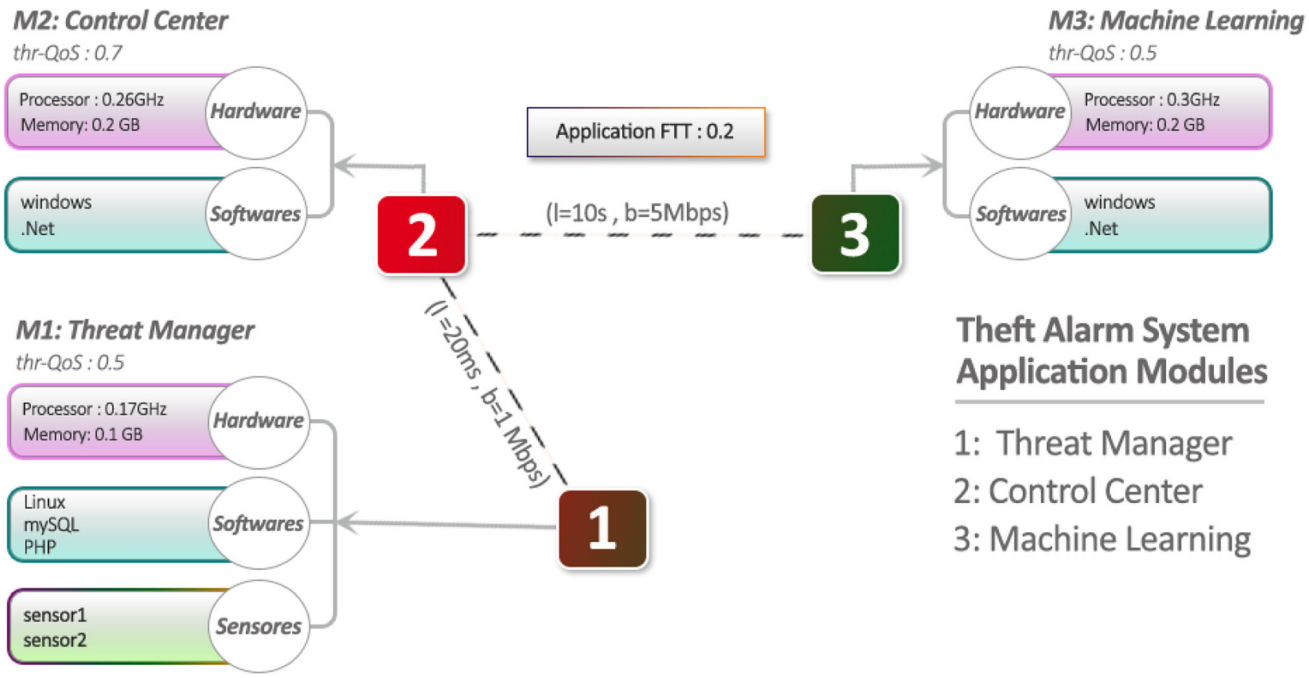


Fig. 6 Application module specifications

resources. This is impossible for a human user to determine favorite deployment once the infrastructure and the number of software modules grow significantly and the search space grows exponentially accordingly. To get these modules work properly and also the application as a consequence, it is necessary to meet resource requirement and requested QoS accurately. In this paper, it is assumed that we are given a set of R IoT applications, each of which $r \in R$ is shown by a vector $r = (FTT, M, \text{Appmodlist})$ where FTT is fault tolerance threshold parameter (c.f. in Sect. 3.4). Each application has M different modules where are enlisted in $Appmodlist$ variable. Accordingly, each module is elaborated in a vector $Appmod_i=(k, \text{h}, s, sensorlist, thr_{QoSscore})$. Note that parameter $thr_{QoSscore}$ is used to indicate the requested QoS which must be met by fog node hosting the considered module. User application is modeled in the form of graph $G = (Appmodlist, T)$ where $Appmodlist = (Appmod_1, Appmod_2, \ldots, Appmod_m)$ is module list and $T = \{t_{ij} | \text{is the amount of traffic between } Appmod_i \text{ and } Appmod_j\}$ is set of traffic pairs. Figure 7 demonstrates communication graph and matrix in Eq. (3) is dedicated for traffic between each pair of application modules.

$$
\begin{array}{c}
\\
Appmod_1 \\
Appmod_2 \\
Appmod_3 \\
\vdots \\
Appmod_m
\end{array}
\begin{array}{c}
\begin{array}{ccccc} Appmod_1 & Appmod_2 & Appmod_3 & \ldots & Appmod_m \end{array} \\
\begin{bmatrix}
t_{11} & t_{12} & t_{13} & \ldots & t_{1m} \\
t_{21} & t_{22} & t_{23} & \ldots & t_{2m} \\
t_{31} & t_{32} & t_{33} & \ldots & t_{3m} \\
\vdots & \vdots & \vdots & \ldots & \vdots \\
t_{m1} & t_{m2} & t_{m3} & \ldots & t_{mm}
\end{bmatrix}
\end{array}
\tag{3}
$$

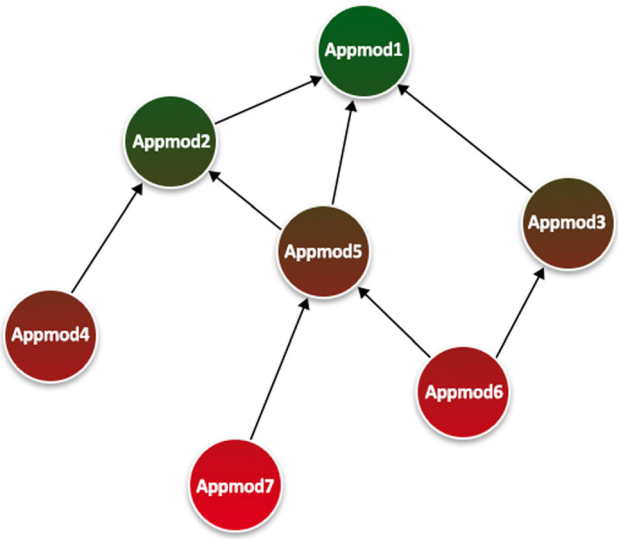


Fig. 7 Traffic matrix and communication network graph of application modules

3.4 Reliability model

Assume that the fog orchestrator component receives R execution requests from fog layer for requested applications concern to different customers. The users requested applications are modeled by Eq. (4)

$$UApp = \{UApp_i | 0 < i \leq R\} \tag{4}$$

In this model, each user application $UApp_i$ is determined by a specification vector which Eq. (5) shows.

$$UApp_i = (FTT_i, M_i, Appmodlist_i) \tag{5}$$

where the parameters M_i and FTT_i are used for the number of modules and fault tolerance threshold determined for an application $UApp_i$. Fault tolerance threshold parameter is the number which the customer submits to the system along with other application's information. In addition, as Eq. (6) calculates, $Appmodlist_i$ specifies list of modules concerning to ith requested application, namely $Application_i$.

$$Appmodlist_i = \{Appmod_k | 0 < k \leq M_i\} \tag{6}$$

where parameter $Appmod_k$ indicates kth module of an application. This parameter is elaborated in Eq. (7).

$$\begin{aligned} Appmod_k = \{&Appmod_k \in Appmodlist_i | Appmod_k \\ &= k, H, S, sensorlist, thr_{QoSscore}\} \end{aligned} \tag{7}$$

The aforementioned parameters requested by a customer are registered in fog orchestrator component. Then, the orchestrator applies this information for decision making of module deployment process. In this paper, we assume the probability of simultaneously crashing more than one fog nodes is very low. In addition, we consider crashing probability is the same for all fog nodes. To investigate the reliability execution of requested application on specific fog node, it is compared with user's submitted FTT parameter. To do so, the average effect of fog nodes' crash on customer's application is compared with the FTT parameter submitted by a customer. To calculate the effect amount of fog nodes' crash on customer's application, Eq. (8) is dedicated [25].

$$FTT_i = \sum_{L}^{n_L} FP_L \times \frac{m_{i,L}}{M_i} \tag{8}$$

The parameter FTT_i is dedicated for the average amount of fault tolerance relevant to customer's application running on fog nodes. In this regard, FP_L is the fault probabilty of each fog node L, and $m_{i,L}$ is the number of modules associated with an application $UApp_i$ deployed on fog node L. In addition, M_i is the number of whole modules in an application $UApp_i$ and $\frac{m_{i,L}}{M_i}$ is the the effect amount which an application $UApp_i$ is affected once the fog node L is crashed. Moreover, the parameter n_L is used for the number of fog nodes in the network incorporating in running an application $UApp_i$. If we consider the same probability of failure for each fog node, namely $FP_L = \frac{1}{n_L}$, consequently, Eq. (6) is simplified in Eq. (9) [25].

$$FTT_i = \frac{1}{n_L} \times \sum_{l}^{n_L} \frac{m_{i,l}}{M_i} = \frac{1}{n_L} \tag{9}$$

As explained earlier, there is a meaningful relationship between the fault-tolerant threshold of an application running on fog nodes and the number of underlying utilized fog nodes. In other words, the minimum number of fog nodes n_L is needed to meet fault-tolerant threshold FTT_i of an application $UApp_i$. Therefore, Eq. (10) indicates the necessary number of fog nodes to meet the fault-tolerant objective for running an application $UApp_i$.

$$n_L > \frac{1}{FTT_i} \tag{10}$$

3.5 Deployment model

The problem of module deployment can be solved in both static and dynamic fashions. As the number of fog nodes is considered fix in a time frame, the decision for deployment of modules is made in predetermined time window. Therefore, the optimization deployment scheme can be periodically re-executed to be commensurate with the dynamic nature of IoT and fog environment. There are two stakeholders in the system under study, namely customer and provider, each of which has its own objectives to be met. The objectives may have conflict, so the proposed scheme must compromise between them. In the proposed paper to meet reliability, which is associated with user perspective for requested application, the minimum number of fog nodes is calculated for module deployment. On the other hand, to minimize the total power consumption, which is associated with provider perspective, the modules must be distributed on calculated fog nodes, but it is not always possible because of variable requirement of modules, heterogeneous fog node specification, and different constraints. So, it may utilize more fog nodes for deployment process which not only increases power consumption, but also increases bandwidth wastage. Therefore, a trade-off is necessary between stakeholders' objectives. To figure out module deployment problem, all full-mesh networks are extracted from an initial fog network graph in which fog nodes in extracted full mesh can meet all requirement of requested applications in terms of delay, bandwidth, and sensors. It is also assumed that if a sensor is not supported for a fog node, other fog nodes can satisfy the limitation via resources existing in extracted full-mesh

subnetwork by a single-hop connection. In this line, decision variable d_{ij} is used to indicate whether the module $Appmod_i$ is deployed on a fog node fn_j or not. To decrease network traffic load, the distance matrix between fog nodes in network graph and traffic matrix between each pair of modules must be calculated. Since there is a limitation in delay and bandwidth between each communication link, the traffic rate between modules of applications are bounded based on communication links capacity. Therefore, Eq. (11) indicates the communication constraint in this domain.

$$\sum_{Appmod_i \in fn_m} \sum_{Appmod_j \in fn_n} b_{ij} \times l_{ij} < B_{mn} \times L_{mn} \tag{11}$$

Among QoS parameters, the latency and bandwidth are taken into consideration. QoS parameters are evaluated in regard to the kind of modules in an application and the distance between requested QoS by customer and presented QoS by provider [26]. To this end, aforementioned attributes of QoS are measured in Eq. (12) which finally gains a score in interval [0..1].

$$Score_{QoS} = \sum_{q=1}^{d} \alpha_q \times Score_{QoS}^{q} \tag{12}$$

In Eq. (12), parameters d and α_q are the number of attributes in QoS and the weight of qth quality attribute in whole QoS. In addition, the score of qth quality, $Score_{QoS}^{q}$, is determined in [0..1] which is calculated via Eq. (13) [27].

$$Score_{QoS}^{q} = \begin{cases} 0 & \text{if } diff\left(r_q, o_q\right) = \infty \\ 1 & \text{if } diff\left(r_q,\ o_q\right) = 0 \\ f\left(diff\left(r_q, o_q\right)\right) & \text{else} \end{cases} \tag{13}$$

where function $diff\left(r_q, o_q\right)$ returns the difference between requested amount of qth quality for a module (r_q) and presented amount of qth quality presented by provider (o_q). It shows that if the difference is zero, the score is one; in case the difference is ultimate, the score is zero. For other cases, the score is gained via the degree of satisfaction fucntion which Eq. (14) calculates [27].

$$f\left(diff\left(r_q, o_q\right)\right) = e^{-diff\left(r_q, o_q\right)} \tag{14}$$

The function $diff(.,.)$ is calculated by Eq. (15).

$$diff\left(r_q, o_q\right) = \begin{cases} \infty & \text{if}\left(\beta = 0 \text{ and } \left(r_q - o_q\right) \times type_q > 0\right) \\ 0 & \text{if}\left(\beta \geq 0 \text{ and } \left(r_q - o_q\right) \times type_q \leq 0\right) \\ \left(r_q - o_q\right) \times type_q & \text{if}\left(\beta > 0 \text{ and } \left(r_q - o_q\right) \times type_q > 0\right) \end{cases} \tag{15}$$

where parameter $\beta \in [0..1]$ is used for level of flexibility associated with a module in regard to attribute qth quality. For instance, $\beta = 0$ means that the special service must be completely met with high degree of obligation. In addition, $type_q \in \{-1, 1\}$ which takes 1 or -1 according to type of qth quality. In other words, it takes 1 for capacity-based quality parameters and takes -1 for time-based quality parameters. For instance, in regard to latency parameter, if 4 ms in terms of latency is necessary for a module and the presented latency by fog is 2 ms, then $\left(r_q - o_q\right) \times type_q = (4 - 2) \times -1 < 0$; it means $diff\left(r_q, o_q\right) = 0$ based on Eq. (15).

In deployment process, the score gained via Eq. (12) which is the capability of fog for execution of application modules must be greater than user determined QoS threshold. This inequality constraint is presented in Eq. (16) [27].

$$Score_{QoS} \geq thr_{QoSscore} \tag{16}$$

In addition, there are some other constraints that will be explained in problem statement section.

4 Problem statement

As explained earlier, the static module deployment issue is formulated to a multi-objective optimization problem with simultaneous minimization of both link wastage rate (LWR) and total power consumption (*TPC*) perspectives. To this end, we present two objective models mathematically.

4.1 Link wastage model

To model link wastage rate (*LWR*), the cost model associated with communication of each pair of fog nodes is calculated by Eq. (17).

$$Cost_{ij} = \sum_{Appmod_m \in fn_i} \sum_{\substack{Appmod_n \in fn_j \\ fn_i \neq fn_j}} d_{ij} \times t_{mn} \tag{17}$$

Therefore, total link wastage rate is equal to sum of all mutual traffics traded between each pair of modules distributed over fog nodes. So, the total link wastage rate is modeled by Eq. (18).

$$\text{LinkWastageRate}(UApp_{Cost}) = \sum_{fn_i \in \text{F}} \sum_{\substack{fn_j \in F \\ i \neq j}} Cost_{ij} \quad (18)$$

4.2 Power consumption model

Generally, several factors effect on total power consumption of a system under study such as computation workloads, communication technology, amount of traded traffic, and distance between each pair of fog nodes. To calculate power consumption of each fog node, power consumption relevant to running of modules on the fog nodes and the power consumed as for information communication between each pair fog nodes are considered. The power consumption of each fog node directly depends on its resource utilization [28]. Therefore, average normalized resource utilization of a fog node fn_i is calculated by Eq. (19).

$$U_{fn_i}^{res} = \frac{W_1 . \sum_j^{fn_i} \frac{r_{Appmod_j}^{CPU}}{R_{fn_i}^{CPU}} + W_2 . \sum_j^{fn_i} \frac{r_{Appmod_j}^{RAM}}{R_{fn_i}^{RAM}}}{2} \quad (19)$$

Note that two real-valued coefficients W_1 and W_2, where $0 \leq W_1 \leq 1, 0 \leq W_2 \leq 1$, and $W_1 + W_2 = 1$, are applied to indicate the importance of resources incorporating in total power consumption. As the big portion of power consumption is relevant to processing units instead of main memory, in this paper the CPU utilization is considered for power consumption, namely $W_1 = 0.9$ and $W_2 = 0.1$ [6]. So, Eq. (20) calculates the power consumption ($P_{fn_i}^{res}$) owing to used resources associated with each node (fn_i) which hosts different modules [6, 28].

$$P_{fn_i}^{res} = ((P_{max} - P_{min}) \times U_{fn_i}^{res} + P_{min}) \quad (20)$$

where parameters P_{min} and P_{max} are used to indicate the minimum and maximum power consumption of each processing node in the lowest and uppest utilization circumstances, respectively. In addition, a decision binary variable y_{fn_i} is applied to imply whether the processing node fn_i is active or not. It is multiplied with parameter $P_{fn_i}^{res}$ in Eq. (23). In addition, the power consumption owing to data transfer via communication links are calculated by Eq. (21).

$$P_{fn_i}^{tr} = \sum_{fn_i \neq fn_j} t_{Appmod_m, Appmod_n} \times P_{tr} \quad (21)$$

The parameter P_{tr} is used for prower consumption unit of traffic exchanges. Note that this power is considered provided the modules are deployed on different computing nodes. Cosequently, the total power consumption is calculated via Eq. (22). The first part is for resource utilization, whereas the second part is for traffic transfer power cost.

$$P_{fn_i} = P_{fn_i}^{res} + P_{fn_i}^{tr} \quad (22)$$

4.3 Multi-objective QoS-aware optimization deployment model

After two objective models has been defined, we formulate module deployment problem to a multi-objective QoS-aware optimization model. In this line, Eqs. (23–24) are dedicated to objective functions, whereas Eqs. (25–33) are presented to indicate problem constraints.

$$\boldsymbol{min\ TPC} = Min \sum_{fn_i \in F} P_{fn_i} \times y_{fn_i} \quad (23)$$

$$\boldsymbol{min\ LWR} = Min \sum_{fn_i \in \text{F}} \sum_{\substack{fn_i \in F \\ i \neq j}} Cost_{ij} \quad (24)$$

Subject to:

$$n_L > \frac{1}{FTT_k}, \quad k = 1, 2, \ldots, R \quad (25)$$

$$\sum_{Appmod_m \in fn_i} \sum_{Appmod_n \in fn_j} b_{mn} \times l_{mn} < B_{ij} \times L_{ij} \quad (26)$$

$$Score_{QoS} \geq \text{thr}_{QoSscore}, \quad \forall \text{Appmod} \in UApp, fn_i, fn_j \in F \quad (27)$$

$$\sum_{\text{Appmod} \in UApp} x_{Appmod, fn_i} \cdot s_{Appmod} \leq S_{fn_i}, \quad \forall fn_i \in F \quad (28)$$

$$x_{Appmod, fn_i} \leq y_{fn_i}, \quad \forall \text{Appmod} \in UApp, fn_i \in F \quad (29)$$

$$\sum_{\text{Appmod} \in UApp} x_{Appmod, fn_i} \cdot r_{Appmod} \leq R_{fn_i}, \quad \forall fn_i \in F \quad (30)$$

$$\sum_{fn_i \in F} x_{Appmod, fn_i} = 1, \quad \forall Appmod \in UApp \quad (31)$$

$$x_{Appmod,fn_i} \in \{0,1\} \tag{32}$$

$$y_{fn_i} \in \{0,1\} \tag{33}$$

Note that Eq. (28) states that available sensors must be more than the requested number of sensors. Equation (29) indicates that the module can be deployed on a fog node providing it is an active node. In addition, the total resources requested by modules deployed on a fog node cannot exceed from its capacity; so, Eq. (30) is dedicated to this issue. Moreover, each module should be only placed on one fog node; so, the constraint of Eq. (31) is dedicated to this issue. Furthermore, Eqs. (32–33) are used for two binary decision variables indicating whether the module is deployed on fog node and whether a fog node is active or not, respectively.

5 Proposed MOGA for module deployment

As the stated multi-objective optimization problem is computationally NP-hard, so finding an optimal solution of large-scale problem is impossible in practice; therefore, we apply a hybrid and effective approach which has two phases to figure it out. Firstly, a heuristic algorithm is utilized to reduce very large search space into a concise district in short time. Secondly, this balanced search space is given to a meta-heuristic algorithm for finding non-dominated solutions because the problem is a multi-objective optimization and there is not any single optimal solution to meet all objectives. To this end, we utilize dominance concept [29]. To solve aforementioned combinatorial optimization problem, we engage genetic algorithm (GA) although we examined several meta-heuristic optimization algorithm. Then, we trust on genetic-based algorithm in this project because of gained results. Since the search space of stated problem is discrete in nature, the GA is more flexible for customizing in discrete search space. In multi-objective domain, famous GA-based NSGA-II had several achievements in solving combinatorial studies [29]. Therefore, we customize multi-objective GA capability in such a way to solve module deployment problem. To do so, we conduct two operators: *crossover* for exploration and *mutation* for exploitation. It this way, it escapes from getting stuck in local optimal. Figure 8 depicts the block diagram of proposed algorithm. This algorithm receives input parameters relevant to deployment modules such as information of requested resources for applications' modules, communication pattern between modules, fog infrastructure configurations, population size, and the maximum number of iterations. Then, it outputs set of non-dominated deployment schemes in regard to objective functions.

In the following, encoding schema, operators, fitness function, non-dominated sorting, and the crowding distance concepts are elaborated and discussed.

5.1 Encoding schema

One of the most important of GA's elements is the genes and chromosomes concepts and how to encode the problem. There may be several encoding schemas for a single problem each of which has its own performance; then, proposing an efficient encoding schema has drastic impact on overall algorithm performance [26, 30, 31]. Each chromosome is an agent of a possible solution. In this line, a chromosome contains $|M|$ genes where M is the number of modules, each of which is used for a module to be deployed on the set of N fog nodes. This is the reason each gene gets an integer value in $[1..|N|]$ interval such as in [6, 26, 32]. Figure 9 demonstrates a feasible solution scheme.

5.2 Preprocessing

Since there exist dependencies between application's modules, it is necessary to select fog nodes for deployment so that they have communication with each other and can meet QoS requirement in network domain. Full-mesh extraction from initial network graph, which is abstracted to famous Clique problem, has multiple advantages; firstly, it shortens search space for extraction of optimal deployment scheme. Secondly, if a sensor is not supported for application's module by a fog node, the required sensors can be available via other fog nodes in full-mesh subnetwork with one-hop connection. To extract full-mesh subnetworks that can support all modules' requirements, Algorithm 1 is designed.

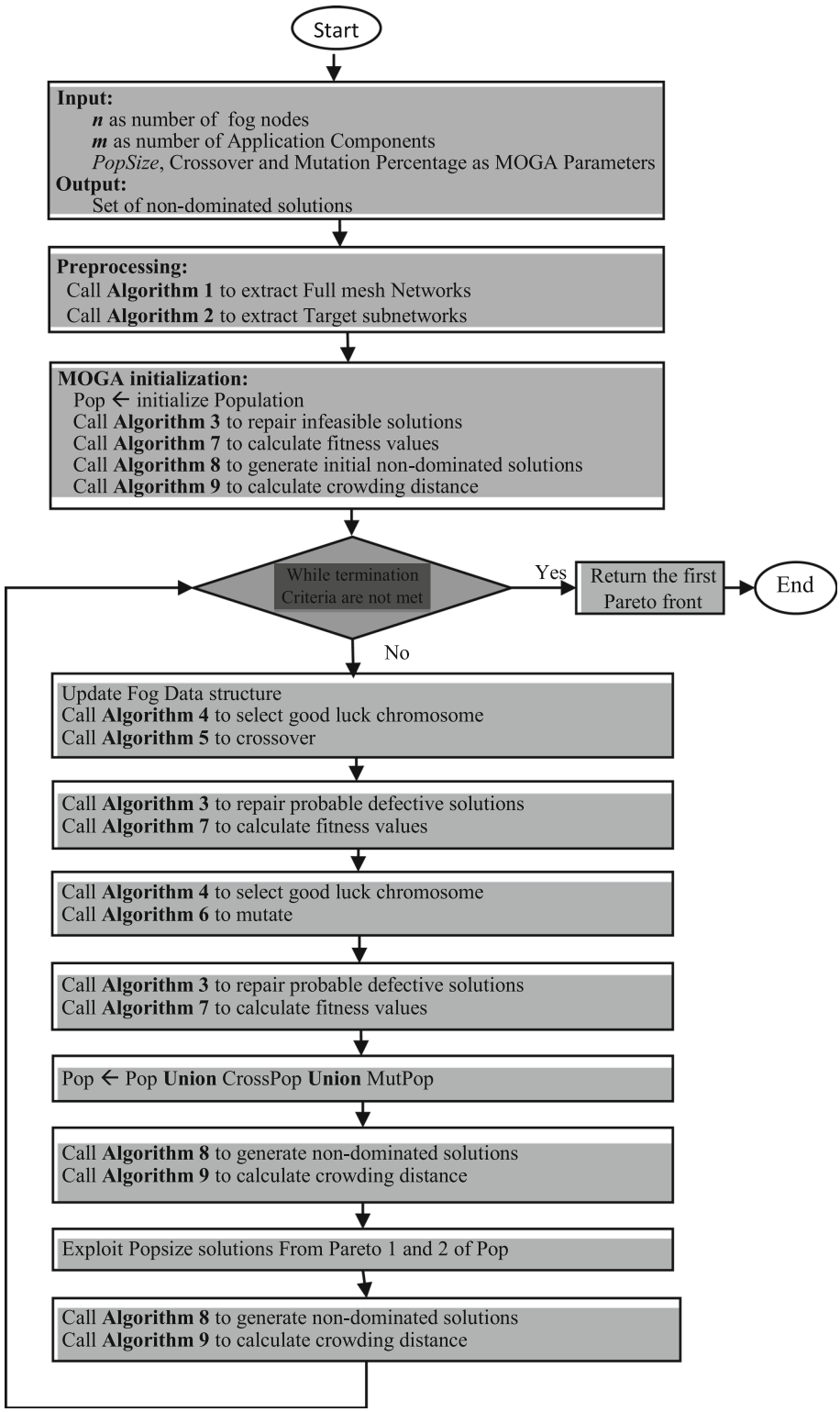


Fig. 8 Block diagram of proposed algorithm

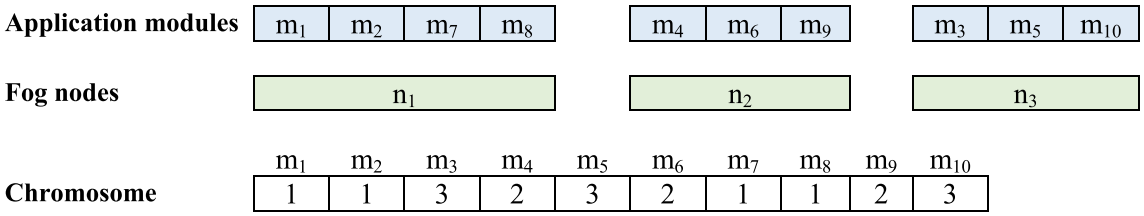


Fig. 9 Example for encoded of feasible solution to a chromosome

Algorithm 1. Fullmesh Subnetworks Exploitation

Input Parameters:
N : number of fog nodes
K : desired clique size;
FN : Set of fog nodes.
FN_CMAT : two dimensional array [1..N, 1..N] of binary;
(* it is used for indicating nodes dependencies *)

Output:
FullMesh _FN_SubNetworks: vertical array [1..N] of LinkedList;
(* Each LinkList[i] can add pair of nodes which is an edge in the graph *)

Initialization:
Define FullMesh_FN_SubNetworks [1..N].LinkedList ← NIL
Define FN_CMAT : an upper triangular matrix

```
1:  Begin
2:    (* Initialize data structure *)
3:    For each fog node F_i ∈ FN (i=1,…,N) Do
4:      For each fog node F_j= ∈ FN (j=i+1,…,N) Do
5:        IF FN_CMAT[F_i][F_j] is connected THEN
6:          FullMesh_FN_SubNetworks [F_i].LinkedList ← FullMesh_FN_SubNetworks [F_i].LinkedList +{(F_i, F_j)}
7:        End-IF
8:      End-For
9:    End-For
10:   (* end of initialization *)
11:   While termination criteria are not met DO
12:     For each fog node F_i ∈ FN (i=1,…,N) Do
13:       For each fog node F_j ∈ FN (j=1,…,N, j≠i) Do
14:         Current_Row= FullMesh_FN_SubNetworks [F_j].LinkedList
15:         IF Fog node F_i is not in Current_Row, but it is connected by all elements in Current_Row according to
      FN_CMAT Then
16:           FullMesh_FN_SubNetworks [F_j].LinkedList ← FullMesh_FN_SubNetworks [F_j].LinkedList +{(F_j, F_i)}
              End-IF
17:       End-For
          Remove duplicate element of FullMesh_FN_SubNetworks [F_j].LinkedList
18:     End-For
19:   End-While
20:   Return FullMesh _FN_SubNetworks
21: End
```

Algorithm 1 receives fog set specifications and returns all full-mesh networks. It applies data structure *FullMesh_FN_SubNetworks* as a vertical array in which its every row has a linked list of elements. In the inception, all pair of fog nodes which directly communicate with each other are added to this data structure. In other words, at the outset for each fog node F_i, the *FullMesh_FN_SubNetworks* [F_i]. LinkedList is filled by direct fog nodes which F_i connects with. Afterward, the algorithm plummets into while-loop. It is iterated until the specified criteria are met. The termination criterion is the number of desired clique size (K). In other words, the main loop is iterated K times. At the next rounds, for each node each row of data structure is investigated. If the node is not in the list, but is connected to all of nodes in the list; then, this node is added to the linked list. After each round, duplicated record is discarded. Finally, all of full-mesh subnetworks are returned. In this way, the search space is declined and

condensed to concise district. Since the effective statements of Algorithm 1 are in the while-loop, its time complexity is O (K.N^2). Up now, we have several full-mesh subnetworks, but a question arises: Which of them is well suited for requested IoT applications' modules? To meet the problem constrains which have been formulated in Eqs. (25–32), after extraction of full-mesh subnetworks, Algorithm 2 selects number of nodes to construct target subnetwork nodes for hosting requested modules provided the constraints are satisfied.

variable. In line 6, different full-mesh permutations with estimated number of fog nodes are extracted from *FullMesh_FN_subNetworks* via *nchoosek(.)* function. To meet constraint of Eq. (26), in line 11, function *Check_Delay_DW(.)* compares minimum bandwidth and maximum delay of subnetwork in Row_i with maximum bandwidth and minimum delay requested for modules. If it is the case, it returns true. Lines 12–15, try to meet other constraint of stated problem, i.e., Eq. (27). If each of *Delay_BW_Status* or *Check_QoS_Score* returns true, the

Algorithm 2. Find Target Fullmesh Subnetwork Nodes

Input Parameters:

define FN={F_i | set of fog nodes,i=1,..,N};
define Appmod={$Appmod_i$ | set of application modules ,i=1,..,M};
define FN_BW: array[1..N,1..N] of bandwidth between fog nodes;
define FN_Delay: array[1..N,1..N] of delay between fog nodes;
define Appmod_BW: array[1.. M,1.. M] of bandwidth between modules;
define Appmod_Delay= array[1.. M,1.. M] of delay between modules;
define App_FTT: Application fault tolerance threshold;
define AppmodDataset: Application modules resource requirements;
define FNDataset: Fog nodes resources;
define FullMesh_FN_SubNetworks : vertical array[1..Rows] of FullMesh subnetworks node;
define Candidate_subNetworks=array() : array of candidate fullmesh subnetworks;

Output:

Candidate_subNetworks

```
1:  Number_of_Fog_Nodes = calculate ceil(1/App_FTT) based on InEq. (25);
2:  Min_Required_Fog_Nodes =
        Calculate_Required_Fog_Nodes(AppmodDataset,FNDataset);
3:  if Number_of_Fog_Nodes less than Min_Required_Fog_Nodes Then
4:      Number_of_Fog_Nodes = Min_Required_Fog_Nodes;
5:  End if
6:  sub_Fullmesh_Networks =
        nchoosek(FullMesh_FN_SubNetworks,Number_of_Fog_Nodes);
7:  //returns a matrix containing all possible combinations of the fog nodes
    of vector FullMesh_FN_SubNetworks taken Number_of_Fog_Nodes at a time;
8:  Appmods_Delay_BW=calculate_Delay_BW(Appmod, Appmod_BW, Appmod_Delay);
9:  Candidate_subNetworks ← NIL
10: For each sub_Fullmesh_Networks as Row_i Do
11:   Delay_BW_status =
        Check_Delay_BW(Appmods_Delay_BW, Row_i,FNDataset,FN_BW,FN_Delay);
        //constrain of Ineq. (26)
12:   Qos_Score_Status=
        Check_Qos_Score(AppmodDataset, Appmod_BW, Appmod_Delay ,row_i
            ,FNDataset,FN_BW,FN_Delay);
        //constrain of Ineq. (27)
13:   If Delay_BW_status or Qos_Score_Status are true Then
14:     Candidate_subNetworks ← Candidate_subNetworks +[Row_i];
15:   End if
16: End For
17: Return Candidate_subNetworks;
```

In the first line of Algorithm 2, the minimum number of requested fog nodes is determined to fulfill condition of Eq. (25). In the second line, the minimum number of fog nodes regarding their existing capacity and requested resources of modules is estimated. If estimated number of fog nodes are more than existing fog nodes, all available fog nodes are considered for *Number_of_Fog_Nodes* full-mesh subnetwork is registered in *Candidate_SunNetworks* and is returned as a candidate full-mesh subnetwork. All calculation and checker functions of Algorithm 2 are simple and belong to O (1) except for function *nchoosek*(.) in line 6 and main loop in line 10 which belong to O (N) and O (M), respectively. Therefore, Algorithm 2's time complexity is O ($N + M$).

5.3 Initialization step

Similar to other meta-heuristic algorithms, our proposed algorithm starts with initial population after preprocessing phase which meets Eqs. (25–27). To this end, initial population is randomly produced regarding satisfying constraints stated in Eqs. (29–31). Note that, during the course of algorithm running for producing possible new solutions, some solutions violate constraints generating infeasible solutions; this is because Algorithm 3 is proposed to repair them for exploiting all population possibilities and capabilities toward final solutions. It finds a fog node which is overloaded; then, it selects an *Appmod* with minimum dependency to other modules; then, it migrates preferably on an active fog node which has sufficient resources to adopt it. Afterward, this solution and related data structure are updated.

Time complexity of Algorithm 3 is O (N.PopSize) because two nested for-loop are the most effective statements.

5.4 Selection strategy

There are several strategies for selection of solutions such as roulette wheel, rank-biased, and tournament. All of them are conducted in such a way that the fittest solution has the high probability to be selected. Algorithm 4 is dedicated for this issue. It firstly selects two group of solutions; namely, one of them is from the first Pareto front and the other is from the second Pareto front of population. Then, one individual is randomly selected from each group. Afterward, the one has upper crowding distance is returned because the crowding distance indicate high potential of searching divergence [29]. It is clear-cut that its time complexity is O (1).

Algorithm 3. Repair Solutions

Input Parameters:
define Fog_Data_Structure : Fog nodes status in each solution;
define Fog_Node_List : node list taken from Algorithm 2;
define Appmod=$\{Appmod_i \mid$ set of application modules ,i=1,..,M$\}$;
define AppmodDataset : Application modules Specification;
define Solutions is all solutions in a population;
define PopSize : Population Size

Output:
Feasible Solutions and Fog_Data_Structure

```
1:  For Each solution in Solutions Do
2:    For Each Fog node in solution as sourceFN Do
3:      If sourceFN Resource overload is true Then
4:        violation = true;
5:        While violation is true Do
6:          find Appmod with minimum dependency to other modules on
            sourceFN and migrate it to other fog node in
            Fog_Node_List with enough resource. Candidate Appmod has
            maximum dependency to modules on destination Fog node;
7:          Update solution;
8:          Update Fog_Data_Structure;
9:          If sourceFN resource overload is false Then
10:           violation = false;
11:         End If
12:       End While
13:     End If
14:   End For Each
15: End For Each
16: Return Solutions and Fog_Data_Structure;
```

Algorithm 4. selection strategy

Input Parameters:
define Population;
Output:
Winner solution in tournament

```
Tournament_Pop = all Pareto front1 and Pareto front2 solution in population;
Solution1 = exploit a solution at random from Tournament_Pop;
Solution2 = exploit a solution at random from Tournament_Pop;
if Solution1.Crowding_Distance >= Solution2.Crowding_Distance Then
    Winner solution = Solution1;
Else
    Winner solution = Solution2;
End if
Return Winner solution;
```

5.5 Crossover Operator

To explore search space, crossover operator is applied to produce new generations. Then, the new generation is added to current population with predetermined probability. In this line, the single-point procedure is done for crossover. Figure 10 exemplifies a single-point crossover application.

Algorithm 5 is presented for crossover (exploration).

It is clear-cut that its time complexity of Algorithm 5 belongs θ (nCrossover) which is *O* (PopSize).

5.6 Mutation Operator

Mutation is applied in GA to avoid pre-saturation and early convergence in course of execution. Despite crossover behavior, mutation only changes one gene randomly. In each generation, some new solutions are added to population which was predetermined by *mutation percentage* parameter. Figure 11 depicts how the mutation operator works. In the main proposed algorithm, one chromosome is selected as a parent by *tournament* method; then, two random genes are selected for exchange in the opted parent. Newborn individual is added to the population. Algorithm 6 is dedicated to mutation operator.

Algorithm 5. crossover operator

Input Parameters:
define Population;
define nCrossover : Number of Parents(Offsprings);
define CPoint : Crossover Point(a random point in solution);
define CrossPop=array() is empty array for insert crossover solutions;
Output:
Crossover Population

```
For Each nCrossover Do
    Parent_ Solution1 = select a solution from Population using Algorithm 4;
    Parent_ Solution2 = select a solution from Population using Algorithm 4;
    Offspring_Solution1 =
        [Parent_ Solution1(1 to CPoint-1), Parent_Solution2(CPoint to end)];
    Offspring_ Solution2 =
        [Parent_ Solution2(1 to CPoint-1), Parent_Solution1(CPoint to end)];
    Insert Offspring_Solution1 and Offspring_Solution2 in CrossPop
End For Each
Return CrossPop;
```

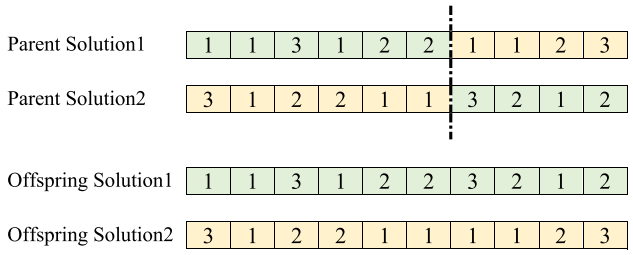


Fig. 10 Application of crossover for exploration

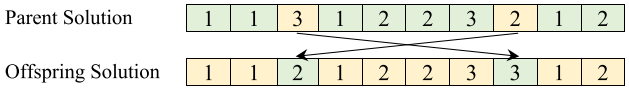


Fig. 11 Application of mutation for exploitation

Algorithm 6. mutation operator

Input Parameters:
define Population;
define nMutation : Number of Parents(Offsprings);
define MutationPop=array() is empty array for insert mutation solutions;
Output:
Mutated Population

```
For Each nMutation Do
    Parent_ Solution = select a solution from Population using Algorithm 4;
    gene1 = random gene in Parent_ Solution;
    gene2 = random gene in Parent_ Solution;
    Offspring_Solution = swap gene1 and gene2 in Parent_ Solution;
    Insert Offspring_Solution in MutationPop
End For Each
Return MutatedPop;
```

It is clear-cut that its time complexity of Algorithm 6 belongs θ (nMutation) which is O (PopSize).

5.7 Fitness function

Generally, one of the most important issues in evolutionary computation is to evaluate competency of each candidate solution. Since each solution is equivalent to one deployment scheme, the objective functions or fitness values of Eqs. (23–24), namely total power consumption and total link wastage rate, are calculated regarding information derived from candidate solution. Algorithm 7 receives a candidate solution; then, it returns two objective function values as a result.

Algorithm 7. Fitness function

Input Parameters:
define Population;
Output:
Update population with solutions' cost

```
For Each solution in Population as solution[i] Do
    Energy = calculate computing and network resource energy consumption
        of all fog node in solution [i] based on Eq. (23);
    LWR = calculate Link Wastage Rate of all fog node in solution [i] based on Eq. (24);
    The fitness cost of solution [i] in Population = [Energy , LWR];
End For Each
Return population(Energy & LWR);
```

It is clear-cut that its time complexity of Algorithm 7 is O (PopSize).

5.8 Non-dominated sorting

In multi-objective meta-heuristic optimization algorithm, population evolution is gained by special strategy with the aim of omitting inferior solutions and preserving superior solutions. In other words, low-level solutions are gradually discarded. In this line, dominance concept is applied and non-dominated solutions are remained in the final result set called Pareto set [29]. Similar to NSGA-II, our algorithm exploits non-dominated sorting algorithm to classify individuals based on their competencies in regard to dominance concept [29]. Note that all individuals in the same class, hereafter called Frontier, do not dominate each other, but they dominate individuals in the upper levels which are inferior. So, the topmost Frontier belongs to the first Frontier. Algorithm 8 calculates all Frontiers of a given population.

Algorithm 8. none dominated sorting strategy

```
Input Parameters:
        define Population: array of Solutions;
        PopSize : Size of Population;
        (* Each Solution [i] has three fields N, S, and Rank *)
        (* numeric field N: is the number of solutions dominate Solution [i] *)
        (* set S: is collection of solutions which Solution[i] dominates them *)
        (* numeric field Rank: is the rank of frontier which Solution [i] belongs to *)
        define F{K}=array() is array of solutions in each subgroup K;
        (* F{K} indicates all solutions in frontier K-th *)
Output:
    Updated Population with subgrouping solutions;
For Each Solution [i] in Population Do
    Solution[i].S= NULL
    Solution[i].N= 0;
    For Each Solution [j] in Population Do
        If Solution [i] dominates Solution [j] Then
            Solution[i].S= Solution[i].S UNION Solution[j]
        Else If Solution [j] dominates Solution [i] Then
            Solution[i].N= Solution[i].N + 1
        End IF
    End For
    If Solution [i].N =0 Then
        Solution [i].Rank=1
        F{1} ← F{1} UNION Solution[i]
    End IF
End For
K=1
While F{K} is not NULL Do
    Temp = NULL
    For each Solution[i] in F{K} Do
        For each Solution[j] in Solution[i].S Do
            Solution[j].N= Solution[j].N +1
            If Solution[j].N=0 Then
                Solution[j].Rank=K+1
                Temp=Temp UNION Solution[j]
            End IF
        End For
    End For
    K=K+1
    F{K} ← Temp
End While
Return F and K
```

Since the effective statements of Algorithm 8 are in nested For-loop, its time complexity is O ($PopSize^2$).

5.9 Crowding distance

In multi-objective domain, another important thing is to how extent the solutions are scattered in search space. The more distribution in search space, the more probability to find better solutions. For this reason, we apply crowding distance strategy which Algorithm 9 draws. This algorithm avoids early convergence of solutions in course of evolutionary process.

Algorithm 9. crowding distance strategy

Input Parameters:
Population : Set of Solutions;
PopSize: Population Size;
Output:
CD[1..PopSize] : Crowding Distance value of Population;
(* CD[i] is crowding distance value of Solution [i] in Population *)

1: **For Each** solution in Population as solution [i] **Do**
2: CD[i] ← 0;
3: **End For**
4: **For Each** objective in stated problem as Obj_j **Do**
5: NewPop ← Sort Population of solutions based upon Obj_j in ascending order;
6: CD[1] = CD[PopSize] = Infinite;
7: **For Each** solution[i] in NewPop from solution [2..PopSize-1] **Do**
8: CD[i] ← CD[i]+ solution[i+1].Obj_j –solution [i-1].Obj_j)/ (max Obj_j of NewPop - min Obj_j of NewPop)
9: **End For**
13: **End For**
14: **Return** Population and CD;

It is clear that the time complexity of Algorithm 9 is O (PopSize. Log (PopSize)) which is mostly relevant to sorting procedure.

5.10 Description of proposed MOGA

The problem of IoT application modules' deployment is solved by calling Algorithm 10. This algorithm receives problem specifications relevant to both application and fog infrastructure; then, it returns non-dominated solutions regarding two defined objective functions. Firstly, it calls Algorithm 1 to extract full-mesh subnetworks. Then, Algorithm 2 is called to refine subnetworks in such a way that to reach the target full-mesh subnetworks which cover all applications requirements. In line 5 of Algorithm 10, the random population is generated. To repair the probable infeasible solutions, Algorithm 3 is called in line 8. To calculate the fitness values of each individual Algorithm 7 is called in line 9. Afterward, Algorithms 8 and 9 are called to find initial non-dominated solutions and balance them in crowding distance. The main loop of Algorithm 10 is started from line 13 and ends in line 33. It is iterated until the termination criteria are met. At the outset of the main loop, Algorithm 4 is called to select good individuals for crossover; also, it is called for mutation. After both of them, Algorithm 3 is run to repair probable defective chromosomes. Then, Algorithms 8 and 9 are called to find non-dominated solution and balance them based on crowding distance. Since the size of population may exceed from its initial size because of adding newborn individuals resulting from crossover and mutation operations, selecting the most suitable individuals is done for next round with the same size as initial size. To do so, the selection of individuals to consider for next round is based on solutions' rankings. It is done from the first Pareto ranking to the worse. In case of selection in the same ranking, the ones which have greater crowding distance value are in high priority to be selected. Thus, the new population is ready for the next round. Once the last iteration is done, the final non-dominated solutions associated with the first front are returned as a Pareto front or set of non-dominated solutions.

Algorithm 10. The proposed Multi-Objective Genetic- Algorithm (MOGA)

Input Parameters:
(*Parameters of Applications And Fog network Modules*)
Datasets : Application Dataset;
Fog Network Dataset: Fog Dataset relevant fog nodes and module delay and bandwidth matrix;
(*Parameters of MOGA *)
PopSize: number of solution in Population;
MaxIteration : Maximum iteration;
pCrossover: Crossover Percentage;
nCrossover: number of parent(Offsprings);
pMutation: Mutation Percentage
nMutation: number of Mutants;
Output:
Solutions in First Pareto front

```
(* Pre-processing *)
Fog_DS = Initial Fog Data Structure using Datasets;
FullMesh_FN_SubNetworks = call Algorithm 1 for Exploit Fullmesh Subnetworks;
Target_Sub_Networks = call Algorithm 2 for Find Target Fullmesh Subnetwork Nodes;
Pop = an initial Random Population(PopSize,ModuleDatasets, Target_Sub_Networks);
Update Fog_DS using Pop information;
(* End of Pre-Processing *)
Call Algorithm 3 for Repair infeasible Solutions in Pop;
Call Algorithm 7 for Calculate Solutions cost in Pop;
(* Multi Objective Operations *)
Call Algorithm 8 for None Dominated Sorting Solutions in Pop;
Call Algorithm 9 for calculate Crowding Distance of Solutions in Pop;
While the termination criteria are not met do
   Fog_DS = initial Fog Data Structure using Datasets;
   Call Algorithm 4 to select individuals for crossover based on pCrossover rate;
   CrossPop = call Algorithm 5 for Crossover Solutions in pervious Pop;
   Update Fog_DS using CrossPop information;
   call Algorithm 3 for Repair infeasible Solutions in CrossPop;
   call Algorithm 7 for Calculate Solutions cost in CrossPop;
   Fog_DS = initial Fog Data Structure using Datasets;
   Call Algorithm 4 to select individuals for mutation based on pMutation rate;
   mutPop = call Algorithm 6 for Mutation Solutions in pervious Pop;
   Update Fog_DS using mutPop information;
   Call Algorithm 3 for Repair infeasible Solutions in mutPop;
   Call Algorithm 7 for Calculate Solutions cost in mutPop;
   Pop = merge ( Pop , CrossPop , mutPop );
   //Multi Objective Operations
   Call Algorithm 8 for None Dominated Sorting Solutions in Pop;
   Call Algorithm 9 for calculate Crowding Distance of Solutions in Pop;
   Pop = Exploit first PopSize solutions from Pop;
   Call Algorithm 8 for None Dominated Sorting Solutions in Pop;
   Call Algorithm 9 for calculating Crowding Distance of Solutions in Pop;
End While
Return First Pareto Front in Pop; //a near optimal solution
```

Now that time complexity of all subalgorithms have been determined, the time complexity of Algorithm 10 is easily calculated. Its initialize phase take $O\,(M + K.N^2)$. In addition, the main loop iterates *MaxIteration* times. For the main loop, we have $MaxIteration \times (N.PopSize + PopSize.\ Log(PopSize) + PopSize^2)$. After simplification, its time complexity is $O\,(M + K.N^2 + MaxIteration.\ PopSize^2)$ which is relatively acceptable time complexity.

6 Simulation and evaluation

To evaluate the proposed algorithm for solving IoT application module deployment on fog computing infrastructure, we conduct extensive scenarios along with considering two prominent criteria, i.e., total power consumption and total link wastage rate. In this line, the proposed algorithm is compared with different state of the art such as MODGWO [33], MODCS [34], and MODPSO [35] in different scenarios. Note that all comparative algorithms associated with each scenario have been run in the fair conditions. In this regard, all of them were run 20

Table 3 Determined Scenarios for Simulation

		Scenario 1	Scenario 2	Scenario 3	Scenario 4
Category 1	Fog nodes #	10	15	20	25
	Application modules #:	20	20	20	20
		Scenario 5	Scenario 6	Scenario 7	Scenario 8
Category 2	Fog nodes #	15	15	15	15
	Application modules #	20	25	30	35
		Scenario 9	Scenario 10	Scenario 11	Scenario 12
Category 3	Fog nodes #	10	15	20	25
	Application modules #	25	30	35	40

times independently. Then, the comprehensive simulation results along with descriptive statistics in terms of min, max, average, and standard deviation (STD) values associated with each scenario have been reported. This report can strongly support the current proposal.

6.1 Experimental settings, datasets, and scenarios

In this section, experimental settings, scenarios, datasets associated with both requested applications and fog infrastructure are presented. To this end, 12 different scenarios are conducted in 3 groups. In the first group, the number of modules are fix, whereas the number of fog nodes are gradually increased. In the second group, the number of fog nodes are fix, whereas the number of modules are gradually increased. Finally, in the third group, number of both modules and fog nodes are gradually augmented. Table 3 depicts the details of different scenarios. Note that all simulations have been executed in a computer specified by Dual core Intel Core i3380M with 2.53 GHz clock rate, four logical processors, and 8 GB RAM.

As fog infrastructure is ad hoc and heterogeneous in nature, there is not abundant dataset in the literature. This is the reason we produce our own dataset and we take its heterogeneity into account. Fog nodes' heterogeneity are in terms of processing power, storage capacity, bandwidth, delay, and power consumption as well. For instance, Tables 4, 5, and 6 are dedicated to specification of a sample fog network which has 10 different heterogeneous processing nodes. Note that CPU capacity, memory, bandwidth, power consumption and delay are based on GHz, GB, Mbps, Watt, and ms units, respectively. As Table 4 shows, threshold of CPU and memory usage, minimum and maximum of power consumption, type of supported sensors, and power consumption of data transfer are completely different.

In this regard, Tables 5 and 6 are used to indicate bandwidth and delay between each pair of fog nodes. To make dimensionless, the values in the tables are normalized in [0..1] interval. In Table 5, the zero means two nodes do not have direct connection, but one means source and destination are the same. Also in Table 6, the one means two nodes do not have direct connection, but zero means two nodes are the same.

Similar to Tables 4, 5, 6, 7, 8, and 9 are dedicated to module specifications in terms of requested resources, recommended bandwidth and delay between modules.

Note that the values of Tables 7, 8, and 9 have been received from processing power and storage capacity of fog nodes based on their architecture and existing computing fog nodes such as personal digital assistant (PDA), smart phones, and internal computers.

In addition, parameters of proposed, MODGWO, MODCS, and MODPSO algorithms are set in such a way Table 10 draws.

6.2 Experimental results

In this section, the result of simulations are analyzed in three categories. To compare performance of comparative algorithms, one of the good solutions of them in each scenario was selected to be depicted. The illustration is relevant to Pareto front, and status of objective functions in each iteration along with their convergence in course of running, the values of objective functions, and elapsed time are utilized. Finally, at the end of this subsection, the descriptive statistics in terms of min, max, average, and STD values relevant to all scenarios are tabulated. As forthcoming figures demonstrate, there are two versions for MODGWO, i.e., MODGWO-I and MODGWO-II. In the former, GA's operators are applied for evolutionary process, whereas in the latter canonical MODGWO's operators are applied. Moreover, for geography place map of fog nodes, random values in normal distribution are generated

for fog nodes' (*X, Y*) coordinates value in its network. In addition, fog nodes' distribution map relevant to each scenario is depicted. In this map, the host nodes which deploy application's modules are placed in the map with blue color.

Table 4 Resource specification of fog nodes

Fog nodes	1	2	3	4	5	6	7	8	9	10
CPU	1.02	1.15	1.38	1.46	1.06	1.21	1.42	1.41	1.09	1.45
RAM	1.3	1.6	1.2	1.4	1.3	1.4	1.3	1.4	1.4	1.4
CPU_Thr	0.98	0.93	0.96	0.94	0.92	0.94	0.94	0.90	0.92	0.91
RAM_Thr	1.00	0.99	0.91	0.92	0.98	0.97	0.93	0.93	0.96	0.97
P_min	94	82	99	81	91	86	93	81	93	84
P_max	133	132	133	147	142	146	133	144	145	130
Sensor	1,2	1,2	1,2	2	1,2	1,2	1	2	1	2
P_tr	0.2	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.2	0.2

Table 5 Bandwidth specification between each pair of nodes

Fog nodes	1	2	3	4	5	6	7	8	9	10
1	1	0.98	0.80	0.89	0.97	0.82	0	0	0	0.87
2	0.84	1	0.82	0.94	0.92	1.00	0	0	0	0.99
3	0.93	0.94	1	0.97	0	0.97	0	0	0	0.90
4	0.88	0.92	0.91	1	1.00	0.96	0.84	0.99	0	0.85
5	0.92	0.99	0	0.90	1	0.85	0	0	0	0.82
6	0.93	0.90	0.87	0.98	0.99	1	0.89	0.90	0	0.81
7	0	0	0	0.83	0	0.96	1	0.94	0.86	0.87
8	0	0	0	0.90	0	0.96	0.94	1	0.94	0.94
9	0	0	0	0	0	0	0.84	0.87	1	0
10	0.92	0.98	0.84	0.98	0.99	0.98	0.92	0.88	0	1

Table 6 Delay specification between each pair of nodes

Fog nodes	1	2	3	4	5	6	7	8	9	10
1	0	0.17	0.19	0.10	0.10	0.14	1	1	1	0.12
2	0.12	0	0.10	0.11	0.18	0.19	1	1	1	0.17
3	0.18	0.14	0	0.12	1	0.15	1	1	1	0.13
4	0.16	0.19	0.14	0	0.14	0.19	0.18	0.13	1	0.18
5	0.11	0.14	1	0.16	0	0.11	1	1	1	0.11
6	0.12	0.17	0.11	0.20	0.13	0	0.12	0.15	1	0.18
7	1	1	1	0.15	1	0.13	0	0.12	0.14	0.18
8	1	1	1	0.15	1	0.15	0.10	0	0.10	0.12
9	1	1	1	1	1	1	0.18	0.11	0	1
10	0.12	0.10	0.12	0.18	0.17	0.11	0.18	0.16	1	0

Table 7 Specification of requested resources for applications' modules

Application modules	1	2	3	4	5	6	7	8	9	10
CPU	0.15	0.19	0.24	0.26	0.29	0.25	0.22	0.16	0.26	0.16
RAM	0.2	0.2	0.1	0.2	0.1	0.2	0.2	0.2	0.1	0.2
Sensor	1	1	2	2	2	1.2	1.2	1.2	1	1.2
$Thr_{QosScore}$	0.75	0.8	0.9	1	0.84	0.95	0.77	0	0	0

Table 8 Specification of bandwidth recommended for applications' modules

Application modules	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0.20	0.29	0.20	0.34
2	0	1	0.33	0	0.32	0.35	0.31	0.35	0	0.25
3	0	0	1	0.31	0.20	0.21	0.27	0.23	0	0.31
4	0	0	0	1	0.39	0.23	0	0	0.40	0.33
5	0	0	0	0	1	0.31	0	0.24	0.27	0
6	0	0	0	0	0	1	0	0.37	0	0.22
7	0	0	0	0	0	0	1	0.22	0.30	0.36
8	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	1

Table 9 Specification of delay recommended for applications' modules

Application modules	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	0.26	0.28	0.20	0.23
2	0	0	0.20	1	0.28	0.25	0.24	0.28	1	0.25
3	0	0	0	0.24	0.21	0.26	0.29	0.22	1	0.21
4	0	0	0	0	0.20	0.22	1	1	0.23	0.24
5	0	0	0	0	0	0.24	1	0.30	0.26	1
6	0	0	0	0	0	0	1	0.21	1	0.23
7	0	0	0	0	0	0	0	0.28	0.28	0.23
8	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0

6.2.1 First category: the number of fog nodes is variable and the number of modules is fixed

In this subsection, four different scenarios are investigated. Figure 12 demonstrates performance comparison of different algorithms for solving the first scenario, in a platform containing 10 fog nodes with 20 requested modules when *FTT* parameter is 0.24. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 2, 4, 6, 8, and 10 which our proposed algorithm determined. Figure 12a compares Pareto fronts associated with different algorithms; as it shows, our proposed algorithm outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 12b, c shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 12d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 12.

Furthermore, the average elapsed time of different algorithms are compared in Table 11. In contrast to other algorithms, our algorithm has favorite time elapsed.

Figure 13 illustrates performance comparison of different algorithms for solving the second scenario, in a platform containing 15 fog nodes with 20 requested modules when *FTT* parameter is 0.24. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 4, 10, 12, 14, and 15 which our proposed algorithm determined. Figure 13a compares Pareto fronts concern to different algorithms; as it draws, our proposed algorithm has dominance against others regarding the quality and number of finding non-dominated solutions. Figure 13b, c indicates that proposed algorithm has better results against others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 13d, e. Figure 13d proves dominance of proposed algorithm in terms of the best result of the first objective function, but Fig. 13e shows that our algorithm is marginally in the third place

Table 10 Parameter settings of algorithms selected for comparison

Algorithms	Specific parameters				Number of objective	Population size	Max iterations
MOGA	Crossover Percentage:		0.7		2	100	100
	Mutation Percentage:		0.4				
MODGWO	Archive size:	100	Alpha:	0.1			
	nGrid:	10	Beta:	4			
			Gamma:	2			
MODCS	Pa:	0.25					
MODPSO	nGrid:	20	W:	0.4			
	maxvel:	5	C1:	2			
	u_mut:	0.5	C2:	2			

Alpha: Grid inflation parameter

Beta: Leader selection pressure parameter

maxvel: Maximum velocity in percentage (search space percentage)

Gamma: Extra repository member selection pressure

u_mut: Uniform mutation percentage

Archive size: Repository size

W: Inertia weight

nGrid: Number of grids per each dimension

C1: Individual confidence factor

Pa: Discovery rate of alien eggs/solutions

C2: Swarm confidence factor

after MODGWO-I and MODCS in terms of the best of second objective function. In addition, our best so far plan is drawn in Fig. 13f.

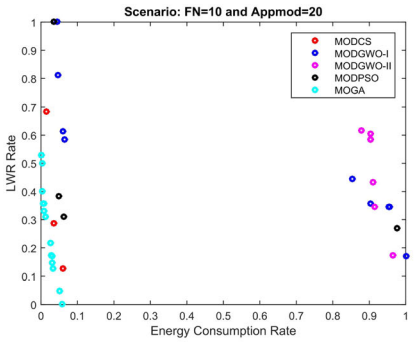
In addition, the average elapsed time of different algorithms are compared in Table 12. In regard to elapsed time, two algorithms MODPSO and MODCS have the shortest value, but in regard to overall objective functions all of them fall behind to our proposed algorithm.

Figure 14 demonstrates performance comparison of different algorithms for solving the third scenario, in a platform containing 20 fog nodes with 20 requested modules when *FTT* parameter is 0.2. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 2, 5, 10, 12, and 14 which our proposed algorithm determined. Figure 14a compares Pareto fronts associated with different algorithms; as it proves, our proposed algorithm outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 14b, c shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In addition, the best results depicted in Fig. 14d, e prove dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 14f.

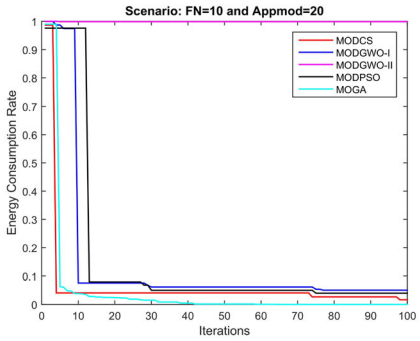
In addition, the average elapsed time of different algorithms are compared in Table 13. In regard to elapsed time, MODPSO has the shortest value, but in regard to overall objective functions our proposed algorithm is in the first place.

Figure 15 shows the performance comparison of different algorithms for solving the fourth scenario, in a platform containing 25 fog nodes with 20 requested modules when *FTT* parameter is 0.2. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 3, 8, 12, 13, and 15 which our proposed algorithm determined. Figure 15a compares Pareto fronts relevant to different algorithms; as it indicates, our proposed algorithm beats others regarding the quality and number of finding non-dominated solutions. Figure 15b, c indicates that proposed algorithm outperforms against others in terms of two objective functions with the increase of iterations. In this line, the best results are illustrated in Fig. 15d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 15f.

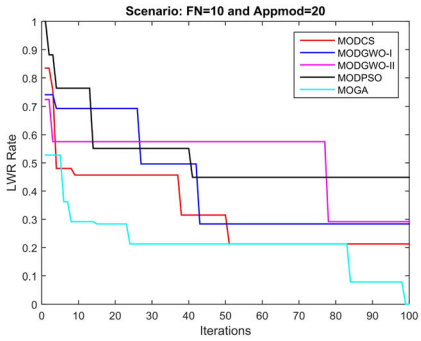
In addition, elapsed time comparison of different algorithms which is placed in Table 14 proves that our proposed algorithm has better performance in terms of execution time.



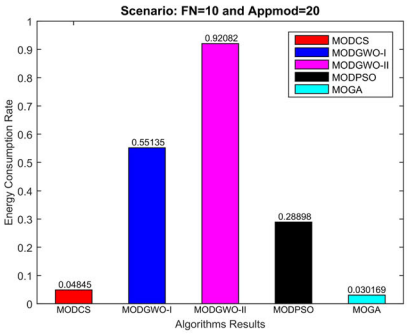
a. Pareto Front Comparison



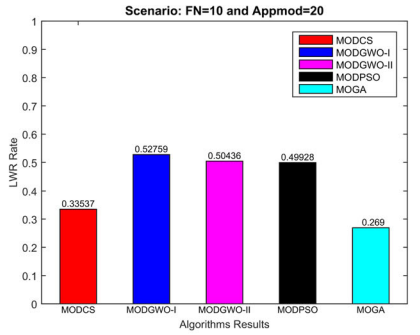
b. Performance comparison in term of total power consumption based on Eq. (23).



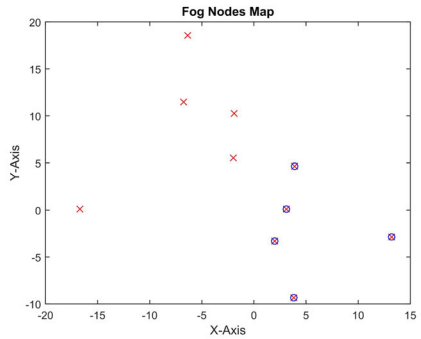
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

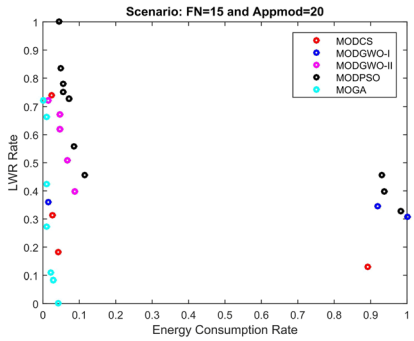
Fig. 12 Simulation results of a scenario in a platform containing 10 fog nodes with 20 requested modules

Table 11 Average elapsed time comparison of studies for a scenario in a platform containing 10 fog nodes with 20 requested modules

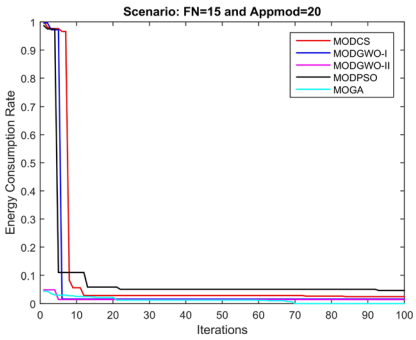
MOGA:	240.67 s	MODGWO-I:	876.99 s	MODPSO:	213.91 s
MODCS:	242.07 s	MODGWO-II:	266.67 s		

6.2.2 Second category: the number of fog nodes is fixed and the number of modules is variable

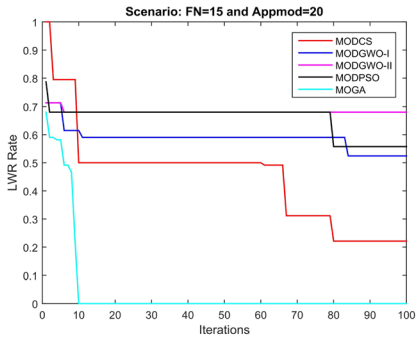
In this subsection, four scenarios numbered 5 through 8 are investigated. Figure 16 demonstrates performance comparison of different algorithms for solving the fifth scenario, in a platform containing 15 fog nodes with 20 requested modules when *FTT* parameter is 0.24. Note that in our simulations there may be similar scenarios, but the values used in datasets are completely different. For instance, the fifth scenario is similar to second one, but the data are different. Anyway, in regard to *FTT* parameter in fifth scenario, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 2, 3, 5, 7, and 8 which our proposed algorithm determined. Figure 16a compares Pareto fronts associated with different algorithms; as it shows, our proposed algorithm significantly outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 16b, c also shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 16d, e which proves dominance of proposed



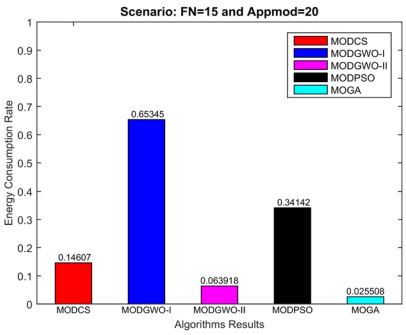
a. Pareto Front Comparison



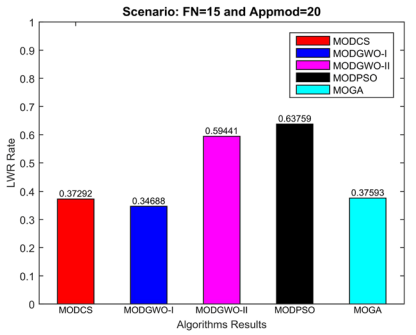
b. Performance comparison in term of total power consumption based on Eq. (23).



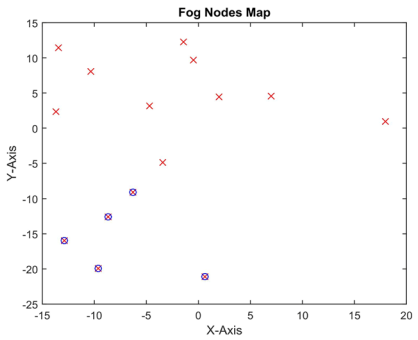
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 13 Simulation results of a scenario in a platform containing 15 fog nodes with 20 requested modules

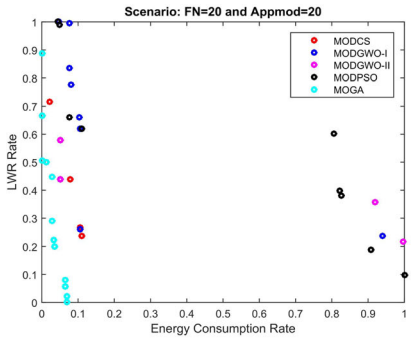
Table 12 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 20 requested modules

MOGA:	268.22 s	MODGWO-I:	968.46 s	MODPSO:	225.21 s
MODCS:	242.92 s	MODGWO-II:	272.28 s		

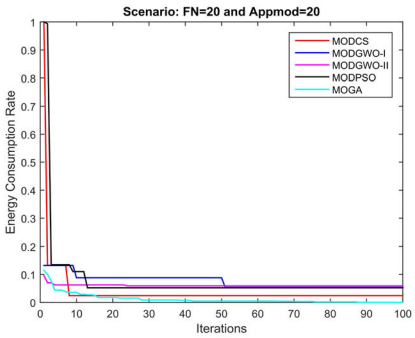
algorithm. In addition, our best so far plan is depicted in Fig. 16f.

In addition, elapsed time comparison of different algorithms which is placed in Table 15 proves that our proposed algorithm has better performance in terms of execution time.

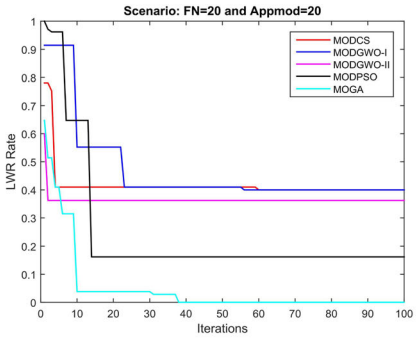
Figure 17 depicts performance comparison of different algorithms for solving the sixth scenario, in a platform containing 15 fog nodes with 25 requested modules when *FTT* parameter is 0.24. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 2, 3, 4, 9, and 13 which our proposed algorithm determined. Figure 17a compares Pareto fronts related to different algorithms; as it shows, our proposed algorithm outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 17b shows that proposed algorithm is marginally in the third place after two competing MODCS and MODPSO algorithm in terms of first objective function, but Fig. 17c indicates our proposed algorithm and MODCS, which having near results, beat others in terms of second objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 17d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 17f.



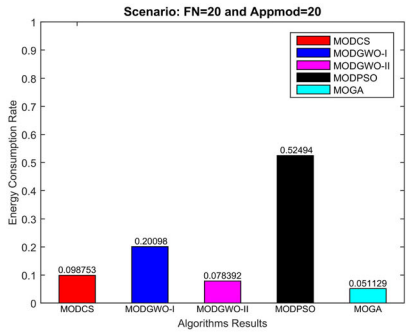
a. Pareto Front Comparison



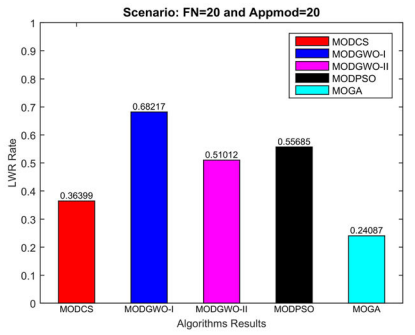
b. Performance comparison in term of total power consumption based on Eq. (23).



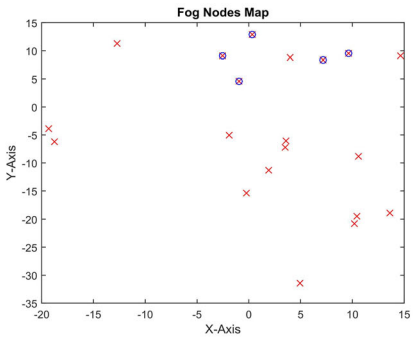
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

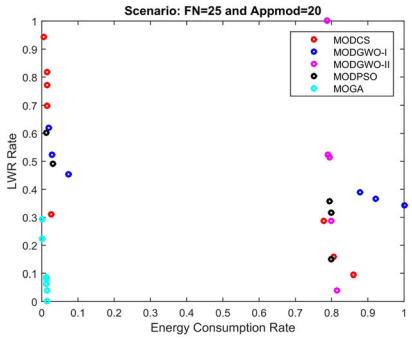
Fig. 14 Simulation results of a scenario in a platform containing 20 fog nodes with 20 requested modules

Table 13 Average elapsed time comparison of studies for a scenario in a platform containing 20 fog nodes with 20 requested modules

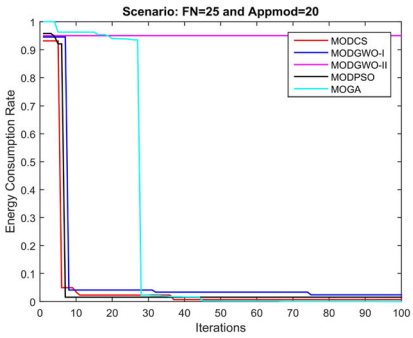
MOGA:	240.62 s	MODGWO-I:	1162.5 s	MODPSO:	219.86 s
MODCS:	248.63 s	MODGWO-II:	288.12 s		

In addition, elapsed time comparison of different algorithms is placed in Table 16. Although our proposed algorithm is ranked in second place after MODCS in terms of execution time, our proposed algorithm has better performance in terms of objective functions.

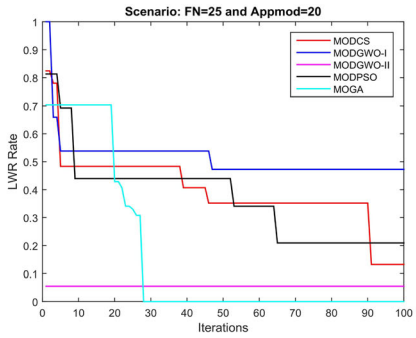
Figure 18 demonstrates performance comparison of different algorithms for solving the seventh scenario, in a platform containing 15 fog nodes with 30 requested modules when *FTT* parameter is 0.21. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 3, 7, 8, 9, 10, and 15 which our proposed algorithm determined. Figure 18a compares Pareto fronts associated with different algorithms; as it shows, our proposed algorithm outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 18b shows that proposed algorithm beats others in terms of first objective function, but Fig. 18c indicates our proposed algorithm after MODCS, which having near results, beat others in terms of second objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 18d, e which proves dominance of proposed algorithm against others in terms of the first objective and second place after MODCS in second objective. In addition, our best so far plan is depicted in Fig. 18f.



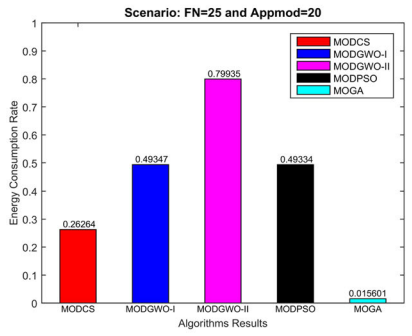
a. Pareto Front Comparison



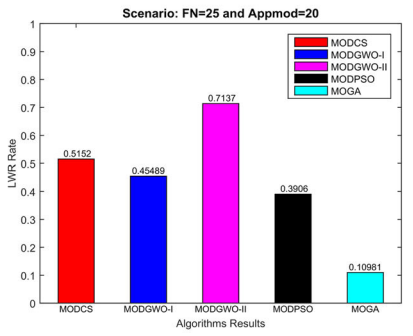
b. Performance comparison in term of total power consumption based on Eq. (23).



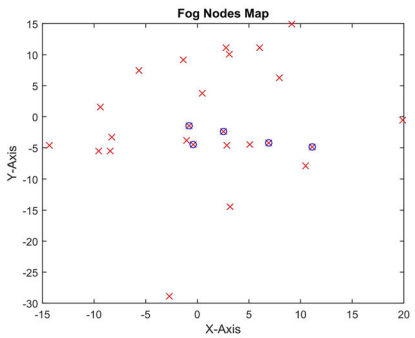
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

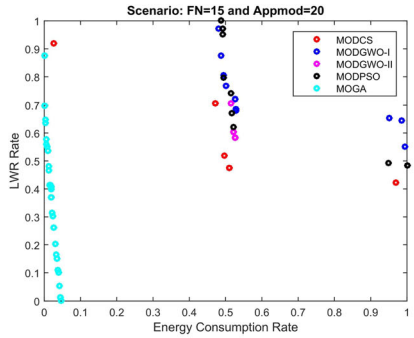
Fig. 15 Simulation results of a scenario in a platform containing 25 fog nodes with 20 requested modules

Table 14 Average elapsed time comparison of studies for a scenario in a platform containing 25 fog nodes with 20 requested modules

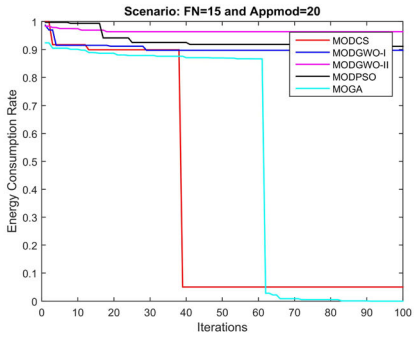
MOGA:	262.75 s	MODGWO-I:	982.53 s	MODPSO:	241.54 s
MODCS:	273.01 s	MODGWO-II:	311.69 s		

In addition, elapsed time comparison of different algorithms is placed in Table 17 which has similar result in comparison to previous scenario. Although our proposed algorithm is ranked in the second place after MODCS in terms of execution time, our proposed algorithm has better performance in terms of objective functions.

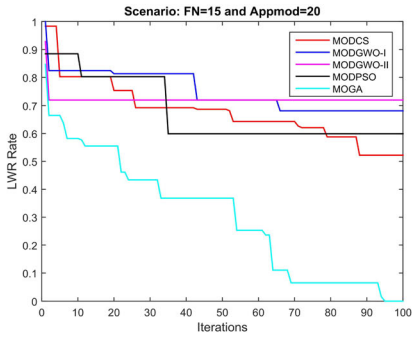
Figure 19 demonstrates performance comparison of different algorithms for solving the eighth scenario, in a platform containing 15 fog nodes with 35 requested modules when *FTT* parameter is 0.23. In regard to *FTT* parameter, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 2, 9, 10, 11, 12, 13, and 14 which our proposed algorithm determined. Figure 19a compares Pareto fronts associated with different algorithms; as it shows, our proposed algorithm outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 19b shows that proposed algorithm beats others in terms of first objective function, but Fig. 19c indicates our proposed algorithm after MODCS, which having near results, beat others in terms of second objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 19d, e which proves dominance of proposed algorithm against others in terms of both objective functions. In addition, our best so far plan is depicted in Fig. 19f.



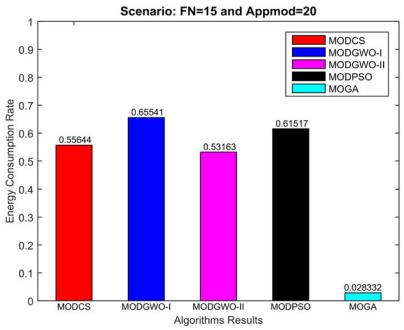
a. Pareto Front Comparison



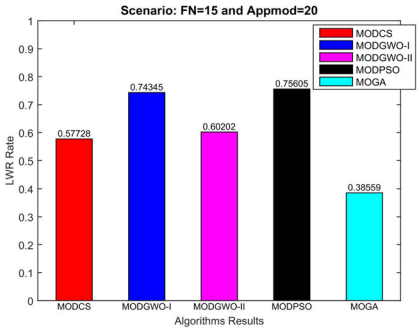
b. Performance comparison in term of total power consumption based on Eq. (23).



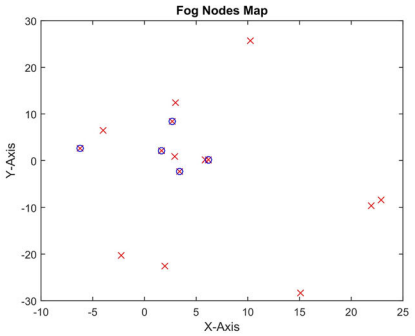
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 16 Simulation results of a scenario in a platform containing 15 fog nodes with 20 requested modules

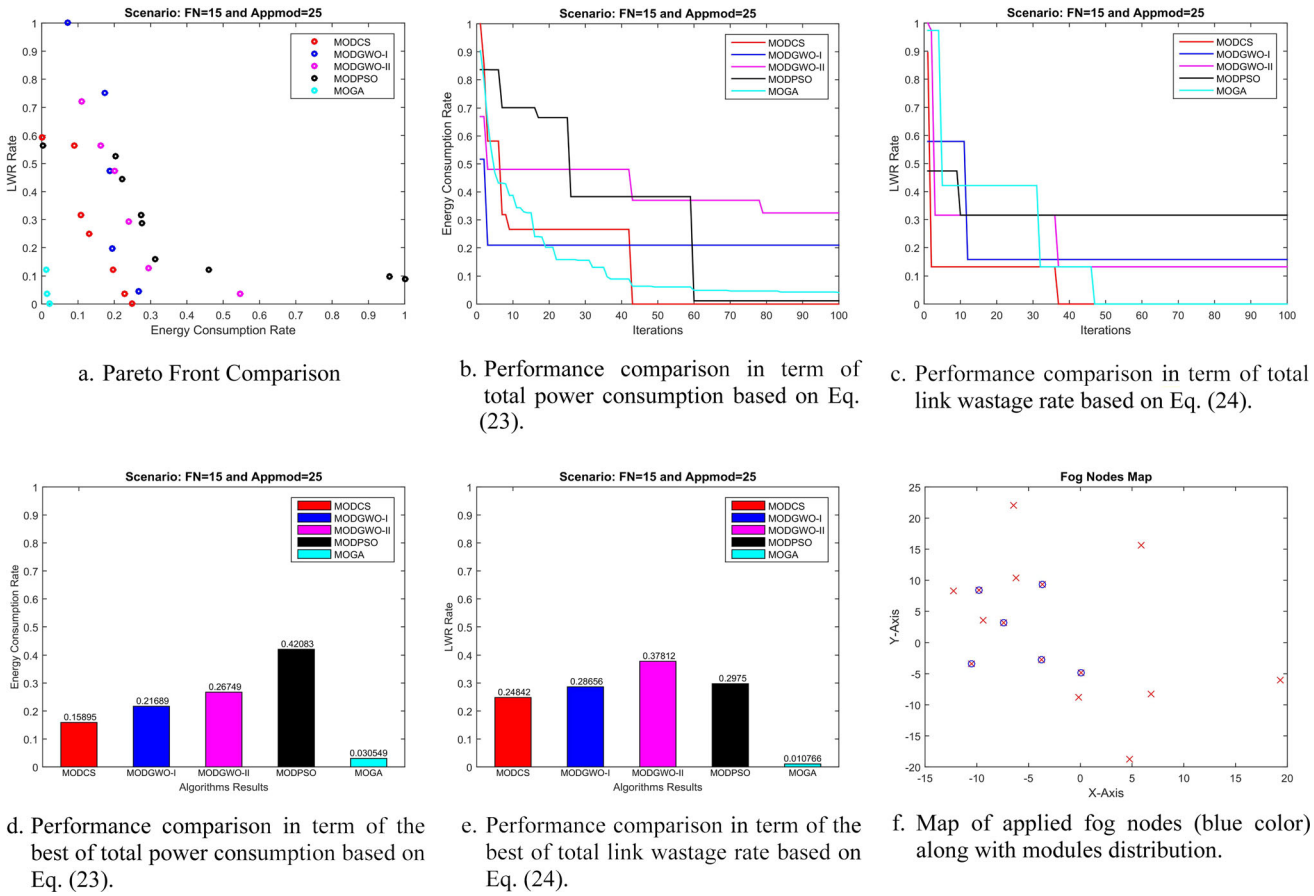
Table 15 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 20 requested modules

MOGA:	240.43 s	MODGWO-I:	926.41 s	MODPSO:	219.84 s
MODCS:	250.48 s	MODGWO-II:	280.72 s		

In addition, elapsed time comparison of different algorithms is placed in Table 18 which has similar result in comparison with the previous scenario. Although our proposed algorithm is ranked in second place after MODCS in terms of execution time, our proposed algorithm has better performance in terms of objective functions.

6.2.3 Third category: the number of fog nodes is variable and the number of modules is variable

In this subsection, four other scenarios numbered 9 through 12 are investigated. In this line, Fig. 20 depicts performance comparison of different algorithms for solving the ninth scenario, in a platform containing 10 fog nodes with 25 requested modules when *FTT* parameter is 0.19. Note that in our simulations there may be similar scenarios, but the values used in datasets are completely different. For instance, the tenth scenario is similar to seventh one, but the data are different. Anyway, in regard to *FTT* parameter in fifth scenario, at least 6 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 2, 4, 6, 9, and 10 which our proposed algorithm determined. Figure 20a compares Pareto fronts associated with different algorithms; as it illustrates, our proposed



a. Pareto Front Comparison

b. Performance comparison in term of total power consumption based on Eq. (23).

c. Performance comparison in term of total link wastage rate based on Eq. (24).

d. Performance comparison in term of the best of total power consumption based on Eq. (23).

e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).

f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 17 Simulation results of a scenario in a platform containing 15 fog nodes with 25 requested modules

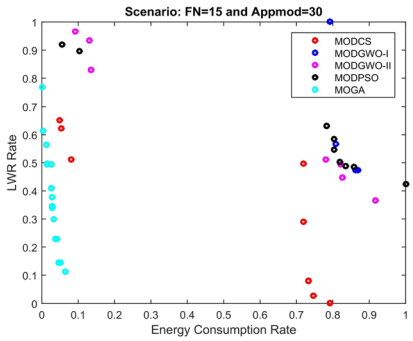
Table 16 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 25 requested modules

MOGA:	324.74 s	MODGWO-I:	1105.6 s	MODPSO:	279.88 s
MODCS:	424.28 s	MODGWO-II:	353.12 s		

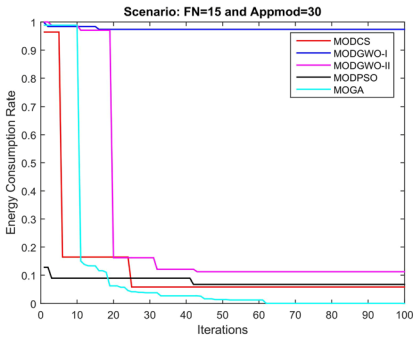
algorithm significantly outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 20b, c also shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 20d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 20f.

In addition, the average elapsed time of different algorithms are compared in Table 19. In contrast to other algorithms, our algorithm has favorite time elapsed.

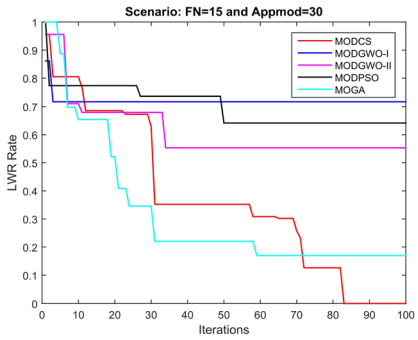
Figure 21 depicts performance comparison of different algorithms for solving the tenth scenario, in a platform containing 15 fog nodes with 30 requested modules when *FTT* parameter is 0.18. In regard to *FTT* parameter in tenth scenario, at least 6 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 3, 6, 8, 9, and 13 which our proposed algorithm determined. Figure 21a compares Pareto fronts associated with different algorithms; as it illustrates, our proposed algorithm significantly outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 21b, c also shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 21d, e which proves dominance of



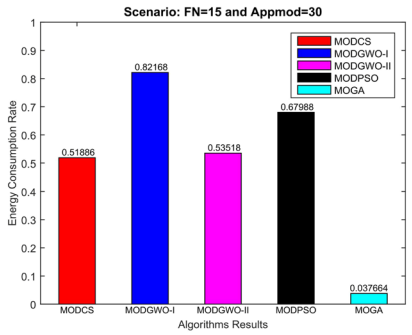
a. Pareto Front Comparison



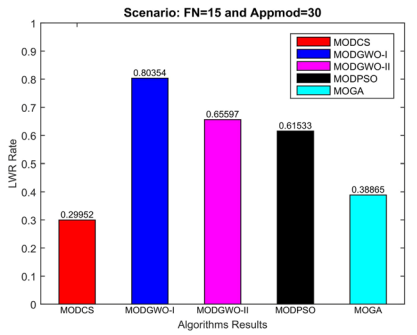
b. Performance comparison in term of total power consumption based on Eq. (23).



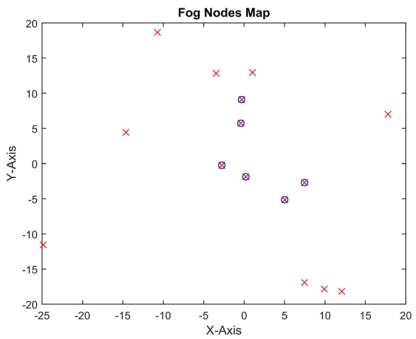
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 18 Simulation results of a scenario in a platform containing 15 fog nodes with 30 requested modules

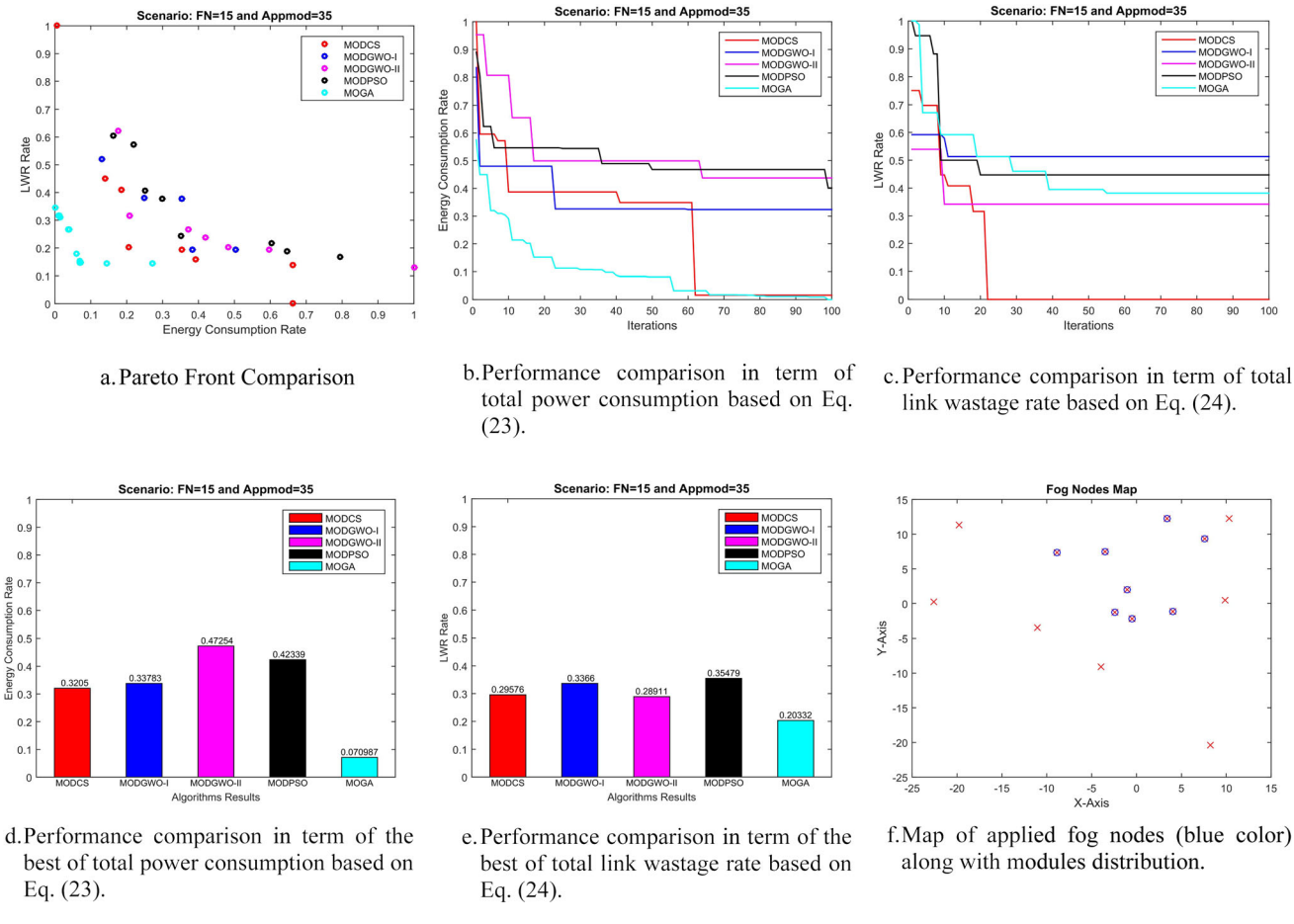
Table 17 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 30 requested modules

MOGA:	397.91 s	MODGWO-I:	1297.5 s	MODPSO:	317.68 s
MODCS:	344.86 s	MODGWO-II:	353.7 s		

proposed algorithm. In addition, our best so far plan is depicted in Fig. 21f.

In addition, the average elapsed time of different algorithms are compared in Table 20. In contrast to other algorithms, our algorithm has favorite time elapsed.

Figure 22 illustrates performance comparison of different algorithms for solving the eleventh scenario, in a platform containing 20 fog nodes with 35 requested modules when *FTT* parameter is 0.24. In regard to *FTT* parameter in tenth scenario, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 4, 8, 10, 15, 17, and 18 which our proposed algorithm determined. Figure 22a compares Pareto fronts associated with different algorithms; as it illustrates, our proposed algorithm significantly outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 22b, c also shows that proposed algorithm beats others in terms of two objective functions with the increase of iterations. In this line, the best results are depicted in Fig. 22d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 22f.



a. Pareto Front Comparison

b. Performance comparison in term of total power consumption based on Eq. (23).

c. Performance comparison in term of total link wastage rate based on Eq. (24).

d. Performance comparison in term of the best of total power consumption based on Eq. (23).

e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).

f. Map of applied fog nodes (blue color) along with modules distribution.

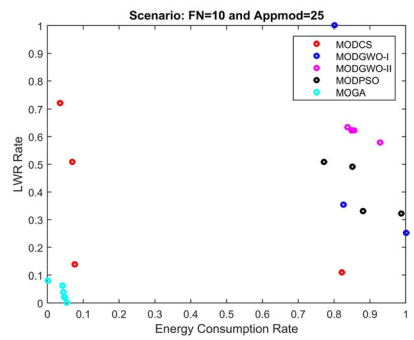
Fig. 19 Simulation results of a scenario in a platform containing 15 fog nodes with 35 requested modules

Table 18 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 35 requested modules

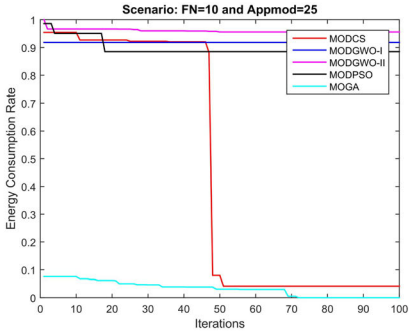
MOGA:	515.6 s	MODGWO-I:	1602.7 s	MODPSO:	421.00 s
MODCS:	445.48 s	MODGWO-II:	407.33 s		

In addition, the average elapsed time of different algorithms are compared in Table 21. In contrast to other algorithms, our algorithm has favorite time elapsed.

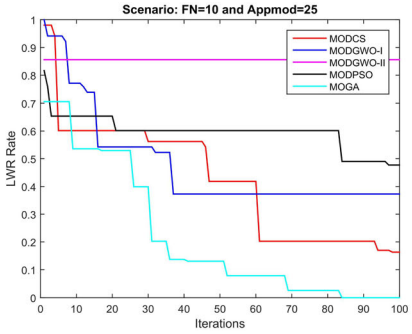
Figure 23 illustrates performance comparison of different algorithms for solving the twelfth scenario, in a platform containing 25 fog nodes with 40 requested modules when *FTT* parameter is 0.20. In regard to *FTT* parameter in tenth scenario, at least 5 fog nodes are needed for module deployment. The best so far solution is to select node numbers 1, 5, 6, 13, 14, 17, 18, 19, 20, and 25 which our proposed algorithm determined. Figure 23a compares Pareto fronts associated with different algorithms; as it illustrates, our proposed algorithm significantly outperforms against others regarding the quality and number of finding non-dominated solutions. Figure 23b shows that after MODCS, the proposed algorithm beats others in terms of first objective function, whereas Fig. 23c shows that proposed algorithm beats others in terms of second objective functions with the increase of iterations. In this line, average results are depicted in Fig. 23d, e which proves dominance of proposed algorithm. In addition, our best so far plan is depicted in Fig. 23f.



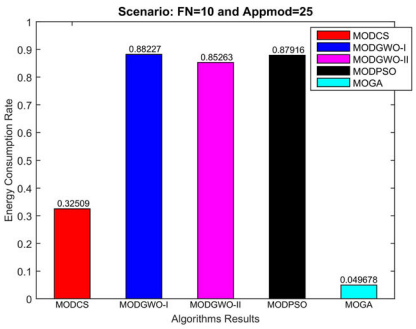
a. Pareto Front Comparison



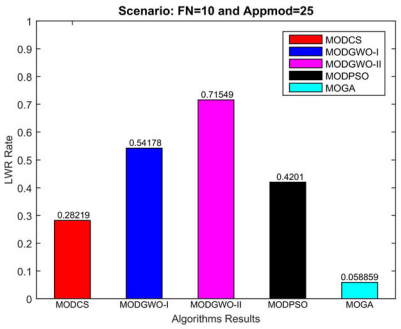
b. Performance comparison in term of total power consumption based on Eq. (23).



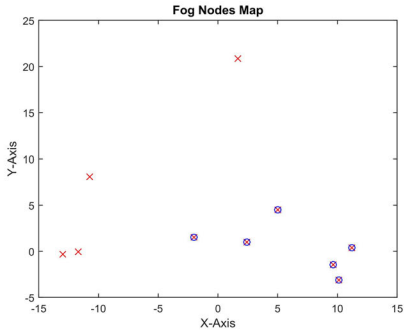
c. Performance comparison in term of total link wastage rate based on Eq. (24).



d. Performance comparison in term of the best of total power consumption based on Eq. (23).



e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).



f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 20 Simulation results of a scenario in a platform containing 10 fog nodes with 25 requested modules

Table 19 Average elapsed time comparison of studies for a scenario in a platform containing 10 fog nodes with 25 requested modules

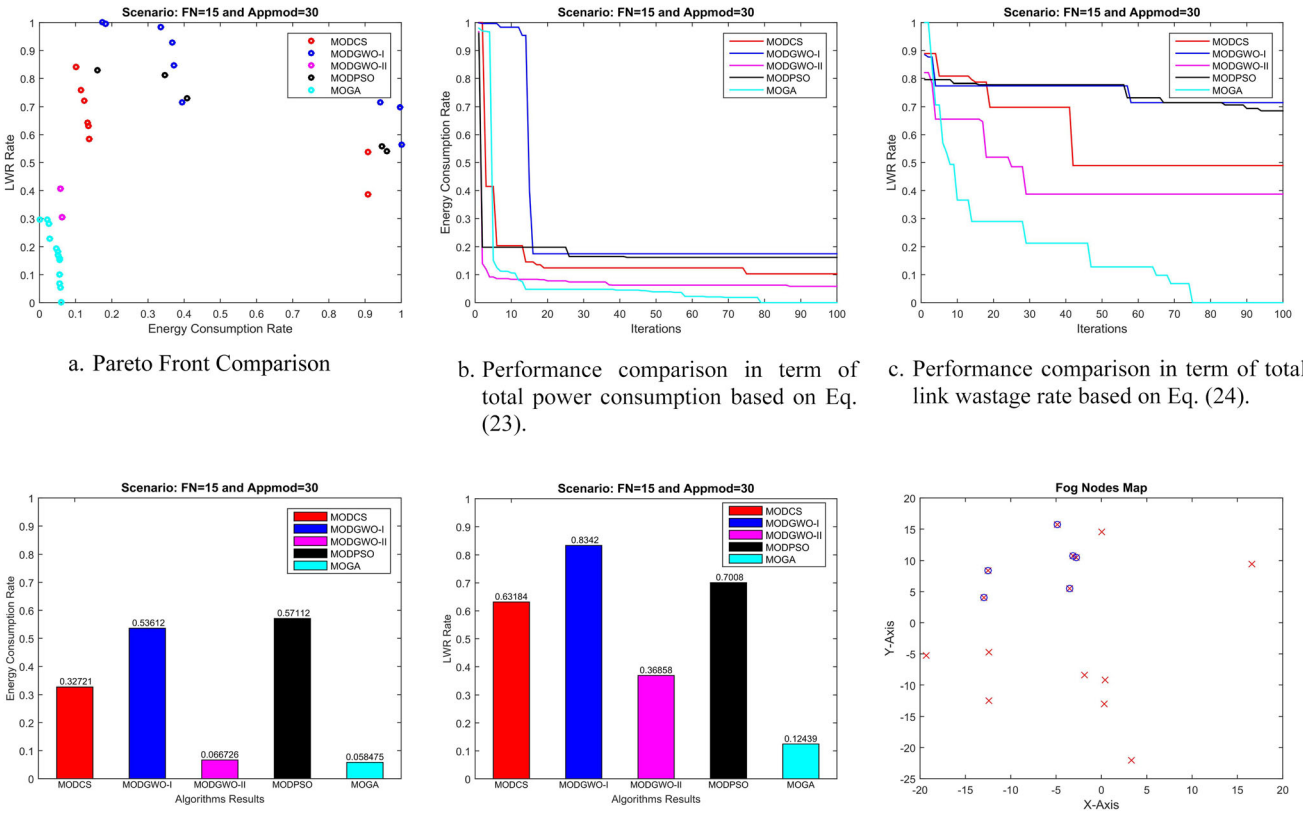
MOGA:	261.69 s	MODGWO-I:	1197.1 s	MODPSO:	284.04 s
MODCS:	277.99 s	MODGWO-II:	304.84 s		

In addition, the average elapsed time of different algorithms are compared in Table 22. In contrast to other algorithms, our algorithm has favorite time elapsed.

To closer look and to have better analysis for evaluation of comparative algorithms in different scenarios, the descriptive statistics gives supportive information. To this end, Tables 23, 24, 25, 26, 27, and 28 are dedicated. Table 23 shows the *min* and *Max* values associated with the first objective (*TPC*) gained from simulation results in different scenarios. Note that the *min* and *Max* values of proposed MOGA is less than others in terms of *TPC* as the first objective function.

Table 24 shows the *min* and *Max* values associated with the second objective (*LWR*) gained from simulation results in different scenarios. Note that the *min* and *Max* values of proposed MOGA is less than others in terms of *LWR* as the second objective function except for some cases for lower search space that MOPSO is better.

Table 25 compares the average values of different algorithm in terms of the first objective function along with relative percentage deviation (*RPD*) [7]. As Table 25 shows, the proposed MOGA has the best average value. In addition, the improvement of the proposed MOGS against other state of the art is separately reported.



a. Pareto Front Comparison

b. Performance comparison in term of total power consumption based on Eq. (23).

c. Performance comparison in term of total link wastage rate based on Eq. (24).

d. Performance comparison in term of the best of total power consumption based on Eq. (23).

e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).

f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 21 Simulation results of a scenario in a platform containing 15 fog nodes with 30 requested modules

Table 20 Average elapsed time comparison of studies for a scenario in a platform containing 15 fog nodes with 30 requested modules

MOGA:	288.78 s	MODGWO-I:	1503.2 s	MODPSO:	294.17 s
MODCS:	327.15 s	MODGWO-II:	392.15 s		

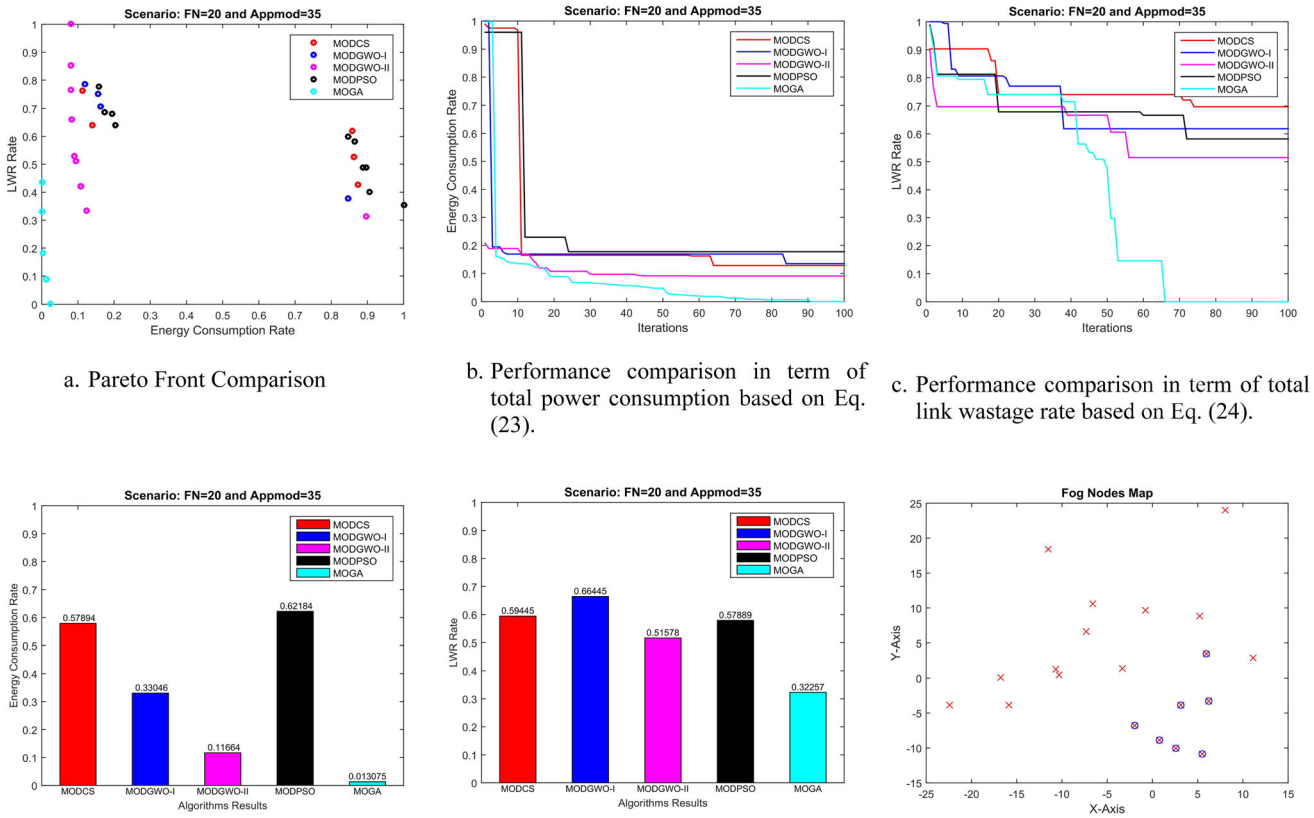
Table 26 compares the average values of different algorithm in terms of the second objective function along with relative percentage deviation (*RPD*). As Table 26 shows, the proposed MOGA has the best average value. In addition, the improvement of the proposed MOGS against other state of the art is separately reported.

To prove the convergence of comparative algorithms, the *STD* value for both objective functions are brought in Tables 27 and 28, respectively.

The lowest *STD* value of proposed MOGA in both objective function associated with all scenarios proves the high rate of convergence and low data skewness.

7 Discussion

In the fog systems, there are two prominent stakeholders which are service providers and resource requesters. Each stakeholder needs to meet their requirement based on its business objectives. One of the most important issues in the reduction in the providers' capital expenditure (CAPEX) is associated with power management [32], the reason why the first objective of the current paper is to minimize the *TPC* during the IoT applications' deployment. Secondly, the IoT applications are typically wireless and their performance are strongly dependent on the common



a. Pareto Front Comparison

b. Performance comparison in term of total power consumption based on Eq. (23).

c. Performance comparison in term of total link wastage rate based on Eq. (24).

d. Performance comparison in term of the best of total power consumption based on Eq. (23).

e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).

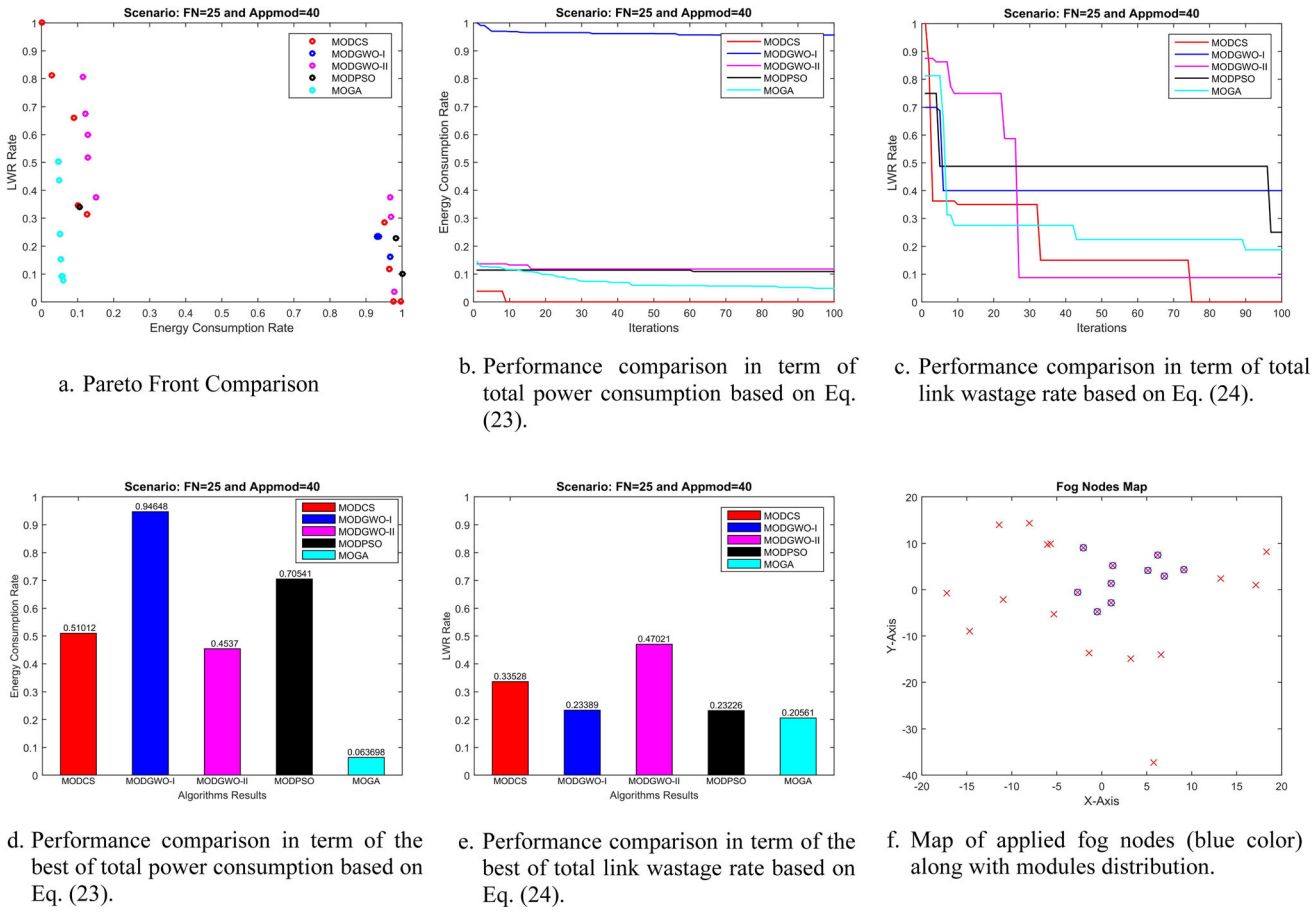
f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 22 Simulation results of a scenario in a platform containing 20 fog nodes with 35 requested modules

Table 21 Average elapsed time comparison of studies for a scenario in a platform containing 20 fog nodes with 35 requested modules

MOGA:	323.22 s	MODGWO-I:	1466.9 s	MODPSO:	338.04 s
MODCS:	366.72 s	MODGWO-II:	404.11 s		

bandwidth usage the reason why the second objective of this paper is to minimize total bandwidth wastage rate. On the other side, the users eventually release cloud/fog service providers which deliver unreliable service provisioning [36]; so, the minimum number of processing nodes which meet the least reliability execution of IoT applications are considered in the constraints of the proposed model. To solve this combinatorial problem, the MOGA model was proposed which takes benefit of several novel operators each of which is exploited in timely manner. The new utilized operators improve the quality of produced solutions. Firstly, it applies the new efficient selection strategy that works better than existing tournament, roulette wheel, and rank-biased strategies. The single-point crossover and random mutation is utilized similar to multi-objective genetic presented by Hosseini et al. in [6]. Also, the crowding distance procedure is customized based on the stated problem (c.f. in Algorithm 9) to produce randomly diverse solutions which contingently lead to efficient solutions. One important novelty of the current solution is to apply two heuristic algorithms 1 and 2 as preprocessing by incorporating the clique concept from graph theory which accurately confines the search space and increase the convergence speed. To evaluate the performance of the



a. Pareto Front Comparison

b. Performance comparison in term of total power consumption based on Eq. (23).

c. Performance comparison in term of total link wastage rate based on Eq. (24).

d. Performance comparison in term of the best of total power consumption based on Eq. (23).

e. Performance comparison in term of the best of total link wastage rate based on Eq. (24).

f. Map of applied fog nodes (blue color) along with modules distribution.

Fig. 23 Simulation results of a scenario in a platform containing 25 fog nodes with 40 requested modules

Table 22 Average elapsed time comparison of studies for a scenario in a platform containing 25 fog nodes with 40 requested modules

MOGA:	517.25 s	MODGWO-I:	1750.5 s	MODPSO:	468.13 s
MODCS:	652.26 s	MODGWO-II:	479.67 s		

proposed MOGA model against other state-of-the-art MODCS, MOGWO-I, MOGWO-II, and MOPSO, 12 extensive scenarios were conducted. All of the aforementioned state of the art were customized based on the stated problem and have been run on the same platform and datasets with the same conditions to reach fair results and comparisons. The result of extensive simulations proves that the proposed MOGA model has 18, 38, 9, and 43 percent of improvement against MODCS, MOGWO-I, MOGWO-II, and MOPSO in terms of *TPC* and it has 6.4, 15.99, 28.15, and 15.43 dominance percent against them in terms of *LWR*, respectively. In terms of real run time, the average execution times of MODCS, MOGWO-I, MOGWO-II, MOPSO, and proposed MOGA models in all 12 scenarios are, respectively, 341.32, 1236.70, 342.87, 293.61, and 323.49 s. Although the convergence speed of MOGA model in solving the stated multi-objective optimization problem is marginally in the second ranking after MOPSO, the closer look in the quality of generated solutions in terms of objective functions shows that the MOGA model beats others. Consequently, it is worth mentioning that the quality of solutions generated by MOGA model is acceptable in comparison with other models with overlooking the little bit slow convergence in regarding MOPSO model.

8 Conclusion and future direction

This paper focused on problem of application module deployment on cloud and fog nodes with regard to two prominent stakeholders' viewpoints, namely users and

Table 23 Obtained range of *TPC* value in terms of minimum and maximum associated with different algorithms

Min–Max							
Scenarios	FN	Appmod	*TPC*				
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*
Category 1	10	20	589.28–590.97	588.29–590.94	593.75–595.08	590.44–592.34	588.32–589.54
	15	20	596.36–602.44	599.98–604.84	606.766–611.70	600.83–604.91	595.79–599.81
	20	20	595.59–600.12	598.75–602.26	602.31–605.78	595.24–602.20	595.25–598.39
	25	20	580.82–583.65	580.68–583.04	586.45–589.95	580.97–583.39	580.19–581.94
Category 2	15	20	591.57–593.74	592.12–594.27	595.24–597.51	592.96–594.93	591.09–592.99
	15	25	689.53–693.33	689.36–694.40	694.42–696.37	691.03–695.57	689.67–691.83
	15	30	728.49–733.58	732.76–735.98	734.25–739.72	732.63–736.70	728.43–731.89
	15	35	957.56–961.41	961.01–962.69	959.24–963.54	959.67–961.61	957.44–958.93
Category 3	10	25	694.93–701.28	693.56–701.99	701.701–705.24	694.25–701.46	693.36–696.96
	15	30	758.03–761.90	760.48–763.72	762.70–767.80	762.05–765.50	755.61–759.42
	20	35	879.16–884.27	879.01–883.20	884.71–887.92	881.43–886.25	878.76–881.61
	25	40	1198.28–1200.93	1199.54–1202.03	1199.41–1201.03	1199.54–1201.75	1195.20–1197.01

Table 24 Obtained range of *LWR* value in terms of minimum and maximum associated with different algorithms

Min–Max							
Scenarios	FN	Appmod	*LWR*				
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*
Category 1	10	20	0.84–1.09	0.84–1.14	1.24–1.49	0.78–1.14	0.81–1.02
	15	20	0.87–1.15	0.92–1.36	1.27–1.61	1.03–1.24	0.82–0.97
	20	20	0.76–1.12	1.15–1.41	1.16–1.69	0.99–1.27	0.73–1.01
	25	20	0.98–1.2	1.1–1.37	1.36–1.62	1.08–1.36	0.84–1.07
Category 2	15	20	1.86–2.16	2.05–2.31	2.51–2.84	1.98–2.22	1.8–2.04
	15	25	0.83–0.85	0.85–0.88	0.95–1.15	0.85–0.88	0.82–0.83
	15	30	1.25–1.47	1.57–1.92	2.07–2.26	1.76–2.04	1.21–1.36
	15	35	2.95–3.07	3–3.17	2.98–3.16	3.06–3.2	2.89–3.01
Category 3	10	25	2.55–2.91	2.37–2.74	3.16–3.66	2.34–2.89	2.31–2.7
	15	30	1.21–1.68	1.72–2.15	1.67–2.44	1.62–2.02	1.11–1.5
	20	35	1.8–2.07	1.92–2.21	2.05–2.38	1.89–2.27	1.7–1.94
	25	40	2.68–3	2.89–3.06	2.86–3.07	2.94–3.16	2.6–2.79

providers. For users viewpoints, the QoS and reliability are the most relevant objectives and for providers the power consumption is one of the most concerns. This is the reason this paper formulates the IoT application deployment modules to a multi-objective QoS-aware reliable optimization problem with minimization of total energy consumption and total link wastage rate inclination. To solve this combinatorial problem, a multi-objective genetic algorithm (MOGA) has been presented. To validate the proposed GA-based algorithm, extensive scenarios were conducted. It was evaluated against some other comparative state of the art in different scenarios. The analysis and assessment were based on descriptive statistics, namely *min*, *Max*, average and *STD* values to support the simulation results. The simulation results gained from extensive scenarios prove the superiority of proposed algorithm in terms of prominent evaluation parameters along with analysis of descriptive statistics. Totally in all 12 scenarios on average, the proposed MOGA model beats other comparatives models with the amount of 27 and 16.49%

Table 25 Performance comparison of different algorithms in terms of *TPC* mean value

Mean value											
Scenarios	FN	Appmod	*TPC*					RPD (%)			
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*	*MODCS*	*MOGWO-I*	*MOGWO-II*	*MOPSO*
Category 1	10	20	590.4	589.97	594.46	591.32	588.93	0.25	0.18	0.93	0.4
	15	20	599.97	602.23	608.91	603.02	598.09	0.31	0.69	1.78	0.82
	20	20	597.41	600.63	604.31	599.66	596.57	0.14	0.68	1.28	0.52
	25	20	582.42	582.13	588.04	582.38	581.14	0.22	0.17	1.17	0.21
Category 2	15	20	592.53	593.47	596.56	593.96	591.84	0.12	0.28	0.79	0.36
	15	25	690.98	692.1	695.26	692.9	690.96	0	0.17	0.62	0.28
	15	30	730.53	734.5	737.41	734.5	730.55	0	0.54	0.93	0.54
	15	35	960.26	961.75	961.24	960.82	958.3	0.21	0.36	0.31	0.26
Category 3	10	25	698.54	699.1	703.21	697.96	695.72	0.4	0.48	1.07	0.32
	15	30	759.05	762.19	764.73	764.04	758.09	0.13	0.54	0.87	0.78
	20	35	881.44	881.22	886.88	883.36	880.76	0.08	0.05	0.69	0.29
	25	40	1199.48	1201.17	1200.39	1200.65	1195.83	0.3	0.44	0.38	0.4
AVG								0.18	0.38	0.9	0.43

Table 26 Performance comparison of different algorithms in terms of *LWR* mean value

Mean value											
Scenarios	FN	Appmod	*LWR*					RPD (%)			
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*	*MODCS*	*MOGWO-I*	*MOGWO-II*	*MOPSO*
Category 1	10	20	0.97	0.96	1.36	0.94	0.89	7.84	6.88	34.07	5.3
	15	20	1	1.11	1.45	1.11	0.9	10	19.06	38.1	18.92
	20	20	0.9	1.27	1.46	1.11	0.85	4.9	32.86	41.59	22.78
	25	20	1.09	1.28	1.55	1.27	0.95	12.82	25.63	38.66	25.16
Category 2	15	20	2.04	2.19	2.71	2.12	1.92	6.16	12.34	29.17	9.7
	15	25	0.83	0.87	1.03	0.87	0.83	0.72	5.26	19.92	5.26
	15	30	1.34	1.73	2.2	1.91	1.28	4.32	25.87	41.58	32.77
	15	35	3.01	3.1	3.06	3.12	2.95	2.26	4.97	3.79	5.64
Category 3	10	25	2.73	2.56	3.4	2.64	2.54	7.04	0.86	25.41	4.01
	15	30	1.46	1.99	2.12	1.82	1.26	13.31	36.42	40.26	30.7
	20	35	1.92	2.09	2.19	2.11	1.83	4.38	12.18	16.27	13.09
	25	40	2.79	2.99	2.96	3.06	2.7	3.09	9.58	8.91	11.88
AVG								6.4	15.99	28.15	15.43

improvement, respectively, in terms of *TPC* and *LWR* as two prominent objective functions. Except for the MOPSO model which delivers poor results, the proposed MOGA model returns more efficient solutions quicker than others in terms of real execution time. For future work, we envisage to present a QoS-aware economic model for dynamic deployment of IoT applications in fog environment.

Table 27 Performance comparison of different algorithms in terms of *TPC* STD value

Standard deviation (STD) value											
Scenarios	FN	Appmod	*TPC*					RPD (%)			
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*	*MODCS*	*MOGWO-I*	*MOGWO-II*	*MOPSO*
Category 1	10	20	0.63	0.88	0.53	0.68	0.39	37.12	55.35	25.39	42.26
	15	20	2.03	1.78	1.87	1.66	1.54	23.93	13.44	17.67	7.31
	20	20	1.58	1.39	1.27	2.33	1.03	34.7	25.75	18.67	55.76
	25	20	0.92	0.88	1.12	0.82	0.71	22.13	18.91	36.35	13.01
Category 2	15	20	0.85	0.81	0.9	0.82	0.73	13.53	10.07	18.63	10.42
	15	25	1.28	2.21	0.69	1.7	0.76	40.14	65.43	11.54	55.1
	15	30	1.68	1.26	2.34	1.35	1.24	25.93	1.14	46.76	7.63
	15	35	1.4	0.76	1.46	0.64	0.54	61.17	28.1	62.78	15.11
Category 3	10	25	2.17	3.02	1.43	2.54	1.26	41.85	58.23	11.68	50.27
	15	30	1.44	1.29	1.8	1.43	1.33	7.83	3.02	26.37	7.24
	20	35	1.95	1.71	1.12	1.72	1.04	46.73	39.5	7.62	39.68
	25	40	1.04	0.86	0.79	0.71	0.64	38.59	25.59	19.26	9.61
AVG								32.8	28.71	25.23	26.12

Table 28 Performance comparison of different algorithms in terms of *LWR* STD value

Standard deviation (STD) Value											
Scenarios	FN	Appmod	*LWR*					RPD (%)			
			MODCS	*MOGWO-I*	*MOGWO-II*	*MOPSO*	*MOGA*	*MODCS*	*MOGWO-I*	*MOGWO-II*	*MOPSO*
Category 1	10	20	0.1	0.1	0.08	0.14	0.08	20.61	26.15	4.19	46.5
	15	20	0.12	0.17	0.12	0.08	0.06	51.92	66.59	50.12	31.71
	20	20	0.12	0.11	0.18	0.11	0.10	19.17	11.8	46.21	12.24
	25	20	0.09	0.1	0.1	0.1	0.09	2.01	6.91	6.59	8.7
Category 2	15	20	0.11	0.09	0.15	0.08	0.08	25.92	9.26	45.51	3.36
	15	25	0.01	0.01	0.07	0.01	0.05	50	66.67	94.04	66.67
	15	30	0.07	0.14	0.07	0.09	0.05	28.29	63.66	24.52	41.14
	15	35	0.04	0.07	0.06	0.05	0.04	8.25	36.9	34.53	17.37
Category 3	10	25	0.15	0.16	0.17	0.24	0.15	3.21	6.15	15.33	39.54
	15	30	0.16	0.15	0.26	0.17	0.14	13.81	9.67	48.05	21.9
	20	35	0.11	0.11	0.15	0.13	0.10	7.52	12.29	33.47	27.8
	25	40	0.12	0.08	0.09	0.07	0.06	48.76	18.24	34.03	13.6
AVG								23.29	27.86	36.38	27.54

Funding There is no funding.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest There is no conflict of interest.

References

1. Azimi S, Pahl C, Shirvani MH (2020) Particle swarm optimization for performance management in multi-cluster IoT edge architectures. In: International cloud computing conference (CLOSER), pp 328–337. https://doi.org/10.5220/0009391203280337.
2. Bonomi F, Milito R, Natarajan P, Zhu J (2014) Fog computing: "a platform for internet of things and analytics". In: Big data and internet of things: a roadmap for smart environments. Springer, Berlin, pp 169–186. https://doi.org/10.1007/978-3-319-05029-4_7
3. Andriopoulou F, Dagiuklas T, Orphanoudakis T (2017) Integrating IoT and fog computing for healthcare service delivery. In: Components and services for IoT platforms. Springer, Berlin, pp 213–232. https://doi.org/10.1007/978-3-319-42304-3_11
4. Shi Y, Ding G, Wang H, Roman HE (2015) The fog computing service for healthcare. In: International Symposium on future information and communication technologies for ubiquitous healthcare, pp 70–74. https://doi.org/10.1109/Ubi-HealthTech.2015.7203325
5. An OpenFog Architecture Overview, OpenFog (2017) https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf. Accessed 2017
6. Farzai S, Hosseini Shirvani M, Rabbani M (2020) Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. Sustain Comput Inform Syst 28:100374. https://doi.org/10.1016/j.suscom.2020.100374
7. Hosseini Shirvani M (2020) A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. Eng Appl Artif Intell 90:103501. https://doi.org/10.1016/j.engappai.2020.103501
8. Taneja M, Davy A (2017) Resource-aware placement of IoT application modules in fog-cloud computing paradigm. In: Proc. of the IFIP/IEEE symposium on integrated network and service management, IM'15. IEEE, pp 1222–1228. https://doi.org/10.23919/INM.2017.7987464
9. Venticinque S, Amato A (2018) A methodology for deployment of IoT application in fog. J Ambient Intell Humaniz Comput 1–22, https://doi.org/10.1007/s12652-018-0785-4
10. Hong HJ, Tsai PH, Hsu CH (2016) Dynamic module deployment in a fog computing platform. In: 18th Asia-Pacific network operations and management symposium (APNOMS), pp 1–6. https://doi.org/10.1109/APNOMS.2016.7737202
11. Ramzanpoor Y, Hosseini Shirvani M, Golsorkhtabaramiri M (2022) Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure. Complex Intell Syst 8:361–392. https://doi.org/10.1007/s40747-021-00368-z
12. Pallewatta S, Kostakos V, Buyya R (2022) QoS-aware placement of microservices-based IoT applications in Fog computing environments. Futur Gener Comput Syst 131:121–136. https://doi.org/10.1016/j.future.2022.01.012
13. Chen L et al (2021) IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning. IEEE Internet Things J 8(16):12610–12622. https://doi.org/10.1109/JIOT.2020.3014970
14. Mahmud R, Ramamohanarao K, Buyya R (2018) Latency-aware application module management for fog computing environments. ACM Trans Internet Technol 1–21. https://doi.org/10.1145/3186592
15. Brogi A, Forti A (2017) QoS-aware deployment of IoT applications through the fog. iEEE Internet Things J 4:1185–1192. https://doi.org/10.1109/JIOT.2017.2701408
16. Yangui S, Ravindran P, Bibani O, Glitho RH, Hadj-Alouane NB, Morrow MJ, Polakos PA (2016) A platform as-a-service for hybrid cloud/fog environments. In: 2016 IEEE international symposium on local and metropolitan area networks (LANMAN), pp 1–7.https://doi.org/10.1109/LANMAN.2016.7548853
17. Ahmadighohandizi F, Systä K (2016) Application development and deployment for IoT devices. In: Proc. 4th Int'l workshop cloud for IoT (CL IoT 16). https://doi.org/10.1007/978-3-319-72125-5_6
18. Chen BL, Huang SC, Luo YC, Chung YC, Chou J (2017) A dynamic module deployment framework for M2M platforms. In: IEEE 7th international symposium on cloud and service computing (SC2). IEEE, pp 194–200. https://doi.org/10.1109/SC2.2017.37
19. Li F, Vögler M, Claeßens M, Dustdar S (2013) Towards automated iot application deployment by a cloud-based approach. In: 6th international conference on service-oriented computing and applications. IEEE, pp 61–68. https://doi.org/10.1109/SOCA.2013.12
20. Saurez E, Hong K, Lillethun D, Ramachandran U, Ottenwalder B (2016) Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In: DEBS 2016, pp 258–269. https://doi.org/10.1145/2933267.2933317
21. Vögler M, Schleicher JM, Inzinger C, Dustdar S (2015) DIANE—dynamic IoT application deployment. In: IEEE International conference on mobile services, pp 298–305. https://doi.org/10.1109/MobServ.2015.49
22. Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Comput Netw 47(4):445–487. https://doi.org/10.1016/j.comnet.2004.12.001
23. Blaglazov A, Buyya R (2011) Optimal online deterministic and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machine in cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420. https://doi.org/10.1002/cpe.1867
24. Arcangeli JP, Boujbel R, Leriche S (2015) Automatic deployment of distributed software systems: definitions and state of the art. J Syst Softw 3:198–218. https://doi.org/10.1016/j.jss.2015.01.040
25. Luo J, Song W, Yin L (2018) Reliable virtual machine placement based on multi-objective optimization with traffic-aware algorithm in industrial cloud. IEEE Access 6:23043–23052. https://doi.org/10.1109/ACCESS.2018.2816983
26. Hosseini Shirvani M, Gorji AB (2020) Optimization of automatic web services composition using genetic algorithm. Int J Cloud Comput 9(4):397–411. https://doi.org/10.1109/IC4.2015.7375538
27. Li H, Zhu G, Zhao Y, Dai Y, Tian W (2017) Energy-efficient and QoS-aware model based resource consolidation in cloud data centers. Clust Comput 20:2793–2803. https://doi.org/10.1007/s10586-017-0893-5
28. Saeedi P, Hosseini Shirvani M (2021) An improved thermodynamic simulated annealing-based approach for resource-skewness-aware and power efficient virtual machine consolidation in cloud data centers. Soft Comput 25:5233–5260. https://doi.org/10.1007/s00500-020-05523-1
29. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi objective genetic algorithm: Nsga-II. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017
30. Hosseini Shirvani M (2021) Bi-objective web service composition problem in multi-cloud environment: a bi-objective time-varying particle swarm optimisation algorithm. J Exp Theor Artif Intell 33(2):179–202. https://doi.org/10.1080/0952813X.2020.1725652
31. Hosseini Shirvani M (2018) Web service composition in multi-cloud environment: a bi-objective genetic optimization algorithm. In: 2018 IEEE (SMC) International conference on innovations in

intelligent systems and applications (INISTA). https://doi.org/10.1109/INISTA.2018.8466267
32. Hosseini Shirvani M, Rahmani AM, Sahafi A (2018) An iterative mathematical decision model for cloud migration: a cost and security risk approach. Softw Pract Exp 48(3):449–485. https://doi.org/10.1002/spe.2528
33. Mirjalili S, Saremi S, Mirjalili SM, Coelho LDS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. J Expert Syst Appl. https://doi.org/10.1016/j.eswa.2015.10.039
34. Yang XS, Deb S (2013) Multiobjective cuckoo search for design optimization. J Comput Oper Res. https://doi.org/10.1016/j.cor.2011.09.026
35. Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 congress on evolutionary computation (CEC'02). IEEE Publications, USA. https://doi.org/10.1109/CEC.2002.1004388
36. Asghari Alaie Y, Hosseini Shirvani M, Rahmani AM (2023) A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. J Supercomput 79:1451–1503. https://doi.org/10.1007/s11227-022-04703-0