Int. J. Nonlinear Anal. Appl. 17 (2026) 7, 157–170
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.31875.4729

Bi-objective IoT applications deployment in fog environment using parallel ACO

Fatemeh Saadian[a], Homayun Motameni[b,*], Mehdi Golsorkhtabaramiri[a]

[a]Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran
[b]Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

(Communicated by Seyed Hossein Siadati)

Abstract

Nowadays, the Internet of Things (IoT) is inevitable in human daily life. Many IoT-based applications are executed on interoperable devices via the IoT-Fog-Cloud continuum to satisfy users' requirements. Such applications are generally time-sensitive, so they must be executed in real-time. The time constraint satisfaction strongly depends on the strategy of application placement on processing nodes. In this paper, a multi-objective optimization strategy for deploying microservices of real-time IoT applications is presented, which is considered among the challenging problems in fog and edge environments. A bi-objective optimization model for three-level deadline-aware application services is proposed, which is solved by using a parallel version of the Ant Colony Optimization (ACO) metaheuristic. The simulation results using IFogSim2, the new release of the famous Fog simulator IFogSim, show the superiority of the proposed approach compared to counterpart algorithms in terms of total resource wastage, total network latency, the main objective function, and execution time.

Keywords: Fog computing, Internet of things, service placement problem, real-time applications
2020 MSC: 97Pxx, 68M11

1 Introduction

With the ever-increasing pervasiveness of the IoT paradigm in every aspect of human life, the number of devices equipped with the IoT is increasing at a tremendous speed. According to predictions, more than 75.44 billion IoT devices will be connected to the Internet by 2025, and this number can exceed 500 billion devices by 2030 [45, 2, 43]. Different types of geographically distributed devices, such as smartphones, smart cameras, vehicles, home appliances, etc. with the ability to connect to the Internet, can be connected and cooperated to process the applications needed by users. In this way, a huge amount of heterogeneous data called metadata is produced that can be processed [43, 1]. Due to resource limitations, energy limitations in battery-powered devices, instability and high mobility like wireless networks, dealing with such a volume of data relying on IoT devices is a big challenge.

One of the new technologies that emerged in response to new processing needs is cloud computing, and due to elastic virtual resources, flexibility in payment pricing scheme, and scalability in computing model, appears useful at

*Corresponding author
Email addresses: saadyan@yahoo.com (Fatemeh Saadian), h_motameni@yahoo.com (Homayun Motameni), golesorkh@baboliau.ac.ir (Mehdi Golsorkhtabaramiri)

Received: August 2024 *Accepted:* October 2024

first sight [19, 26, 28, 29]. However, the shortcomings of cloud computing have been revealed in the face of a large number of IoT devices and a large volume of production data. Since most IoT applications are latency-sensitive (such as emergency response, healthcare systems, intelligent transportation systems, and interactive mobile games), some problems arise with processing in the cloud environment. First, massive data transmission over the network between IoT devices and centralized cloud data centers can increase communication latency, which acts as a bottleneck for network and application performance in the face of bandwidth limitations. Second, the processing of such time-sensitive applications through cloud data centers may lead to service time crises and thus increase the rate of service level agreement violations [32, 40, 23, 31]. To overcome the mentioned drawbacks of cloud computing, Cisco Systems in 2012 started fog computing as a decentralized middle computing layer between cloud and fog devices [17, 7, 8]. Fog computing places to network and storage resources on nodes (eg, smart gateways, cellular base stations, switches, routers, wireless access points, micro data centers, etc.) near the edge of the network and close to users' devices [7, 8, 21]. Therefore, IoT application services can be processed near end devices. In addition, the collection, analysis and filtering of the required data is done near the respective end device. Therefore, the goal of fog computing is: to reduce the power consumption of end devices; reduce traffic problems in the main cloud and fog networks; Minimizing response time and latency (due to traffic management and multi-hop physical distance management between the end user and the cloud data center); It also enhances privacy and security, while location and context-aware decision-making become more accurate and resource heterogeneity is properly managed [28, 5, 36, 25, 12].

Despite all the positive points mentioned, deploying applications in a fog environment requires some considerations. Typically, IoT applications are decomposed into many interdependent services that are placed in the fog environment to be processed in a distributed style, known as the fog service placement problem (FSPP) [14, 15, 42]. Communication between these interdependent services, especially for services with high communication rates, can easily saturate the limited bandwidth between host nodes [6, 16, 44]. This situation may fundamentally threaten the timing aspects of latency-sensitive IoT applications, which can lead to performance degradation for the entire application. At the same time, the availability of resources in the fog environment is not as good as that provided by the cloud [13]. Therefore, it is necessary to choose a suitable policy for providing resources while considering application deadlines. Therefore, an efficient approach is needed to optimize FSPP that: simultaneously optimizes resource utilization and network congestion. Since FSPP belongs to the NP-hard time complexity class [40, 9], it is unlikely to find an exact solution for large-sized real-world problems [34]. Therefore, the next concern is to make the problem solvable by using an efficient approximate solution while achieving the desired solution.

This paper proposes a two-partite trade-off between resource utilization and traffic management about the priority of applications deadlines [22] by using a Parallel Bi Objective ACO (PBOACO) algorithm with Master-slave structure which is solved in a reasonable time. Therefore, the contributions of this research are as follows:

- Proposing a prioritization policy based on the type of deadline for real-time IoT applications.
- Proposing an algorithm to calculate the delays caused by the communication between services related to applications, to manage traffic congestion.
- Utilizing a parallel version of ACO to solve the problem according to the optimization model.

The rest of the paper is arranged as follows. Section 2 has an overview of related works. Section 3 is dedicated to the proposed scheme, including the system framework, optimization model, and PBOACO solution. Evaluation and comparison are described in section 4; section 5 concludes the paper and proposes future directions.

2 Related works

The issue of FSPP has attracted a lot of effort as a hot research topic from different perspectives due to the significant impact on the performance of fog systems. In this chapter, some related studies in the literature are discussed considering their goals and proposed solutions to solve the problem.

In [28], an approach to jointly optimize load distribution and service deployment for IoT and edge environments was proposed, focusing on reducing SLA violations due to deadline, operational cost, and unavailability, which was solved using a multi-objective GA based on Biased Random Key Genetic Algorithm (BRKGA) and NSGA-II considering the solutions close to Pareto optimal frontier. Optimization of resources and communication issues were not included in this study. The authors in [40] proposed a genetic-based service deployment algorithm to reduce network utilization and application delay suitable for a cloud-fog environment. The simulation results in the iFogSim simulator showed a reduction in program delay, network utilization, energy consumption, and cost compared to GA-PSO and random

deployment approaches. The use of resources is not considered in this work a multi-objective two-tier resource provisioning framework was proposed in [31], which used Docker-container technology to deploy IoT applications in a fog environment. The proposed model was solved using the Elite Genetic Algorithm (EGA). However, bandwidth usage, traffic management and inter-node communications were not considered in that work. Research [14] compared the effectiveness of three evolutionary algorithms for optimizing the place of services in fog architectures: Weighted Sum Genetic Algorithm (WSGA), Non-Dominated Sorting Genetic Algorithm (NSGA-II) and Decomposition Based Multi-Objective Evolutionary Algorithm (MOEA/D). The paper focused on optimizing network latency, service deployment, and resource utilization. The results showed that the NSGA-II approach provides the best results compared to others. A lightweight decentralized service deployment approach was proposed in [15] qualified for fog architecture to optimize network utilization and reduce delay by managing the distance between interdependent services. The main idea of this work was that the best devices to host the service are those that are as close as possible to the clients' gateways. The proposed policy was evaluated using iFogSim against the Edge-ward approach and the results showed that the proposed approach is effective in terms of network utilization and service delay in most requested services. The authors in [42] proposed a fog conceptual framework and defined a formal model for the FSPP to optimize fog resources while considering the heterogeneity of applications and resources. The proposed architecture problem was solved using the greedy first-fit (FF) heuristic approach, the genetic meta-heuristic algorithm and an exact approach implemented in the IBM CPLEX library. The results showed that approaches based on genetic algorithms and exact methods do not violate the deadline of users' requests. The exact approach showed a better use of resources, while the genetic algorithm played a better role in reducing communication delays. The authors in [6] presented a precise and innovative delay-aware algorithm to minimize the delay between nodes and the overall delay in the deployment process of IoT application modules in the fog environment. The exact approach was defined as an integer linear programming (ILP) model that was solved using the CPLEX solver. The result of the exploratory approach was obtained through iFogSim. The evaluation results showed that both approaches have their own advantages and disadvantages. The exact approach produced a globally optimal solution with high execution time, while the heuristic algorithm led to a near-optimal solution, which in turn saves time. In [16], a simulation called iFogSim, which is an extension of its previous tool, CloudSim, was proposed for IoT systems. Among the most key capabilities of this simulator is the possibility of evaluating the impact of resource management on various criteria such as electricity consumption, network consumption and delay. In this work, in addition to introducing iFogSim, a virtual machine deployment policy called Edge-ward placement was proposed to be applied in a cloud-fog environment. The simulation results proved the high efficiency of the Edge-ward deployment compared to the cloud-only mode, in terms of energy, network density and delay reduction. A heuristic module mapping algorithm was proposed by [44] to reduce energy consumption, network utilization and application delay in deploying IoT applications in cloud-fog infrastructure. The simulation results obtained through iFogSim showed the superiority of the proposed approach in the fog-cloud topology in comparison with the traditional cloud infrastructure. Authors in [33] proposed a resource management technique by applying multi-objective VMs placement in a dynamic cloud environment, to reduce energy consumption and SLA violation, where resources balance. they solved multi-objective problems by applying heuristics and meta-heuristic algorithms. A priority, power, and traffic-aware IoT application deployment model in the form of deploying qualified virtual machines for cloud data centers was presented in [34], where applications were classified into two categories based on importance. For critical applications, minimizing traffic and power was the main goal. While for non-critical applications, the goal was to reduce the loss of resources and electricity consumption. Although, the proposed approach performed better than others in many scenarios, the proposed schemes similar to the proposed method by [33] were not designed for active fog environments. In [37], a VM-based task deployment method was proposed for IoT applications in the fog environment to reduce energy consumption through resource management. This model was solved using a combination of ACO and GA to achieve high exploration in the search space, while an acceptable convergence rate was obtained. Simulation experiments showed that the combined method has better results compared to ACO and GA in terms of power, time and cost reduction. A two-stage multi-criteria FSPP optimization approach was developed in [35] suitable for decentralized network edge microclouds. Computational experiments showed that the proposed method performs better than both the random location strategy and the basic bandwidth-aware method based on the optimization of bandwidth consumption and resource consumption. In [3], the authors developed a multi-objective service placement optimization model to reduce construction time, energy consumption and cost in fog environments. The proposed model was followed by usinga genetic algorithm to solve the problem. The simulation results using the Python-based YAFS tool showed that the proposed genetic algorithm performed better than random deployment in all the studied criteria. In [18], a network and energy-aware service deployment policy, called MinRE, was proposed for the fog-cloud environment, which aimed to optimize the deployment of IoT services in two ways based on application priority. In this work, the problem was formulated as a mixed integer linear programming model and then the model was solved using two heuristic algorithms separately for critical and normal services. The simulation results showed that the proposed approach performs better than the standard policies in most scenarios. Research [4] proposed a four-phase

autonomous IoT service deployment method called MADE to optimize service delay, fog usage, and the total cost of computing and communication. The evaluation results showed that MADE performed better than multi-objective particle swarm optimization and NSGAII in general comparison. The authors in [39] presented a conceptual framework used in fog-cloud control middleware, called MADE-k, to optimize the deployment of IoT service in fog. The objectives of the proposed approach were to reduce service latency and cost while maximizing the use of fog resources. MADE-k consisted of four steps similar to those proposed by [4], using a common knowledge base to solve the problem. The PSO algorithm was used for the decision-making process in the third step of the approach. The evaluation results indicated the higher efficiency of MADE-k compared to other counterparts. Another meta-heuristic algorithm, the Cuckoo Search Algorithm, was implemented in MADE-k in [24]. Evaluations showed that CSA showed better results compared to other leading algorithms, especially PSO implemented in [39] in terms of response time, service delay, non-violation of SLA, use of fog, cost and energy consumption. A distributed automatic service deployment strategy was developed in [20], which optimizes the energy consumption of fog nodes while reducing the communication overhead of applications. Next, the model was solved using the approximate method based on the Markov chain. The evaluation results against some centralized and distributed heuristics indicate that the proposed method works better than others in many scenarios. An application deployment strategy focusing on bandwidth utilization and delay reduction was proposed in [11], which classified IoT applications into critical and non-critical classes. The main policy was to deploy critical dependent modules in a fog node or at least in close proximity. The simulations performed with iFogSim showed that the proposed method performs better than the competing approaches in terms of delay and reduction of network usage. The authors [38] proposed an independent service placement optimization strategy called MAPE-k to increase efficiency in relation to execution costs, which was implemented using the Gray Wolf Optimization approach. The simulation results showed that the proposed approach has achieved an acceptable performance compared to other counterparts from the perspective of total scheduled time, average waiting time, number of successfully deployed services and number of failed services. The authors [10] proposed a simulation-based approach using the Monte Carlo method to select the appropriate deployment of multi-component applications on a fog-enabled IoT infrastructure. In addition, a multi-objective optimization algorithm was proposed to optimize the placement of the application according to the issue of energy, deployment cost and quality of service factors. However, maximizing the use of resources is not considered in this study.

According to our studies, the present research is the first in the fog environment that minimizes resource wastage and network traffic between nodes according to the priority class of real-time schedule deadline (soft/moderate/hard) using a master-slave parallel version of the ACO metaheuristic.

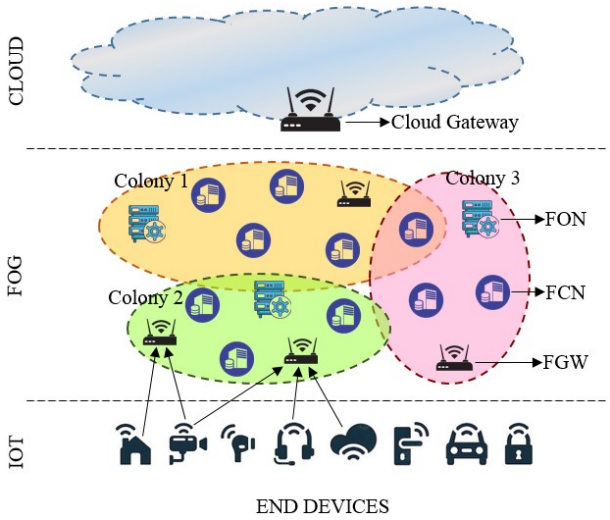

Figure 1: Proposed Fog computing framework.

3 Parallel ACO based placement policy

3.1 Fog landscape

A conceptual architecture of our fog computing framework is shown in Figure 1 with three distinct layers in a hierarchical structure: the cloud layer at the top; the fog layer in the middle; and the IoT layer at the bottom of

the architecture. The IoT layer, which consists of endpoint devices, generally requests a service, receives a response, generates data (eg, collected by sensors) to be stored, or consumes the data to act upon it. Each IoT device can be routed to the fog layer through a Fog GateWay (FGW) node using related protocols such as Zigbee, Bluetooth, WiFi, etc. [18]. In addition to FGWs as entry points, there are two types of nodes, generally network devices, in the fog layer: fog control nodes (FONs) and fog computing nodes (FCNs). FONs continuously monitor topology dynamics to provide resources and coordinate services to deploy services according to the chosen placement strategy. FCNs are independent nodes with different types and amounts of resources that are used to host services (in the form of containers or virtual machines) and data [31, 14].

Fog nodes form clusters called fog colonies. In other words, each fog colony is a micro data center that includes an arbitrary number of FCNs, FGWs and a FON that controls the operation of the colony nodes. We assume that the nodes of a colony form a complete network as in [4] and the connections between colonies in the fog layer form the structure of the graph. Each FON must be aware of its neighbour colonies' status. In the deployment process, the IoT device sends the service request to its FGWs. If the FGW has enough resources to handle the request, it processes the request and sends the response to the device. Otherwise, the FGW forwards the request to the corresponding FON to decide whether the service is hosted by the appropriate FCN. In our fog structure, similar to [18] and [30], all fog nodes can play the role of computing and packet forwarding. Figure 2 shows a scheme of services of an IoT application placed on fog nodes in a colony.

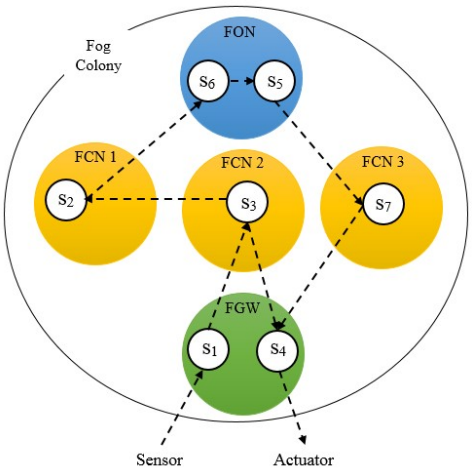

Figure 2: Service placement in a fog colony.

3.2 Applications prioritization

The real-time application must run within an expected time frame. The final moment of the time period is called the deadline. Obviously, the result must be produced before the deadline expires. Programs are classified into three categories in terms of deadline [22, 41]: *Hard*: In hard real-time applications, which are essentially safety-critical, all deadlines must be met. This means that even one case of missing the deadline will lead to severe consequences such as complete failure. It is clear that the utility of the result after the deadline is zero. Train signalling systems, flight control systems and nuclear power plant monitoring systems are included in this category. *Moderate*: In moderate real-time applications, a few rare missed deadlines are tolerable. In other words, a major disaster (e.g. failure) will not occur if a deadline is missed. However, the utility of the result after the deadline is zero. Also, QoS may decrease. Video conferencing, interactive online games and online image processing are examples of this class. *Soft*: In soft real-time applications, the result after the timeout is not useless, but its usefulness decreases and it also reduces QoS. Web browsing and ticket reservation systems are in this class.

For high-priority applications, it must be ensured that the deadline is never violated. Therefore, as a constraint, the application's round-trip time (RTT) must be less than or at most equal to the request deadline. For moderate-priority applications, several rare deadline violations are tolerable. Therefore, two limitations should be considered at this level: i) The number of non-compliance with the deadline should not exceed the maximum number of allowed violations. ii) In addition, the sequence of violations is not allowed. Therefore, if a violation occurs in this class, our approach temporarily increases the priority level of the program to "high" and when the first deadline is met, the priority level returns to its default. This strategy ensures that the chain of violations never occurs. For low-priority

classes, meeting the deadline is not a strict constraint. However, it is still considered an important optimization cornerstone in our model.

3.3 Service model

Let us define a number of applications (a) as set (3.1).

$$A = \{A_i^{P_i D_i} | i = 1, 2, \ldots, a\} \tag{3.1}$$

where $P_i \in P = \{1(High), 2(Moderate), 3(Low)\}$ and D_i indicate the priority level and the deadline associated with i-th application respectively. In FSPP, each application is decomposed into a number (s) of services, which is shown as set (3.2).

$$A_i = \{S_{ij} | i = 1, 2, \ldots, a \quad , \quad j = 1, 2, \ldots, s\}. \tag{3.2}$$

We have c colonies in our Fog environment, as set (3.3), each of which has f Fog nodes to accommodate services. So, we can define Fog nodes of k-th colony as set (3.4)

$$C = \{C^k | k = 1, 2, \ldots, c\} \tag{3.3}$$

$$C^k = \{F^{kw} | k = 1, 2, \ldots, c \quad , \quad w = 1, 2, \ldots, f\}. \tag{3.4}$$

All Fog nodes inside a colony include FGWs, FCNs, and the FON are mesh connected. The FON is directly connected to FONs of neighbor colonies. The connections may be wired or wireless. Each connection that connects two different Fog nodes has a bandwidth (BW) which determines the traffic volume passing over the link. We define BW matrix among Fog nodes of each colony C^k as equation (3.5). Note that we consider the FON of each colony as the head-node and the first node of that colony (node id = 1).

$$\begin{aligned} BW^k &= \left(bw^k_{ww'}\right)_{f \times f} \\ bw^k_{ww'} &= 0 \text{ iff } w = w' \\ bw^k_{ww'} &= bw^k_{w'w} \quad w, w' = 1, 2, \ldots, f \ , \quad \forall k \in (1, 2, \ldots, c) \end{aligned} \tag{3.5}$$

where all the elements on the main diagonal are equal to zero which means any Fog node has no communication link to itself. Each colony has a FON that is connected to the FONs of neighbor colonies. For simplicity, the BW of inter-FONs connections are considered to be same and is shown equation (3.6).

$$bw\_FON_{kk'}, \ \forall k, k' \in (1, 2, \ldots, c)\,. \tag{3.6}$$

In service deployment process, we use a decision variable x^{kw}_{ij} that is defined as equation (3.7).

$$x^{kw}_{ij} = \begin{cases} 1 & \text{if } S_{ij} \text{ placed on } F^{kw} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

After placing services, inter-dependent services belonging to application A_i establish some communications to reach the common goal. Let us define the traffic matrix among services of A_i as equation (3.8).

$$\begin{aligned} Traf^i &= \left(traf^i_{jj'}\right)_{s \times s} \\ traf^i_{jj'} &= 0 \text{ iff } j = j' \quad j, j' = 1, 2, \ldots, s \ , \quad \forall i \in (1, 2, \ldots, a)\,, \end{aligned} \tag{3.8}$$

where all the elements on main diagonal are equal to zero. It means that obviously any of services does not require to communicate with itself.

3.4 Resource wastage

In FSPP, resource provisioning is the first consideration. Applications' services have different resource requirements that should be provided. Service resource requirements such as CPU and RAM must not exceed the resources capacity of host node. So, for each Fog node $F^{k,w}$ we have 2 constraints denoted by equations (3.9) and (3.10) corresponding

to CPU and RAM capacities respectively. For simplicity of calculations, we assume there are sufficient capacities of other resources such as storage and memory bandwidth in host nodes.

$$\sum_{i=1}^{a}\sum_{j=1}^{s} S_{CPU\_req}^{ij} \times x_{ij}^{kw} \leq F_{CPU\_cap}^{kw} \quad \forall k \in (1,2,\ldots,c), \forall w \in (1,2,\ldots,f) \tag{3.9}$$

$$\sum_{i=1}^{a}\sum_{j=1}^{s} S_{RAM\_req}^{ij} \times x_{ij}^{kw} \leq F_{RAM\_cap}^{kw} \quad \forall k \in (1,2,\ldots,c), \forall w \in (1,2,\ldots,f), \tag{3.10}$$

where $S_{CPU\_req}^{ij}$, $S_{RAM\_req}^{ij}$, $F_{CPU\_cap}^{kw}$, and $F_{RAM\_cap}^{kw}$ are CPU request of S_{ij}, RAM request of S_{ij}, CPU capacity of F^{kw}, and RAM capacity of F^{kw} respectively. In application deployment process, enhancing resources utilization of Fog node is a desired goal which leads to sprawl reduction in nodes' resources, and consequently, it implicitly yields to power consumption reduction due to hibernating unused Fog nodes. To reach this end, services of different applications are tried to be placed compactly in less number of Fog nodes, so that resource usage to be increased and wasted amount of resources to be reduced. The total resource usage (TRU) in our Fog landscape, similar to [14], can be obtained according to equation (3.11).

$$TRU = \frac{\sum_{i=1}^{a}\sum_{k=1}^{c}\sum_{w=1}^{f}\sum_{j=1}^{s}\left[\left(S_{CPU\_req}^{ij} + S_{RAM\_req}^{ij}\right) \times x_{ij}^{kw}\right]}{\sum_{k=1}^{c}\sum_{w=1}^{f}\left[F_{CPU\_cap}^{kw} + F_{RAM\_cap}^{kw}\right]}. \tag{3.11}$$

So, the total resource wastage (TRW) amount could be calculated via equation (3.12), that it is tried to be minimized.

$$TRW = 1.0 - TRU \tag{3.12}$$

3.5 Communication latency

Latency is a challenge which can seriously threaten QoS for hard real time applications and services. Network latency is highly affected by some technical factors such as inter-nodes traffic congestion, nodes distance, network BW, packet propagation, serialization, queuing, and switching delays. This paper attempts to keep RTT in an acceptable range based on applications' priority level RTT is defined as summation of application's processing time and network latency. So, the equation (3.13) should be minimized.

$$RTT(A_i) = T_{proc}(A_i) + L(A_i) \quad \forall i \in (1,2,\ldots,a) \tag{3.13}$$

where, $T_{proc}(A_i)$ and $L(A_i)$ are the total processing time and the total data transmission time of services belonging to A_i , respectively. As denoted in [31], processing time of application A_i is calculated via equation (3.14) by summation processing time of all services S_{ij} belonging to A_i.

$$T_{proc}(A_i) = \sum_{k=1}^{c}\sum_{w=1}^{f}\sum_{j=1}^{s} \frac{S_{ij}^{size}}{F_{proc\_cap}^{kw}} \times x_{ij}^{kw} \quad \forall i \in (1,2,\ldots,a) \tag{3.14}$$

where,$F_{proc\_cap}^{kw}$ is processing capacity of Fog node F^{kw}. The other focus of this section is on latency, which is attempted to be reduced by placing inter-dependent services as close as possible, so that, traffic among these services to be transmitted by passing the shortest possible path over the network. In this regard, high-dependency services are tried to be placed on the same Fog node, in the next choice on two different Fog nodes in the same colony, and as the last choice, on two different Fog nodes in different colonies aim to reduce the number of hops between nodes. Each service from each application may be placed on FGW, FON, or on a suitable FCNa in a colony based on the decision made by the colony's FON. All Fog nodes inside a colony are fully mesh connected, and the FON of each colony is connected to the FONs of other colonies. So, if a service S_{ij} placed on Fog node F^{kw} transfers the traffic to a service S_{ij} placed on Fog node $F^{k'w'}$, the latency of traffic transmission is calculated using Algorithm 1.

Note that we assume the FON of each Fog colony is the first node (id = 1) of that colony. So, the communication latency of application A_i and the total network latency (TNL) can be obtained via equations (3.15) and (3.16) respectively.

$$L(A_i) = \sum_{k=1}^{c}\sum_{k'=1}^{c}\sum_{w=1}^{f}\sum_{w'=1}^{f}\sum_{j=1}^{s}\sum_{j'=1}^{s} x_{ij}^{kw} \times x_{ij'}^{k'w'} \times L_{jj'}^{i} \quad \forall i \in (1,2,\ldots,a) \tag{3.15}$$

$$TNL = \sum_{i=1}^{a} L(A_i) \tag{3.16}$$

Algorithm 1 :Inter-services communication latency

input: BW^k, $BW^{k'}$, $traf^i_{jj'}$, k, w, k', w'
output: communication latency (L) between S_{ij} and $S_{ij'}$
01. % ———— $k = k'$ ————
02. **if** $w = w'$ then $L^i_{jj'} = 0$
03. **if** $w \neq w'$ then $L^i_{jj'} = traf^i_{jj'}$ / $bw^k_{ww'}$
04. % ———— $k \neq k'$ ————
05. **if** $w = 1$ then
06. **if** $w' = 1$ then $L^i_{jj'} = \left(traf^i_{jj'} \text{ / } bw\_FON_{kk'}\right)$
07. **if** $w' \neq 1$ then $L^i_{jj'} = \left(traf^i_{jj'} \text{ / } bw\_FON_{kk'}\right) + \left(traf^i_{jj'} \text{ / } bw^{k'}_{1w'}\right)$
08. **if** $w \neq 1$ then
09. **if** $w' = 1$ then $L^i_{jj'} = \left(traf^i_{jj'} \text{ / } bw^k_{w1}\right) + \left(traf^i_{jj'} \text{ / } bw\_FON_{kk'}\right)$
10. **if** $w' \neq 1$ then $L^i_{jj'} = \left(traf^i_{jj'} \text{ / } bw^k_{w1}\right) + \left(traf^i_{jj'} \text{ / } bw\_FON_{kk'}\right) + \left(traf^i_{jj'} \text{ / } bw^{k'}_{1w'}\right)$
11. **return** $L^i_{jj'}$

3.6 Bi-objective model

With considering the set A of real time IoT applications each of which has associated specifications of deadline and priority level, we want to place applications' services on Fog colonies consisting Fog nodes, so that, the summation of total resource wastage and total network latency to be minimized as defined according to equation (3.17).

$$\min(TRW + TNL). \tag{3.17}$$

Constraints: Equations (3.9) and (3.10) that guarantee the resources requirements of services placed on a Fog node, would not exceed that node's resources capacities. Each service S_{ij} should be placed exactly on one Fog node, which is implied by equation (3.18).

$$\sum_{k=1}^{c} \sum_{w=1}^{f} x^{kw}_{ij} = 1 \ , \quad \forall S_{ij} \in A_i \ , \quad \forall A_i \in A. \tag{3.18}$$

In addition to constraints above, we should consider deadline constraint for each priority level of applications. For high priority applications, any deadline violation is not permitted. Hence, the constraint (3.19) must be taken into account, which guarantees that deadline never would be missed.

$$(3.13) \leq D_i \ , \quad \forall A_i \in A. \tag{3.19}$$

For moderate priority applications, constraint (3.19) should be satisfied for utility of application's response. However, a number of infrequent deadline violations are tolerable. Thus, the constraint (3.19) along with (3.20) should be considered for this category.

$$D_{miss} \leq D^{\max}_{miss}, \tag{3.20}$$

where, D_{miss} is the number of deadline violations happened currently, and D^{max}_{miss} is the maximum number of allowed violations. Finally, for low priority applications, the application's response is useful after deadline, but its usefulness degrades over a tolerable time interval tol^{max}. So, we define constraint (3.21) to satisfy this deadline criteria.

$$(3.13) \leq D_i + \Delta t \ , \quad \forall A_i \in A \ , 0 \leq \Delta t \leq tol^{\max}. \tag{3.21}$$

Note that, it is desired that the value of elapsed time Δt be as small as possible to keep the response usefulness. Obviously, if Δt passes the tol^{max}, the application's response will be useless. Then, we can represent our multi-objective

optimization model via equations (3.17), (3.9), (3.10), and (3.18)-(3.21) as represented in the following:

$$\min(TRW + TNL) \quad s.t:$$
$$\sum_{i=1}^{a}\sum_{j=1}^{s} S^{ij}_{CPU\_req} \times x^{kw}_{ij} \leq F^{kw}_{CPU\_cap}, \quad \sum_{i=1}^{a}\sum_{j=1}^{s} S^{ij}_{RAM\_req} \times x^{kw}_{ij} \leq F^{kw}_{RAM\_cap}$$
$$\sum_{k=1}^{c}\sum_{w=1}^{f} x^{kw}_{ij} = 1, \qquad RTT(A_i) \leq D_i$$
$$D_{miss} \leq D^{\max}_{miss}\ ,\ P_i = 2$$
$$RTT(A_i) \leq D_i + \Delta t\ ,\ P_i = 3 \quad \forall A_i \in A\ , 0 \leq \Delta t \leq tol^{\max}.$$

3.7 PBOACO meta-heuristic

The pseudo-code of the proposed PBOACO is depicted in algorithm 2. This algorithm works as follows: Our main ACO procedure employs three slave procedures in parallel, each of which is dedicated to the applications of a priority level. For each priority level, we consider some coefficients for elements of the objective function (eq. 3.17) to determine their importance in fitness calculation. For high priority, we consider 0.2 and 0.8, which give the most importance to reduce latency. Resource wastage is reduced the in second level. For moderate priority, we consider 0.3 and 0.7, which give importance to latency and resource, respectively. For low priority, we consider 0.4 and 0.6, which give more importance to resources in comparison with high priority levels. However, because of the real-time nature of all categories, latency reduction is the most crucial objective and consequently receives the greatest weight. Note that the summation of coefficients is equal to 1.0.

In an initialization phase of ACO, the parameters are initialized and all the pheromone trails are set to τ_0. In the next part, the procedure receives all requests and starts assigning services to nodes. A local pheromone update is performed once an artificial ant has built a movement. Similarly to the general implementation of ACO algorithms, PBOACO starts with a pheromone trails matrix and a heuristic information matrix.

Algorithm 2 :PBOACO meta-heuristic

input: list of applications, services, deadlines, priority levels, colonies, Fog nodes
output: service allocation [ ]
begin // —— *initializing by master* ——-
01. sort Fog nodes inside of each colony based on connections' BW descending; // *FON is fixed*
02. sort applications based on deadline ascending;
03. sort services of each application based on traffic-interdependency descending;
04. categorize applications based on their associated priority (P);
05. coeff_S = [coeff1, coeff2];
06. **for** P = 1 to 3 **do in parallel** // *slave 1 to slave 3*
07. sol_S ← random solution of applications with priority level P;
08. sol_S_size ← number of individuals in sol_S;
09. fitness_function = coeff_S(1).TRW + coeff_S(2).TNL
10. fit_S ← fitness_function (sol_S);
11. sol_S_sorted ← sort (sol_S, fit_S);
12. pheromone = update (pheromone);
13. service allocation_S [ ] ← the best individual of sol_S_sorted;
14. **for** iteration = 1 to maximum_iteration **do**
15. new_sol = next solutions (pheromone);
16. new_sol_fit = fitness (new_sol);
17. new_pheromone = update (pheromone);
18. sol_S = new_sol;
19. pheromone = new_pheromone;
20. fit_S ← fitness_function (sol_S);
21. sol_S_sorted ← sort (sol_S, fit_S);
22. service allocation_S [ ] ← the best individual of sol_S_sorted;
23. **endfor**
24. **endfor**
25. **for** P = 1 to 3 **do**
26. service allocation [ ] ← concatenate (service allocation [ ], service allocation_S [ ]);
27. **endfor**
28. **return** service allocation [ ]
end

The quality of an ACO implementation depends greatly on the definition of the meaning of the pheromone trail. Another important factor in an ACO application is the choice of a good heuristic, which will be used in combination with the pheromone information to build solutions. It guides the probabilistic solution construction of ants with problem-specific knowledge.

4 Evaluation results

We simulated the proposed framework by utilizing the iFogSim2 simulator [27] for 3 Fog colonies, each of which covers 10 Fog nodes. Fog nodes, including FONs, FGWs, and FCNs are considered from 5 types with their resource capacities, similar to Fog cells specifications described in [24]. Table 1 shows details of Fog nodes. Generally, FGWs and FONs are from high capacity types to handle services eliminating deploying them on other FCNs.

Table 1: Details of Fog nodes

FN type	CPU_cap (MIPS)	RAM_cap (MB)
FCN 1	100	256
FCN 2	200	512
FCN 3	300	1024
FCN 4	1400	2048
FCN 5	1600	4096

15 IoT applications are considered with random deadlines, and from three priority levels, i.e. high: hard real-time; moderate: firm real-time; and low: soft real-time. Each application consists of 5 to 10 services. For simplicity, we considered 5 applications per priority level. Types and specifications of services are shown in Table 2. As considered in [24], we consider 5 types of services, i.e. sense, actuate, and three processing services, with their resource demands.

Table 2: Details of services

Service type	CPU_req (MIPS)	RAM_req (MB)
Sense	50	30
Actuate	50	20
Process 1	200	10
Process 2	200	20
Process 3	100	30

We define 3 scenarios to evaluate the performance of the proposed PBOACO algorithm against other counterparts: random placement, first fit decreasing (FFD) heuristic, bi-objective ACO (BOACO), and bi-objective genetic algorithm (BOGA) approach. In the first scenario, we observe outcomes of 1 application per each priority level, a total of 3 applications, to be deployed on 1 Fog colony. In the second scenario, we compare outcomes of 3 applications per priority level, i.e. 9 applications with different priorities to be deployed on 2 Fog colonies. In the third scenario, we deploy all 15 applications on 3 Fog colonies.

4.1 Total resource wastage

Total resource wastage (TRW) is the first metric in our evaluation in which the smaller value indicates the better performance of the algorithm. TRW obtained from PBOACO along with other counterparts are shown in Figure 3. As illustrated in Figure 3, results obtained from meta-heuristics compete, in comparison with random and FFD approaches. Although TRW values obtained from meta-heuristics in the first scenario appear in a near range, in the second and third scenarios the proposed PBOACO outperformed the others with an increase in the number of services and Fog nodes.

4.2 Total network latency

TNL as an effective objective on QoS, influences the RTT of applications which are defined in 3 real-time levels. Obviously, real-time application performance heavily depends on adhering to the associated deadline. Network latency, which is defined as the time that data is in flight from the source node to the destination node, seriously affects the possibility of a deadline meeting. Figure 4 illustrates the performance of proposed and rival algorithms on TNL. As we see in Figure 4, the proposed algorithm shows superiority against other algorithms in this important metric.

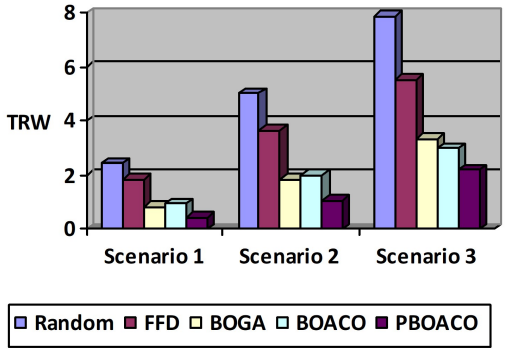

Figure 3: TRW evaluation.

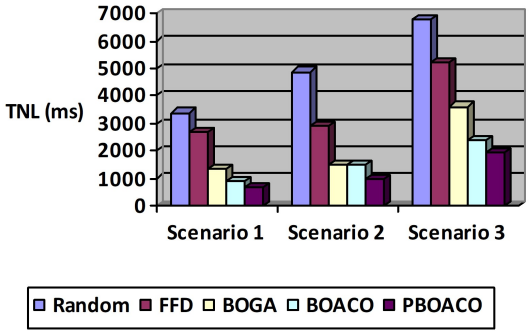

Figure 4: TNL evaluation.

4.3 Main objective function

The main objective of our model is defined as a weighted summation of TRW and TNL. As we considered a high weight for TNL in the objective function to ensure adhering to applications' deadlines, the value of TNL plays the most important role in calculating the value of the main objective function. Figure 5 illustrates the values of the main objective provided by algorithms. As shown, the value of the main objective of PBOACO is minimized in all scenarios.

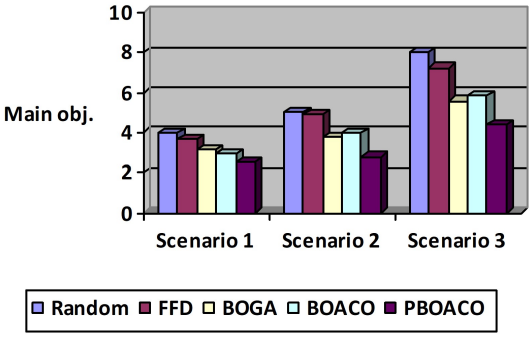

Figure 5: Main obj. fun. evaluation.

4.4 Execution time

Generally, a number of services and a number of Fog nodes increase the execution time of the placement algorithms. As indicated in Figure 6, the execution times of the random placement approach and pure FFD heuristic are significantly less than meta-heuristics. This time distance is at the expense of degrading the result's quality and fitness value. In such a situation, the execution time superiority is not the matter. Among meta-heuristics, the proposed algorithm shows superiority against BOACO, which is well-known for the property of fast convergence.

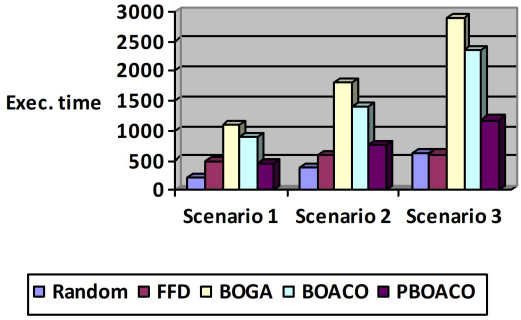

Figure 6: Execution time evaluation.

5 Conclusion

With the pervasiveness of IoT applications in humans' daily lives, the Fog computing paradigm appeared as a complement facility to the Cloud capabilities. The distance between IoT devices and powerful servers located in centralized Cloud data centers, was a great challenge for QoS parameters desired for users according to the service level agreement. This challenge became bolder for time-sensitive real-time applications. Hence, these applications were preferred to be processed in Fog nodes, closer to the user's device location. Although this close neighbourhood was a significant help in avoiding deadline violations, the limited resources of Fog nodes were an issue. In this regard, this paper proposed a deadline-aware bi-objective optimization model for FSPP to optimize resource wastage and network latency, that reducing the resource wastage implicitly yields power consumption reduction. Real-time applications were categorized into three priority levels: high, moderate, and low; corresponding to hard, firm, and soft real-time, respectively. Our main focus was on deploying interdependent services in the closest proximities as possible. The problem is solved using a parallel ACO algorithm which deploys services of all types of applications, by considering their associated deadline, in parallel. Experimental results indicate that the proposed PBOACO outperformed its counterparts: Random, FFD, BOGA, and BOACO strategies in terms of optimizing TRW, TNL, and reducing algorithm running time. As a future direction, we plan to investigate time-related prioritization properties in horizontal and vertical deployment of services in the IoT-Fog-Cloud hierarchy.

References

[1] E. Ahmed, I. Yaqoob, I.A. Targio Hashem, I. Khan, A. Ibrahim Abdalla Ahmed, M. Imran, and A.V. Vasilakos, *The role of big data analytics in Internet of Things*, Comput. Networks **129** (2017), 459–471.

[2] A.H. Alavi, P. Jiao, W.G. Buttlar, and N. Lajnef, *Internet of Things-enabled smart cities: State-of-the-art and future trends*, Measurement **129** (2018), 589–606.

[3] H.K. Apat, K. Bhaisare, B. Sahoo, and P. Maiti, *A nature-inspired-based multi-objective service placement in fog computing environment*, Intell. Syst.: Proc. ICMIB 2020, Springer, 2021, pp. 293–304.

[4] M. Ayoubi, M. Ramezanpour, and R. Khorsand, *An autonomous IoT service placement methodology in fog computing*, Software: Practice Exper. **51** (2021), no. 5, 1097–1120.

[5] B. Barzegar, H. Motameni, and A. Movaghar, *Eatsdcd: A green energy-aware scheduling algorithm for parallel task-based application using clustering, duplication and DVFS technique in cloud datacenters*, J. Intell. Fuzzy Syst. **36** (2019), no. 6, 5135–5152.

[6] A.R. Benamer, H. Teyeb, and N. Ben Hadj-Alouane, *Latency-aware placement heuristic in fog computing environment*, On the Move to Meaningful Internet Systems. OTM Conf.: Confederated Int. Conf.: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proc. Part II, Springer, 2018, pp. 241–257.

[7] K. Bilal, O. Khalid, A. Erbad, and S.U. Khan, *Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers*, Comput. Networks **130** (2018), 94–120.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, *Fog computing and its role in the internet of things*, Proc. First Edit. MCC Workshop on Mobile Cloud Comput., 2012, pp. 13–16.

[9] A. Brogi, S. Forti, C. Guerrero, and I. Lera, *How to place your apps in the fog: State of the art and open challenges*, Software: Practice Exper. **50** (2020), no. 5, 719–740.

[10] A. Brogi, S. Forti, and A. Ibrahim, *Optimising qos-assurance, resource usage and cost of fog application deployments*, Cloud Comput. Serv. Sci.: 8th Int. Conf., CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Revised Selected Papers 8, Springer, 2019, pp. 168–189.

[11] M. Dadashi Gavaber and A. Rajabzadeh, *Badep: Bandwidth and delay efficient application placement in fog-based IoT systems*, Trans. Emerg. Telecommun. Technol. **32** (2021), no. 8, e4136.

[12] S. Fatehi, H. Motameni, B. Barzegar, and M. Golsorkhtabaramiri, *Energy aware multi objective algorithm for task scheduling on DVFS-enabled cloud datacenters using fuzzy NSGA-II*, Int. J. Nonlinear Anal. Appl. **12** (2021), no. 2, 2303–2331.

[13] R. Fayos-Jordan, S. Felici-Castell, J. Segura-Garcia, J. Lopez-Ballester, and M. Cobos, *Performance comparison of container orchestration platforms with low cost devices in the fog, assisting Internet of Things applications*, J. Network Comput. Appl. **169** (2020), 102788.

[14] C. Guerrero, I. Lera, and C. Juiz, *Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures*, Future Gen. Comput. Syst.**97** (2019), 131–144.

[15] C. Guerrero, I. Lera, and C. Juiz, *A lightweight decentralized service placement policy for performance optimization in fog computing*, J. Ambient Intell. Human. Comput. **10** (2019), 2435–2452.

[16] H. Gupta, A.V. Dastjerdi, S.K. Ghosh, and R. Buyya, *iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments*, Software: Practice Exper.**47** (2017), no. 9, 1275–1296.

[17] M. Haghi Kashani, A.M. Rahmani, and N. Jafari Navimipour, *Quality of service-aware approaches in fog computing*, Int. J. Commun. Syst. **33** (2020), no. 8, e4340.

[18] H.O. Hassan, S. Azizi, and M. Shojafar, *Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments*, IET Commun. **14** (2020), no. 13, 2117–2129.

[19] A. Hedhli and H. Mezni, *A survey of service placement in cloud environments*, J. Grid Comput. **19** (2021), no. 3, 23.

[20] P. Kayal and J. Liebeherr, *Autonomic service placement in fog computing*, IEEE 20th Int. Sympos. A World of Wireless, Mobile Multimedia Networks, IEEE, 2019, pp. 1–9.

[21] W.Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, *Edge computing: A survey*, Future Gen. Comput. Syst. **97** (2019), 219–235.

[22] H. Kopetz and W. Steiner, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Springer Nature, 2022.

[23] M. Laroui, B. Nour, H. Moungla, M.A. Cherif, H. Afifi, and M. Guizani, *Edge and fog computing for IoT: A survey on current research activities and future directions*, Comput. Commun. **180** (2021), 210–231.

[24] C. Liu, J. Wang, L. Zhou, and A. Rezaeipanah, *Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm*, Neural Process. Lett.**54** (2022), no. 3, 1823–1854.

[25] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, *Mobile edge cloud system: Architectures, challenges, and approaches*, IEEE Syst. J. **12** (2017), no. 3, 2495–2508.

[26] R. Mahmud, R. Kotagiri, and R. Buyya, *Fog computing: A taxonomy, survey and future directions*, Internet of everything: algorithms, methodologies, technologies and perspectives, Springer, 2018, pp. 103–130.

[27] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, *IFogSim2: An extended IFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments*, J. Syst. Software **190** (2022), 111351.

[28] A.M. Maia, Y. Ghamri-Doudane, D. Vieira, and M.F. de Castro, *An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in Edge computing*, Comput. Networks **194** (2021), 108146.

[29] J. Masoudi, B. Barzegar, and H. Motameni, *Energy-aware virtual machine allocation in DVFS-enabled cloud data centers*, IEEE Access **10** (2021), 3617–3630.

[30] S. Misra and N. Saha, *Detour: Dynamic task offloading in software-defined fog for IoT applications*, IEEE J. Selected Areas Commun. **37** (2019), no. 5, 1159–1166.

[31] B.V. Natesha and R.M.R. Guddeti, *Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment*, J. Network Comput. Appl. **178** (2021), 102972.

[32] Z.M. Nayeri, T. Ghafarian, and B. Javadi, *Application placement in fog computing with AI approach: Taxonomy and a state of the art survey*, J. Network Comput. Appl. **185** (2021), 103078.

[33] B. Nikzad, B. Barzegar, and H. Motameni, *Sla-aware and energy-efficient virtual machine placement and consolidation in heterogeneous DVFS enabled cloud datacenter*, IEEE Access **10** (2022), 81787–81804.

[34] Sh. Omer, S. Azizi, M. Shojafar, and R. Tafazolli, *A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers*, J. Syst. Architect. **115** (2021), 101996.

[35] J. Panadero, M. Selimi, L. Calvet, J.M. Marquès, and F. Freitag, *A two-stage multi-criteria optimization method for service placement in decentralized Edge micro-clouds*, Future Gen. Comput. Syst. **121** (2021), 90–105.

[36] Zh. Peng, B. Barzegar, M. Yarahmadi, H. Motameni, and P. Pirouzmand, *Energy-aware scheduling of workflow using a heuristic method on green cloud*, Sci. Program. **2020** (2020), no. 1, 8898059.

[37] X. Ren, Zh. Zhang, and S.M. Arefzadeh, *An energy-aware approach for resource managing in the fog-based internet of things using a hybrid algorithm*, Int. J. Commun. Syst. **34** (2021), no. 1, e4652.

[38] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, *Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment*, Software: Practice Exper. **51** (2021), no. 8, 1745–1772.

[39] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, *An evolutionary multi-objective optimization technique to deploy the IoT services in fog-enabled networks: An autonomous approach*, Appl. Artific. Intell. **36** (2022), no. 1, 2008149.

[40] N. Sarrafzade, R. Entezari-Maleki, and L. Sousa, *A genetic-based approach for service placement in fog computing*, J. Supercomput. **78** (2022), no. 8, 10854–10875.

[41] A.K. Shukla, R. Sharma, and P.K. Muhuri, *A review of the scopes and challenges of the modern real-time operating systems*, Int. J. Embedded Real-Time Commun. Syst. **9** (2018), no. 1, 66–82.

[42] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and Ph. Leitner, *Optimized IoT service placement in the fog*, Serv. Orien. Comput. Appl. **11** (2017), no. 4, 427–443.

[43] Statista, *Internet of Things-number of connected devices worldwide 2015-2025*, `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`, 2025.

[44] M. Taneja and A. Davy, *Resource aware placement of IoT application modules in fog-cloud computing paradigm*, IFIP/IEEE Symp. Integ. Network Service Manag. (IM), IEEE, 2017, pp. 1222–1228.

[45] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J.P. Jue, *All one needs to know about fog computing and related edge computing paradigms: A complete survey*, J. Syst. Archit. **98** (2019), 289–330.