IJICTR
International Journal of Information & Communication Technology Research

Volume 15- Number 3 – 2023 (31 -42)

An Efficient Application Deployment in Fog

Masomeh Azimzadeh
Department of Computer Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
masomeh.azimzadeh@srbiau.ac.ir

Ali Rezaee*
Department of Computer Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
alirezaee@srbiau.ac.ir

Somayyeh Jafarali Jassbi
Department of Computer Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
s.jassbi@srbiau.ac.ir

Mehdi Esnaashari
Faculty of Computer Engineering
K. N. Toosi University of Technology University
Tehran, Iran
esnaashari@kntu.ac.ir

Received: 15 May 2023 – Revised: 2 August 2023 - Accepted: 28 August 2023

Abstract—**Fog computing emerged to meet to the needs of modern IoT applications, such as low latency, high security, etc. To this end, it brings the network resources closer to the end user. The properties of fog computing, such as heterogeneity, distribution, and resource limitations, have challenged application deployment in this environment. Smart service placement means deploying services of the IoT applications on fog nodes in a way that their service quality requirements are met and fog resources are used effectively. This paper proposes an efficient application deployment method in fog computing using communities. In contrast to previous research, the proposed method uses more factors than topological features to distribute network capacity more evenly between communities. This results in efficient use of network resources and better fulfillment of application requirements. In addition, according to our argument, using multiple criteria to prioritize applications will lead to better deployment and more effective use of resources. For this purpose, we use the number of application requests besides the deadline factor for application prioritization. Extensive simulation results showed that the proposed method significantly outperforms the state-of-the-art methods in terms of meeting deadlines, decreasing delays, increasing resource utilization, and availability by about 17, 33, 7, and 11 percent, respectively.**

Keywords: Fog Computing, Application Deployment, Evolutionary Computation.

Article type: Research Article

Publisher: ICT Research Institute

* Corresponding Author

[DOI: 10.61186/itrc.15.3.31]

I. INTRODUCTION

To meet the needs of the end user fog computing emerged at the edge of the network. The main purpose of this technology is to support the specific needs of delay-sensitive applications, such as augmented reality and IoT applications, which generate large amounts of data. Service management is a challenging task of fog computing because of the characteristics of the fog environment, such as resource limitations, heterogeneity, and dynamism of the environment on the one hand, and the complex and multi-component nature of the IoT applications on the other hand.

Application service deployment as an important field of service management means the proper placement of services on fog nodes. An efficient application deployment must meet the quality of service and use fog resources efficiently. One approach in this field to face the challenges of service placement is organizing the fog nodes as communities. Newman's theory considers the modularity of complex systems in such a way that detecting important structural patterns in a network causes a deeper understanding of a system. So, this theory has motivated many studies in complex networks, such as fog computing [3-6].

The research in the community detection field categorizes approaches from different perspectives. For example, community detection algorithms could be classified into three categories: node-based, group-based, and network-based. With the node-based method, the basis for creating communities can be based on characteristics such as reachability and membership grade. In the group-based methods, characteristics such as group density are the basis for creating communities, and in the network-based community identification methods, the entire network is divided into separate sets of nodes. Network connections are global in these algorithms. Newman's theory falls into the third category [7]. Another view of community recognition approaches is the creation of separate or overlapping communities [8-9]. For example, Xie et al. introduce an algorithm to determine relationships between individuals, groups, and observable interactions. They use this relationship to create overlapping communities [9].

The service placement methods proposed in the fog computing field consider node density and topology as the primary criteria for community building [10-14]. Just paying attention to the topology leads to the forming of unbalanced communities. Unbalanced communities introduce complexity and delay in deciding how to place them. An application's services distribution in different communities leads to increasing communication delays and decreasing availability. Considering network capacity from different aspects lead to the acquisition of more knowledge about the environment and the creation of high-quality communities.

This paper proposes a method called Community based Fog Service Placement (CFSP). The focus of this method is the balanced distribution of the network capacities between communities based on the genetic algorithm. For this purpose, the network capacity is considered from three aspects: 1) the amount of fog node resources, including the amount of Random Access Memory (RAM), Central Processing Unit (CPU), and storage space, 2) the number of fog nodes, and 3) the topology and network connections.

Creating balanced communities leads to speeding up decision-making regarding the placement of multi-component applications related to multiple requests in the environment. Also, by placing each multi-component application in a single community, the delay between the services of an application is reduced and their availability is increased. The policy of the CFSP method in the deployment step is placing of service in the nearest community to the end user. In this way, the delay between the end user and the requested application is reduced and leads to a better supply of deadline and response time.

To improve the CFSP method, some enhancements are made in both phases of creating deployment infrastructure and application service deployment. To improve the phase of creating deployment infrastructure, a version called CFSP.v1 is being developed. In this version, more connection and adjacency criteria are used to create communities. This leads to faster convergence of the genetic method towards the high-value communities with higher connectivity. Also, this approach causes the creation of overlapping communities. So, they can share some services or resources, which help to use resources more efficiently. We must mention that in the CFSP method the communities were separate and did not overlap.

In the application service deployment phase, application prioritization is improved. In this version of the CFSP method, called CFSP.v2, the number of application requests is taken into account in addition to the application deadlines. This policy prioritizes applications with shorter deadlines and more requests.

We should note that different versions of the method have an evolutionary nature. The CFSP.v1 version improves the community creation process of the base method, CFSP, and the CFSP.v2 improves the application prioritization of the CFSP.v1. That is to say, the CFSP.v2 is the best method among the others.

The innovations of the proposed method include:

- Development of an efficient application deployment approach: In the CFSP method, communities are formed based on the genetic method, considering various parameters that affect a balanced distribution of network capacities. This approach results in the efficient use of resources and an increase in service quality. The CFSP method places all services of an application in the nearest community, so it improves the availability and response time of applications.
- Improvement of the CFSP method in the deployment infrastructure phase: Community formation is improved with the creation of more connected and overlapping communities in the CFSP.v1 version, resulting in better resource sharing and utilization.
- Extension of the CFSP method in the application service deployment phase: To prioritize applications for delivery, in addition to using the deadline as a prioritization factor in the CFSP.v2 version, the

[DOI: 10.61186/itrc.15.3.31]

number of application requests is also taken into account. This results in placing more applications and increasing their Quality of Service (QoS).

In this paper, the proposed CFSP method is described and evaluated. The rest of this paper is organized as follows: In Section II, a brief literature survey is presented. The proposed method is described in Section III. In Section IV, the proposed method is evaluated using extensive simulation studies. Finally, Section V concludes the paper.

II. Literature Review

The proposed methods in the field of fog service placement fall into two categories. In the first category, only some optimizations are made for the deployment of the applications on fog nodes, but the node distribution in the network or the network topology is ignored [17-20]. The second category organizes the placement infrastructure by grouping nodes in the first phase, and deploying applications on top of them in the second phase [10-14][21-25]. The grouping of nodes is based on criteria such as distance from a central node or placement in a service domain, etc. [10-14] [22, 24]. In community-based approaches, nodes are clustered based on connection density and organized as communities [21, 24, 25].

A. Policy based application service deployment

Abbasi et al. [17] develop a genetic algorithm for workload allocation in a fog-cloud. They try to improve energy consumption and reduce delays. Reddy et al. [18] by intelligent sleep and wake-up cycles of the fog nodes, follow the energy minimization at the fog layer. So, they respond to requests with a minimal number of active fog nodes. Natasha and Guddeti [19] present a multi-objective optimization solution for service placement in the fog environment. Their objectives were to minimize service delay, cost, and energy consumption. Al-Tarawneh [20] uses a genetic algorithm to place application modules on fog devices. To do this, they consider the criticality levels of the applications and the security requirements. Vijouyeh et al. [27] formulate the placement and routing problem using an Integer Linear Programming (ILP) approach to deploy applications in the infrastructure network and direct traffic from end devices to deployed applications. The primary goals were to meet different user requirements and to maximize the profit of the infrastructure provider. Sriraghavendra et al. [28] propose a method to use fog resources while meeting time constraint of applications. They use a genetic algorithm for service placement in the fog environment. This research analyzes the response time of service placement in different layers and decides about the service placement of IoT applications in different layers of the fog-cloud architecture.

B. Group based application service deployment

Yousefpour and Ishigaki [13] group fog nodes based on the operational domain or specific needs of applications. They propose a delay-minimizing collaboration and offloading policy for fog capable devices that aims to reduce the service delay for IoT applications. Skarlat et al. [10] cluster the nodes as fog colonies for service placement. They formalize an optimization problem to provide delay-sensitive utilization of available fog-based computational resources. Kimovski et al. [22] use the graph theory for service placement. They define a fog architecture, to increase the speed of decision-making and provide adaptive resource management. Lera et al. [24] model the environment through centrality indices to determine the fog devices that are close to the sensors to enhance resource usage criterion. Skarlat et al. [12, 26] use fog colonies to allocate fog resources to IoT services. They consider colonies as micro data centers made up of an arbitrary number of fog cells. Baranwal and Vidyarthi [29] propose a service placement method based on the selection of some fog orchestrator nodes. They use distributed fog orchestrator nodes to allocate computing resources fairly among the fog nodes to improve the quality of service (QoS). Elkhatib et al. [11] also apply micro-cloud computing capabilities to deliver fog services to reduce latency.

C. Community based application service deployment

Filiposka et al. [25] propose the first method for using the community for resource management. They use hop count between virtual machines to reduce the distance between them. The aim is to improve communication efficiency and reduce power consumption. Lera et al. [23] apply the features of the complex network to organize the communities of fog nodes for service placement. The betweenness centrality measure is used to create a set of well-connected devices to improve service availability. Velasquez et al. [21] create groups of nodes based on the possibility of sharing gateway load between the nodes. To do this, they rank the nodes to form communities that contain nodes with the highest transition probability.

III. Proposed Method

This section introduces the proposed method. For this purpose, the environment definition is presented in Subsection A. This section describes the architecture and other components of the environment. After that, the proposed CFSP method and its enhanced versions are presented in Subsection B.

A. The environment definition

Fig. 1 shows the architecture of the proposed CFSP method in three layers: 1) IoT, 2) fog and 3) Cloud. The user requests come into the environment from the IoT layer. Communities are created in the Fog layer, which sits on top of the IoT layer to form the placement infrastructure to deliver the services. The resources in the cloud layer, which is above the fog layer, are used in situations where the fog resources do not meet user needs. Several gateways were considered for communication and information exchange between different layers.

We consider the fog environment as a graph G with the fog devices as nodes, denoted by the FN set, and the connections between the devices as edges, denoted by the EN set (Equation 1).

[DOI: 10.61186/itrc.15.3.31]

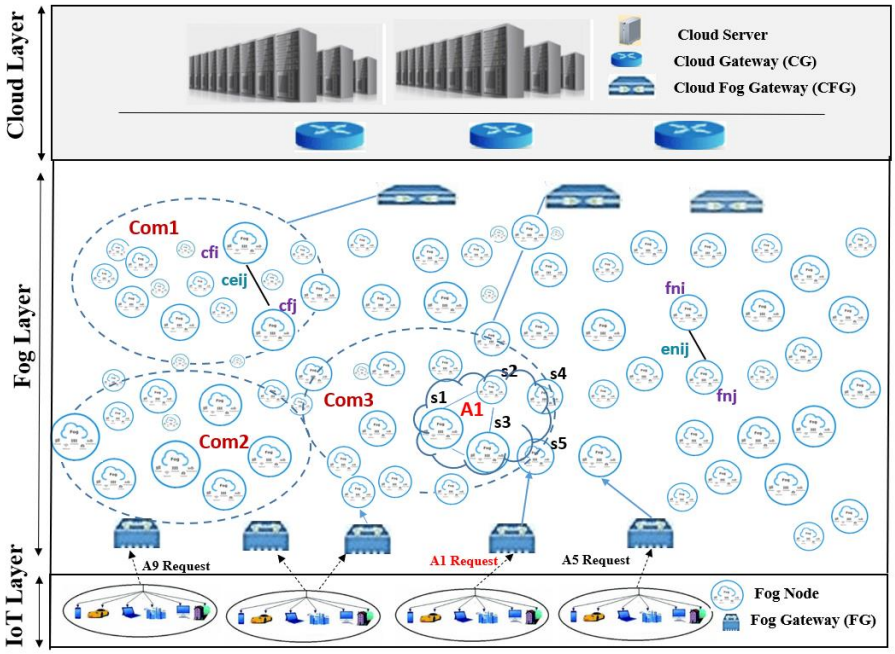

Figure 1. Architecture of the proposed CFSP method

$$G=(FN,EN) \quad (1)$$

According to Equation 2, FN contains all fog nodes (fn_i) that exist in a fog environment.

$$FN = \{fn_1, fn_2, \ldots, fn_m\} \quad (2)$$

The edges of the graph are also defined as EN in Equation 3. Each edge (en_{ij}) is a connection between two fog nodes fn_i and fn_j.

$$EN = \{ en_{ij} \mid i,j \epsilon [1,m] , i \neq j , en_{ij} = <fn_i, fn_j> \} \quad (3)$$

Communities are also considered subgraphs whose union forms the main graph. Equation 4 represents a set of k communities.

$$Com = \{Com_1, Com_2, \ldots, Com_k\} \quad (4)$$

According to Equation 5, each community is a subgraph of the main graph that includes a set of nodes (CF_i) and a set of edges (CE_i).

$$Com_i = \{(CF_i, CE_i) \mid CF_i \subset FN , CE_i \subset EN\} \quad (5)$$

Equation 6 represents the nodes of the CF_i community. This community contains cf_i nodes as a subset of fog nodes.

$$CF_i = \{cfi_1, cfi_2, \ldots, cfi_s\} \quad (6)$$

Also, Equation 7 shows the links between the nodes of each community.

$$CE_i = \{ ce_{ij} \mid i,j \epsilon [1,s] , i \neq j , ce_{ij} = <cf_i, cf_j> j\} \quad (7)$$

In the problem space, besides the fog graph and communities, a set of applications and their corresponding requests are defined. Each application may be requested by several users. Therefore, in Equation 8, the set of applications comprising p numbers of A_i is defined.

$$Applications = \{A_1, A_2, A_3, \ldots, A_p\} \quad (8)$$

Each A_i includes multiple services related to each other according to Equation 9.

$$A_i = \{s_1, s_2, s_3, \ldots, s_r\} \quad (9)$$

Equation 10 shows a request set of all applications, which is considered r requests. According to this equation, each request in the environment is associated with one application (A_x).

$$Requests = \{(R_i, A_x) \mid A_x \text{ is Requested by } R_i , i \epsilon [0, r]\} \quad (10)$$

B. Proposed CFSP method: Community based Fog Service Placement

In this section, the proposed CFSP method is described in terms of two phases: 1) creating deployment infrastructure and 2) application service deployment. Then, the enhanced versions of these two phases are presented as the CFSP.v1 version and the CFSP.v2 version.

1. Phase1: Creating deployment infrastructure

In the proposed method, the genetic algorithm is used to form communities. Within the framework of the CFSP method, each problem solving chromosome contains a set of communities. So, the aggregation of chromosomes means that the population is a set of solutions for creating communities. To initialize the chromosomes, each gene is randomly assigned a community identifier. After that, this initialized set of chromosomes enters the genetic cycle and the best ones are chosen as fathers. The father's chromosomes undergo crossover and mutational operators to produce new chromosomes as children. It should be noted that two-point crossover and one-point mutation operators are applied for this purpose. A set of the best chromosomes from fathers and children is then used to create the next population. The mentioned process continues until the best chromosome is reached, which represents the best set of created communities.

- Crossover operation

[DOI: 10.61186/itrc.15.3.31]

Two-point crossover operator is used as the first process for creating children. For this purpose, the population is divided into two equal parts, and in each stage, one chromosome is randomly selected from each part of the population. Then two intersection points are randomly selected. In this way, each chromosome can be divided into three sections. Finally, the contents of the chromosomes are shifted in the three selected sections.

- Mutation operation

Mutation operator is also applied as second operation for creating children. For this purpose, one point is randomly selected on each chromosome. Then the community identifier of this position is replaced by a randomly selected community identifier.

- Fitness function

As already mentioned, the phase of deployment infrastructure creation is about using network capacities efficiently. For this purpose, the network capacity parameters are defined in three categories:

1) Resource Allocation (RA): the average amount of resources allocated to each community
2) Node Distribution (ND): average number of nodes distributed among communities
3) Community Connectivity (CC): ratio of connected nodes of a community to the total number of community nodes.

To distribute the network capacity evenly among the communities, network capacity parameters are used in the fitness function, which we will describe below.

To calculate the average resources assigned to each community, the average RAM, CPU, and storage of each community are calculated respectively according to Equation 11, Equation 12, and Equation 13, respectively. The ratio of the resource of a resource type which was allocated to a community to the total resources of the same type was estimated by each of these equations.

$$\mathrm{Com_i}^{RAM} = \frac{\sum_{j=0}^{|\mathrm{CF_i}|} \mathrm{cf_{ij}}^{RAM}}{\sum_{m=0}^{|FN|} \mathrm{fn_m}^{RAM}} \quad (11)$$

$$\mathrm{Com_i}^{TB} = \frac{\sum_{j=0}^{|\mathrm{CF_i}|} \mathrm{cf_{ij}}^{TB}}{\sum_{m=0}^{|FN|} \mathrm{fn_m}^{TB}} \quad (12)$$

$$\mathrm{Com_i}^{IPT} = \frac{\sum_{j=0}^{|\mathrm{CF_i}|} \mathrm{cf_{ij}}^{IPT}}{\sum_{m=0}^{|FN|} \mathrm{fn_m}^{IPT}} \quad (13)$$

Finally, the average of resources used by each community is calculated according to Equation 14.

$$RA = \frac{\sum_{i=0}^{|Com|} (w^{RAM} * \mathrm{Com_i}^{RAM} + w^{TB} * \mathrm{Com_i}^{TB} + w^{IPT} * \mathrm{Com_i}^{IPT})}{|\mathrm{Com}|} \quad (14)$$

The node distribution is the second parameter for the balanced distribution of the network capacity. To estimate this parameter, we first calculate a balance criterion in Equation 15 by dividing the total number of nodes by the number of communities to get the balance criterion (α+). Then the node distribution is calculated by computing the sum of the ratio of the number of nodes in each community to the balance criterion and then dividing it by the number of communities (Equation 16).

$$\alpha^{+} = \frac{|FN|}{|Com|} \quad (15)$$

$$\mathrm{ND} = \frac{\sum_{i=0}^{|Com|} \frac{Com_k}{\alpha^{+}}}{|\mathrm{Com}|} \quad (16)$$

The largest connected component is found to check the connectivity of a community, then the membership rate of the assigned nodes of the community in the connected component is calculated. If a community does not contain a connected component, we ignore it as a community. Also, having more assigned nodes in the connected component increases the score of that community. Therefore, we calculate the ratio of the number of connected nodes in a community to the total number of those community nodes. Then the results of all communities are summed and divided by the number of communities (Equation 17).

$$CC = \frac{\sum_{i=0}^{|Com|} \frac{|connected-component(Com_k)|}{Com_k}}{|\mathrm{Com}|} \quad (17)$$

Finally, the fitness function is calculated according to Equation 18 as a weighted combination of three parameters: 1) the average of allocated resources, 2) the average number of assigned nodes, and the rate of community connectivity.

$$f(x) = \lambda_1 * RA + \lambda_2 * \mathrm{ND} + \lambda_3 * CC \quad (18)$$

2. Phase2: application service *deployment*

In this phase, applications are prioritized according to deadlines. Then, an application with the minimum deadline is selected and the nearest community to the user with sufficient resources is found for service placement. In Algorithm 1, the placement method is presented.

Algorithm 1: Application Placement
Input: genetic based communities, Applications, Requests
Output: Application placement results, List of ranked communities

```
1)  # Prioritize applications based on their deadline in ascending order
2)  PA ← Prioritize (applications, key = deadline, ascending)
3)  # Place each application request in a suitable community
4)  for appId in PA do:
5)      # Extract the list of Application's requests with appId key
6)      appReqs = Requests (appId)
7)      for Req_i in appReqs do:
8)          Placed =False
9)          # Calculate community distance for each user/ request
10)         ComRank = comm-distance (Req_i , communities,
12)                   key=neighbors )
13)         for Com_j in ComRank do:
14)             if appId in Com_j then:
15)                 Placed = True
16)                 Break;
17)             # Compare required resources with current resources
18)             if currentResource (Com_j) > = requiredResource (appId)
19)                 then:
20)                 placeApp (Com_j)
21)                 UpdateComResources (Com_j)
22)                 Placed = True
23)                 Break;
24) Return placeApp, ComRank
```

According to line 2, the applications are sorted in ascending order by the deadline. Then, from the list of sorted applications, each application is selected in turn (line 4) and the list of its requests is extracted (line 6). Next, according to lines 7 to 10, a list of the ranked

[DOI: 10.61186/itrc.15.3.31]

communities must be extracted for each request of the selected application. So, the neighborhood distance with different communities is calculated for each request and then a list of communities is ordered according to the neighborhood distance (line 10). Neighborhood distance means the number of hops from the requester to the community. In the next step, we first check for each candidate community whether a version of the application has already been made available there. If the version is available, the requestor is given access to it. Also, the placement process for this request ends. These steps are shown in lines 13 to 16. If the existing version is not available to the user, the amount of available capacity in the candidate community is compared to the resource requirements of the selected application. If there are enough resources, the application is placed in that community and according to lines 18 to 23 the community resources are updated. Otherwise, the next closest community in the list is selected and the previous steps are repeated. Lines 11 to 20 show these steps.

3. *Enhancement of Phase1*

In the problem definition, each chromosome is viewed as an array with the index of each element corresponding to the identifier of a fog node. In the CFSP method, a random community identifier is assigned to each chromosome gene, leading to a time-consuming convergence to a solution in the genetic cycle. To solve this problem, the CSFP.v1 version uses the connectivity information of the fog graph. To do this, the identifier of one of the corresponding neighbors of the fog node is assigned to each chromosome gene. This neighbor is randomly selected from the neighbors of this node. Algorithm 2 represents an improved initialization method. The input of this algorithm is the connection graph of fog nodes, and the output is the initialized chromosomes. In this algorithm, the entire population is initialized. To do this, the following steps are performed for each chromosome of the population.

Algorithm 2: Chromosome Initialization
Input: topology (G)
Output: Initialized chromosomes

```
1) # Assign a random neighbor identifier to genes in each chromosome
2)  for each chromosome in population do:
3)      for fn in G.Fn do:
4)          nodeId = identifier (fn)
5)          # Find a list of neighbors of the fn
6)           Neighbors = Find_neighbors (fn)
7)           #select a neighbor of fn from neighbor list
8)           rand_Id = Select_ random (Neighbors)
9)           # Fill the content of gene with random neighbor
10)          Chromosome[nodId] =  rand_Id
11)    # Find more frequent identifiers of and set then as headers
12)    Comm_heads = Frequent-nodes (Chromosome)
13)    # Form some clusters based on community header
14)     j=0
15)    for comhead in Comm_heads do:
16)          Community [j] = Find-neighbors (comhead)
17)          j = j + 1
18)          # check if all nodes are assigned to a community
19)     for fn in G.Fn do:
20)          if fn not in Community then:
21)               Near_comm = Find_nearest_community (Community)
22)               Add (fn, Near_comm)
23) Return initialized chromosomes
```

For each gene (identifier of the candidate nodes), its neighbors are searched according to line 6 and listed in the array named Neighbors. Then, according to line 8, the identifier of a neighbor is randomly selected from the Neighbors list. Then the content of the corresponding gene of the candidate node is filled with the identifier of the selected neighbor (line 10). These steps are repeated for all genes of the chromosome. After the initialization of the chromosome, the most frequent neighboring nodes in the chromosome are chosen as the community head. This is shown in lines 11 and 12. In the next step, according to lines 14 to 18, clusters of communities are formed around each of these central nodes. Finally, in this step, an attempt is made to map all existing nodes to some heads (lines 19 to 22).

The output of this algorithm is fed into the genetic cycle of the proposed CFSP method. Finally, after applying the relevant operators, a solution to the problem is extracted, including a set of balanced communities.

4. *Enhancement of Phase2*

After identifying a chromosome as a solution that represents a set of communities, it's time to place each application on the communities. The placement process is improved in the CFSP.v2 version by considering the number of requests criterion as an additional parameter for prioritizing applications.

Algorithm 3 shows how application prioritization is improved in the CFSP.v2 method. According to lines 2 to 6 of this algorithm, for each application, the criteria deadline (dl) and the number of requests (nr) are first calculated. After that, the initial list of applications in the Initial_priority list is sorted in descending order based on the number of requests (line 7).

Algorithm 3: Application Prioritization
Input: Applications, Requests
Output: Prioritized Applications

```
1)    # For application in application's list, calculate two metrics (dl, nr)
2)    for appId in applications do:
3)        req_list = requests (appId)
4)        # Calculate the number of application's requests
5)        nr (appId) = len (req_list)
6)        dl (appId) = deadline (appId)
7)    Initial_priority = Sort (applications, key= nr, descending)
8)    Final_priority = []
9)    for appId in Initial-priority do:
10)       appId_min = Minimum (applications, key = deadline)
11)       dl_rate = dl (appId_min) / dl (appId)
12)       nr_rate = nr (appId_min) / nr (appId)
13)       if (w_dl* dl_rate + w_nr* nr_rate) < = threshold then:
14)           Final-priority.append (appId)
15)           Initial-priority.remove (appId)
16)       else:
17)           Final_priority.append (appId_min)
18)           Initial_priority.remove (appId_min)
19)   Return Final_priority
```

In the next step, a decision is made on the final priority list of applications and stored in Final_priority. To do this, an application with the lowest deadline is searched and extracted from the list (line 10), called the minimum deadline application. Then, according to line 11, the ratio of the deadline of the minimum deadline application and the first application in the Initial_priority list (candidate application) is calculated. This ratio is also calculated for the number of requests for these two applications, as shown in line 12.

[DOI: 10.61186/itrc.15.3.31]

According to line 13, the weighted combination of these two criteria is then calculated and compared with the decision threshold value. If the value of this combination is less than the threshold, the candidate application is added to the Final_priority list and removed from the initial list. This is shown in lines 14 and 15. As in 17 and 18, the application with the lowest deadline is added to the Ininal_priority list and removed from the Initial_priority list.

IV. EXPERIMENTAL EVALUATION

In this section, a brief explanation of the simulation environment is given first in Subsection A. Then the evaluation criteria are explained in Subsection B. In Subsection C the evaluation and comparison results of the methods are presented.

A. Simulation Environment

To evaluate the proposed method the YAFS simulation software is used. This software is a fog environment simulator and has been used in related works. YAFS also has graph-based features and supports critical features of fog environments. It is also open source and its source code is accessible [36]. Table I shows the setting of the parameter's value of links, nodes, and applications, according to the Partition method [23].

TABLE I. Parameter setting of Fog

	Parameter	Value	Description
Link	BW	$6*10^6$-$6*10^7$	**Bandwidth(bit/s)**
	PD	3-5	**Propagation time (ms)**
Node	Fog	100	**Numbers**
	RAM	10-25	**Resources (MB)**
	IPT	100-1000	**Speed (Instr/ms)**
	TB	(0.2-100)	**Terabyte**
Application	Deadline (ms)	2600-6600	**Deadline (ms)**
	Service(number)	2-10	**Service (number)**
	Resource	1-6	**res. Units**
	Packet	1,500,000–4,500,000	**size (bytes)**

Other parameters related to the CFSP method are listed in Table II.

TABLE II. Parameter setting of proposed CFSP method

	Parameter	Value	Description
Fitness Function	λ_1	0.4	**Resource Allocation weight**
	λ_2	0.25	**Node Distribution weight**
	λ_3	0.35	**Community Connectivity Weight**
Application prioritization	Wdl	0.85	**Deadline factor**
	Wnr	0.15	**Number of requests factor**
	threshold	0.9	**Decision parameter for prioritization**

In this research, the workload consists of the number of applications and their corresponding execution requirement. This workload has been borrowed from [23]. However, in [23] authors have considered a fixed workload (20 applications), whereas in this manuscript, we have changed the range of workload from 10 to 80 applications in different evaluations. The detailed specification of the workload is given in Table I.

Other parameter related to the CFSP method are listed in Table II. The values of the parameters in Table II have been chosen empirically and based on several evaluations.

B. Evaluation Metrics

An efficient application deployment method should lead to efficient use of resources and meet the needs of applications. Therefore, we evaluate the efficiency of the proposed method from the following two aspects: resource usage and meeting application's needs.

To evaluate the method from the resource usage aspect, we consider some metrics, such as the average resource usage, placed application usage and the rate of application placement in the cloud.

In addition, to evaluate from the aspect of meeting application's need, we use some other metrics such as the delay, the deadline and, the availability. The deadline metric determines the rate of applications that receive responses before the deadline, the delay metric indicates the average time that a requester waits for receiving a response from an application, and the availability metric shows the availability of applications for their requesters.

1. Average Resource Usage:

The amount of resource use is evaluated as the average amount of resources used by different communities. To do this, the average memory, storage, and processing speed of each community are calculated. Equation 19 shows the average amount of memory used by a community, Equation 20 shows the average amount of storage space used, and Equation 21 shows the average amount of processing speed of the community. In these equations, ucf_{ij} means the amount of resource usage of node f_{ij}. Finally, the average resources used by all formed communities or the average used resources are calculated as presented in Equation 22.

$$uCom_i^{RAM} = \frac{\sum_{j=0}^{|CF_i|} ucf_{ij}^{RAM}}{\sum_{m=0}^{|FN|} fn_m^{RAM}} \quad (19)$$

$$uCom_i^{TB} = \frac{\sum_{j=0}^{|CF_i|} ucf_{ij}^{TB}}{\sum_{m=0}^{|FN|} fn_m^{TB}} \quad (20)$$

$$uCom_i^{IPT} = \frac{\sum_{j=0}^{|CF_i|} ucf_{ij}^{IPT}}{\sum_{m=0}^{|FN|} fn_m^{IPT}} \quad (21)$$

$$\varphi = \frac{\sum_{i=0}^{|Com|} (w^{RAM} * uCom_i^{RAM} + w^{TB} * uCom_i^{TB} + w^{IPT} * uCom_i^{IPT})}{|Com|} \quad (22)$$

[DOI: 10.61186/itrc.15.3.31]

2. Placed application usage:

This metric calculates the number of satisfied requests that are the answers of using the existing deployed applications to the total number of requests shown in Equation 23.

$$\theta = \frac{\sum_{i=0}^{r} R_i^{already}}{r} \quad (23)$$

$$R_i^{already} = \begin{cases} 1 \; if \; A_x \text{ exist in the community} \\ 0 \; if \; A_x \text{ not exist in the community} \end{cases} \quad (24)$$

Where $R_i^{already}$ is defined in Equation 24.

3. Meeting application deadline:

In calculating, the ratio of applications answered before their deadline to all applications this criterion is used (Equation 25).

$$\mu = \frac{\sum_{i=0}^{p} x_i}{p} \quad (25)$$

$$x_i = \begin{cases} 1 \; if \; responsetime(A_i) \leq deadline(A_i) \\ 0 \; if \; responsetime(A_i) > deadline(A_i) \end{cases} \quad (26)$$

Where Equation 26 indicates whether the application deadline is met or not.

4. availability

The availability metric measures the application ratio available to the associated request (R_{A_i}) (Equation 27)

$$\Phi = \frac{(\sum_{i=0}^{|p|} R_{A_i})}{r} \quad (27)$$

$$R_{A_i} = \begin{cases} 1 \; if \; all \; services \; of \; A_i \; are \; accessible \; for \; R_{A_i} \\ 0 \; if \; all \; services \; of \; A_i \; are \; not \; accessible \; for \; R_{A_i} \end{cases} \quad (28)$$

Where Equation 28 shows application availability as if the associated request could access all services of the desired application.

5. Delay

The delay metric is calculated based on two criteria: 1) the Delay between the Requester and the application (DR) and 2) the Delay between the application's Services (DS) (Equation 29).

Definition 1: delay$_{(DR)}$ is delay between requester and an application

Definition 2: delay$_{(DS)}$ is delay between services of an application

$$\boldsymbol{\varepsilon} = delay_{(DR)} + delay_{(DS)} \quad (29)$$

$$\rho(fn_i, fn_j) = en_{ij}^{PD} + \frac{Packet_{size}}{en_{ij}^{BW}} \quad (30)$$

Where Equation 30 calculates the delay between two devices on the shortest path between the source device and the destination device.

C. Experimental results

In this section, the proposed CFSP method is compared to other methods, such as Partition and ILP. Partition method [23] is selected from the community-based placement category as one of recent method in this category. ILP method is chosen to represent the policy-based placement category because it is widely used in related research as an optimization method [29-35]. In addition, different versions of the proposed CFSP method are evaluated. The CFSP.v1 version enhances the proposed CFSP method in terms of the placement infrastructure aspect, and the CFSP.v2 version improves the proposed CFSP method in terms of the application services placement aspect.

One of the important success factors of the placement method is the extent of utilization of the resources. As Fig. 2 shows, the proposed method is better than the ILP and Partition methods because of the creation of communities as placement infrastructure and the balanced network capacity's distribution between them. Also, the CFSP.v2 version consumes fewer resources and compared to other methods has a better performance. This topic shows the impact of improving communities and using multiple criteria to prioritize applications.

Fig. 3 compares the methods using the existing placement usage criterion. As the figure shows, the proposed CFSP method outperformed the ILP and Partition methods and demonstrated the impact of the way the communities are created and used. Fig. 3 also represents that the CFSP.v1 version uses the existing placements better than CFSP and finally the CFSP.v2 version uses most of the existing placements. This shows the effect of the improvements made in both phases on the further development of the proposed method.

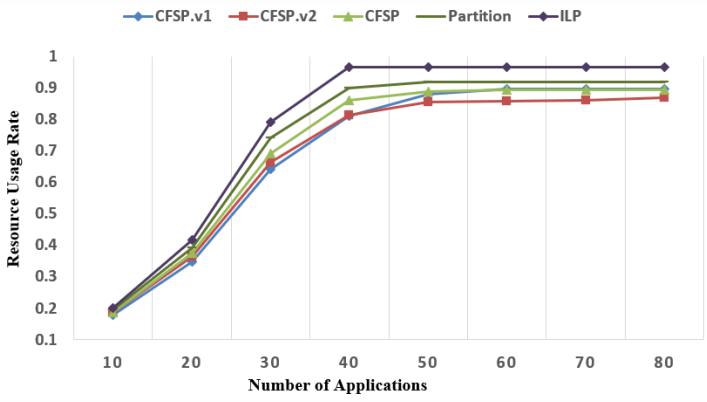

Figure 2. Resource usage rate

[DOI: 10.61186/itrc.15.3.31]

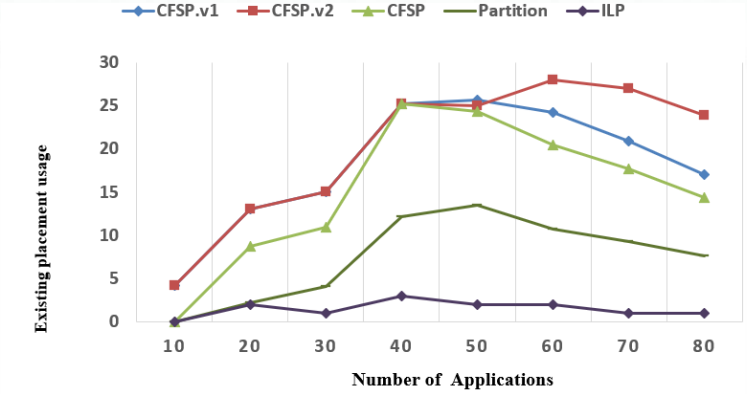

Figure 3. Existing placement usage

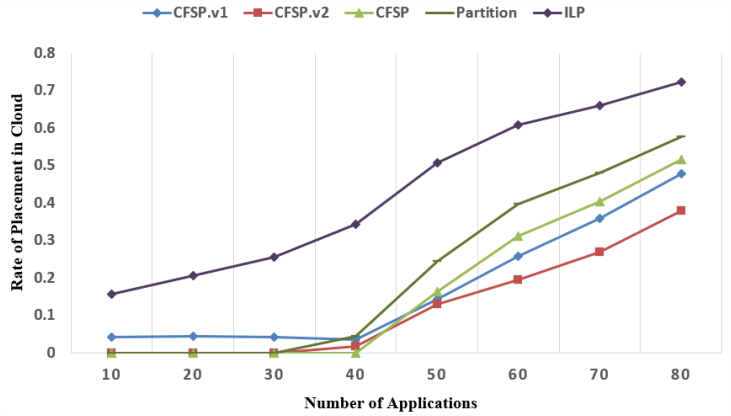

Figure 4. Rate of placement in cloud

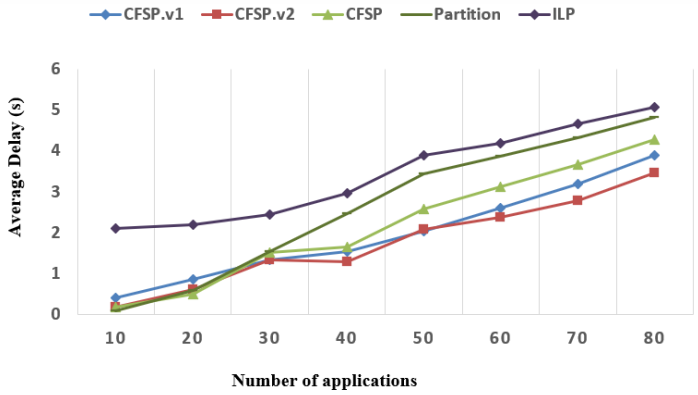

Figure 5. Average delay

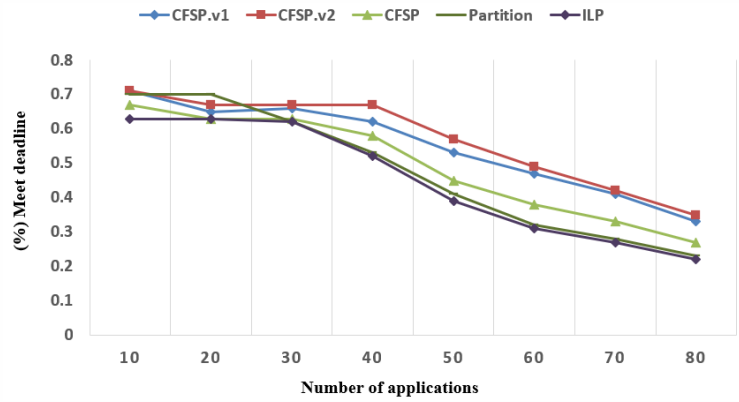

Figure 6. Meet the deadline

[DOI: 10.61186/itrc.15.3.31]

Reducing the amount of application placement in the cloud is an important criterion for evaluating the performance of the placement method. As shown in Fig. 4, the proposed CFSP method places fewer applications in the cloud and the improved versions of the method also have better performance than others. In particular, the CFSP.v2 version causes significant performance differences compared to the CFSP method and the CFSP.v1 version of it.

Fig. 5 shows the performance of methods by the delay criterion. According to this figure, the CFSP method has significantly less delay than the previous methods. In addition, the CFSP.v_2 performed better than other methods, especially as the number of applications increased.

One of the most important requirements for IoT applications is to meet their deadline. The proposed CFSP method performed better on this criterion than previous methods such as ILP and Partition as shown in Fig. 6. With the increase in applications, the CFSP.v2 version of the proposed method also provided the deadline for more applications. Additionally, the better results of the CFSP.v1 version compared to the CFSP method show the impact of the way communities are created.

The failure rate of nodes is changed from 10% to 80% to evaluate the method performance by the availability criterion. In addition, two scenarios were considered which are related to the number of applications were considered. The first scenario refers to the case where the system faces a normal workload, i.e., 20 applications. The second scenario is related to the condition when a higher load has entered the system, i.e., 50 applications.

Fig. 7 shows the availability rate of applications under a normal workload. As this figure shows, the CFSP method has a similar performance to the Partition method in this criterion, while the improvement of CFSP.v1 and CFSP.v2 led to the superiority of the proposed method over the Partition method.

In the situation where the system load has reached 50 applications, the CFSP.v2 version of the proposed method still maintains its superiority over other methods (Fig. 8). It must be mentioned that the high resource degradation limits the planning of their use by different methods. So that when this failure has reached 50%, the behavior of the methods becomes more similar.

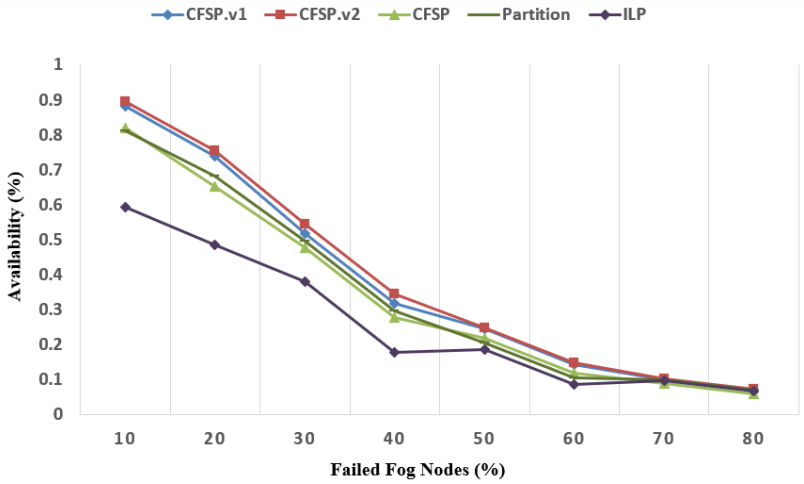

Figure 7. The availability rate in normal workload condition

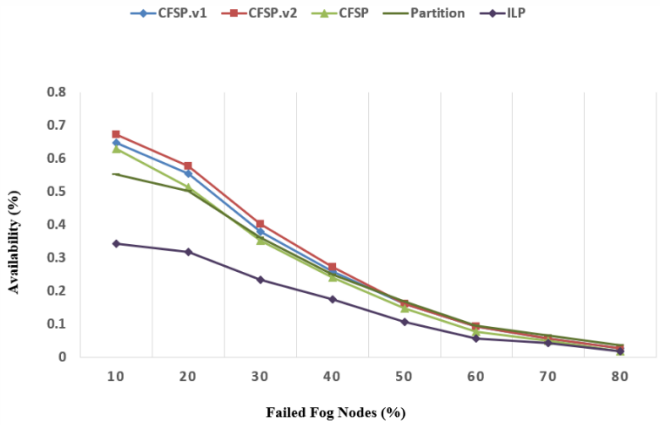

Figure 8. The availability rate in high workload condition

[DOI: 10.61186/itrc.15.3.31]

V. Conclusion

In this paper, we proposed a novel Community-based Fog Service Placement method called CFSP in the fog environment. This method wells manage resources and increase the QoS of the IoT applications.

The CFSP method, along with the connection properties, considered the even distribution of the resource and fog nodes in the phase of creating the deployment infrastructure. This resulted in efficient use of network capacity and meeting the need for more applications. In addition, the proposed CFSP method used the number of requests metric along with the deadline metric for application prioritization in the application service deployment phase. This is unlike the previous investigations that they have applied a single metric for application prioritization. The result of considering multiple criteria is increasing the QoS of applications and placing more applications.

Different versions of the method had an evolutionary nature. The CFSP.v1 version improved the community creation process of the base method, CFSP, and the CFSP.v2 improved the application prioritization of the CFSP.v1. So, the CFSP.v2 had the highest performance among the others.

The result showed that the proposed CFSP method totally performs better in various criteria by applying an appropriate prioritization approach for service placement and focusing on the optimal use of network capacity and resources. Extensive simulation results showed that the CFSP method significantly outperforms the method related to the state-of-the-art across various criteria. For example, the proposed method increased the meeting deadlines by about 17 percent, resource utilization by approximately 7 percent, and availability by about 11 percent.

In future work, we intend to use the community concept in the condition of fog node mobility. Also, we will apply more criteria for prioritizing applications.

References

[1] Ayoubi, M., Ramezanpour, M., and Khorsand, R., "An autonomous IoT service placement methodology in fog computing." Software: Practice and Experience 51, no. 5 (2021): p. 1097-1120.

[2] Shooshtarian, L., Lan, D., and Taherkordi, A. "A clustering-based approach to efficient resource allocation in fog computing." In International Symposium on Pervasive Systems, Algorithms and Networks, p. 207-224. Springer, Cham, 2019.

[3] Schaub, M.T., Delvenne, J.C., Rosvall, M. and Lambiotte, R., 2017. The many facets of community detection in complex networks. Applied network science, 2(1), p.1-13.

[4] Ahuja, M., R. Kaur, and D. Kumar, Trend towards the use of complex networks in cloud computing environment. Int J Hybrid Inf Technol, 2015. 8(3): p. 297-306.

[5] Cazabet, R. and G. Rossetti, Challenges in community discovery on temporal networks, in Temporal Network Theory. 2019, Springer. p. 181-197.

[6] [6] Lei, Y. and S.Y. Philip, Cloud service community detection for real-world service networks based on parallel graph computing. IEEE Access, 2019. 7: p. 131355-131362.

[7] Chandusha, K., Chintalapudi, S.R. and Krishna Prasad, M.H.M., 2019. An empirical study on community detection algorithms. In Smart Intelligent Computing and Applications , p. 35-44, Springer, Singapore.

[8] Wang, W., Liu, D., Liu, X. and Pan, L., 2013. Fuzzy overlapping community detection based on local random walk and multidimensional scaling. Physica A: Statistical Mechanics and its Applications, 392(24), p.6578-6586.

[9] Xie, J., Kelley, S. and Szymanski, B.K., 2013. Overlapping community detection in networks: The state-of-the-art and comparative study. Acm computing surveys (csur), 45(4), p.1-35.

[10] Skarlat, O., S. Schulte, M. Borkowski and P. Leitner. Resource provisioning for IoT services in the fog. in 2016 IEEE 9th international conference on service-oriented computing and applications (SOCA). 2016. IEEE.

[11] Elkhatib, Y., et al., On using micro-clouds to deliver the fog. IEEE Internet Computing, 2017. 21(2): p. 8-15.

[12] Skarlat, O., M. Nardelli, S. Schulte, M. Borkowski and P. Leitner, Optimized IoT service placement in the fog. Service Oriented Computing and Applications, 2017. 11(4): p. 427-443.

[13] Yousefpour, A., G. Ishigaki, R. Gour, and J. P. Jue, On reducing IoT service delay via fog offloading. IEEE Internet of things Journal, 2018. 5(2): p. 998-1010.

[14] Guerrero, C., I. Lera, and C. Juiz. On the influence of fog colonies partitioning in fog application makespan. in 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). 2018. IEEE.

[15] Chunaev, P., Community detection in node-attributed social networks: a survey. Computer Science Review, 2020. 37: p. 100286.

[16] Interdonato, R., et al., Feature-rich networks: going beyond complex network topologies. Applied Network Science, 2019. 4(1): p. 1-13.

[17] Abbasi, M., E.M. Pasand, and M.R. Khosravi, Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm. Journal of Grid Computing, 2020. 18(1): p. 1-14.

[18] Reddy, K., AK Luhach , B. Pradhan, JK Dash and DS Roy, A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. Sustainable Cities and Society, 2020. 63: p. 102428.

[19] Natesha, B. and R.M.R. Guddeti, Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment. Journal of Network and Computer Applications, 2021. 178: p. 102972.

[20] Al-Tarawneh, M.A., Bi-objective optimization of application placement in fog computing environments. Journal of Ambient Intelligence and Humanized Computing, 2021. 12(2): p. 1-24.

[21] Velasquez, K., DP Abreu, L. Paquete, M. Curado, and E. Monteiro. A rank-based mechanism for service placement in the fog. in 2020 IFIP Networking Conference (Networking). 2020. IEEE.

[22] Kimovski, D., et al. Adaptive nature-inspired fog architecture. in 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC). 2018. IEEE.

[23] Lera, I., C. Guerrero, and C. Juiz, Availability-aware service placement policy in fog computing based on graph partitions. IEEE Internet of Things Journal, 2018. 6(2): p. 3641-3651.

[24] Lera, I., C. Guerrero, and C. Juiz. Comparing centrality indices for network usage optimization of data placement policies in fog devices. in 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC). 2018. IEEE.

[25] Filiposka, S., A. Mishev, and C. Juiz, Community-based VM placement framework. The Journal of Supercomputing, 2015. 71(12): p. 4504-4528.

[26] Skarlat, O., M. Nardelli, S. Schulte, and S. Dustdar. Towards qos-aware fog service placement. in 2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC). 2017. IEEE.

[27] Vijouyeh, L. N., Sabaei, M., Santos, J., Wauters, T., Volckaert, B., & De Turck, F., Efficient application deployment in fog-enabled infrastructures. In 2020 16th International Conference on Network and Service Management (CNSM), 2020, p. 1-9. IEEE.

[DOI: 10.61186/itrc.15.3.31]

[28] Sriraghavendra, M., Chawla, P., Wu, H., Gill, S.S. and Buyya, R., DoSP: A Deadline-Aware Dynamic Service Placement Algorithm for Workflow-Oriented IoT Applications in Fog-Cloud Computing Environments. In Energy Conservation Solutions for Fog-Edge Computing Paradigms, 2022, p. 21-47, Springer, Singapore.

[29] Baranwal, G. and D.P. Vidyarthi, FONS: a fog orchestrator node selection model to improve application placement in fog computing. The Journal of Supercomputing, 2021: p. 1-28.

[30] Baranwal, G. and D.P. Vidyarthi, FONS: a fog orchestrator node selection model to improve application placement in fog computing. The Journal of Supercomputing, 2021: p. 1-28.

[31] Gasmi, K., Dilek, S., Tosun, S. and Ozdemir, S., "A survey on computation offloading and service placement in fog computing-based IoT", the Journal of Supercomputing, 78(2), 2022, p.1983-2014.

[32] Heng L, Yin G, Zhao X. Energy aware cloud ‐ edge service placement approaches in the Internet of Things communications. International Journal of Communication Systems. 2022 Jan 10;35(1):e4899.

[33] Velasquez, K., DP Abreu, M. Curado and E. Monteiro, Service placement for latency reduction in the internet of things. Annals of Telecommunications, 2017. 72(1-2): p. 105-115.

[34] Salaht, F., F. Desprez, A. Lebre, C. Prud'Homme, and M. Abderrahim Service placement in fog computing using constraint programming. in 2019 IEEE International Conference on Services Computing (SCC). 2019. IEEE.

[35] Arkian, H.R., A. Diyanat, and A. Pourkhalili, MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. Journal of Network and Computer Applications, 2017. 82: p. 152-165.

[36] Yang, L., J. Cao, G. Liang, and X. Han, Cost aware service placement and load dispatching in mobile cloud systems. IEEE Transactions on Computers, 2015. 65(5): p. 1440-1452.

[37] Lera, I.a.C.G., YAFS, Yet Another Fog Simulator.

Masomeh Azimzadeh received her B.Sc. degree in Software Engineering from the Kharazmi University, Tehran, Iran, in 2001, and the M.Sc. degree in Software Engineering from the Islamic Azad University South Branch, Tehran, Iran, in 2007. Now she is a Ph.D. student in Software Engineering in the Science and Research Branch of Islamic Azad University, Tehran, Iran. Her research interests are in the areas of Fog Computing, IoT, Learning System and Information Retrieval.

Ali Rezaee is an Assistant Professor with the Science and Research Branch of IAU. He is the Chair of Distributed Systems Lab. and Formal Verification lab. at IAU. He is program leader for graduate studies in Computer Engineering in United Arab Emirates Branch of IAU. Also, He is with Shakhes CPA as the Head of R&D, Senior Data Scientist and Solution Architect. His main research interests include Software Architecture, Formal Modeling and Verification of Dynamically Reconfigurable Distributed Systems.

Somayyeh Jafarali Jassbi received the M.Sc. and Ph.D. degrees in Computer Architecture Engineering from the Islamic Azad University Science and Research branch, Tehran, Iran, in 2007 and 2010 respectively. In 2010, she joined the Department of Computer Engineering, Islamic Azad University, Science and Research Branch. Her current research interests include Cloud Computing, Internet of Things (IoT), Wireless Sensor Network and Computer Architecture. She was head of Computer Department in 2012. She was also an active member of young researcher club from 2004. She has written, translated and published several professional books and papers in her fields. She is currently an Assistant Professor at Science and Research Branch of Islamic Azad University (SRBIAU).

Mehdi Esnaashari received the B.Sc., M.Sc. and Ph.D. degrees in Computer Engineering from the Amirkabir University of Technology in Iran, in 2002, 2005, and 2011 respectively. He worked at Iran Telecommunications Research Center as an Assistant Professor from 2012 to 2016. Currently, he is an Assistant Professor in Computer Faculty of K. N. Toosi University of Technology. His research interests include Computer Networks, Learning Systems, Soft Computing and Information Retrieval.

[DOI: 10.61186/itrc.15.3.31]