Service Placement in a Fog Computing Environment with Knowledge of Service Quality

Yousef Abofathi[1], Babak Anari[2], Mohammad Masdari[3]
[1,3]Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
[2] Department of Computer Engineering, Shabestar Branch, Islamic Azad University, Shabestar, Iran
Email: anari322@gmail.com

Received: 03 January 2024 Accepted: 22 August 2024 Published:15 February 2025

Abstract

Fog computing has been proposed to meet the growing demands of Internet of Things users with low latency and high bandwidth. Fog computing has extended cloud computing services to the edge of the network. In this research work, the methods of deploying services in the fog computing environment have been investigated with a focus on service quality. Considering challenges such as resource constraints, heterogeneous environments, and dynamic network conditions, a new framework for deploying services aware of multifaceted aspects of service quality, including response time, availability, reliability, and latency, is proposed. In this article, a new method called DLA-SPSQ is used for optimal placement of clustered services based on service quality criteria. The proposed algorithm is a combination of FCS and DLA-FMP algorithms. The fuzzy FCS algorithm clusters IOT user requests based on the quality-of-service criteria. The quality of clusters is validated based on evaluation criteria. A single objective cost function is used to evaluate the loop delay of modules/services. The results show the improvement of the proposed method in clustering services compared to the case of not clustering them.

Keywords: fog computing, distributed learning automata, quality of service, fuzzy clustering of services, placement of services.

1. Introduction

Cisco announced that more than 75 billion devices will be connected to the Internet by 2025, generating massive amounts of data[1]. Processing large amounts of data with limited resources in real-time is practically impossible. Cloud computing has been used to solve the problem of data processing. Cloud computing also faces increased latency, bandwidth limitations, privacy, and security issues. Fog computing was used to overcome the problems of cloud applications. Fog computing has extended cloud computing services to the network's edge, processing, analysing, and storing data at locations close to users (IoT). Using fog computing results in reducing the amount of data sent to the cloud, reducing delay and calculation costs, and increasing scalability. With all its benefits, fog computing has challenges such as optimal placement of modules/services, resource management, network interference and latency, data management, adaptation to changing environments, security issues, and privacy protection. Optimum placement of services in the fog environment due to the heterogeneous nature and limited capacity of most fog nodes (limited resources), environment dynamics, creation and removal of resources in the fog network, moving people, changing infrastructure and application information over time (e.g., workload change) and geographic distribution of fog devices on an extensive infrastructure is a complex issue. Due to the mentioned reasons, the problem of placing services/modules of Internet of Things applications can be considered an NP-Hard combinatorial optimization problem[2, 3].

In this research work, optimal placement is done on services selected in fuzzy clustering. Fuzzy clustering is performed on them based on service quality criteria. The selected service quality criteria are response time, availability, reliability, and delay. The working method is that the services are initially clustered based on the service quality criteria with the proposed DLA-SPSQ algorithm after removing outliers from the loaded data set. Clustering evaluation criteria have been used to validate the quality of clusters. According to the need of the problem and the acceptable values for the evaluation criteria, clustering with three clusters has been chosen in this research. The XB and average evaluation criteria in each cluster have been used to determine the priority among the three clusters for implementation. After clustering the data set, the services of each cluster are optimally placed in a distributed learning automata system according to the DLA-FMP algorithm. Most of the presented methods for solving the problem of placement of services/modules are based on evolutionary algorithms and without clustering the requests of Internet of Things users, which are unsuitable for online applications due to the time-consuming search for the solution space.

A local and global search framework is designed to locate the modules in the fog topology to implement the DLA-FMP algorithm. To map fog topology to distributed learning automata, a proposed framework has been modelled in the paper x. The proposed algorithm can be used in dynamic environments, and local and global search can be performed simultaneously in the fog environment. The proposed method can be applied to any topology with a high convergence speed. With this method, it is possible to make maximum use of the available capacity of fog nodes in the lower layers and close to the Internet of Things layer, and it is possible to achieve the minimum delay and reduce the execution time of tuples and network consumption

The main contribution of this article is as follows:

- Designing a framework for fuzzy clustering of services based on service quality criteria.
- Providing a new approach for optimal service placement according to service quality criteria.
- Ability to implement the proposed method for homogeneous and heterogeneous topology.
- Validation of the proposed algorithm regarding service execution delay criteria, tuple executiondelay, and network usage reduction.

This paper is further organized as follows: Section 2 provides a summary of related studies, and Section 3 provides concepts related to fuzzy clustering of services, module/service placement problems, learning automata, distributed learning automata, and the evaluated criteria given in the optimal placement of services. The proposed method is explained in Section 4. In section 5, the simulation results and their analysis are given. Finally, Section 6 includes conclusions and future directions.

2. Related Works

This section summarizes several studies related to fuzzy clustering strategies of services and their optimal placement in the fog environment. These studies are motivated by fuzzy system clustering and optimization goals such as minimizing delay or improving service quality. The work related to the fuzzy clustering of services has been done in this article. Most of the studies about optimization have been done in the first article of this research, and here, they are listed in a categorized manner to complete the scope of the proposed algorithm.

A) Several articles of the work done about fuzzy clustering of systems are summarized below:

In[4], the authors address some common challenges in the Internet of Things, such as managing large volumes of data and the need for real-time processing, with the possibility of more efficient and accurate data clustering. This is particularly important for IoT applications where data classification is fundamental to decision-making processes, such as smart cities, healthcare monitoring systems, and industrial automation.

In[5], the researchers propose an innovative algorithmic solution that concurrently optimizes for latency, bandwidth, throughput, cost, and energy consumption to make deployment decisions. They incorporate a Pareto optimization technique, which allows stakeholders to understand the trade-offs and make informed choices about their deployment strategies.

In [6], the authors present an innovative mechanism for embedding Internet of Things (IoT) services in a fog computing environment that prioritizes Quality of Service (quality of service). Acknowledging the importance of meeting quality of service criteria such as latency, throughput, and reliability, the authors have proposed a new service placement strategy using an open-source development model.

In [7],a set of developed algorithms for managing the deployment process is described. These algorithms consider the different QoS requirements of different IoT applications and balance these requirements against the current capabilities and load on cloud and fog resources. Key to this process is the dynamic assessment of network conditions, user demands, and service importance to ensure that QoS objectives are consistently met.

In[8], various algorithms for service placement and resource management, which are designed to maintain quality of service standards, have been discussed. These algorithms focus on minimizing latency, maximizing bandwidth, and ensuring reliability and fault tolerance in IoT services.

The model proposed in[9]considers different quality of service requirements in the context of IoT, such as ensuring low latency, high throughput, and fixed network stability. It features a new architectural solution that uses fog nodes—gateways, routers, or other edge devices—to distribute the computing load between the cloud and end devices. For this, it uses an adaptive mechanism that can dynamically evaluate the service demands of IoT applications in real-time and align them with the capabilities of fog nodes. For example, an intelligent traffic system requires immediate data processing for effective traffic control, which this model prioritizes with efficient use of edge computing resources.

The authors in [10]propose a new approach that combines fuzzy logic with meta-heuristic algorithms for resource provisioning. Fuzzy logic is applied to provide a more flexible decision-making process that can handle the imprecision and uncertainty inherent in cloud environments. It adjusts resource provisioning more subtly than binary logic, which is either too aggressive or too conservative. In this article, fuzzy clustering is used to classify the demand for input resources into different fuzzy categories, and a meta-heuristic algorithm optimizes the allocation of resources in each cluster.

In[11], fuzzy clustering can be used to improve data analysis, resource allocation, and load balancing by assigning workloads to the most appropriate computing layers (cloud canter or edge devices in the fog) and managing inherent uncertainty and variability. The performance and capacity of these resources have been investigated.

In [12], a fuzzy approach for deploying IoT applications in cloud computing environments better aligns with real-world deployment scenarios' dynamic and complex nature and outperforms rigid, rule-based systems.

B)Here, based on the optimization strategies used, a classification of advanced studies is presented, which are (i) mathematical optimization, (ii) heuristic techniques, (iii) meta-heuristics, (iv) machine learning, and (v) Other materials and methods.

Mathematical optimization techniques

Mathematical optimization is finding the best value for an objective function in a permissible set, calculated as the maximum or minimum value based on some criteria. In this method, unlike the complex problem of module placement, more minor problems can be solved because examining the entire solution space in complex problems requires high execution time. Different mathematical optimization models that have been used in fog computing can be called integer linear programming[13], mixed integer linear programming[14], and non-linear integer programming methods [15, 16] which are done as discrete and continuous optimization. Mathematicians have long used this method to solve optimization problems in all sciences, including computers and engineering.

Heuristic techniques

The module placement problem is computationally very complex due to the dynamic nature of the fog infrastructure, and analyzing the entire solution space can be more practical. In this case, heuristic techniques are often used to arrive at the answer. Heuristic techniques use information from previous experiences with similar problems to solve problems. Heuristic techniques include rules that make it easy to implement a practical solution to complex problems but have no guarantee of performance[3, 14].

Metaheuristic techniques

Today, more meta-heuristic methods are used to solve complex and challenging problems. These methods are inspired by nature and are high-level techniques for modeling and optimization. Meta-heuristic methods are optimized by selecting random solutions and acting as a black box. Effective and efficient exploration of the search space avoids local optimization due to the stochastic nature of the method. It improves better solutions in a reasonable amount of time through an iterative search process[17-19].

Machine learning techniques

In several cases, machine learning techniques have been used to solve the problem of placing modules. This method uses data to improve performance among a set of tasks. Based on sample data (training data), it makes a model for decision-making and necessary predictions without explicit planning. Machine learning algorithms have been used in various cases, such as speech recognition, health care[20], machine vision, etc. Developing conventional algorithms to perform the required tasks is impossible and difficult.

Other techniques

Several other strategies have also been reviewed in the literature. A new method called multi-fog placement (MFP) has been introduced [21]to place resources in the Internet of Things systems in the fog environment. In this paper, the authors have used multi-region fog architecture, including several fog nodes, to reduce delay and energy consumption. Comprehensive studies on service placement algorithms have been done[22-26].

3. Preliminaries

This section describes the basic elements underlying the challenge of fuzzy clustering of services/modules and their optimal placement, the principles of machine learning, and the framework of distributed learning machines.

3-1- Fuzzy clustering of services

The concept of fuzzy clustering of Internet of Things (IoT) users' service quality criteria using Fuzzy C-Means (FCM) in a fog network environment entail creating groups (clusters) of IoT service quality experiences and preferences that are not sharply defined but overlap. This approach recognizes the subjective and varied nature of user experiences in IoT applications and the importance of latency-sensitive and context-aware processing provided by fog networking. Here is a summary of how FCM applies to IoT quality of service metrics in a fog network deployment:

- **Quality of Service Data Collection:**

This involves collecting various quality of service metrics from IoT devices spread throughout the cloud network. These metrics can include response time, availability, reliability, latency, and others, which may vary in importance depending on the specific IoT context and application.

- **Application of FCM for Clustering:**

Deploy the FCM algorithm to analyze the service quality data. Since FCM allows for fuzzy membership, each IoT user or device can belong to multiple clusters that represent different service quality profiles or experiences.

- ✓ **Initialization:** Choose the number of clusters and initialize the cluster centres.
- ✓ **Membership Assignment:** Compute the membership degree of each IoT user's service quality data for each cluster.
- ✓ **Centroid Update:** Update the cluster centroids based on the calculated membership degrees and the service quality data.
- ✓ **Iterative Process:** Iterate the assignment and update steps until the centroids stabilize within a small tolerance limit.

➢ **Interpretation of Clusters:**

Assess the resulting clusters to understand different categories or levels of service quality experienced by users across the fog network. It can be insightful to identify clusters that reflect high satisfaction, moderate satisfaction, and low satisfaction, for example.

➢ **Action based on Cluster Analysis:**

Utilize the insights from cluster analysis to optimize resource distribution, improve service delivery, and forecast future demands or the need for infrastructure adjustments in the fog network.By using FCM, the fog network can reliably interpret the nuanced, user-reported experiences of service quality, accommodating the inherently imprecise and overlapping evaluation that different IoT users might have. Furthermore, the localized data analytics capability of fog computing allows for real-time or near-real-time clustering and analysis, whichis crucial for promptquality of service adjustments and enhancements in IoT systems.

3-2- Module Placement ProblemDefinition

Suppose an array of IoT applications for processing as$App\{App_1, \dots, App_k, \dots, App_n\}$ defined.App_k representing a collection of m modules, as $\{M_{(k,1)}, \dots, M_{(k,t)}, \dots, M_{(k,m)}\}$. These modules are mandated to execute within the virtualized spheres of either the cloud or the fog on demand from a user. The optimal placement of each module unto an appropriate fog node is contingent upon a specific cost metric. Modules are characterized by the Equation (1).

$$M_{k,t} = \{M_{k,t}^{CPU}, M_{k,t}^{RAM}, M_{k,t}^{Disk}, M_{k,t}^{BW}, M_{k,t}^{DL}, M_{k,t}^{size}\} \quad (1)$$

In Equation (1), $M_{k,t}^{CPU}$, $M_{k,t}^{RAM}$, $M_{k,t}^{Disk}$, $M_{k,t}^{BW}$, $M_{k,t}^{DL}$, and $M_{k,t}^{size}$ correspond to the module's requisites for processing capability, RAM, main memory, network traffic to serve the module, execution deadline (in milliseconds), and the volume of module instructions (in MIPS),respectively. The process of assigning modules,$M_{k,t}$ where t belongs *to t ε{1,…,* $|App_k|$ *}* from applicationsApp_k, *k* ∈*{1,…, n}* to fog nodes $Fog_{(i,j)}$, *iε{0,…, L}, jε{0,…, nᵢ}* is performed such that the best possible solution can be reached with the lowest value of the cost function without violating the service level agreement (SLA) and disrupting the QoS. It is known as the service/module placement problem.

3-3- Learning Automata

The method of automata-based learning involves selecting the most suitable action from a range of possible actions. An automaton selects an action from its limited action set to use in a random environment. This environment then the chosen action and provides feedback to the automaton. With the feedback, the automaton refines its action-selection mechanism, specifically its probability distribution of actions. This process — from action selection by the automaton to environmental application and assessment to the adjustment of the action probability distribution —iterates until the automaton achieves a predefined goal state. An automaton that adapts to a random and unknown environment and enhances its performance is called a learning automaton. This concept was introduced by [27, 28]. Learning automata have found utility across diverse areas, such as computer networks[29],fuzzy logic systems[30], image analysis[31], chaos theory[32], structure identification in Bayesian Networks[33, 34], advanced reinforcement learning[35], and the dissection and interpretation of speech [36]. Figure 1 illustrates the interaction between a learning automaton and its stochastic environmental context.

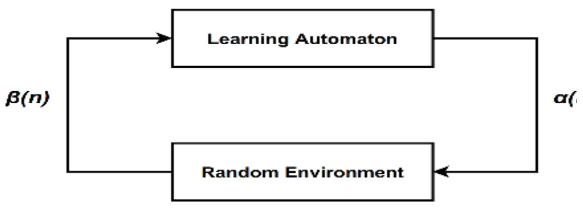

Fig.1. Mutual interaction between learning automata and the random environment

Learning automata are classified into fixed-structure learning automata (FSLA) and variable-structure learning automata (VSLA). Within a fixed-structure automaton, the probabilities associated with selecting actions and the probabilities governing transitions between states are static. Conversely, in a variable-structure automaton, these same probabilities—the action selection likelihoods and state transition frequencies—are dynamic, evolving as time progresses. The variable-structure learning automaton (VSLA) can be conceptualized as LA={α,β,P,T}, where it encapsulates the permitted actions available to the automaton, represents the environmental feedback (reinforcement signals) in reaction to the automaton's selected action, constitutes the set of probabilities for each action (Element Pi indicates the probability of choosing action αi). T denotes the learning algorithm updating the automaton's action probability vector based on the received environmental feedback. In binary environments, the feedback from the environment to the automaton's action is interpreted as either positive/rewarding (β=0) or negative/ penalty (β=1). The most fundamental learning algorithm is linear, which is formalized in Equations (2) and (3). When the automaton's action is rewarded by the environment, the action probability vector gets updated as specified in Equation (2). Conversely, if the environment penalizes the automaton's action, the update is carried out per Equation (3). Within Equations (2) and (3), the variables 'a' and 'b' act as tuning parameters for rewards and penalties, respectively, affecting the increment and decrement rates of the action probabilities.

$$p_j(n+1) = \begin{cases} p_j(n) + a\left(1 - p_j(n)\right), & if\ j = i \\ (1-a)p_j(n), & if\ j \neq i \end{cases} \tag{2}$$

$$p_j(n+1) = \begin{cases} (1-b)p_j(n), & if\ j = i \\ \dfrac{b}{r-1} + (1-b)p_j(n), & if\ j \neq i \end{cases} \tag{3}$$

3-4- Distributed learning automata

A distributed learning automata (DLA) is a new model of interconnected automata in which a set of automatons cooperate to solve a specific problem. Choosing an action by an automaton in the network activates the automata corresponding to this action. Only one automaton is active in the network at any time. A DLA with n learner automatons can be defined by a directed graph (A, E) where A= {A_1, A_2,⋯,A_n}is the set of automata and E⊂A×A is the set of edges such that the edge (i,j) corresponds to the jth action from automata Ai to automata A_j. $A_j$will be activated when the learning automata action A_j is selected. The number of actions of the learning automata A_k, $k = 1,2,3,\ldots,n$,equals the degree of the output of the node corresponding to the learning automata A_k (Figure 2).

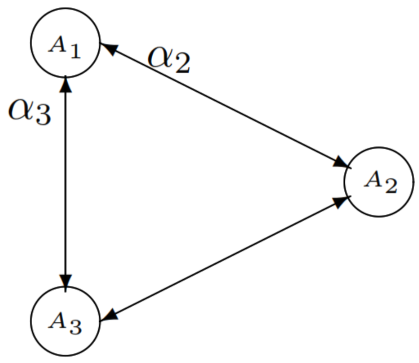

Fig. 2. Distributed learning automata with three automatons

3-5- Application Loop Delay

Each fog node has a waiting queue in which the tuples are placed as soon as they arrive at the nodes. In fog nodes, each tuple is removed from the queue and processed with the default FIFO policy. The application execution loop delay (D) between the user and the deployed fog nodes is calculated using Equation (4).

$$D=\sum_{\text{for each APP}_j}^{\infty}(T^{s\to f}+\sum_{i=1}^{|APP_j|-1}\left(PD_{f\to f'}+T^{Trans}_{f\to f'}+T^{Queue}_{f}+T^{process}_{f}\right)+T^{f\to a}) \quad (4)$$

where $T^{s\to f}$ is the transfer time from the sensor to the first fog node, and $T^{f\to a}$ is the transfer time from the last fog node to the actuator; moreover, $T^{Tran}_{f\to f'}$, The transmission delay is based on the length of the tuples and the network bandwidth between two end-to-end nodes, which is calculated using equation (5).

$$T^{Trans}_{f\to f'}=\frac{\text{TupleNwLength}}{\text{Bandwidth}} \quad (5)$$

In equation (5), Bandwidth and tupleNwLength, respectively, indicate the link's Bandwidth and the size of the tuple used between two end-to-end fog nodes in the network. The waiting time of each fog node is calculated using equation (6).

$$T_f^{Queue}=\sum_{ɨ=1}^{n}T_ɨ^{wait} \quad (6)$$

where $T_ɨ^{wait}$,ɨε{1,2, 3,…,n}is the waiting time for the tuple ɨthin the waiting queue. The execution time at each fog node is calculated using equation (7).

$$T_f^{process}=\sum_{ɨ=1}^{n}T_ɨ^{process} \quad (7)$$

In the above equation, $T_ɨ^{process}$ is the execution time of the ɨth tuple in each fog and iscalculated as Equation (8).

$$T_{\mathrm{f}}^{process} = \frac{TupleCpuLength}{Total\_MIPSoffog} \quad (8)$$

In equation (8), *TupleCpuLength* indicates the processing required to execute a tuple, and *Total_MIPS* of the fog indicates the total MIPS allocated to the processing element node of each fog (PE). The propagation delay is calculated using equation (9) between two adjacent fog nodes.

$$PD_{f\to f'} = \frac{D_{f\to f'}}{PS} \quad (9)$$

The end-to-end distance between two adjacent fog nodes is indicated by $D_{f\to f'}$, PS indicates the speed of light, and its size is equal to$3 * 10^{8^{\mathrm{m/s}}}$.

3-6- Tuple CPU Execution Delay criterion

The execution delay of tuple processing is calculated using Equation (10).

$$Tuple\ Cup\ Execution\ Delay= \sum_{for\ All\ Tuple\ Type} AverageCpuTime_{(TupleType)} \quad (10)$$

where *AverageCpuTime* indicates the average execution time of each type of entered tuple.

3-7- Network Usage criterion

IOT input devices transmit the data required for processing in multiple byte units to fog or cloud nodes in higher layers. The amount of data sent and received by the network is called network usage. Network utilization is calculated using equation(11). Network usage depends on the size and overall delay of the request between the source device and the user's desired destination.

$$Network\ usage= Latency*TupleNwSize \quad (11)$$

In equation (11), Latency indicates the sending delay of each node in the fog network, and TupleNwSize indicates the file size of each sent tuple.

4. The Proposed DLA-SPSQ Algorithm

The proposed Distributed Learning Automata-Service Placement based Service Quality (DLA-SPSQ) method is performed in two steps, which are:

1. Fuzzy clustering of services
2. Optimum placement of services based on distributed learning automata

Fuzzy clustering of services is done by FCS Algorithm and optimal placement of services is done based on distributed learning automata by DLA-FMP method, each of the methods are explained in order below.

4-1- Fuzzy clustering of services (FCS)

Web service quality of service (QoS) is essential to the overall user experience. This includes response time, availability, reliability, and latency. Due to the wide variation and uncertainty in these parameters, the fuzzy logic approach is often used. Fuzzy logic enables more flexible and realistic modeling of complex systems by handling fuzzy and ambiguous data. Fuzzy-based clustering is a classification method in machine learning where data elements are grouped based on their similarity. In the QoS of web services, fuzzy-based clustering can separate services into different quality levels or groups based on specific QoS characteristics. This allows easy management and prioritization of services.

Fuzzy-based QoS clustering of web services has been widely used and researched. Several models and techniques have been proposed to improve service selection, matching, and composition in web services. Continued development in this area promises more efficient and user-centric web services.

```
AlgorithmFCS
1.   Load dataset.
2.   Remove extraneous data from dataset.
3.   fuzzy clustering.
4.   Calculate below the Evaluation criteria of clustering for validating cluster quality.
     A)   PC (Partition Coefficient)
     B)   CE (Classification Entropy)
     C)   XB (Xie-Beni Index)
     D)   avgSilhouette
5.   Determining the priority of clusters for placing services according to the XB values of each cluster and the average value of clustering quality criteria in each cluster.
```

Each of the steps of the FCS algorithm is explained in order below.

- **Load dataset**

In this section, the method of loading datasets in different formats for fuzzy clustering of web services requested by Internet of Things users to run in the fog network by the Load dataset algorithm is explained in two modes.

```
Algorithm Load dataset
Input:your dataset
    1.   Load datasetfrom mat file:
         //Path and file name with extensionmat.
         path='D:\aaaFuzzy110\fuzzy classification\datasetName.mat'
         Data=load(path);
    2.   Load datasetfrom xlsx file:
         /Path and file name with extensionxlsx.
         file Path = 'C:\Users\Home\Desktop\xlsxFileName.xlsx'
         Data = xlsread(filePath);
```

- **Remove extraneous data from dataset.**

Outliers can significantly bias clustering results because they may be inappropriately attributed to the degree of membership in a cluster. To remove outliers, you should consider preprocessing your data.

While MATLAB does not have a built-in function specifically to detect and remove outliers for direct FCM, it is certainly possible to combine several steps and techniques to detect outliers before performing FCM. Use statistics, including:

1) **Standard deviation method**: If the data is normally distributed, about 68% of the data values fall within one standard deviation of the mean and 95% fall within two standard deviations. Data points that lie beyond a certain threshold may be considered outliers.
2) **Interquartile range (IQR):** The spread of the middle 50% of values. Anything more than 1.5 times the IQR above the third quartile and below the first quartile can be considered an outlier.

3) **Standard Deviation Method:** If data is normally distributed, then around 68% of data values will lie within one standard deviation of the mean, and 95% within two standard deviations.

4) **Boxplot Analysis:** Using boxplots can help visualize potential outliers.

5) **Proximity Based Methods:** Such as DBSCAN, where data points that do not fall within a cluster are considered outliers.

This section explains how to remove outliers from a data set using the IQR method by the remove outlier's algorithm.

```
Algorithm remove outliers
Input: dataset
Output: filteredData
% Compute the lower and upper bounds for each feature
Q1 = quantile (dataset, 0.25);
Q3 = quantile (dataset, 0.75);
IQR = Q3 - Q1;
lowerBound = Q1 - 1.5 * IQR;
upperBound = Q3 + 1.5 * IQR;
% Initialize a logical index vector assuming all values are not outliers
nonOutlierIdx = true (size (dataset, 1), 1);
% Check for outliers in each feature dimension
for i = 1: size (Data.data10, 2)
nonOutlierIdx = nonOutlierIdx& ...
        (dataset (: i) >lowerBound(i)) & ...
        (dataset (: i) <upperBound(i));
end
% Use the nonOutlierIdx to filter the non-outliers
filteredData = dataset (nonOutlierIdx, :);
```

The standard methods of outlier detection might incorrectly remove non-outliers, especially when the data is not normally distributed or when the "outliers" actually represent valuable extremes that are of interest.

FCM is less sensitive to outliers than hard clustering methods because it assigns a degree of belonging to each cluster rather than absolute membership. But the presence of outliers can still affect the centroids and therefore the results of the clustering process.

- **Fuzzy clustering**

The FCM is a soft clustering method that allows a single web service to belong to multiple clusters to varying degrees. This characteristic is particularly beneficial in fog environments where web services exhibit varying quality-of-service (QoS) attributes and may not fit strictly into a single category. For instance, a video streaming service might rank highly in bandwidth but lower in response time, fitting into different clusters for different QoS assessments. By applying FCM, fog nodes can categorize services into clusters based on criteria like latency, bandwidth, reliability, and throughput.

This section explains how to use the FCM (Fuzzy C-Means) function for fuzzy clustering of web services based on service quality criteria using a fuzzy clustering algorithm.

```
Algorithm fuzzy clustering
Input: dataset, number of clusters
Output:clusters, U,centers
X= dataset
[centers, U] = fcm (X,numberofclusters)
maxU=max(U); // U is the membership matrix from fcm
// Identifies data points that are most strongly associated with each cluster.
$index_k$=find (U (k, :)==maxU); □ k=1,2, 3, … ,numberofclusters
//In this step, a subset of the dataset is created.
$cluster_k$= X ($index_k$, :) □ k=1,2, 3,…, numberofclusters
```

- **Calculate Evaluation criteria of clustering for validating cluster quality**

Calculating cluster validity indices is an essential step in evaluating the performance of a clustering algorithm in MATLAB. Cluster validity indices can help assess the quality of cluster formation—for example, how distinct they are, how dense they are, and how well they fit the underlying data distribution.

For fuzzy clustering, such as that performed by the FCM (Fuzzy C-Means Clustering) function in MATLAB, standard validity indices are:

PC (Partition Coefficient)

This method is specific to fuzzy clustering. The PC is calculated for different numbers ofclusters, and a higher value indicates a better clustering structure.It can be calculated using equation 12.

PC = sum (sum (U.^2)) / size (X, 1) (12)

where U is the membership matrix from FCM, and X is dataset.

CE (Classification Entropy)

Evaluates the distribution of membership values across clusters. Lower values are generally better, indicating a more defined partition.It can be calculated using equation 13.

CE = -sum(sum((U.*log(U)))) / size (X, 1) (13)

XB (Xie-Beni Index)

Provides a ratio of compactness and separation of clusters; a lower value of the Xie-Beni index indicates a better partition due to compact and well-separated clusters. FCM algorithm can be run for different numbers of clusters and XB index can be calculated for each partition. The number of clusters with the lowest XB index can be considered as optimal clustering, which can be calculated using the Xie-Beni Index algorithm.

AlgorithmXie-Beni Index

```
Input: dataset,centers, U //U is the membership matrix
from FCM
Output:XB
X= dataset
sum_dist = sum(sum(sum((U.^2))).* pdist2(X, centers).^2);
min_dist = inf;
for i = 1:2
    for j = i+1:3
        d = norm (centers (i, :) - centers (j, :), 2);
min_dist = min (min_dist, d);
    end
end
XB = sum_dist / (size(X,1) * min_dist^2);
```

- **Silhouette Coefficient**

Calculating the silhouette coefficient in fuzzy clustering, especially in Fuzzy C-Means (FCM), can be tricky since each data point has a degree of belonging to every cluster, not just a single one. The silhouette coefficient measures how similar a point is to its own cluster compared to other clusters, which is straightforward in hard clustering methods. Still, for fuzzy clustering, one must first determine the degree to which a data point is assigned to its clusters.

Here's how to calculate the silhouette coefficient for a clustered dataset, which the FCM function in MATLAB can calculate.

crispy the Clusters:

Although in FCM, each data point has a membership degree to each cluster, to calculate the silhouette coefficient we need a “hard” assignment. Assign each data point to the cluster for which it has the highest membership value.

Calculate a(i)Within-ClusterDissimilarity:

For each data point i, calculate the average distance from i to all other points in its assigned cluster.

Calculate b(i)Between-Cluster Dissimilarity:

For the same data point i, calculate the average distance from i to all points in the next nearest cluster that it is not assigned to. One way to define the “next nearest” cluster could be finding the cluster for which the average distance to all its points is minimal, excluding the cluster to which i is assigned.

Compute the Silhouette Value:

The silhouette value s(i) for each data point is calculated byequation 14:

$$s(i) = (b\_i - a\_i) / \max (a\_i, b\_i) \qquad (14)$$

Here, a(i) represents the average distance from i to all other points in its own cluster, and b(i) represents the smallest average distance from i to points in any other cluster that i is not a part of.

Average Silhouette Coefficient:

To find the overall silhouette coefficient for the dataset, average s(i) over all data points i.

In MATLAB, the silhouette algorithm is designed for hard clustering outputs. It is necessary to rewrite the existing function to implement the silhouette coefficient in fuzzy mode. It is explained here under the Silhouette Coefficient Algorithm.

AlgorithmSilhouette Coefficient

```
Input: dataset, U //U is the membership matrix from FCM
Output:avgSilhouette//Average silhouette value of the dataset
X= dataset
[~, hardClustering] = max (U, [], 1);
% Peral locate silhouette values array
s = zeros (size (X, 1), 1);
 for i = 1: size (X, 1)
% within-cluster distance
a_i = mean (pdist2(X(i, :), X(hardClustering == hardClustering(i), :)));
      % between-cluster distances excluding the own cluster
b_i = inf;
for k = setdiff(unique(hardClustering), hardClustering(i))
distToOtherCluster = mean (pdist2(X (i, :), X(hardClustering == k, :)));
b_i = min (b_i, distToOtherCluster);
end
s(i) = (b_i - a_i) / max (a_i, b_i);
end
 avgSilhouette = mean(s)
```

This simplified algorithm does not handle some cases (such as when a_i is zero), so this must be modified for the specific application and data set.

4-2- DLA-FMP Algorithm

After the data set's clustering, each cluster's services are optimally placed in a distributed learning automata system according to the DLA-FMP algorithm[37]. With the optimal placement performed by the Distributed Learning Automata – FogModule/Service Placemen(DLA-FMP) algorithm, it is possible to significantly reduce the delay in the execution of services in clusters with higher priority and respond to the requested users' services who have paid more money at a suitable time. Provide higher quality services to increase user satis faction. The general steps of the DLA-FMP algorithm are illustrated by the diagram presented in Figure 3.

This section describes the main concepts of the algorithm proposed in article x. The DLA-FMP algorithm's working method is that the required parameters are first defined and then quantified. In the next step, a fog topology is created using the iFogSim tool after the fuzzy clustering service. To perform optimal placement, a distributed learning automata is mapped on the defined fog network so that each of the fog nodes in the fog network corresponds to a learning automaton.

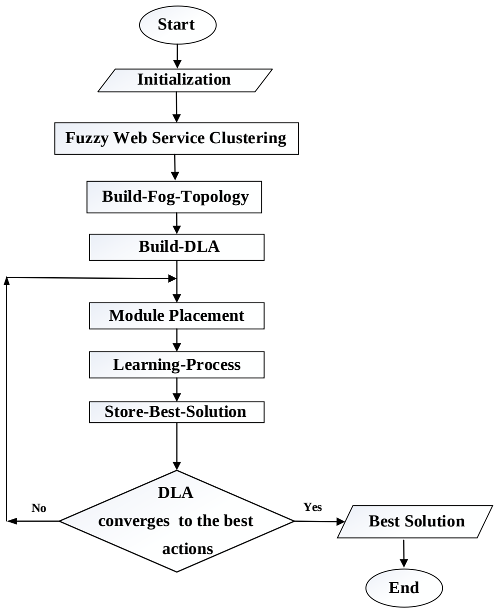

Fig. 3. DLA-FMP algorithm steps diagram

This section describes the main concepts of the algorithm proposed in article x. The DLA-FMP algorithm's working method is that the required parameters are first defined and then quantified. In the next step, a fog topology is created using the iFogSim tool after the fuzzy clustering service. To perform optimal placement, a distributed learning automata is mapped on the defined fog network so that each of the fog nodes in the fog network corresponds to a learning automaton.

The proposed algorithm for solving the module placement problem uses three main phases in each iteration: Phase 1: Module placement, Phase 2: Learning process, and Phase 3: Cost function evaluation phase. These three phases are explained below:

Phase1-Module Placement: In this phase, each application is placed in a module in one of the fog nodes, starting from the edge-level fog nodes. Each edge-level automaton chooses one action from its set of actions to place each module. After selecting the action, the capacity of the fog node corresponding to the selected action is checked using Equation (15).

$$M_{(k,t)}^{RAM} \leq Fog_{(i,j)}^{RAM}, M_{(k,t)}^{cpu} \leq Fog_{(i,j)}^{cpu} \qquad (15)$$

The desired module is deployed if the above conditions apply to the selected fog node.Otherwise, the learning automaton corresponding to the selected action will be responsible for placing that module. This process continues until the module can be placed in one of the fog nodes or that module is placed in the cloud. The important point in this process is that the cloud can only choose one action. In this phase, learning automata in DLA finds a suitable place for each desired module of each application.

Phase 2 - Learning process: The output of Phase1 indicates that each application module is located in which fog node. In Phase 2, we check whether the action selected by the automaton corresponding to each fog node could minimize the objective function. If that automaton has minimized the objective function, it is inferred that the action chosen by that automaton is suitable. Thus, that action will be rewarded according to Equations (2) and (3); otherwise, it will be penalized. This learning process is repeated until DLA converges to the best actions.

Phase 3 - The cost function evaluation and Store-Best-Solution: The cost function is used to evaluate the rewards or penalties in the learning process. The cost function (D) is considered as a single objective function in the proposed method. Therefore, the goal is to reduce the average delay of IoT user services (D) in all fog nodes during the execution of applications.In this phase, the best solution is saved after learning and evaluating the cost function.

These three main phases continue until the distributed learning automata converges to the best solution.

5. Evaluation and Experimental Analysis

The evaluation, similar to the proposed DLA-SPSQ method, is done in two stages, which are:

5-1- Fuzzy clustering of services

In these stages, the selected data set is clustered by Fuzzy C-Means (FCM) in MATLAB after removing extraneous data. Clustering evaluation criteria have been calculated to validate the quality and optimal number of clusters. High-priority clusters are determined by calculating the average service quality indicators in each cluster. The data set (QWS Dataset) has been used for clustering services[39]. From the service quality criteria, four criteria have been selected according to the purpose of the problem, which are given in Table 1.

Table 1. Selected service quality criteria

Title	Description
Response Time	Time taken to send a request and receive a response
Availability	Number of successful invocations/total invocations
Reliability	Ratio of the number of error messages to total messages
Latency	Time taken for the server to process a given request

To remove outlier data from the data set, the algorithm (remove outliers) mentioned in section 4-1 is used; also, for easy work, the fuzzy k-means method can be used. The calculation results of the PC, CE andavgSilhouetteof clustering evaluation criterion for validating the quality of the clusters are given in Table 2.According to the need of the problem and the acceptable values for the evaluation criteria, clustering with three clusters has been chosen in this research work (row 2 of Table 2).

Table 2. Selected clustering from clustering evaluation criterion

Clusters	PC	CE	avgSilhouette
2 Cluster	0.964617	0.064732	0.872612
3 Cluster	0.92457999	0.1478412	0.7893497
4 Cluster	0.808208	0.347637	0.574982
5 Cluster	0.774274	0.41459	0.522598
6 Cluster	0.736786	0.492458	0.479991
⋮	⋮	⋮	⋮

The calculation results of the XB clustering evaluation criterion for validating the quality of the clusters are given in Table 3.

Table 3. Selected clustering from XB criteria

Clusters	XB1	XB2	XB3	XB4	XB5	XB6
2 Cluster	164	2335				
3 Cluster	2498	928	21345			
4 Cluster	198006	8608	7893	33356		
5 Cluster	107645	11261	31998	12773	316640	
6 Cluster	19119	830225	388270	85181	22014	25598
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In this research work, the priority of the clusters can be determined according to the lowest amount of response time, delay, reliability, XB criterion and the highest amount of availability.

Table 4.Selected cluster from the average value of QoS criteria in each cluster

Evaluation Criteria	Cluster1	Cluster 2	Cluster 3
Response Time	1214.233	237.0746	3269.776
Availability	77.1105	81.5802	77.12698
Reliability	71.67403	69.56739	72.11111
Latency	216.926	32.65847	379.0551
XB	2498	928	21345
Priority	2	۱	3

In Table 4, the lowest average result for the evaluation criteria of response time, reliability, delay, XB and the highest average availability are obtained for cluster 2. Therefore, with certainty, the first priority can be assigned to cluster 2. (lighted with yellow color). According to the values obtained for other clusters, clusters 1 and 3 are placed in the second and third priorities, respectively.

5-2- Simulation and evaluation of the proposed method

In these stages, the DLA-FMP method has been used to calculate *AppLoopDelay*, *Tuplecupexecutiondelay*, and *Networkusage* criteria in high-priority clusters of different sizes, and the results of the proposed DLA-FMP method have been evaluated and compared in two modes of clustering and without clustering of services.

5-2-1- Simulation Environment

The iFogSim simulator tool and Java programming language have been used to simulate the proposed method. The specifications of the system used are given in Table 5. The iFogSim simulator uses a tree structure to generate fog topology. Since the proposed fog topology structure is a graph, the iFogSim simulator has been developed to simulate the proposed method so that any graph can be defined and created.

Table 5. Specifications of the simulation system

Name	Description
CPU	Core i7-3720QM CPU @ 2.60GHZ
RAM	32.0GB
Memory	1TB+ 128GB SSD
OS	Window10-64 bit

5-2-2- Parameter Setting

The proposed method considers the reward (a) and penalty (r) parameters of 0.3 and 0.003, respectively. The values assigned to their parameters in related articles have been used in implementing the compared algorithms.

5-2-3- Fog Device Characteristic

In the simulation of the proposed method, a heterogeneous topology with 63 fog nodes, which has 32 edge nodes and is shown as $T_{63}(32)$, is used. The CPU and RAM specifications of fog nodes of topology $T_{63}(32)$ are given in Table 6. In addition, the specifications related to the power consumption of each fog node in the idle state (idlePower), high bandwidth (UpBW) and low bandwidth (downBW) are given in Table 7.

Table 6. Topology with 63 fog nodes $T_{63}(32)$[37]

Level	**Fog$_{(i,j)}$$(\boldsymbol{CPU}, \boldsymbol{RAM})$**
0	Fog$_{(0,0)}$(44800,40000)
1	Fog$_{(1,0)}$(22400,20000),Fog$_{(1,1)}$(33600,30000)
2	Fog$_{(2,0)}$(18000,16000),Fog$_{(2,1)}$(17000,15000), Fog$_{(2,2)}$(19000,17000),Fog$_{(2,3)}$(20000,18000)
3	Fog$_{(3,0)}$(8400,8000),Fog$_{(3,1)}$(8400,8000), Fog$_{(3,2)}$(5600,6000),Fog$_{3,3)}$(5600,6000), Fog$_{(3,4)}$(16000,14000),Fog$_{(3,5)}$(15000,13000), Fog$_{(3,6)}$(14000,12000),Fog$_{(3,7)}$(12000,10000)
4	Fog$_{(4,00)}$(8400,8000),Fog$_{(4,01)}$(8400,٨000), Fog$_{(4,02)}$(18000,16000),Fog$_{(4,03)}$(8600,6000), Fog$_{(4,04)}$(12000,10000),Fog$_{(4,05)}$(13000,8000) , Fog$_{(4,06)}$(16000,14000),Fog$_{(4.07)}$(9400,6000), Fog$_{(4,08)}$(16000,10000),Fog$_{(4,09)}$(5600,6000), Fog$_{(4,10)}$(4200,2000),Fog$_{(4,11)}$(5600,4000), Fog$_{(4,12)}$(14000,٨000),Fog$_{(4,13)}$(12000,10000) , Fog$_{(4,14)}$(8000,6000),Fog$_{(4,15)}$(6000,4000),
5	Fog$_{(5,00)}$(4000,2000),Fog$_{(5,01)}$(4000,2000) , ... ,Fog$_{(5,31)}$(4000,2000)

Table 7. Fog nodes parametervalues

Fog node	**UpBW, downBW, idlePower**
Cloud	100, 10000, 1332
Fog nodes	10000, 10000,83.4333

5-2-4- Application Characteristic

The iFogSim simulator tool was used to create programs and related modules. The general structure of the program used in the iFogSim simulator is shown in Figure 4. It includes three modules: ***Object_Detector***, ***Motion_Detector***, and ***Object_Tracker***.One ***Object_Detector*** module is considered for App_1, and therequired ***Object_Detector*** modules are considered for applications $App_2,\ App_3\ and\ \ App_4$ according to Table 8.

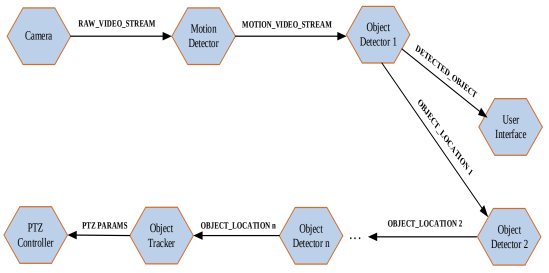

Fig. 4. Modules and edges of the smart monitoring application [38]

Table 8. Specifications of the generated applications

Application Name	**Modules**	$\boldsymbol{App_k}("\boldsymbol{M^{CPU}_{(K,1)}}, \boldsymbol{M^{RAM}_{(K,1)}}", \ldots, "\boldsymbol{M^{CPU}_{(K,m)}}, \boldsymbol{M^{RAM}_{(K,m)}}")$
App_1	3	App_1(“50,10”,”400,10”,”200,10”)
App_2	6	App_2(“50,10”,”400,10”,”200,10”, ”600,10”,”200,10”,”400,10”)
App_3	9	App_3(“50,10”,”400,10”,”200,10”, ”600,10”,”200,10”,”400,10”, ”200,10”,”600,10”,”200,10”)
App_4	12	App_4(“50,10”,”400,10”,”200,10”, ”600,10”,”200,10”,”400,10”, ”200,10”,”600,10”,”200,10”, ”300,10”,”400,10”,”300,10”)

5-3- Experiments

In this section, the results related to the simulation and evaluation of each parameter in both clustering and non-clustering modes of services have been reviewed and compared

5-3-1-Experiment1

In this experiment, the application loop delay is calculated by the DLA-FMP algorithm in the two modes of clustering and non-clustering of services by Equation 4. The simulation results in both cases are given in Figure 5. According to this figure, due to the optimal placement of high-priority cluster services in resources close to the network's edge, they are more efficient than running the service randomly. As a result, the proposed method performs better in reducing the delay of the program loop.

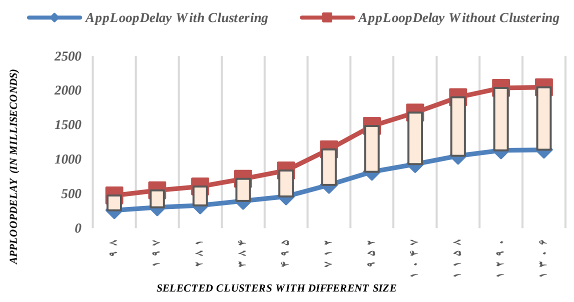

Fig. 5. Comparison of AppLoopDelay in two modes of clustering and non-clustering of services

According to the results obtained in Figure 5 and comparing the results in both cases, the proposed algorithm has improved the App Loop Delay parameter in the service clustering mode compared to the service non-clustering mode. On average, the application loop delay of applications is an 18.5% reduction.

5-3-2-Experiment 2

This experiment aims to investigate the amount of delay caused by the processing of tuples for services in both clustered and non-clustered modes during the simulation process. According to Figure 6, the average processing delay of tuples in the clustering mode of services compared to the non-clustering mode is acceptable by increasing the number of services in the selected cluster and has experienced an improvement of 17.78 percent. Due to the optimal placement of the modules, the increase in the delay in the processing of tuples in the proposed method has been less in the clustering mode than in the non-clustering mode. Therefore, the processing delay of tuples is significantly increased with the increase of the number of modules in non-clustering mode.

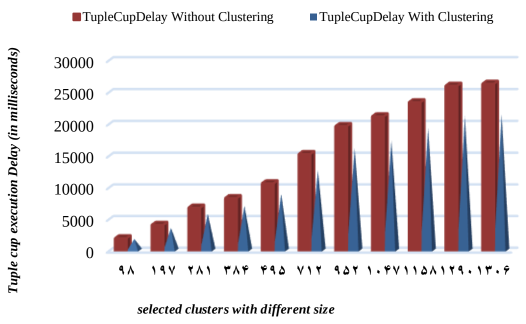

Fig. 6. Comparison of tuple cup execution delay in two modes of clustering and non-clustering of services

5-3-3-Experiment 3

In this test, the amount of network usage is calculated in two modes of clustering and non-clustering services during the simulation process. This parameter depends on the number of links in the fog network, and with the increase in the number of links, the size of the network usage increases. As shown in Figure 7, the use of the network in the service clustering mode performs better than the service non-clustering mode. Due to the optimal placement of the modules, there has been a significant reduction in the delay in creating communication links between the modules/services in the source and destination devices, and the proposed algorithm has experienced a 15.82% reduction.

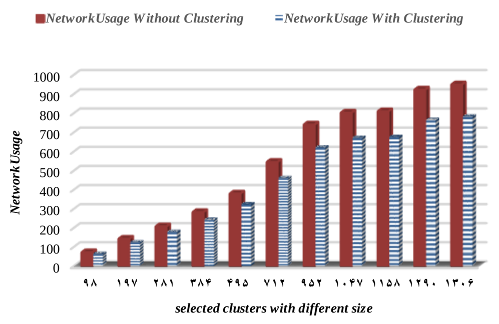

Fig. 7. Comparison of network usage in two modes of clustering and non-clustering of services

5-3-4-Experiment 4

In this section, the p values obtained from the statistical analysis for the evaluated parameters such as *AppLoopDelay*, Tuple cup execution delay, and Network usage are given in Table 9. A value of $p \leq 0.05$ indicates that H0 can be rejected with a confidence interval of 95%. In other words, p-values smaller than 0.05 show that the proposed method is better than service execution in non-clustering mode due to module/services loop delay, tuples processing delay, and network usage rate.

Table 9. Statistical analysis for the proposed algorithm

Metric	**Algorithms**	**P-value**	**Significance**
AppLoopDelay	Non-clustering mode:clusteredmode	P = 0.0028	extremely statistically
Tuple cup execution Delay	Non-clustering mode:clusteredmode	P = 0.0189	very statistically
Network usage	Non-clustering mode:clusteredmode	P = 0.0382	Statistically

6. Conclusion

We can determine the optimal service placement in the fog network based on the clustering results, minimize latency, and improve service responsiveness by considering the proximity between IoT devices and fog nodes. In addition, we can allocate services according to the availability of resources and the capacity of fog nodes and ensure the efficient use of resources.This article proposes a DLA-SPSQ method based on fuzzy clustering of services and distributed learning automata for optimal placement of services/modules in heterogeneous fog nodes. The DLA-SPSQ algorithm combines the FCS algorithm and the DLA-FMP algorithm. Fuzzy clustering of services, the optimal number of clusters, and the determination of high-priority clusters based on service quality criteria have been done using cluster evaluation criteria. The optimal placement of services in two modes, clustered and non-clustered, is based on the cost function provided by the DLA-FMP. This article proposes a method called DLA-SPSQ based on fuzzy clustering of services and distributed learning automata for optimal placement of services/modules in heterogeneous fog nodes. The DLA-SPSQ algorithm combines the FCS algorithm and the DLA-FMP algorithm. The FCS algorithm has done fuzzy clustering of services, the optimal number of clusters, and the determination of high-priority clusters based on service quality criteria. The optimal placement of services in two modes, clustered and non-clustered, is based on the cost function provided by the DLA-FMP algorithm. For the optimal placement of services, the topology of the fog network is modelled by a directed graph and mapped to a DLA. DLA performs the deployment of services from the edge nodes upwards in the fog topology with the cooperation of automatons in an optimal and hierarchical manner. In this search method, due to the use of the maximum capacity of the edge level fog nodes, the cost function (service execution delay), tuple processing execution delay, and network usage have decreased by 18.5%, 17.78%, and 15.82%, respectively. P-Value ≤ 0.05 in the results of Experiment 4 confirms the efficiency and improvement of the proposed algorithm in the state of clustering of services compared to the state of not clustering them.

future works:

Research areas related to the proposed topic that can be done in future works are:

- ✓ Creating multi-objective optimization models for optimal placement of services considering QoS requirements and resource constraints simultaneously.
- ✓ Research the placement of sustainable services considering renewable energy sources in fog nodes
- ✓ Investigating the impact of load balancing on service quality in fog nodes with heterogeneous workloads.
- ✓ Investigating security and privacy in the optimal deployment of QoS-based services.
- ✓ Creating criteria and standards for service quality evaluation in cloud computing environments.
- ✓ Investigate economic models for billing based on the QoS provided, such as premium pricing for higher service levels.
- ✓ Investigating the effect of network conditions on the performance of real-time analytical programs in fog environments.

References

[1] Y. Al Mtawa, A. Haque, and B. Bitar, "The mammoth internet: Are we ready?," *IEEE Access*, vol. 7, pp. 132894-132908, 2019.

[2] P. Maiti, B. Sahoo, A. K. Turuk, A. Kumar, and B. J. Choi, "Internet of Things applications placement to minimize latency in multi-tier fog computing framework," *ICT Express*, vol. 8, no. 2, pp. 166-173, 2022, doi: http://dx.doi.org/10.1016/j.icte.2021.06.004.

[3] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5595-5608, 2020, doi: http://dx.doi.org/10.1109/TCYB.2020.2989309.

[4] S. Gorikapudi and H. K. Kondaveeti, "A novel clustering model via optimized fuzzy C- means algorithm and sandpiper optimization with cycle crossover process in IoT," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 23, p. e7776, 2023.

[5] M. Hosseini Shirvani and Y. Ramzanpoor, "Multi-objective QoS-aware optimization for deployment of IoT applications on cloud and fog computing infrastructure," *Neural Computing and Applications*, vol. 35, no. 26, pp. 19581-19626, 2023.

[6] D. Zhao, Q. Zou, and M. Boshkani Zadeh, "A QoS-aware IoT service placement mechanism in fog computing based on open-source development model," *Journal of Grid Computing*, vol. 20, no. 2, p. 12, 2022.

[7] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185-1192, 2017.

[8] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service- aware approaches in fog computing," *International Journal of Communication Systems*, vol. 33, no. 8, p. e4340, 2020.

[9] H. K. Apat, B. Sahoo, and S. Mohanty, "A Quality of Service (QoS) aware Fog Computing model for intelligent (IoT) applications," in *2021 19th OITS International Conference on Information Technology (OCIT)*, 2021: IEEE, pp. 267-272.

[10] A. N. Al-Masri and M. Nasir, "A novel fuzzy clustering with metaheuristic based resource provisioning technique in cloud environment," *Fusion: Practice and Applications*, vol. 6, no. 1, pp. 08-8-16, 2021.

[11] M. Kupriyanov, I. Holod, and A. Shorov, "Fuzzy Clustering Based on Cloud and Fog Computing," in *2019 XXII International Conference on Soft Computing and Measurements (SCM))*, 2019: IEEE, pp. 1-5.

[12] F. Tavousi, S. Azizi, and A. Ghaderzadeh, "A fuzzy approach for optimal placement of IoT applications in fog-cloud computing," *Cluster Computing*, pp. 1-18, 2022.

[13] 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," *Computer Networks*, vol. 194, p. 108146, 2021, doi: http://dx.doi.org/10.1016/j.comnet.2021.108146.

[14] M. Ghobaei-Arani and A. Shahidinejad, "A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment," *Expert Systems with Applications*, vol. 200, p. 117012, 2022, doi: http://dx.doi.org/10.1016/j.eswa.2022.117012.

[15] F. Khosroabadi, F. Fotouhi-Ghazvini, and H. Fotouhi, "Scatter: Service placement in real-time fog-assisted iot networks," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, p. 26, 2021.

[16] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *2017 IEEE international conference on edge computing (EDGE)*, 2017: IEEE, pp. 17-24.

[17] 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 Processing Letters*, vol. 54, no. 3, pp. 1823-1854, 2022, doi: http://dx.doi.org/10.1007/s11063-021-10708-2.

[18] B. Natesha 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*, vol. 178, p. 102972, 2021, doi: http://dx.doi.org/10.4108/eai.22-2-2022.173492.

[19] 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," *Applied Artificial Intelligence*, pp. 1-34, 2022, doi: http://dx.doi.org/10.1080/08839514.2021.2008149.

[20] F. M. Calisto, N. Nunes, and J. C. Nascimento, "Modeling adoption of intelligent agents in medical imaging," *International Journal of Human-Computer Studies*, vol. 168, p. 102922, 2022, doi: http://dx.doi.org/10.2139/ssrn.4116048.

[21] M. Dadashi Gavaber and A. Rajabzadeh, "MFP: an approach to delay and energy-efficient module placement in IoT applications based on multi-fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 7, pp. 7965-7981, 2021, doi: https://link.springer.com/article/10.1007/s12652-020-02525-7.

[22] G. Baranwal and D. P. Vidyarthi, "FONS: a fog orchestrator node selection model to improve application placement in fog computing," *The Journal of Supercomputing*, vol. 77, no. 9, pp. 10562-10589, 2021, doi: https://link.springer.com/article/10.1007/s11227-021-03702-x.

[23] M. Masdari, A. B. Sangar, and K. Majidzadeh, "A Hybrid Multi-objective Algorithm for Imbalanced Controller Placement in Software-Defined Networks," *Journal of Network and Systems Management*, vol. 30, no. 3, pp. 1-54, 2022, doi: http://dx.doi.org/10.1007/s10922-022-09650-y.

[24] Z. M. Nayeri, T. Ghafarian, and B. Javadi, "Application placement in Fog computing with AI approach: Taxonomy and a state of the art survey," *Journal of Network and Computer Applications*, vol. 185, p. 103078, 2021, doi: http://dx.doi.org/10.1016/j.jnca.2021.103078.

[25] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1-35, 2020, doi: http://dx.doi.org/10.1145/3391196.

[26] E. Torabi, M. Ghobaei-Arani, and A. Shahidinejad, "Data replica placement approaches in fog computing: a review," *Cluster Computing*, pp. 1-29, 2022, doi: http://dx.doi.org/10.1007/s10586-022-03575-6.

[27] K. Narendra and M. Thathachar, "Learning Automata: An Introduction Prentice-Hall," *New Jersey*, 1989.

[28] M. A. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 32, no. 6, pp. 711-722, 2002, doi: https://doi.org/10.1109/tsmcb.2002.1049606.

[29] K. S. Narendra and M. A. Thathachar, "On the behavior of a learning automaton in a changing environment with application to telephone traffic routing," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 5, pp. 262-269, 1980, doi: https://doi.org/10.1109/TSMC.1980.4308485.

[30] Z. Anari, A. Hatamlou, and B. Anari, "Automatic Finding Trapezoidal Membership Functions in Mining Fuzzy Association Rules Based on Learning Automata," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 7, no. 4, 2022, doi: https://doi.org/10.1142/S0218001421590266.

[31] B. Anari, J. A. Torkestani, and A. M. Rahmani, "Automatic data clustering using continuous action-set learning automata and its application in segmentation of images," *Applied Soft Computing*, vol. 51, pp. 253-265, 2017, doi: https://doi.org/10.1145/3391196.

[32] B. Zarei and M. R. Meybodi, "Improving learning ability of learning automata using chaos theory," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 652-678, 2021, doi: https://doi.org/10.1007/s11227-020-03293-z.

[33] K. Asghari, M. Masdari, F. Soleimanian Gharehchopogh, and R. Saneifard, "A fixed structure learning automata- based optimization algorithm for structure learning of Bayesian networks," *Expert Systems*, vol. 38, no. 7, p. e12734, 2021, doi: https://doi.org/10.1111/exsy.12734.

[34] F. Farahbakhsh, A. Shahidinejad, and M. Ghobaei- Arani, "Multiuser context- aware computation offloading in mobile edge computing based on Bayesian learning automata," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4127, 2021, doi: https://doi.org/10.1002/ett.4127.

[35] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani, "Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8265-8284, 2021, doi: https://link.springer.com/article/10.1007/s12652-020-02561-3.

[36] N. Kumar, J.-H. Lee, and J. J. Rodrigues, "Intelligent mobile video surveillance system as a Bayesian coalition game in vehicular sensor networks: Learning automata approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1148-1161, 2014, doi: http://dx.doi.org/10.1109/TITS.2014.2354372.

[37] Y. Abofathi, B. Anari, and M. Masdari, "A learning automata based approach for module placement in fog computing environment," *Expert Systems with Applications*, vol. 237, p. 121607, 2024.

[38] U. Arora and N. Singh, "IoT application modules placement in heterogeneous fog–cloud infrastructure," *International Journal of Information Technology*, vol. 13, no. 5, pp. 1975-1982, 2021, doi: http://dx.doi.org/10.1007/s41870-021-00672-4.

[39] https://qwsdata.github.io/