TYPE Original Research
PUBLISHED 14 December 2023
DOI 10.3389/fcomp.2023.1293209

Machine learning-driven task scheduling with dynamic K-means based clustering algorithm using fuzzy logic in FOG environment

Muhammad Saad Sheikh[1]*, Rabia Noor Enam[1] and Rehan Inam Qureshi[2†]

[1]Computer Engineering Department, Sir Syed University of Engineering and Technology, Karachi, Pakistan, [2]Stanley College, Perth, WA, Australia

OPEN ACCESS

EDITED BY
Manlio Bacco,
National Research Council (CNR), Italy

REVIEWED BY
Saira Bano,
National Research Council (CNR), Italy
Achilles Machumilane,
National Research Council (CNR), Italy

*CORRESPONDENCE
Muhammad Saad Sheikh
msaad@ssuet.edu.pk

†PRESENT ADDRESS
Rehan Inam Qureshi,
Functional Analyst (Strategic Projects),
University of Southern Queensland,
Toowoomba, QLD, Australia

RECEIVED 16 September 2023
ACCEPTED 20 November 2023
PUBLISHED 14 December 2023

CITATION
Sheikh MS, Enam RN and Qureshi RI (2023) Machine learning-driven task scheduling with dynamic K-means based clustering algorithm using fuzzy logic in FOG environment. *Front. Comput. Sci.* 5:1293209. doi: 10.3389/fcomp.2023.1293209

Fog Computing has emerged as a pivotal technology for enabling low-latency, context-aware, and efficient computing at the edge of the network. Effective task scheduling plays a vital role in optimizing the performance of fog computing systems. Traditional task scheduling algorithms, primarily designed for centralized cloud environments, often fail to cater to the dynamic, heterogeneous, and resource-constrained nature of Fog nodes. To overcome these limitations, we introduce a sophisticated machine learning-driven methodology that adapts task allocation to the ever-changing Fog environment's conditions. Our approach amalgamates K-Means clustering algorithm enhanced with fuzzy logic, a robust unsupervised learning technique, to efficiently group Fog nodes based on their resource characteristics and workload patterns. The proposed method combines the clustering capabilities of K-means with the adaptability of fuzzy logic to dynamically allocate tasks to fog nodes. By leveraging machine learning techniques, we demonstrate how tasks can be intelligently allocated to fog nodes, resulting in reducing execution time, response time and network usage. Through extensive experiments, we showcase the effectiveness and adaptability of our proposed approach in dynamic fog environments. Clustering proves to be a time-effective method for identifying groups of jobs per virtual machine (VM) efficiently. To model and evaluate our proposed approach, we have utilized iFogSim. The simulation results affirm the effectiveness of our scheduling technique, showcasing significant enhancements in execution time reduction, minimized network utilization, and improved response time when compared to existing machine learning and non-machine learning based scheduling methods within the iFogSim framework.

KEYWORDS
machine learning, fog computing (FC), Internet of Things (IoT), symbiotic organism search algorithm (SOS), task scheduling, K-mean clustering algorithm, fuzzy logic, fog nodes (FD)

1 Introduction

The advent of the Internet of Things (IoT) has ushered in an era of pervasive data generation, enabling smart cities, industrial automation, healthcare monitoring, and a myriad of other applications (Kumar et al., 2019). However, the efficient processing and analysis of this vast volume of data at the edge of the network have posed significant challenges. Fog computing, an extension of cloud computing (Arooj et al., 2021), offers a promising solution by bringing computation and data storage closer to the data sources, thereby reducing latency, bandwidth usage, and enabling real-time decision-making. In this paradigm, a fundamental aspect that demands attention is the scheduling of tasks on fog nodes, which is pivotal in ensuring timely execution and resource optimization (Goniwada, 2022).

Fog computing introduces a dynamic and heterogeneous environment where resources, such as computing power and network bandwidth, can vary significantly across fog nodes. Moreover, the tasks to be executed often exhibit diverse characteristics, including varying execution times and real-time requirements (Guo and Chen, 2023). The challenge lies in orchestrating these resources efficiently to meet stringent task execution deadlines while optimizing resource utilization.

Traditional cloud computing scheduling algorithms are ill-suited for the fog computing landscape due to the inherent differences in resource proximity and the real-time constraints imposed by IoT applications. While centralized cloud environments benefit from ample resources and stable conditions, fog nodes operate on the edge, where resources are constrained, and environmental conditions are volatile.

One critical aspect of effective task scheduling in fog computing is the accurate prediction of task execution times. Unlike cloud data centers (Zhang et al., 2023) where resource performance is relatively stable, fog nodes are susceptible to resource fluctuations due to factors such as network congestion, device failures, and environmental conditions. Consequently, execution times for tasks can exhibit considerable variability.

Fog computing serves as an intermediary layer positioned between end users and cloud data centers. Its utility becomes particularly pronounced when dealing with applications demanding minimal latency and immediate responses, contingent on the origin of the data source. Within this stratum, a multitude of virtual servers can be seamlessly deployed to efficiently manage incoming requests.

Resource allocation, as described by Kreuzberger et al. (2023), is a methodical process involving the distribution of accessible resources to internet-based Cloud clients. The precise timing and sequence of these allocations hold paramount importance, as they play a pivotal role in optimizing the advantages derived from virtual server utilization. This strategic allocation has the potential to enhance system throughput without imposing undue charges on customers. Furthermore, ensuring the availability of resources for high-priority tasks is essential to prevent them from languishing at the end of the task queue. Neglecting this aspect could lead to inefficient utilization of virtual servers and potential financial losses for a company. Consequently, the prioritized allocation of resources for profit maximization emerges as a critical and promising research area.

Machine learning (ML) (Aqib et al., 2023; Li et al., 2023), a pivotal field, has witnessed remarkable progress across various academic domains. Numerous research endeavors have been undertaken to explore the utilization of machine learning in addressing challenges within the realm of fog computing. Recent years have seen a notable surge in the application of machine learning (ML) to enhance fog computing applications and provide a spectrum of fog-related services. These services encompass efficient resource management, security enhancement, latency reduction, energy conservation, and traffic modeling, among others. Within the diverse landscape of fog computing, various devices, sensors, and objects contribute copious amounts of data that necessitate processing. Real-time processing not only holds the potential to enhance operational efficiency but, in some scenarios, becomes imperative. To ensure optimal resource utilization, sensors, devices, and objects often engage in resource-intensive interactions (Aqib et al., 2023). Consequently, the management of resources in fog computing demands careful consideration and implementation (Alguliyev et al., 2023). This section of the study delves into investigations that harness ML algorithms for the purpose of resource management within the domain of fog computing.

In this paper, we introduce a novel Scheduling Algorithm designed for task allocation at the fog computing level. Our algorithm efficiently assigns tasks to virtual machines (VMs) responsible for executing the request/response model within the fog computing environment. To achieve this, we harness the K-Means clustering algorithm (Awad et al., 2023; You et al., 2023) with fuzzy logic (Ranjan and Sharma, 2023) for scheduling fog devices. The primary contributions of our work are as follows:

1. Introduction of K-means Clustering Scheduling enhanced with fuzzy logic within the fog computing paradigm.
2. Implementation of the proposed algorithm within the iFogSim simulator (Gupta et al., 2016; Saad et al., 2023).
3. Substantial reduction in execution time, response time and network usage signifying an improvement in operational efficiency.

The remaining parts of the paper are partitioned into the following hierarchy: Section 2 explores relevant literature, in Section 3 we discuss problem statement and proposed solution, Section 4 discuss application and model in detail, Section 5 elaborates suggested workflow, Section 6 presents our experimental results and comparative analysis, validating the effectiveness of our approach, Section 7 concludes the paper and summarizes future work.

2 Literature survey

The many scheduling methods now in use may be placed in one of three major categories: stochastic (Bhaumik et al., 2016; Guo, 2017), deterministic (Hosseinzadeh et al., 2023; Reddy and Sudhakar, 2023), or hybrid (Saif et al., 2023; Tran-Dang and Kim, 2023). Non-deterministic algorithms, often known as stochastic algorithms, are those that create a result depending on a degree of randomness and an objective function. Stochastic algorithms are also known as evolutionary algorithms. This category also contains algorithms that use heuristic and metaheuristic approaches. Heuristic algorithms find answers by trial and error (Saad et al., 2023), but they cannot ensure that the results will be the best possible solution. Metaheuristic approaches, on the other hand, are more generic. Then there are hybrid algorithms, which are the mixture of many algorithms from the aforementioned scheduling algorithm types. Deterministic algorithms derive the answer exclusively from the input and do not make use of chance or randomness in any way. A literature overview of task scheduling algorithms that use one of these three techniques is presented below in this section.

The field of fog computing is one that is now undergoing significant development. The research community is gradually beginning to pay greater attention to task scheduling in fog computing, despite the fact that there is not a great deal of research on the subject. In distributed systems, the metaheuristic method is often used to schedule tasks and handle resources (Jeyaraj et al., 2023). But not much study has been done on how scheduling methods can be used in fog computing architecture.

In the research carried out and presented by Li et al. (2023), many distinct strategies for the placement of services have been investigated in the context of cloud computing, and edge computing. The atomization idea and both simultaneous and sequential processing have been tried for service location. Gupta and Singh (2023) showed and talked about fog computer designs and other tools that work. They have told us more about organizing and allocating resources in the fog environment, as well as the different modeling tools that can be used for fog computing.

The Bee Life Algorithm was applied to the challenge of task scheduling in the research that Vispute and Vashisht (2023) presented to the scientific community. Both the entire amount of RAM and the execution speed of the CPU have been used as parameters. They compared their effort to the work done by GA and PSO in order to highlight how far things had come. For the purpose of allocation, a static method of task scheduling is used rather than a dynamic one. A cloud-fog infrastructure-based job scheduling approach that was developed by Agarwal et al. (2023) and was based on the genetic algorithm (GA) mechanism was also presented. Comparing the proposed method to the Bee Life algorithm revealed that the proposed method is superior. In this paper, the authors demonstrate efficacy using a limited data set. In Bakshi et al. (2023) by Skarlat et al., a technique for fog device-based service placement in fog colonies is developed. After determining a suitable service deployment sequence, this method optimizes the sequence using the GA method. However, the technique has a flaw in that they have only compared the developed technique to the GA and first fit techniques.

EPSO is a method of resource scheduling that was created by He and Bai (2023) for use in a fog computing network. In order to make the convex issues more manageable, it combined the PSO approach with the proximal gradient technique. In order to bring the local and global solutions closer together, the approach that was devised makes use of additional gradient parameters. Comparisons of total time and makespan metrics have been carried out in order to demonstrate the usefulness of the established approach. The newly discovered approach shortens the amount of time needed to complete a large number of jobs, but at the expense of an increased amount of energy usage.

In the research that was reported by Singhrova (2023), the fireworks method was used to schedule jobs in an environment that had a variety of different types of resources. In order to get the most effective scheduling sequences, it makes use of the components of the fireworks evolutionary approach. The methodology was evaluated in light of the genetic algorithm, and the researchers saw a rapid convergence. The fireworks were chosen using the tournament-based selection approach, however the algorithm did not have a cloud or fog environment built for it.

The approach of task scheduling known as the fireworks algorithm was presented in the work that was done by Shen et al. (2023). They have developed a detecting method for explosions caused by fireworks. In addition to this, the authors have implemented the task clustering mechanism, then proceeded to implement the method. Memory and CPU utilization indicators have been included in their load utilization model, however the time aspect has not been taken into consideration. Consumption of both energy and time has been regarded as the most important elements in the research carried out by Hosseinzadeh et al. (2023). The authors have successfully resolved the issues with scheduling and commitment by using the fireworks method.

The implementation of cloudlets in mobile edge computing has reportedly been successfully completed, as stated in Abadi et al. (2023) research. The placement was carried out with a focus on minimizing times and taking total time of ownership into account. Mishra and Chaturvedi (2023) presented the methodology for cloud-based work distribution as their contribution. This approach used two distinct tactics in order to maximize performance while also reducing the amount of inappropriate job allocation. They have improved the scheduling of cloud-based resources as well as the load allocation across those resources.

The authors Atiq et al. (2023) established a novel paradigm for resource allocation and task scheduling that concurrently takes into consideration three separate criteria. These elements are referred to as the resource balancing factor, task completion time, and throughput. They have made use of the Lyapunov drift methodology in order to optimize and cut down on the amount of time spent waiting for tasks, but it does not include a modern optimization method. Kumar and Karri (2023) suggested a way in which scheduling is intended to function on different parameters by integrating the allocation and selection of resources in a fog environment. In this approach, scheduling is meant to work on multiple parameters. The suggested approach selects a methodology for rapidly completing tasks in order to lower the task loading factor; however, it does not include an optimization technique, which would be helpful in improving the method's currently poor outcome.

Numerous researchers have tried to use various numbers of goals to tackle the multi objective optimization issue of the workflow applications. To plan the workflow tasks across the available resources, a hybrid meta-heuristic GA-PSO method is suggested in this research (Yin et al., 2023). The proposed method uses the properties of heterogeneous tasks and nodes but unable to accomplish multi-objective optimization.

When the total number of repetitions of the process is raised above a certain threshold, the results generated by the algorithms that are based on IGA are, in a nutshell, superior than those produced by other algorithms.

Increasing the total number of repetitions of the process, on the other hand, will result in the IGA-based meta-heuristic algorithm taking a longer amount of time to arrive at the best possible answer (Meng et al., 2023; Sun et al., 2023). Besides that, the IPSO-based meta-heuristic algorithms provide superior outcomes in a shorter amount of time when compared to the other methods. However, since the PSO-based algorithms converged on a solution so quickly, the reliability of the results could be jeopardized as a result of this. This rapid convergence may result in the algorithms being stuck in the locally optimum solution (Gomathi et al., 2023; Pirozmand et al., 2023).

In the research presented in Chaplot et al. (2023), the Ant Colony Algorithm is employed for the purpose of scheduling grouped tasks onto virtual machines. This method encompasses three crucial steps in its execution: the initial grouping of jobs based on factors like time and time, the subsequent prioritization of tasks, and finally, the selection of the most suitable virtual machine (VM) through the utilization of the Ant Colony Algorithm. This particular approach is further implemented and simulated within the iFogSim framework. The notable outcome of this integration is an enhancement in the simulation results, achieved while consuming minimal energy resources. The utilization of the Ant Colony Algorithm in the context of task scheduling contributes significantly to the optimization of resource allocation and the overall performance of the iFogSim simulation environment.

In the paper presented by the authors in Alhijawi and Awajan (2023), they introduce an innovative scheduling approach that merges knapsack optimization with Symbiotic Organism Search (SOS). Additionally, they discuss a distinctive application focused on identifying health issues among elderly individuals through the use of Elderly Human Activity Detection (EAHD) in smart home environments. The SOS process involves three distinct stages. The first stage is Mutualism, which is followed by the random selection of organisms. This combination of knapsack optimization and SOS introduces a novel approach to task scheduling, particularly within the context of health monitoring in smart homes for the elderly population. The second is commensalism, and the third is parasitism. This formula, 1/(TUC+BW), is used to compute fitness value at each phase. Two case studies employing iFogsim DCNS and EAHD that outperform FCFS and Knapsack algorithms.

The GKS algorithm is a scheduling approach proposed in Cerf et al. (2023) that uses knapsack to deploy modules on fog devices. Using knapsack approaches, PEs are allocated to modules. In order to fill a knapsack with extra items (modules), the GKS algorithm is utilized to maximize profit while reducing weight. In iFogsim, this strategy outperforms energy consumption-based algorithms.

The researchers presented in their work a knapsack-based strategy to scheduling concurrent video transfers in a cloud context, with an emphasis on minimizing the completion time, known as the Least Completion Time (MCT) (Paparella et al., 2023). They used the "maximum min" strategy, which involves selecting the most powerful machines in the cloud infrastructure and allocating them to a certain amount of video segments. The MCT method was then used to schedule these portions. The study findings found that, when both execution time and segment count were considered, the "maximum min" approach beat the MCT technique. This shows that the "maximum min" technique is more efficient and effective, resulting in shorter execution durations and a lower total segment count.

Many fields are now using machine learning. In domains like speech recognition, data classification, and face recognition, the robots are intelligently functioning. RL is a technique for finding the best answer without the assistance of the outside environment (Kober et al., 2013). An essential algorithm for machine learning is found in the artificial intelligence (AI) branch known as reinforcement learning (RL). Until the session is over, this process is repeated (Roy et al., 2017). Due to the quick growth of AI and RL as well as the drawbacks of other task scheduling techniques, RL is integrated with AI to solve optimization challenges through task scheduling. SARSA, TD-learning, and Q-learning are three different traditional RL Algorithms (Peng et al., 2015). The Q-learning method is the most effective task scheduling algorithm (Wei et al., 2017). One method for leveraging the Markov decision process (MDP) to solve different issues when all the information is not available is Q-learning.

The research cited in reference Alexandrescu (2023), Alguliyev et al. (2023), Aqib et al. (2023), and Jamshed et al. (2023), upon which I have anchored my foundational problem statement for this entire study, has a notable limitation. Specifically, it fails to account for various operational time parameters, among others. Furthermore, the research does not incorporate the utilization of fuzzy logic to enhance the adaptability of task allocation within fog nodes across different degrees of membership.

The proposed algorithm is compared with Symbiotic Organism Search (SOS) (Hosseinioun et al., 2022) and k-means (Mtshali et al., 2019) algorithm. Symbiotic Organism Search (SOS) is a metaheuristic algorithm inspired by the symbiotic relationships between organisms in an ecosystem. Symbiotic Organism Search (SOS) can be used in task scheduling in fog computing to find optimal task assignments to fog nodes, while minimizing certain objectives such as makespan, cost, and energy consumption. SOS can be slow to converge, especially for complex problems as well as sensitive for initial population. The K-means algorithm can be used to group tasks together based on their characteristics such as task length and resource requirements. Once the tasks have been grouped, they can be assigned to fog nodes based on the resources available at each node. For example, tasks that require a lot of resources can be assigned to fog nodes with more resources, while tasks that have a tight deadline can be assigned to fog nodes that are closer to the data source. K-means is suitable for scenarios where data points naturally belong to distinct clusters and there is little ambiguity in cluster assignments.

3 Problem statement

Use of k-mean algorithm in task scheduling is only effective when each data point is assigned exclusively to one cluster. It uses crisp, binary membership assignments where a point belongs to a cluster or it doesn't. K-means is suitable for scenarios where data points naturally belong to distinct clusters and there is little ambiguity in cluster assignments.

But when there is an uncertainty or ambiguity in assigning tasks to fog nodes, when there is a requirement that tasks can be partially allocated to multiple fog nodes in order to take benefit from the resources of multiple nodes simultaneously, when in fog computing environments, the workload and resource availability of fog nodes fluctuate, when QoS requirements are not strictly binary but have varying degrees of importance or compliance, when task scheduling involves optimizing multiple conflicting objectives, such as minimizing energy consumption while maximizing task execution speed or reliability, when fog nodes have diverse capabilities, including computational power, memory, and network bandwidth, when the fog computing system needs to adapt rapidly to changing task requirements or node conditions, in all these scenario k-mean clustering will not give optimal results.

3.1 Proposed solution

In order to get optimal results, use of K-means clustering using fuzzy logic is valuable in fog computing scenarios where there is uncertainty, variability, and ambiguity in task scheduling and assignment. It uses soft, probabilistic assignments with degrees of membership. It enables flexible task allocation decisions that consider partial membership of tasks to fog nodes and varying degrees of suitability, helping to optimize resource utilization and meet diverse QoS requirements.

4 Application and system models

Fog computing, a variant of cloud computing, operates applications on fog devices situated between end-user devices and the distant cloud. This architectural approach capitalizes on the benefits of both cloud and edge computing, making it particularly advantageous for scenarios involving distributed data and low-latency requirements. At the foundational layer of this architecture, IoT sensors play a pivotal role in the reception and transmission of data via gateways to the higher layers. Concurrently, actuators are responsible for system control tasks. Fog computing, as a practical implementation, empowers edge devices to preprocess and analyze data, enhancing overall system efficiency. It's important to note that each fog network application boasts a distinct topology. To facilitate the creation of custom and predefined topologies, the iFogSim simulator offers a user-friendly Graphical User Interface (GUI) module (Gupta et al., 2016). This GUI empowers users to incorporate various elements such as sensors, actuators, fog, cloud components, and connection elements into their topological designs.

To illustrate the practical application of these concepts, we utilize iFogSim in a case study centered around car parking, demonstrating the creation and utilization of a new topology for this specific scenario. Other modules within the simulator are designed to interpret and execute these customized topologies. Fog computing stands as a pivotal enabler across several critical domains within IoT applications, such as smart car parking system.

In the context of fog computing, innovative solutions emerge. For instance, in the realm of traffic management, fog computing can be harnessed to facilitate dynamic road access based on real-time cues, such as flashing lights. This technology adeptly identifies pedestrians and cyclists while accurately gauging the speed and proximity of oncoming vehicles. Consequently, when motion is detected, sensor-based lighting systems are triggered, and conversely, they switch off when activity ceases (Gupta et al., 2016). Smart traffic lights, a quintessential fog computing component, can be envisaged as fog nodes that seamlessly communicate with one another, effectively transmitting warning signals to nearby vehicles. This collaboration draws upon various communication technologies, including zigbee, 4G, smart traffic signals, and roadside infrastructure, which collectively optimize interactions between fog computing entities and vehicular access points (Gupta et al., 2016).

This case study revolves around a distributed monitoring camera system that spans multiple domains, including healthcare, transportation, security, and manufacturing (Kumar et al., 2019). Within this context, the application model comprises five distinct functions:

1. Raw Video Processing: The system processes raw video feeds to identify specific objects and detect movements occurring in front of the cameras.
2. Object Tracking: An object tracking module is employed to calculate configurations necessary for effective monitoring.
3. PTZ Configuration: The Pan-Tilt-Zoom (PTZ) setup is utilized to adjust the camera's position, enhancing its surveillance capabilities. Both physical cameras and actuators play a role in this process.
4. User Interface Interaction: The system includes a user interface component responsible for transmitting relevant information about tracked objects to the user's device.

The physical topology of this case study, denoted as "case study-A", is represented by the DCNS video camera application architecture (Gupta et al., 2016), as illustrated in Figure 1. All components enclosed within the dotted line are integral to this architectural setup, with fog devices contained within a designated box. "M1" denotes Module 1 within the architecture.

5 Proposed algorithm

5.1 Number of clusters selection

In the scenario where you have only one objective, which is sending images as data or tasks with varying number of instructions, input and output file size as task parameters and with different CPU length (500, 2,000) in MIPS, with different RAM (500, 2,000) in MB, and with different uplink and download bandwidth as fog nodes parameters, the number of clusters would be 9. The reason for this is that the CPU length, RAM, and bandwidth are the three factors that are affecting the execution time, response time and network usage of the tasks. Since the number of instructions, input and output file size can vary, the CPU length, RAM, and bandwidth requirements of the tasks can also vary. The nine clusters could be:

Cluster 1: Fog nodes with CPU lengths in the range (500, 1,000) MIPS, RAM in the range (200, 1,000) MB, and bandwidth in the range (10, 100) Mbps (Low CPU, low RAM, low bandwidth).

Cluster 2: Fog nodes with CPU lengths in the range (1,000, 1,500) MIPS, RAM in the range (1,000, 1,500) MB, and bandwidth in the range (10, 100) Mbps (Medium CPU, medium RAM, low bandwidth).
Cluster 3: Fog nodes with CPU lengths in the range (1,500, 2,000) MIPS, RAM in the range (1,500, 2,000) MB, and bandwidth in the range (10, 100) Mbps (High CPU, high RAM, low bandwidth).
Cluster 4: Fog nodes with CPU lengths in the range (500, 1,000) MIPS, RAM in the range (200, 1,000) MB, and bandwidth in the range (100, 1,000) Mbps (Low CPU, low RAM, medium bandwidth).
Cluster 5: Fog nodes with CPU lengths in the range (1,000, 1,500) MIPS, RAM in the range (1,000, 1,500) MB, and bandwidth in the range (100, 1,000) Mbps (Medium CPU, medium RAM, medium bandwidth).

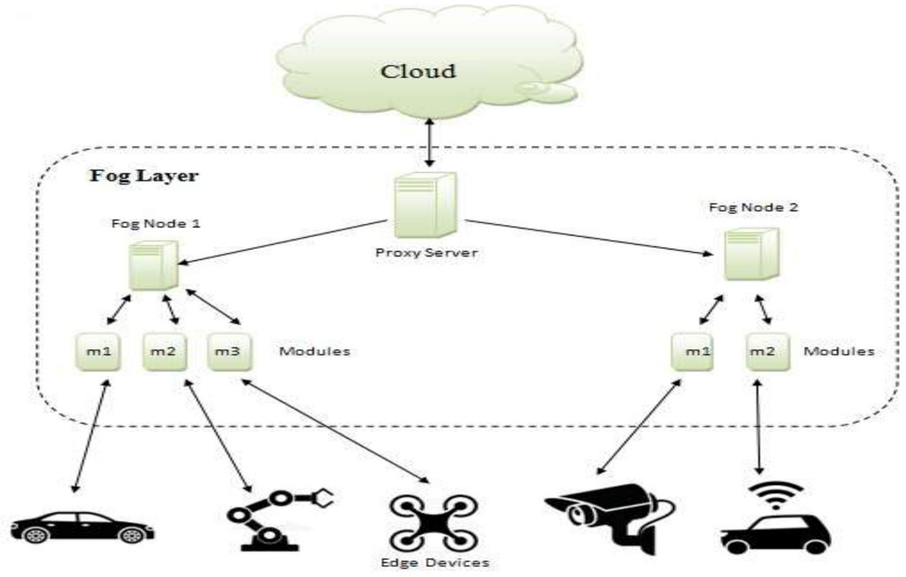

FIGURE 1
Case study A-video surveillance camera.

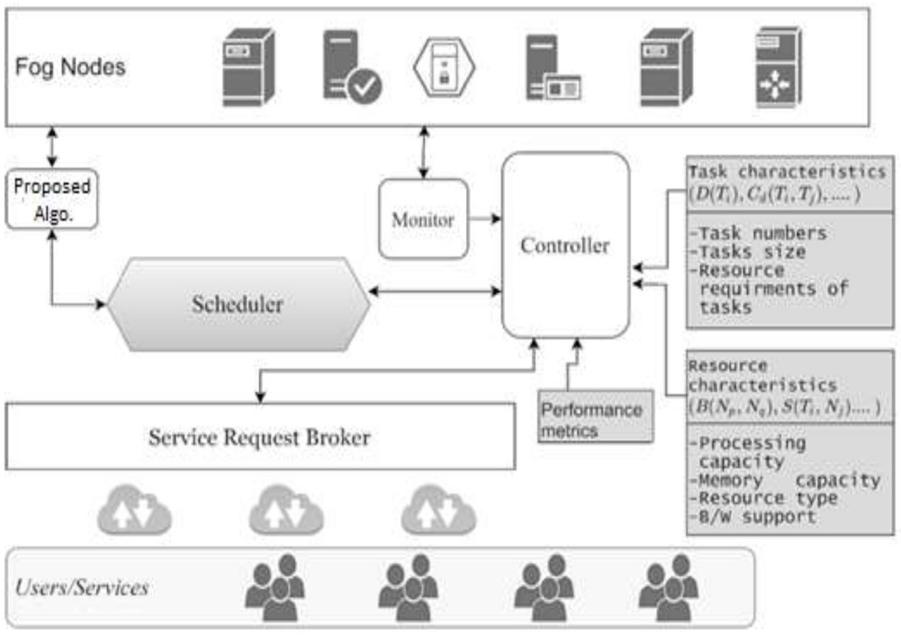

FIGURE 2
iFogSim application operational sequence.

```
Input:
- fogNodes: Array of fog nodes, each with
  attributes (e.g., execution time, response
  time, bandwidth)
- tasks: Array of tasks, each with attributes
- numClusters: Number of fog clusters for K-means
  clustering
- fuzzinessFactor: Fuzziness factor for fuzzy
  logic (greater than 1)
- maxIterations: Maximum number of iterations for
  K-means and fuzzy logic
Output:
- taskAssignments: An array indicating which fog
  node each task is assigned to
1. Apply K-means clustering to the fog nodes with
   the specified number of clusters
   (numClusters). This step results in fog
   clusters and centroids.
2. Initialize an empty array taskAssignments of
   size equal to the number of tasks.
3. Repeat the following steps until convergence
   or the maximum number of iterations
   (maxIterations) is reached:
  a. Calculate membership degrees for each task
     to fog clusters based on fuzzy logic.
    - For each task in tasks:
     i. Calculate the membership degrees to each
        fog cluster using the fuzzy logic equation
        with attributes (e.g., execution time,
        response time, bandwidth).
  b. Update the fog clusters' centroids based on
     the tasks' membership degrees.
    - For each fog cluster:
     i. Calculate the new centroid based on the
        weighted sum of task attributes and their
        membership degrees.
4. For each task in tasks:
  a. Initialize an empty array fogClusterScores
     for fog cluster suitability scores, one for
     each fog cluster.
  b. For each fog cluster:
     i. Initialize a variable totalScore to 1.0.
     ii. For each attribute (e.g., execution time,
         response time, bandwidth):
    - Calculate the combined membership degree for
      each fuzzy set associated with the attribute
      based on the task's fuzzy attribute values.
    - Update the totalScore by multiplying it with
      the combined membership degree.
     iii. Calculate the overall suitability score
          for the fog cluster by considering all
          attributes.
     iv. Store the suitability score in the
         fogClusterScores array.
     c. Determine the fog cluster with the highest
        suitability score for the task.
      - Select the fog cluster index with the
        maximum score in the fogClusterScores array.
      d. Assign the task to the fog cluster with
         the highest suitability score.
5. Repeat steps 4a to 4d for all tasks in tasks.
6. Assign tasks within each fog cluster to the
   specific fog node (centroid) of that cluster.
7. Return the taskAssignments array, which
   indicates which fog node each task is assigned
   to based on K-means clustering and fuzzy
   logic.
End Algorithm
```

Algorithm 1. K-means clustering and fuzzy logic.

Cluster 6: Fog nodes with CPU lengths in the range (1,500, 2,000) MIPS, RAM in the range (1,500, 2,000) MB, and bandwidth in the range (100, 1,000) Mbps (High CPU, high RAM, medium bandwidth).

Cluster 7: Fog nodes with CPU lengths in the range (500, 1,000) MIPS, RAM in the range (200, 1,000) MB, and bandwidth in the range (1,000, 10,000) Mbps (Low CPU, low RAM, high bandwidth).

Cluster 8: Fog nodes with CPU lengths in the range (1,000, 1,500) MIPS, RAM in the range (1,000, 1,500) MB, and bandwidth in the range (1,000, 10,000) Mbps (Medium CPU, medium RAM, high bandwidth).

Cluster 9: Fog nodes with CPU lengths in the range (1,500, 2,000) MIPS, RAM in the range (1,500, 2,000) MB, and bandwidth in the range (1,000, 10,000) Mbps (High CPU, high RAM, high bandwidth).

5.2 Fuzzy logic in task allocation

In task allocation for fog computing, fuzzy logic is used to assign tasks to fog nodes with varying degrees of membership. This approach allows for more flexibility and adaptability when deciding which fog node is best suited for a given task. Here's an expanded explanation with equations:

Step 1—Fuzzy sets and membership functions: In fuzzy logic, attributes are represented as fuzzy sets, where each set is characterized by a membership function. These membership functions determine the degree to which an element belongs to a set. In task allocation, each fog node and task attribute (e.g., execution time, response time, and bandwidth) can be represented as fuzzy sets. The membership functions, denoted as $\mu(x)$, assign a membership degree between 0 and 1 for each element x. For example, if we have a task with an execution time attribute, we can define a fuzzy set "LowExecutionTime" with a membership function μ(LowExecutionTime), where μ(LowExecutionTime)(x) represents the degree to which the execution time of the task is "low".

```
Input:
- MaxAreas: Maximum number of geographic areas
- MaxCameras: Maximum number of surveillance
  cameras
- FogDeviceParams: Parameters for Fog Devices
  (node name, MIPS, RAM, Storage, max BW, min BW,
  Busy power, idle power)
- Application: Modules, Edges, Tuples, Workflow
- MaxFogDevices: Maximum number of fog devices
Output:
- Energy cost and allocation results
1. Initialize a Fog Broker.
2. Create an Application with Modules, Edges,
   Tuples, and Workflow:
  a. Define the application structure, including
     modules, edges connecting them, tuples, and
     the workflow.
3. For i = 1 to MaxAreas (representing
   geographic areas):
  a. For j = 1 to MaxCameras (representing
     surveillance cameras within each area):
    i. Create a Fog Device with parameters:
     - Node name
     - MIPS (Million Instructions Per Second)
     - RAM (Random Access Memory)
     - Storage
     - Maximum available bandwidth (max BW)
     - Minimum required bandwidth (min BW)
     - Busy power consumption
     - Idle power consumption
    ii. End for
  b. End for
4. Initialize a module mapping.
5. Submit the Application to the Fog Broker.
6. Start the iFogSim simulation.
7. For i = 1 to MaxFogDevices:
  a. Add Modules to Fog Device(i).
8. Perform K-Means Clustering and Fuzzy Logic
   with Tuples and Fog Devices:
  a. Implement the enhanced k-means cluster
     scheduling algorithm, taking into account
     tuple characteristics and available fog
     devices.
9. Allocate Processing Elements (PE) to Modules
   using AppModuleAllocationPolicy:
  a. Implement the allocation policy to
     efficiently assign PEs to modules.
10. Update the Energy Cost based on device power
    consumption, considering execution time,
    response time, and bandwidth usage.
11. Stop the iFogSim simulation.
12. Output the Energy Cost and Allocation
    Results.
Main Program:
1. Initialize parameters and input data.
2. Execute the Fog Broker and iFogSim simulation
   algorithm to optimize task allocation based
   on execution time, response time, and bandwidth
   constraints.
```

Algorithm 2. DCNS with K-means clustering and fuzzy logic.

Step 2—Membership degree calculation: To allocate tasks to fog nodes, we calculate the membership degree of each task to each fog node based on their attributes. This degree represents how suitable a fog node is for executing a particular task. The membership degree μ(A)(x) of an element x belonging to fuzzy set A can be calculated using various methods, such as the Gaussian function, triangular function, or trapezoidal function. Here's a general equation using a Gaussian membership function:

$$\mu(A)(x) = \exp(-0.5^{*}((x-c)/\sigma)^{2})$$

- A represents the fuzzy set (e.g., "Low Execution Time").
- x is the value of the attribute (e.g., execution time of a task).
- c is the center of the fuzzy set (representing a typical value for that set).
- σ (sigma) controls the spread or width of the membership function.

Step 3—Multiple attributes and fuzzy rules: Task allocation often considers multiple attributes, such as execution time, response time, and bandwidth. Each attribute can have its fuzzy sets and membership functions.

Fuzzy rules can be defined to determine how these attributes collectively affect the membership degree. For example, a fuzzy rule might state: "If execution time is low AND response time is fast, then the membership degree for fog node A is high."

Step 4—Aggregation of membership degrees: Once membership degrees for various attributes are calculated, they are aggregated to determine the overall membership degree of a task to a fog node. This aggregation can be done using operators like the minimum (AND) or maximum (OR) operators, depending on the inference method used.

5.3 The enhanced K-means algorithm

The proposed approach enhances the traditional K-means algorithm with fuzzy logic to perform task scheduling in fog computing environments. The key steps of this enhanced K-means algorithm are as follows:

Step 1—Initialization: Initialize cluster centroids (representing fog nodes) and set the fuzziness parameter.
Step 2—Membership calculation: Calculate the membership degrees for each task with respect to each fog node based on criteria such as proximity, resource availability, and historical performance.
Step 3—Centroid update: Update the fog node centroids based on the weighted average of the tasks' characteristics and their membership degrees.

TABLE 1 Task input characteristics.

Parameters	Units	Value
No. of instructions	Instructions	$(1, 100)*10^9$
Size of input file	MB	(10, 100)
Size of output file	MB	(10, 100)

TABLE 2 Fog node characteristics.

Parameters	Units	Fog node values
CPU Length	MIPS	(500, 2,000)
Ram	MB	(500, 2,000)
Uplink bandwidth	Mbps	(10, 10,000)
Download bandwidth	Mbps	(10, 10,000)

TABLE 3 Experiment parameters.

Experiment	Purpose	Data input parameters	Fog node parameters
1	Heterogenous task and dynamic nodes	(100, 100, 800)	40

Step 4—Convergence check: Iterate steps 2 and 3 until convergence criteria are met.

5.4 Simulation workflow

iFogsim is a Java-based simulator designed for the development and testing of fog computing scenarios, topologies, and applications (Gupta et al., 2016). The workflow within iFogsim is depicted in Figure 2. Initially, the FogBroker class is responsible for creating the fog computing environment. Subsequently, the createApplication method is employed to establish a case study or any application to evaluate the performance of fog nodes, sensors, and actuators within the fog environment. The createFogDevice method is utilized to generate a specified number of fog devices, each with distinct attributes and capacities.

These attributes encompass essential hardware details like node name, MIPS (Million Instructions Per Second), RAM (Random Access Memory), uplink and downlink bandwidth, level, ratePerMips, busyPower, and idlePower. Upon application submission, our proposed scheduling algorithm, termed: "K-Means Clustering and Fuzzy Logic", is invoked through the Clustering (Tuples, FD) method. This method clusters tasks/tuples in accordance with the approach outlined in Algorithm 1. Subsequently, it assigns virtual machines (VMs) to the created clusters, with the appAllocationPolicy class responsible for handling the scheduling as shown in Algorithm 2. Following the scheduling phase, the simulation incorporates updates pertaining to execution times, response time and network usage. The simulation results are then presented for further analysis and evaluation.

TABLE 4 Actual execution times, response time, and network utilization.

No. of data input (cameras)	DCNS SOS	DCNS SOS	DCNS SOS	DCNS K-means clustering	DCNS K-means clustering	DCNS K-means clustering	DCNS K-means clustering using fuzzy logic	DCNS K-means clustering using fuzzy logic	DCNS K-means clustering using fuzzy logic
	Execution time (ms)	Network usage (kb)	Response time (ms)	Execution time (ms)	Network usage (kb)	Response time (ms)	Execution time (ms)	Network usage (kb)	Response time (ms)
100	20,432	10,678	1,557.14	15,851	6,582	817.37	10,852	5,583	408
200	24,720	12,048	4,448.42	14,410	6,872	3,422.93	11,209	7,572	1,701
300	21,681	11,030	10,749.93	14,812	6,648	8,042.06	11,012	6,448	4,156
400	26,606	11,357	19,924	17,738	8,052	14,310	13,450	6,578	7,345
500	26,357	11,202	37,520.06	16,849	7,077	22,506.54	13,105	6,410	11,255
600	22,126	11,316	45,202	14,833	6,655	26,402	10,959	5,675	13,401
700	20,387	10,072	54,121	16,466	7,340	32,306	1,112	5,400	16,351
800	20,221	10,010	53,125	14,541	7,101	32,012	1,011	5,109	15,153

In Algorithm 1, we first apply K-means clustering to the fog nodes to group them into clusters with centroids. Then, we use fuzzy logic to calculate membership degrees for tasks to each fog cluster, update the cluster centroids based on tasks' membership degrees, and finally assign tasks to fog clusters. Subsequently, tasks within each fog cluster are assigned to the specific fog node (centroid) of that cluster. This hybrid approach leverages both K-means clustering and fuzzy logic to make informed task allocation decisions in fog computing environments.

In Algorithm 2, we initiate the creation of the fog broker and the application. For every camera and specific area, a fog device is generated. These applications are then introduced to the fog broker alongside the creation of fog devices. Subsequently, the application is incorporated into the fog broker during the FD creation phase. The mapping of modules takes place, followed by the commencement of iFogSim, which includes the scheduling of modules for VM allocation through Clustering (Algorithm 1).

6 Experiment and results

This research utilized the iFogsim library for conducting simulations. iFogsim, a Java-based library, comprises modules and classes specifically designed for simulating fog computing scenarios

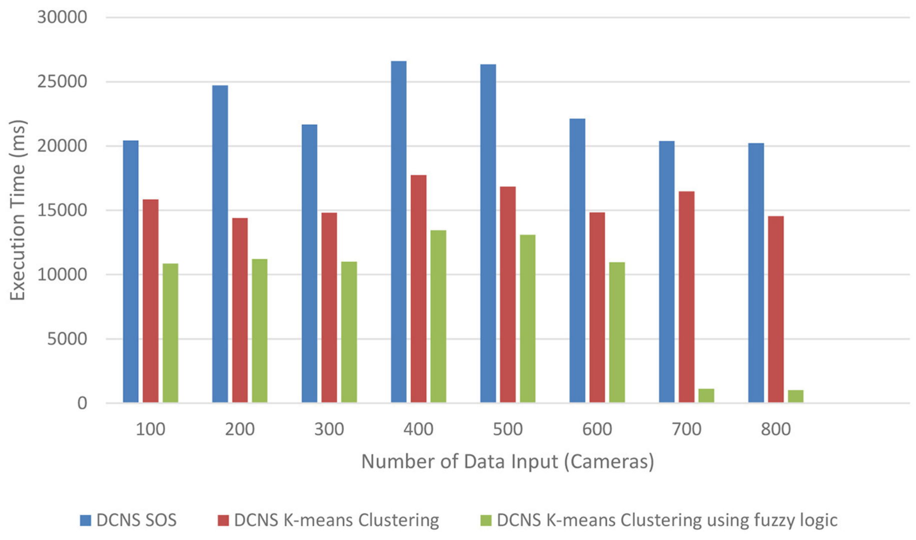

FIGURE 3
Comparison of execution times (in milliseconds) between DCNS (SOS), DCNS K-means clustering and the proposed DCNS K-means clustering using fuzzy logic.

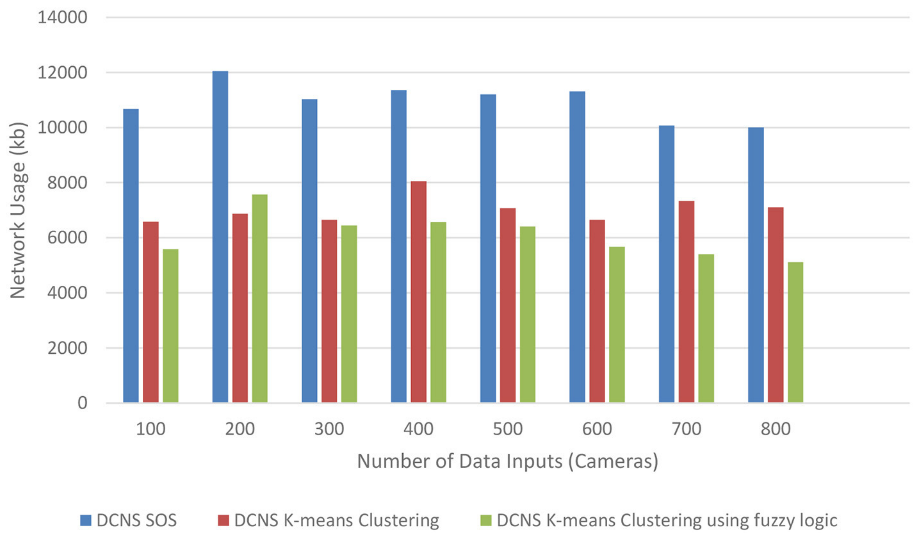

FIGURE 4
Comparison of network usage (in kilobytes) between DCNS (SOS), DCNS K-means clustering and the proposed DCNS K-means clustering using fuzzy logic.

(Gupta et al., 2016). Users familiar with the Cloudsim library will find the iFogsim package, along with its associated classes, essential for their work. To execute the program successfully, a computer with the following specifications is required: an Intel Core i5 processor, a minimum of 3 gigabytes of RAM, and the Microsoft Windows 10 operating system. For implementing the new scheduling algorithm, we will employ two distinct case studies as part of the experimentation process. Tables 1, 2 shows the initialization values for AppModule and FogDevice used as iFogsim entities.

The experiment parameters are summarized in Table 1, which shows that for experiment 1 we have created eight scenarios and in every scenario the number of data input (cameras) are increased with the difference of 100 and throughout this whole experiment, number of fog nodes are fixed to 40 and we got the results as shown in Table 4.

To assess the effect of the proposed method on the workflow scheduling issue relative to existing algorithms, we conducted comprehensive tests on real-world workflow application that is efficient car parking using the simulation settings in terms of number of data input and fog node characteristics listed in Table 3. These factors were used to determine the characteristics of the fog node and workflow application utilized in the research.

We devised a scenario in which high-definition, intelligent cameras photograph parking spots. Following that, the images are sent to the fog node. The fog node analyses the pictures to assess the condition of the parking slot and displays parking space visuals on a Wi-Fi-connected smart LED attached to the fog node. The link between the fog nodes and the cloud server is established through a proxy server. In the simulation, we established variables for parking areas and the number of cameras. In our experimental situation, we established eight parking lots. One hundred to eight hundred cameras were originally deployed to each parking lot to gather photos of the parking area.

It is very important to point out that we produced at least one fog node for each individual region and later on we increased the number of fog nodes in order to analyze the result gathered from different settings. We increased the number of cameras so that we could analyze the data gathered from a variety of settings and to evaluate the effects on the execution time, response time and network usage use in a fog node.

We conducted a total of eight simulation runs for the two scenarios (differing in the number of cameras) and three methodologies (Symbiotic Organism Search SOS, K-means Clustering and K-means Clustering using fuzzy logic) within the DCNS (Gupta et al., 2016) framework. Our comparison primarily relies on identifying the best results achieved under identical configurations for the case studies. Table 4 presents the actual execution time, response time and network utilization for three case studies. In the provided Table 4, it is evident that our proposed scheduling method, DCNS K-means Clustering using fuzzy logic, proves to be more effective compared to the existing scheduling methods that is DCNS (SOS) and DCNS K-means Clustering.

In Table 4, we have gathered the results of the DCNS case study, collected from eight simulation runs using iFogsim. These results include the execution time in milliseconds (ms), response time in milliseconds (ms) and network usage in kilobytes (KB). To facilitate a comparison between the existing DCNS (SOS) and DCNS K-means Clustering and the proposed K-means Clustering using fuzzy logic scheduling method.

Figure 3 represents the execution time comparison between DCNS with the existing scheduling (SOS), K-means Clustering and the proposed K-means Clustering using fuzzy logic Scheduling. It is evident that DCNS proposed K-means Clustering using fuzzy logic Scheduling exhibits significantly lower execution times compared to DCNS with the existing scheduling (SOS) and K-means Clustering scheduling.

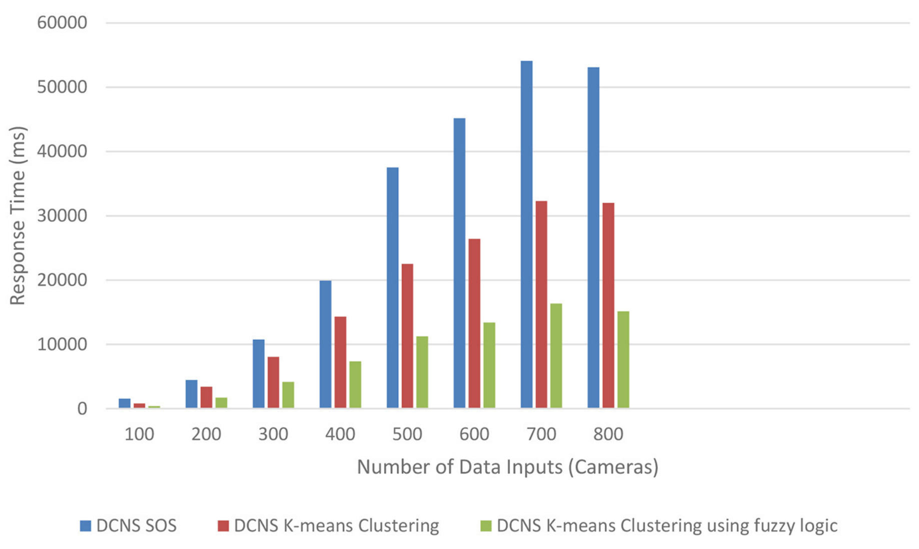

FIGURE 5
Comparison of response time (in milliseconds) between DCNS (SOS), DCNS K-means clustering and the proposed DCNS K-means clustering using fuzzy logic.

Figure 4 represents the network usage comparison between DCNS with the existing scheduling (SOS), K-means Clustering and the proposed K-means Clustering using fuzzy logic Scheduling. It is evident that DCNS proposed K-means Clustering using fuzzy logic Scheduling exhibits significantly lower network usage compared to DCNS with the existing scheduling (SOS) and K-means Clustering scheduling.

Figure 5 represents the response time comparison between DCNS with the existing scheduling (SOS), K-means Clustering and the proposed K-means Clustering using fuzzy logic Scheduling. It is evident that DCNS proposed K-means Clustering using fuzzy logic Scheduling exhibits significantly lower response time compared to DCNS with the existing scheduling (SOS) and K-means Clustering scheduling.

7 Conclusion

In this research, we presented a novel approach to task scheduling in fog computing environments using the K-means clustering algorithm enhanced with fuzzy logic. The combination of clustering capabilities with adaptability to uncertainty and variability provided by fuzzy logic proved to be effective in improving task scheduling efficiency, reducing execution time, response time and network usage and enhancing resource utilization. Leveraging machine learning techniques, our approach demonstrates remarkable reductions in execution time, response time and network usage in dynamic fog environments. This approach adheres to the strategy of task clustering and the subsequent allocation of these optimal clusters to fog devices to identify more suitable virtual machines for allocation. The experimental evaluation of this proposed model is conducted using the iFogsim toolkit. The results provide compelling evidence that the proposed clustering algorithm surpasses the existing scheduling schemes (SOS) and k-means in iFogsim, notably in terms of reducing execution time, response time and network usage. Task scheduling remains a critical challenge in fog computing, and the integration of K-Means clustering with fuzzy logic offers an exciting avenue for further exploration and development. Future research avenues may focus on further enhancing the scalability and robustness of our proposed approach, as well as its applicability in various fog computing scenarios. Overall, the proposed task scheduling algorithm is a promising approach for fog computing applications. However, the proposed algorithm is still under development and has not been evaluated in all possible fog computing environments. It's a very difficult task to design a fuzzy logic rules considering fog nodes capabilities, task requirements etc. otherwise this will limit the proposed algorithm to get the optimal solution.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

MS: Writing – original draft. RE: Supervision, Writing – review & editing. RQ: Supervision, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abadi, Z. J. K., Mansouri, N., and Khalouie, M. (2023). Task scheduling in fog environment—challenges, tools and methodologies: a review. *Comp. Sci. Rev.* 48, 100550. doi: 10.1016/j.cosrev.2023.100550

Agarwal, G., Gupta, S., Ahuja, R., and Rai, A. K. (2023). Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog–cloud computing. *Knowl. Based Syst.* 272, 110563. doi: 10.1016/j.knosys.2023.110563

Alexandrescu, A. (2023). Parallel processing of sensor data in a distributed rules engine environment through clustering and data flow reconfiguration. *Sensors* 23, 1543. doi: 10.3390/s23031543

Alguliyev, R. M., Aliguliyev, R. M., and Alakbarov, R. G. (2023). Constrained k-means algorithm for resource allocation in mobile cloudlets. *Kybernetika* 59, 88–109. doi: 10.14736/kyb-2023-1-0088

Alhijawi, B., and Awajan, A. (2023). Genetic algorithms: theory, genetic operators, solutions, and applications. *Evolut. Intellig.* 1–12. doi: 10.1007/s12065-023-00822-6

Aqib, M., Kumar, D., and Tripathi, S. (2023). Machine learning for fog computing: review, opportunities and a fog application classifier and scheduler. *Wireless Pers. Commun.* 129, 853–880. doi: 10.1007/s11277-022-10160-y

Arooj, A., Farooq, M. S., Akram, A., Iqbal, R., Sharma, A., and Dhiman, G. (2021). Big data processing and analysis in internet of vehicles: architecture, taxonomy, and open research challenges. *Arch. Comput. Methods Eng.* 1–37. doi: 10.1007/s11831-021-09607-5

Atiq, H. U., Ahmad, Z., Uz Zaman, S. K., Khan, M. A., Shaikh, A. A., and Al-Rasheed, A. (2023). Reliable resource allocation and management for IoT transportation using fog computing. *Electronics* 12, 1452. doi: 10.3390/electronics12061452

Awad, F. H., Hamad, M. M., and Alzubaidi, L. (2023). Robust classification and detection of big medical data using advanced parallel K-means clustering, YOLOv4, and logistic regression. *Life* 13, 691. doi: 10.3390/life13030691

Bakshi, M., Chowdhury, C., and Maulik, U. (2023). Cuckoo search optimization-based energy efficient job scheduling approach for IoT-edge environment. *J. Supercomput.* 1–29. doi: 10.1007/s11227-023-05358-1

Bhaumik, H., Bhattacharyya, S., Nath, M. D., and Chakraborty, S. (2016). Hybrid soft computing approaches to content based video retrieval: a brief review. *Appl. Soft Comput.* 46, 1008–1029. doi: 10.1016/j.asoc.2016.03.022

Cerf, S., Doerr, B., Hebras, B., Kahane, Y., and Wietheger, S. (2023). The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem. *arXiv preprint* arXiv:2305.13459. doi: 10.24963/ijcai.2023/613

Chaplot, N., Pandey, D., Kumar, Y., and Sisodia, P. S. (2023). A comprehensive analysis of artificial intelligence techniques for the prediction and prognosis of genetic disorders using various gene disorders. *Arch. Comput. Methods Eng.* 30, 3301–3323. doi: 10.1007/s11831-023-09904-1

Gomathi, B., Lokesh, S., and Antony Vijay, J. (2023). "Task scheduling algorithm using improved PSO in dew computing," in *Micro-Electronics and Telecommunication Engineering: Proceedings of 6th ICMETE 2022*, Singapore: Springer Nature Singapore, 317–324.

Goniwada, S. R. (ed.). (2022). "Cloud native services," in *Cloud Native Architecture and Design* (Berkeley, CA: Apress), 27–54.

Guo, C., and Chen, J. (2023). "Big data analytics in healthcare," in *Knowledge Technology and Systems: Toward Establishing Knowledge Systems Science*, eds Y. Nakamori and H. Nomiya (Singapore: Springer Nature Singapore), 27–70.

Guo, Q. (2017). Task scheduling based on ant colony optimization in cloud environment. *AIP Conf. Proc.* 1834, 1. doi: 10.1063/1.4981635

Gupta, H., Dastjerdi, A., Ghosh, S., and Buyya, R. (2016). iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. *Softw. Pract. Exp.* 47, 1275–1296. doi: 10.1002/spe.2509

Gupta, S., and Singh, N. (2023). "Resource management with load balancing strategies in Fog-IoT computing environment: trends, challenges and future directions," in *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)* (Greater Noida: IEEE).

He, J., and Bai, W. (2023). "Computation offloading and task scheduling based on improved integer particle swarm optimization in fog computing," in *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)* (Qingdao: IEEE).

Hosseinioun, P., Kheirabadi, M., Kamel Tabbakh, S. R., and Ghaemi, R. (2022). aTask scheduling approaches in fog computing: a survey. *Trans. Emerg. Telecommun. Technol.* 33, e3792. doi: 10.1002/ett.3792

Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H., et al. (2023). Task scheduling mechanisms for fog computing: a systematic survey. *IEEE Access*. doi: 10.1109/ACCESS.2023.3277826

Jamshed, M. A., Ismail, M., Pervaiz, H., Atat, R., Bayram, I. S., and Ni, Q. (2023). Reinforcement learning-based allocation of fog nodes for cloud-based smart grid. *e-Prime-Adv. Electr. Eng. Electr. Energy* 4, 100144. doi: 10.1016/j.prime.2023.100144

Jeyaraj, R., Balasubramaniam, A., Ma, A. K., Guizani, N., and Paul, A. (2023). Resource management in cloud and cloud-influenced technologies for internet of things applications. *ACM Comput. Surv.* 55, 1–37. doi: 10.1145/3571729

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: asurvey. *Int. J. Robotics Res.* 32, 1238–1274. doi: 10.1177/0278364913495721

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine learning operations (mlops): overview, definition, and architecture. *IEEE Access*. doi: 10.1109/ACCESS.2023.3262138

Kumar, M. S., and Karri, G. R. (2023). Eeoa: time and energy efficient task scheduling in a cloud-fog framework. *Sensors* 23, 2445. doi: 10.3390/s23052445

Kumar, S., Tiwari, P., and Zymbler, M. (2019). Internet of Things is a revolutionary approach for future technology enhancement: a review. *J. Big Data* 6, 111. doi: 10.1186/s40537-019-0268-2

Li, Y., Zhang, X., Zeng, T., Duan, J., Wu, C., Wu, D., et al. (2023). Task placement and resource allocation for edge machine learning: a gnn-based multi-agent reinforcement learning paradigm. *arXiv preprint* arXiv:2302.00571. doi: 10.1109/TPDS.2023.3313779

Meng, L., Cheng, W., Zhang, B., Zou, W., Fang, W., and Duan, P. (2023). An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem. *Sensors* 23, 3815. doi: 10.3390/s23083815

Mishra, P. K., and Chaturvedi, A. K. (2023). "State-of-the-art and research challenges in task scheduling and resource allocation methods for cloud-fog environment," in *2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)* (Jaipur: IEEE).

Mtshali, M., Dlamini, S., Adigun, M., and Mudali, P. (2019). "K-means based on resource clustering for smart farming problem in fog computing," in *2019 IEEE Africon* (IEEE).

Paparella, V., Anelli, V. W., Nardini, F. M., Perego, R., and Di Noia, T. (2023). *Post-hoc* selection of pareto-optimal solutions in search and recommendation. *arXiv preprint* arXiv:2306.12165. doi: 10.1145/3583780.3615010

Peng, Z., Cui, D., Zuo, J., Li, Q., Xu, B., and Lin, W. (2015). Random task schedulingscheme based on reinforcement learning in cloud computing. *Clust. Comput.* 18, 1595–1607. doi: 10.1007/s10586-015-0484-2

Pirozmand, P., Jalalinejad, H., Hosseinabadi, A. A. R., Mirkamali, S., and Li, Y. (2023). An improved particle swarm optimization algorithm for task scheduling in cloud computing. *J. Ambient Intellig. Human. Comput.* 14, 4313–4327. doi: 10.1007/s12652-023-04541-9

Ranjan, V., and Sharma, L. (2023). "Real-time task scheduling and resource scheduling in fog computing using deep learning techniques," in *2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (IEEE).

Reddy, P. B., and Sudhakar, C. (2023). An osmotic approach-based dynamic deadline-aware task offloading in edge–fog–cloud computing environment. *J. Supercomput.* 1–23. doi: 10.1007/s11227-023-05440-8

Roy, A., Xu, H., and Pokutta, S. (2017). "Reinforcement learning under modelmismatch," in *Paper Presented at: 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 3043–3052.

Saad, M., Qureshi, R. I., and Rehman, A. U. (2023). "Task scheduling in fog computing: parameters, simulators and open challenges." in *2023 Global Conference on Wireless and Optical Technologies (GCWOT)* (Malaga: IEEE).

Saif, F. A., Latip, R., Hanapi, Z. M., and Shafinah, K. (2023). Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. *IEEE Access* 11, 20635–20646. doi: 10.1109/ACCESS.2023.3241240

Shen, X., Xu, D., Song, L., and Zhang, Y. (2023). Heterogeneous multi-project multi-task allocation in mobile crowdsensing using an ensemble fireworks algorithm. *Appl. Soft Comput.* 145, 110571. doi: 10.1016/j.asoc.2023.110571

Singhrova, A. (2023). Levy flight firefly based efficient resource allocation for fog environment. *Intellig. Autom. Soft Comput.* 37, e035389. doi: 10.32604/iasc.2023.035389

Sun, L., Shi, W., Wang, J., Mao, H., Tu, J., and Wang, L. (2023). Research on production scheduling technology in knitting workshop based on improved genetic algorithm. *Appl. Sci.* 13, 5701. doi: 10.3390/app13095701

Tran-Dang, H., and Kim, D. S. (2023). Fog computing: fundamental concepts and recent advances in architectures and technologies. *Cooper. Distrib. Intellig. Comput. Fog Comput. Concepts Architect. Fram.* 1–18. doi: 10.1007/978-3-031-33920-2_1

Vispute, S. D., and Vashisht, P. (2023). Energy-efficient task scheduling in fog computing based on particle swarm optimization. *SN Comp. Sci.* 4, 391. doi: 10.1007/s42979-022-01639-3

Wei, Z., Zhang, Y., Xu, X., Shi, L., and Feng, L. (2017). A task scheduling algorithmbased on Q-learning and shared value function for WSNs. *Comput. Netw.* 126, 141–149. doi: 10.1016/j.comnet.2017.06.005

Yin, C., Li, H., Peng, Y., Fang, Q., Xu, X., and Dan, T., et al. (2023). *An Optimized Resource Scheduling Algorithm Based on GA and ACO Algorithm*.

You, Y., Liu, Z., Liu, Y., Peng, N., Wang, J., Huang, Y., et al. (2023). K-means module division method of FDM3D printer-based function–behavior–structure mapping. *Appl. Sci.* 13, 7453. doi: 10.3390/app13137453

Zhang, X., Xu, M., Su, J., and Zhao, P. (2023). Structural models for fog computing based internet of things architectures with insurance and risk management applications. *Eur. J. Operat. Res.* 305, 1273–1291. doi: 10.1016/j.ejor.2022.07.033