ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20240602001731
Original scientific paper

Optimized Resource Allocation for IoT Networks Using Genetic Particle Swarm Optimization and Enhanced K-Nearest Neighbor Algorithms

G. KALINGARANI, P. SELVARAJ*

Abstract: The proliferation of Internet of Things (IoT) devices has surged, with projections estimating their numbers to reach fifty billion. This growth underscores the imperative need for technologies that prioritize minimal power consumption and computational costs. However, the heightened latency inherent in cloud-based services presents significant challenges. This paper proposes a novel resource allocation strategy employing Genetic Particle Swarm Optimization (GPSO) and an optimized K-Nearest Neighbor (OKNN) algorithm to address these challenges. The GPSO algorithm models each network node as a particle, drawing inspiration from natural behaviors observed in birds. Through iterative updates, it optimizes the position and velocity of these particles, enhancing resource allocation efficiency. Additionally, the OKNN algorithm facilitates data classification into confidential and non-confidential categories, refining data processing mechanisms further. We conduct a comprehensive analysis, evaluating minimum, maximum, and average response times alongside data center processing times. Comparative assessments of CPU usage and energy consumption are performed against established algorithms like round-robin and genetic algorithms. Our findings reveal significant enhancements in both efficiency and performance across various IoT applications. This research advances IoT device management and operation, with implications for improving overall data processing mechanisms. The proposed resource allocation strategy offers a promising avenue for addressing the unique challenges posed by the evolving landscape of IoT technologies.

Keywords: energy efficiency; genetic particle swarm optimization (GPSO); IoT resource allocation; latency reduction; optimized K-nearest neighbor (OKNN)

1 INTRODUCTION

The proliferation of Internet of Things (IoT) devices has led to a surge in data generation and processing requirements. Traditional cloud-based approaches, while effective for certain applications, often face limitations such as high latency, data privacy concerns, and limited scalability. To address these challenges, fog computing has emerged as a promising paradigm, offering decentralized processing closer to the edge of the network [1]. Effective mechanisms for allocating resources are becoming increasingly important as organizations continue to increase their use of fog computing. This is because effective mechanisms for allocating resources are necessary for improving the responsiveness and efficiency of systems. Particle Swarm Optimization (PSO) and K-Nearest Neighbours (KNN) are the two components that make up the PKNN algorithm, which works to achieve the best possible outcomes. This technology aims to improve the efficiency of Fog Computing environments distributing available resources.

The K-Nearest Neighbours (KNN) algorithm is a robust machine learning algorithm well-known for its accuracy and simplicity in the context of regression and classification tasks [2]. In this innovative approach, the strengths of KNN and PSO, a meta-heuristic optimization algorithm influenced by swarm behavior, are combined. PKNN intends to combine these two successful approaches to construct a robust and adaptable system. Within the context of fog computing, the optimization of performance and the enhancement of service quality are the objectives of this system's resource allocation [3]. This paper proposes a novel resource allocation strategy for fog computing environments, specifically tailored for healthcare applications. This approach leverages Genetic Particle Swarm Optimization (GPSO) and Optimized K-Nearest Neighbor (OKNN) algorithms to optimize resource utilization, minimize latency, and enhance data security. By combining these techniques, the aim is to improve the overall performance and efficiency of IoT-based healthcare systems.

2 LITERATURE REVIEW

The Internet of Things (IoT) and edge computing are expected to continue gaining popularity, increasing the demand for computing and storage resources. The concept of cloud computing is brought to the network's periphery through this distributed computing model. In order to accomplish this goal, the distance between the devices and the end-users can be reduced through the integration of computation, storage, and networking strategies. In order to achieve optimal performance and responsiveness, fog computing systems need to allocate resources efficiently. This is because the volume of data generated at the edge constantly grows daily [4]. Making the most of what is available in fog computing means "allocating resources". This is accomplished by strategically distributing computational tasks and managing resources efficiently to meet the varied requirements of applications. An additional layer of complexity is added to this already difficult task by the dynamic nature of edge environments, characterized by unpredictable workloads, resource constraints, and urgent needs [5-7]. In situations where such challenges are present, conventional methods of resource distribution frequently fail to meet expectations. Existing research on resource allocation in fog computing highlights the challenges of managing diverse workloads, dynamic environments, and resource constraints. While various approaches have been proposed, there is a need for more tailored solutions that address the specific requirements of healthcare applications.

Consequently, researchers are investigating innovative approaches to make fog computing systems more effective and flexible. PKNN, the Particle Swarm Optimization algorithm, is a relatively new technique garnering much attention. In order for the combination to be successful, it is necessary to combine the K-Nearest Neighbors (KNN) algorithm with the Particle Swarm Optimization (PSO) algorithm [8]. Particle Swarm Optimization (PSO), which is based on the cooperative behavior of particles, is one optimization method that finds the best solutions. PSO is a

method that finds the best solutions. One of the most well-known algorithms for machine learning is called K-Nearest Neighbors (KNN), which is excellent for handling data classification. The KNN algorithm considers the proximity of instances seen before [9-11]. Utilizing a synergistic approach that incorporates both K-Nearest Neighbors (KNN) and Particle Swarm Optimization (PSO) is a promising strategy for resource allocation in fog computing.. In this method, the power of PSO for optimization and the capability of KNN for pattern recognition are combined to improve decision-making in uncertain and dynamic environments at the edge. The purpose of this literature review is to investigate the approaches researchers have taken to the problem of resource allocation in fog computing [12-14]. More specifically, we will investigate how the PKNN algorithm, a combination of PSO and K-Nearest Neighbors, has been investigated. In order to provide a comprehensive overview, this review will concentrate on the current state of PKNN in fog computing resource allocation, as well as its limitations and potential future directions [15]. This is accomplished through the integration of data that was collected in the past, the discovery of information that is missing, and the resolution of issues. This research aims to shed light on how to allocate resources in a resilient and flexible manner to improve the efficiency and reliability of fog computing systems [16].

3 THE PROPOSED METHOD

Fog computing can be used to improve the quality of service. IoT-based devices in cloud computing make it challenging to provide resource allocation improvement - the delay during allocation is significant and results in low resource usage efficiency. The proposed method puts forth a resource allocation strategy for fog computing based on the genetic particle swarm optimization algorithm and optimized k nearest neighbour algorithm. In the proposed work, the user is given access to choose the needed resources among the resources that have been pre-allocated. The various results obtained through simulation have been tabulated and depicted through graphical representations. The results show that the proposed method is better than the existing ones. Fig. 1 shows the flow diagram of the proposed method. The hospital-based data in the form of texts is the input data for the proposed model. This data is pre-processed to remove unnecessary content. The next step involves the selection of the fog node. Once the fog node has been selected, the frequency spectrum selection is based on the clustering method. Once the clusters are formed, bandwidth allocation for each node present is done.

The nodes are allocated with the required resources. This helps to reduce queuing and thereby minimizes network traffic. After this step, the final text file is obtained with the frequency level. This received file is converted to CSV from the existing text format using Python coding. Pre-processing and feature extraction are carried out in the converted file. The next step is the transfer learning technique. During the transfer learning technique, the training mate is created, the model is formed, and the testing phase occurs. The obtained result is optimized to get the final result.

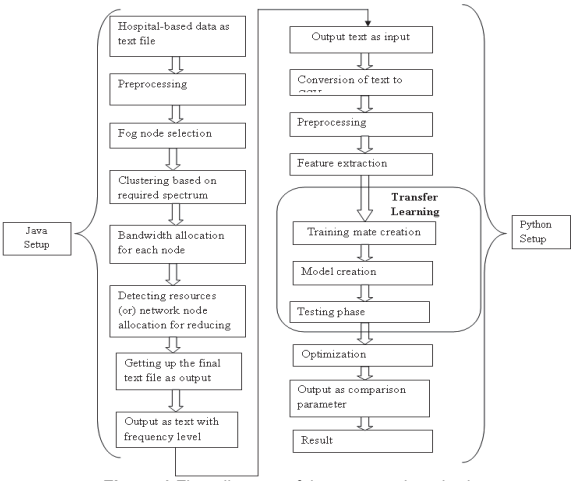

Figure 1 Flow diagram of the proposed method

3.1 Genetic PSO Algorithm

The genetic particle swarm optimization algorithm (GPSO) is based on the intelligence of the swarm. It is a heuristic algorithm. It has been developed from the inspiration of animals and human behavior. The optimal solution can be determined through the PSO algorithm. GPSO is a metaheuristic optimization algorithm inspired by the behavior of bird flocks and fish schools. It is a population-based algorithm that iteratively updates the position and velocity of individuals (particles) in the search space to find optimal solutions. Implementing the PSO algorithm is more straightforward, and the parameters to be adjusted are also less. The goal of this algorithm is the convergence of the fitness value after every iteration. In the proposed work, the virtual machines, i.e., the users are considered particles. Fig. 2 gives the flow diagram of resource allocation through PSO.

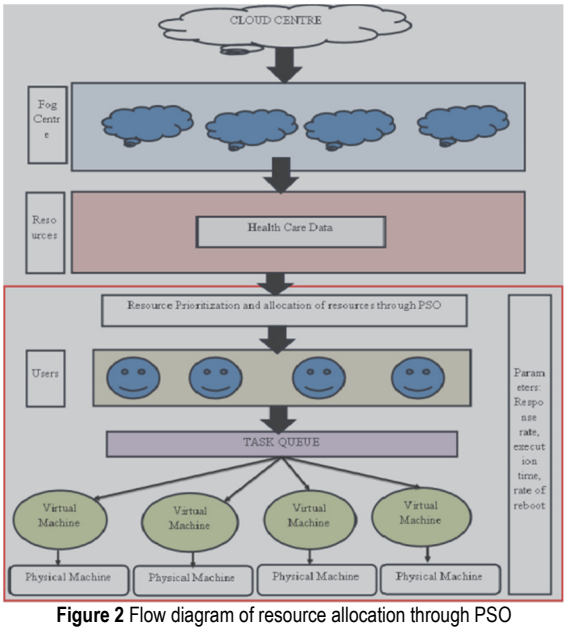

Figure 2 Flow diagram of resource allocation through PSO

3.2 Generation of the Population in a Random Manner

The first step involves selecting the virtual machines with the fittest value. The virtual machines with low fitness values will be discarded. The number of tasks allocated in the virtual machine can be used to calculate the fitness value.

$$Total\ Fitness = \sum_{j=1}^{m} Total\ alloated\ tasks \quad (1)$$

where *m* represents the count of the virtual machines in the group.

Input: *virtual machine group with priority selection*
Output: *Virtual machines with the best fitness value*
Choose the count of the virtual machines (m)
For j =0 to m
{
Return fitness value of the virtual machine group
Choose the virtual machine with the best fitness value
}

Cross Over:

The crossover operation in GPSO combines the genetic material of two parent particles to create offspring. In this approach, it randomly selects two parent particles and exchanges portions of their chromosomes (i.e., resource allocation decisions) to generate new candidate solutions.

In the cross-over, two virtual machines can combine to produce a new virtual machine. A better fitness value of the virtual machine can be obtained. The virtual machine created after the cross-over is called the child virtual machine.

Input: *two virtual machines of the group with the best fitness value*
Output: *new virtual machine (or) child virtual machine*
For j = 0 to m/2
{
Two Fittest virtual machines
For j = 0 to m
{
ChildVirtualMachine_1 = Parent_1.sub-string(0,m/2) + P2.sub-string(m/2,m)
ChildVirtualMachine_2 = Parent_2.sub-string(0,m/2) + P12.sub-string(m/2,m)
}
}

3.3 Mutation

Mutation is the changes in the characteristics of the newly produced virtual machine. The mutation operator is applied to the newly produced virtual machine at a lower rate to get the required characteristics. One or more characteristics of the parent virtual machines can be selected to modify the child virtual machine.

Input: *Novel group of virtual machines*
Output: *Changed virtual machine group after mutation*
If (mutation == T)
{
Set number = Length of child virtual machine
Ind = (integer)(math.rand() x number)
Novel bit = flip (child virtual machine. Sub-string(ind, ind+1))
Novel bit string = child virtual machine. Sub-string(ind, ind+1) + novel bit + child virtual machine . sub-string (ind+1, child virtual machine. Length ())
}

Every individual is considered a particle in the particle swarm optimization algorithm. The population refers to the total number of particles generated during the initial stage. Every particle is capable of moving in a space of multiple dimensions. Their movements can be analyzed using two different parameters, namely, the position and the velocity. During the searching process of the particles, the velocities will be adjusted based on specific criteria. They shift their position to a better one. Two variables are being stored while moving from one position to the other. They are the best local and the best global. Local best refers to the particle's position individually. Global best refers to the best position of the particle as a group.

3.4 Position and Velocity Update

The local and global best are used to update the particle's velocity. The velocity is updated after each iteration through the following relation.

$$Velocity_j^{\,d}(T+1) = Velocity_j^{\,d}(T) + C_1R_1(L_{best\,j}^{\,d}(T) - Y_j^{\,d}) + C_2R_2(G_{best\,j}^{\,d}(T) - Y_j^{\,d}) \quad (2)$$

$$Y_j^{\,d}(T+1) = Y_j^{\,d}(T) + Velocity_j^{\,d}(T+1) \quad (3)$$

C_1 and C_2 are the coefficients of acceleration. R_1 and R_2 are numbers that are uniformly distributed between 0 and 1.

Update of velocity:
Input: *Modified particle group after the mutation process*
Output: *Particle fitness value update*
For j = 1 to n // (n → particle count)
{

$$Velocity_j^{\,d}(T+1) = Velocity_j^{\,d}(T) + +C_1R_1(L_{best\,j}^{\,d}(T) - Y_j^{\,d}) + C_2R_2(G_{best\,j}^{\,d}(T) - Y_j^{\,d}) \quad (4)$$

}
Update of position:
Input: updated particle fitness value
Output: Global best position of the particle, which depends on velocity
For j = 1 to n // (n → particle count)
{

$$Y_j^{\,d}(T+1) = Y_j^{\,d}(T) + Velocity_j^{\,d}(T+1)$$

}

3.5 Optimized KNN (K Nearest Neighbor) Algorithm

OKNN is a machine learning algorithm used for classification and regression tasks. It classifies new data points based on the majority class of their K nearest neighbors in the training set. OKNN can contribute to data privacy by classifying sensitive information and restricting access to authorized users. By using k-nearest neighbors, the algorithm can identify patterns and anomalies in the data, which can be used to detect potential security threats. The proposed KNN algorithm can be segmented into two

categories. The first part deals with the classification of the data sets. The next part deals with using asymmetric cryptographic algorithms for secure data transmission. The utilization of the resources is minimized, and the data processing time is also less by using this method.

4 ANALYSIS RESULT AND DISCUSSION

4.1 Performance Evaluation

To analyze latency, throughput, and other metrics extensive simulations have been conducted in order to evaluate the performance of the proposed method. Compared with existing approaches, the proposed method shows traditional resource allocation algorithms, such as round robin and genetic algorithms. The results show that the method outperforms these approaches in terms of resource utilization and response time. To evaluate scalability the number of IoT devices and the workload have been increased. The results indicate that the method is scalable and can handle large-scale deployments.

4.2 Security Analysis

Implementation of encryption, access control mechanisms, and regular security audits to ensure the confidentiality and integrity of sensitive healthcare data. This paper identifies potential security threats, such as unauthorized access and data breaches, and implements appropriate countermeasures. To Propose mitigation strategies this paper recommends ongoing security monitoring and updates to address emerging threats and vulnerabilities.

4.3 Case Study

The proposed method is applied to a real-world healthcare scenario involving a remote patient monitoring system. The results demonstrate significant improvements in patient monitoring efficiency and response time. The proposed method effectively allocated resources to ensure timely processing of critical data. Fig. 3 shows the distribution of the fog resources to the users.

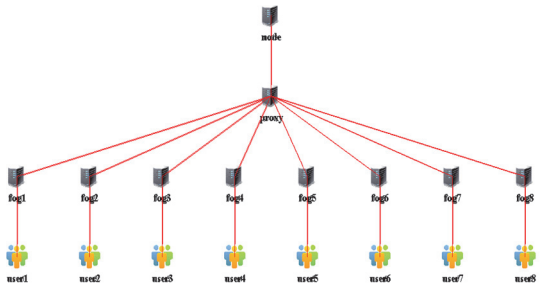

Figure 3 Allocation of fog resources to the users

Here, the proxy server is connected to the node. The fogs are present in the proxy server. Each user that tries to access the proxy server will be allocated with a distinct fog. Tab. 1 gives the simulation results for the scheduled average, minimum, and maximum with the data center processing time of 0.28, 0.02, and 0.88, respectively. The overall response time of the scheduled average, scheduled minimum, and scheduled maximum are 295.40 ms, 38.13 ms, and 627.64 ms, respectively. The average response time of the UB 1 is 299.315 ms.

Table 1 Schedule average, minimum, and maximum with the data center processing time of 0.28 ms, 0.02 ms, and 0.88 ms, respectively

Overall Response Time		295.40	38.13	627.64
Data centre processing time		0.28	0.02	0.88
S. No	Tasks	Avg / ms	Min / ms	Max / ms
1.	UB1	299.315	223.637	376.639
2.	User 1	199.672	156.143	241.145
3.	User 2	497.394	392.64	607.607
4.	User 3	49.863	38.134	60.631
5.	User 4	500.084	375.145	605.142
6.	User 5	200.152	159.145	247.142
7.	User 6	200.624	157.141	244.15
8.	User 7	200.238	151.08	244.139

The average reply duration of user 2 is 497.394 ms. The lowest reply duration and the highest reply duration of user 2 for the number of tasks are 392.64 ms and 607.607 ms, respectively. The average reply duration of the user 3 for the number of tasks is 49.863 ms. The lowest reply duration and the highest reply duration for user 3 for the number of tasks are 38.134 ms and 60.631 ms, respectively. The average reply duration of the user 4 for the number of tasks is 500.084 ms. The lowest reply duration and the highest reply duration for user 4 for the number of tasks are 375.145 ms and 605.142 ms, respectively. The average reply duration of the user 5 for the number of tasks is 200.152 ms. The lowest reply duration and the highest reply duration for user 5 for the number of tasks are 159.45 ms and 247.142 ms, respectively. The average reply duration of the user 6 for the number of tasks is 200.624 ms. The lowest reply duration and the highest reply duration for user 6 for the number of tasks are 157.141 ms and 244.15 ms, respectively.

The average response time of the user 7 for the number of tasks is 200.238 ms. The lowest reply duration and the highest reply duration for user 7 for the number of tasks are 151.08 ms and 244.139 ms, respectively. The average reply duration of the user 8 for the number of tasks is 502.471 ms. The lowest reply duration and the highest reply duration for user 8 for the number of tasks are 402.641 ms and 627.638 ms, respectively.

Tab. 2 shows the simulation results for schedule average, minimum, and maximum with the data center processing time of 0.50 ms, 0.01 ms, and 0.90 ms, respectively. The average, lowest, and highest schedules are 50.15 ms, 38.13 ms, and 64.13 ms, respectively. The average, lowest, and highest schedules of user 1 are 50.164 ms, 38.129 ms, and 60.283 ms, respectively. The average, lowest, and highest schedules of user 2 are 50.05 ms, 39.88 ms, and 64.13 ms, respectively. The average, lowest, and highest schedules of user 3 are 50.043 ms, 39.397 ms, and 61.346 ms, respectively.

Table 2 Schedule average, minimum, and maximum with the data center processing time of 0.50 ms, 0.01 ms, and 0.90 ms, respectively

Overall Response Time		50.15	38.13	64.13
Data centre processing time		0.50	0.01	0.90
S. No	Tasks	Avg / ms	Min / ms	Max / ms
1.	User 1	50.164	38.129	60.283
2.	User 2	50.05	39.88	64.13
3.	User 3	50.043	39.397	61.346
4.	User 4	50.302	38.634	61.134
5.	User 5	50.213	40.9	61.392
6.	User 6	50.228	38.908	61.158
7.	User 7	50.172	39.383	62.158
8.	User 8	50.027	39.381	60.635

The average, lowest, and highest schedules of user 4 are 50.302 ms, 38.634 ms, and 61.134 ms, respectively. The average, lowest, and highest schedules of user 5 are 50.213 ms, 40.9 ms, and 61.392 ms, respectively. The average, minimum, and maximum schedules of user 6 are 50.228 ms, 38.908 ms, and 61.158 ms, respectively. The average, minimum, and maximum schedules of user 7 are 50.172 ms, 39.383 ms, and 62.63 ms, respectively. The average, lowest, and highest schedules of user 8 are 50.027ms, 39.381ms, and 60.635ms, respectively. Tab. 3 shows the simulation results for schedule average, lowest, and highest with the data center processing time of 0.50 ms, 0.02 ms, and 0.90 ms, respectively. The overall average, lowest, and highest schedules are 50.03 ms, 37.63 ms, and 62.64 ms, respectively.

Table 3 Schedule average, minimum, and maximum with the data center processing time of 0.50 ms, 0.02 ms, and 0.90 ms, respectively.

Overall Response Time		50.03	37.63	62.64
Data center processing time		0.50	0.02	0.90
S.No	Tasks	Avg / ms	Min / ms	Max / ms
1.	User 1	49.968	38.131	59.883
2.	User 2	49.832	37.634	59.561
3.	User 3	50.345	38.391	62.635
4.	User 4	50.234	38.9	60.642
5.	User 5	49.804	40.132	59.619
6.	User 6	50.158	38.922	60.659
7.	User 7	49.899	41.034	61.383
8.	User 8	50.005	39.637	60.382

The average, lowest, and highest schedules of user 1 are 49.968 ms, 38.131 ms, and 59.883 ms respectively. The average, lowest, and highest schedules of user 2 are 49.832 ms, 37.634 ms, and 59.561 ms, respectively. The average, lowest, and highest schedules of user 3 are 50.345 ms, 38.391 ms, and 62.635 ms respectively. The average, lowest, and highest schedule of user 4 is 50.234 ms, 38.9 ms, and 60.642 ms, respectively.

The average, lowest, and highest schedule of user 5 is 49.804 ms, 40.132 ms, and 59.619 ms, respectively. The average, lowest, and highest schedules of user 6 are 50.158 ms, 38.922 ms, and 60.659 ms, respectively. The average, lowest, and highest schedules of user 7 are 49.899ms, 41.034 ms, and 61.383 ms, respectively. The average, lowest, and highest schedules of user 8 are 50.005 ms, 39.637 ms, and 60.38 ms, respectively. Fig. 5 depicts the window with the options for canvas, execution, and exit. The shortcut for canvas is control + U, the shortcut for execution is control + I, and the shortcut for exit is control + W.

Tab. 4 shows the computational cost with security measures after the completion of the simulation. The total virtual cost of the machine is about $0.50, the total cost of the data transfer is about $0.51, and the total is about $1.01. The security measure after the prediction of the load balancing rate is shown in Tab. 5.

Table 4 Computational cost

Data Center	Cost of the virtual machine in total / $	Cost of data transfer in total / $	Overall Total / $
DCI	0.502	0.513	1.014

Table 5 Security measure per interval

Data Center	Encryption Execution Duration / ms	Decryption Execution Duration / ms
DCI	617	11775

The encryption execution time and the decryption execution are 617 and 11775 milliseconds, respectively. Tab. 6 shows the comparative analysis of CPU usage and execution delay of different algorithms. Fig. 4 shows the graphical representation of CPU usage of different algorithms. Fig. 5 gives the graphical representation of the execution delay of different algorithms.

Table 6 Comparative analysis of CPU usage and execution delay of different algorithms

Count of edge devices	CPU usage / %			Delay during execution / s		
	Round Robin	Genetic Algorithm	PKNN Algorithm	Round Robin	Genetic Algorithm	PKNN Algorithm
100	4.3	4.9	5.3	1.7	2.0	1.9
150	6.3	7.1	6.1	2.1	2.4	2.1
200	8.4	9.1	6.9	2.5	2.7	2.2
250	8.7	9.5	8.4	2.7	2.9	2.5
300	9.0	9.9	10.2	3.0	3.0	2.8
350	9.1	10.2	11.9	3.5	3.4	3.3
400	9.2	10.5	13.2	4.0	3.7	3.6

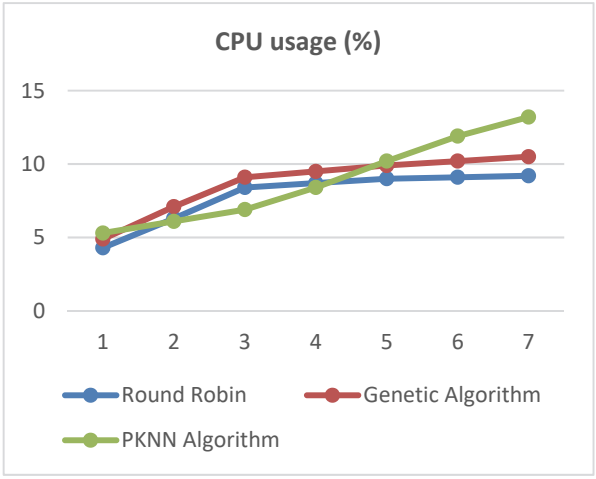

Figure 4 Graphical representation of CPU usage of different algorithms

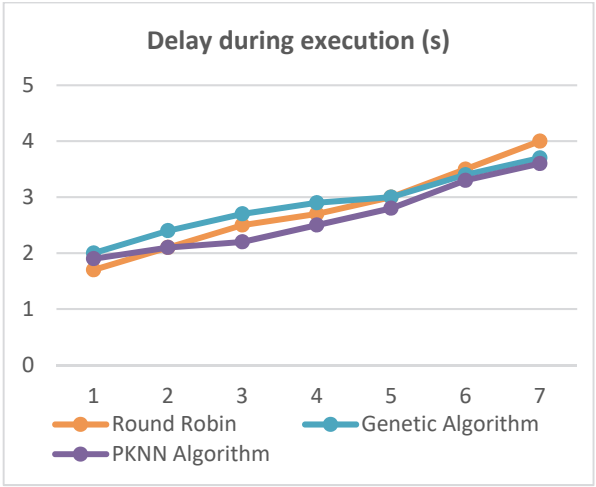

Figure 5 Graphical representation of execution delay of different algorithms

The analysis has been made considering different devices ranging from 100 to 400. The percentage of CPU usage of the round-robin algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 4.3 percent to a maximum of 9.2 percent. The percentage of CPU usage of the genetic algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 4.9 percent to a maximum of 10.5 percent. The percentage of CPU usage of the PKNN

algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 5.3 percent to a maximum of 13.2 percent.

The execution delay of the round-robin algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 1.7 seconds to a maximum of 4.0 seconds. The execution delay of the genetic algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 2.0 seconds to a maximum of 3.7 seconds. The execution delay of the PKNN algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 1.9 seconds to a maximum of 3.6 seconds.

Tab. 7 gives a comparative analysis of different algorithms' waiting duration and energy consumption. Fig. 6 gives the graphical representation of the waiting duration of different algorithms, and Fig. 7 gives the graphical representation of the energy consumption of different algorithms.

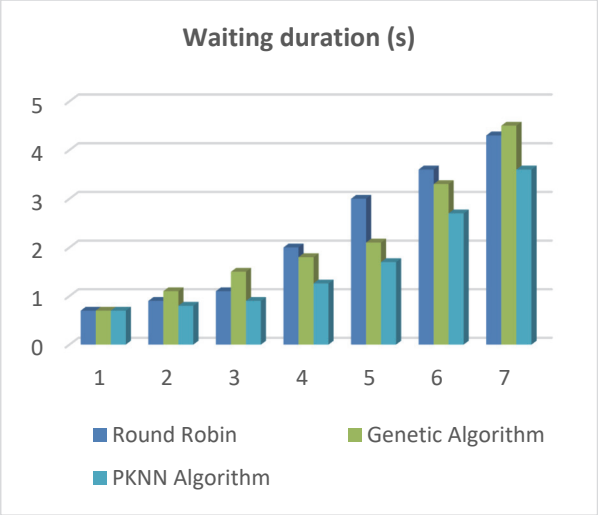

Figure 6 Waiting duration of different algorithms

The waiting duration of the round-robin algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.7 seconds to a maximum of 4.3 seconds. The waiting duration of the genetic algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.7 seconds to a maximum of 4.5 seconds. The waiting duration of the PKNN algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.7 seconds to a maximum of 3.6 seconds.

Table 7 Comparative analysis of waiting duration and energy consumption of different algorithms

Count of edge devices	Waiting duration / s			Consumption of energy / watt/hour		
	Round Robin	Genetic Algorithm	PKNN Algorithm	Round Robin	Genetic Algorithm	PKNN Algorithm
100	0.7	0.7	0.7	0.060	0.060	0.070
150	0.9	1.1	0.8	0.073	0.080	0.075
200	1.1	1.5	0.9	0.080	0.085	0.078
250	2.0	1.8	1.26	0.087	0.090	0.085
300	3.0	2.1	1.7	0.091	0.092	0.094
350	3.6	3.3	2.7	0.097	0.099	0.098
400	4.3	4.5	3.6	0.130	0.102	0.098

The energy consumption of the round-robin algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.060 watts per hour to a maximum of 0.130 watts per hour. The waiting duration of the genetic algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.060 watts per hour to a maximum of 0.102 watts per hour. The waiting duration of the PKNN algorithm as the number of devices is increased from 100 to 400 ranges from a minimum of 0.070 watts per hour to a maximum of 0.098 watts per hour.

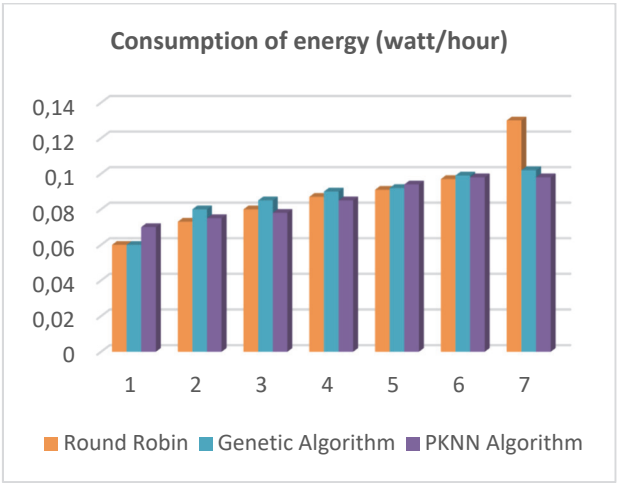

Figure 7 Energy consumption of different algorithms

5 CONCLUSION

This work presents a novel approach to data processing using cognitive Fog computing, marking a significant shift in information system architecture. The proposed framework addresses limitations of traditional Cloud-centric healthcare systems by leveraging edge devices and incorporating features like location awareness, low latency, portability, and real-time response. By implementing a scalable resource allocation mechanism and dynamic queuing technique, the performance of IoT Fog-based applications is notably enhanced. While the proposed method shows promising results, it has certain limitations. For example, the performance may be affected by the complexity of the healthcare applications and the availability of resources in the fog network. Simulation results showcased in this study illustrate the efficacy of the proposed framework in improving the efficiency of real-time healthcare applications. The Fog layer, situated between the periphery and the Cloud within the data center, serves as an intermediary platform facilitating direct connections between end devices and designated fog nodes. This strategic placement enables efficient service delivery. Additionally, the study outlines the architectural framework for utilizing the Fog layer to support real-time healthcare applications while assessing its limitations in the medical field context. Emphasizing the significance of adopting Fog computing as a practical solution for evolving IoT application requirements, particularly in healthcare, this research underscores the potential of the suggested architecture to enhance data processing efficiency, minimize delays, and improve overall system performance. As a result, this work contributes to technological advancements in critical fields that have a substantial impact on human lives. Looking ahead, the integration of Fog computing represents a pivotal advancement in ensuring the smooth and effective functioning of IoT-supported applications in real-world

scenarios. Future research could explore the integration of machine learning techniques for more intelligent resource allocation, investigate the scalability of the proposed method for large-scale deployments, and address emerging security challenges in fog computing.

6 REFERENCES

[1] Basir, R., Qaisar, S., Ali, M., Aldwairi, M., Ashraf, M. I., Mahmood, A., & Gidlund, M. (2019). Fog computing enabling industrial internet of things: State-of-the-art and research challenges. *Sensors*, *19*(21), 4807. https://doi.org/10.3390/s19214807

[2] Wang, W., Liang, B., & Li, B. (2014). Multi-resource fair allocation in heterogeneous cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, *26*(10), 2822-2835. https://doi.org/10.1109/TPDS.2014.2381643

[3] Kumar, K. S., Radhamani, A. S., & Sundaresan, S. (2021). Proficient approaches for scalability and security in IoT through edge/fog/cloud computing: a survey. *International Journal of Data Science*, *6*(1), 33-44. https://doi.org/10.1504/IJDS.2021.113745

[4] Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. *Swarm and Evolutionary Computation*, *62*, 100841. https://doi.org/10.1016/j.swevo.2021.100841

[5] Elbes, M., Alzubi, S., Kanan, T., Al-Fuqaha, A., & Hawashin, B. (2019). A survey on particle swarm optimization with emphasis on engineering and network applications. *Evolutionary Intelligence*, *12*, 113-129. https://doi.org/10.1007/s12065-019-00210-3

[6] Laroui, M., Nour, B., Moungla, H., Cherif, M. A., Afifi, H., & Guizani, M. (2021). Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications*, *180*, 210-231. https://doi.org/10.1016/j.comcom.2021.09.013

[7] Sonmez, C., Ozgovde, A., & Ersoy, C. (2019). Fuzzy workload orchestration for edge computing. *IEEE Transactions on Network and Service Management*, *16*(2), 769-782. https://doi.org/10.1109/TNSM.2019.2906510

[8] Hosni, M., Idri, A., Abran, A., & Nassif, A. B. (2018). On the value of parameter tuning in heterogeneous ensembles effort estimation. *Soft Computing*, *22*, 5977-6010. https://doi.org/10.1007/s00500-018-3138-0

[9] Agrawal, R. (2014). K-nearest neighbor for uncertain data. *International Journal of Computer Applications*, *105*(11), 13-16. https://doi.org/10.5120/18404-9610

[10] Shakarami, A., Shakarami, H., Ghobaei-Arani, M., Nikougoftar, E., & Faraji-Mehmandar, M. (2022). Resource provisioning in edge/fog computing: A comprehensive and systematic review. *Journal of Systems Architecture*, *122*, 102362. https://doi.org/10.1016/j.sysarc.2021.102362

[11] Gedeon, J., Brandherm, F., Egert, R., Grube, T., & Mühlhäuser, M. (2019). What the fog? edge computing revisited: Promises, applications and future challenges. *IEEE Access*, *7*, 152847-152878. https://doi.org/10.1109/ACCESS.2019.2946974

[12] Firouzi, F., Farahani, B., & Marinšek, A. (2022). The convergence and interplay of edge, fog, and Cloud in the AI-driven Internet of Things (IoT). *Information Systems*, *107*, 101840. https://doi.org/10.1016/j.is.2022.101840

[13] Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. *Swarm and Evolutionary Computation*, *62*, 100841. https://doi.org/10.1016/j.swevo.2021.100841

[14] Farid, M., Latip, R., Hussin, M., & Abdul Hamid, N. A. W. (2020). A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing. *Symmetry*, *12*(4), 551. https://doi.org/10.3390/sym12040551

[15] Chaudhary, S., Sharma, V. K., Thakur, R. N., Rathi, A., Kumar, P., & Sharma, S. (2023). Modified Particle Swarm Optimization Based on Aging Leaders and Challengers Model for Task Scheduling in Cloud Computing. *Mathematical Problems in Engineering*, *2023*. https://doi.org/10.1155/2023/6846765

[16] Agarwal, M. & Srivastava, G. M. S. (2021). Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *12*(10), 9855-9875. https://doi.org/10.1007/s12652-021-03200-7

Contact information:

G. KALINGARANI
Department of Computing Technologies,
College of Engineering and Technology, School of Computing,
SRM Institute of Science and Technology,
Kattankulathur, Tamil Nadu, Chennai, 603203, India

P. SELVARAJ, Associate Professor
(Corresponding author)
Department of Computing Technologies,
College of Engineering and Technology, School of Computing,
SRM Institute of Science and Technology,
Kattankulathur, Tamil Nadu, Chennai, 603203, India
E-mail: selvarajsrmist@hotmail.com