Optimized Resource Allocation in IoT using Fuzzy Logic and Bio-Inspired Algorithms

Deepak Kumar Sharma
Netaji Subhas University of Technology
Jahanavi Mishra
Netaji Subhas University of Technology
Aeshit Singh
Netaji Subhas University of Technology
Raghav Govil
Netaji Subhas University of Technology
Krishna Kant Singh (krishnaiitr2011@gmail.com)
Jain University
Akansha Singh
Amity University

Research Article

Keywords: Load Balancing, IoT, Cloud Computing, Optimization, Bio-inspired Algorithms, Firefly Algorithm, GWO, Fuzzy Logic

Posted Date: December 3rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-813780/v1

Optimized Resource Allocation in IoT using Fuzzy Logic and Bio-Inspired Algorithms

Deepak Kumar Sharma[1], Jahanavi Mishra[1] ,Aeshit Singh[1], Raghav Govil[1], Krishna Kant Singh[2], Akansha Singh[3]
[1]Department of Information Technology, Netaji Subhas University of Technology, New Delhi, India.
[2]Faculty of Engineering & Technology, Jain (Deemed-to-be University), Bengaluru, India
[3]Department of CSE, ASET, Amity University Uttar Pradesh, Noida, India
Email: dk.sharma1982@yahoo.com, jahanavimishra7799@gmail.com, aeshit2000@gmail.com, raghavgovil17@gmail.com, krishnaiitr2011@gmail.com, akanshasing@gmail.com

Abstract— IoT smart devices are a confluence of microprocessors, sensors, power source and transceiver modules to effectively sense, communicate and transfer data. Energy efficiency is a key governing value of the network performance of smart devices in distributed IoT networks. Low and discrete power and limited amount of memory and finite number of resources form some major bottlenecks in the workflow. Dynamic load balancing, reliability and flexibility are heavily relied upon by cloud computing for its accessibility. Resources are dynamically provided to the end client in an as-come on-demand fashion with the global network that is the Internet. Proportionally the need for services is increasing at a rate that is astonishing compared to any other forms of development. Load balancing seems a major challenge faced due to the architecture and the modular nature of our cloud environment. Loads need to be distributed dynamically to all the nodes. In this paper, we have introduced a technique that combines fuzzy logic with various nature inspired algorithms - grey wolf algorithm and firefly algorithm in order to effectively balance the load in a network of IoT devices. The performances of various nature inspired algorithms are compared with a brute force approach on the basis of energy efficiency, network lifetime maximization, node failure rate and packet delivery ratio.

Keywords— Load Balancing, IoT, Cloud Computing, Optimization, Bio-inspired Algorithms, Firefly Algorithm, GWO, Fuzzy Logic

Funding
No funding received.

Informed Consent Statement
None

Ethical Statement
None

Author Contribution
All authors provided critical feedback and equally contributed to shaping the research, analysis, and manuscript. Deepak Kumar Sharma, Jahanavi Mishra, Aeshit Singh, Raghav Govil, Dr. Krishna Kant Singh and Dr. Akansha Singh contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript. Deepak Kumar Sharma, Jahanavi Mishra, Aeshit Singh, Raghav Govil, Dr. Krishna Kant Singh and Dr. Akansha: Conceptualization, Methodology, Formal Analysis and Software. Deepak Kumar Sharma, Jahanavi Mishra, Aeshit Singh, Raghav Govil, Dr. Krishna Kant Singh and Dr. Akansha: Data Curation, Validation, Supervision, Writing - Review & Editing, and Project Administration.

Data Availability
The data that support the findings of this study are available from the corresponding author upon reasonable request.

Optimized Resource Allocation in IoT using Fuzzy Logic and Bio-Inspired Algorithms

Deepak Kumar Sharma[1], Jahanavi Mishra[1], Aeshit Singh[1], Raghav Govil[1], Krishna Kant Singh[2], Akansha Singh[3]
[1] Department of Information Technology, Netaji Subhas University of Technology, New Delhi, India.
[2] Faculty of Engineering & Technology, Jain (Deemed-to-be University), Bengaluru, India
[3] Department of CSE, ASET, Amity University Uttar Pradesh, Noida, India
Email: dk.sharma1982@yahoo.com, jahanavimishra7799@gmail.com, aeshit2000@gmail.com, raghavgovil17@gmail.com, krishnaiitr2011@gmail.com, akanshasing@gmail.com

***Abstract*— IoT smart devices are a confluence of microprocessors, sensors, power source and transceiver modules to effectively sense, communicate and transfer data. Energy efficiency is a key governing value of the network performance of smart devices in distributed IoT networks.Low and discrete power and limited amount of memory and finite amount of resources form some major bottlenecks in the workflow.Dynamic load balancing, reliability and flexibility are heavily relied upon by cloud computing for its accessibility.Resources are dynamically provided to the end client in an as-come on-demand fashion with the global network that is the Internet. Proportionally the need for services is increasing at a rate that is astonishing compared to any other forms of development. Load balancing seems a major challenge faced due to the architecture and the modular nature of our cloud environment. Loads need to be distributed dynamically to all the nodes. In this paper, we have introduced a technique that combines fuzzy logic with various nature inspired algorithms - grey wolf algorithm and firefly algorithm in order to effectively balance the load in a network of IoT devices. The performances of various nature inspired algorithms are compared with a brute force approach on the basis of energy efficiency, network lifetime maximization, node failure rate and packet delivery ratio.**

***Index Terms*— Load Balancing, IoT, Cloud Computing, Optimization, Bio-inspired Algorithms, Firefly Algorithm, GWO, Fuzzy Logic**

I. INTRODUCTION

Smart farming is a capital-intensive and hi-tech system of growing food cleanly and sustainable for the masses. Scaling up the process has been a huge market for innovation and establishments. IoT-based smart farming is highly efficient when compared with the conventional approach. In terms of environmental issues, IoT-based smart farming can provide great benefits including more efficient water usage, or optimization of inputs and treatments. This paper leverages technology by creating a fundamental digital environment to efficiently regulate and automate functions. This helps farmers to analyse better the state of their fields concerning the use of resources and equipment.

IoT networks consist of various smart devices deployed in a wide area. Each of these devices can be referred to as nodes. This leads to a network increasing in complex arrangement against regular wireless networks. The data is sensed by the sensors attached to each node of the IoT devices. An optimised mechanism for balancing data among various resources and its routing is essential since a fixed path from end-to-end between the source node and the destination node need not necessarily exist. Other restrictions in an IoT environment include limited wireless connectivity, number of sensors, amount of data being sensed and limitations in terms of resources such as limited battery energy, limited power and limited memory. This leads to an increase in processing time and issues such as inefficient load management, node failure, excessive energy consumption,etc.

In this paper, we have introduced a technique that combines fuzzy logic with various nature inspired algorithms such as firefly algorithm and grey wolf algorithm. The performances of these algorithms along with fuzzy logic are compared to a brute force approach. Initially a fuzzy logic controller is designed as per the Fuzzy Logic inference rules. Fuzzy logic is based on human-like decision making and has the properties essential for detecting chaotic behaviour. The Fuzzy Logic Controller (FLC) is implemented to categorize regions of observation into three classes of IOT devices and then optimized with the firefly and grey wolf algorithms so as to select the fittest nodes. We form the basis of our optimisation comparing it to a brute-force approach. We also compare the relative performances of the two nature inspired algorithms.

In our proposed model, we cluster the nodes into spatially correlated regions of interest (based on the similarity and mobility) using K-means and rather than collecting data or load from all the sensors, we select an optimum number of sensors. Low energy consumption is facilitated by avoiding collection of data from all spatially correlated nodes and doing an optimized selection of nodes. Establishing sufficient sensors' involvement in data detection and reporting helps in achieving high accuracy levels. The selection of clusters is performed through the Fuzzy Logic Controllers(FLC) which work on the Fuzzy Logic-Based Decision System (FLBDS). The fuzzy-based controlled decisions help in traffic rejection and have the potential to make optimisation decisions to maximise network lifetime and minimise energy consumption

without using any complex modeling in mathematics. The bio-inspired algorithm help with the selection of nodes from these clusters. The clusters which have updated data and maximum residual energy are allotted preference values using FLCs. The bio-inspired algorithms adjust the path of nodes on the basis of the value of their fitness function. The conditions for fitness are proposed in a manner so as to prevent exhaustion and subsequent failure of individual nodes. It does this by selecting nodes with maximum energy and updated data. This helps to prevent nodes from collapsing due to low residual energy. It also helps in preventing the transfer of redundant data as sensors with updated data are chosen each time.

In IoT, bio-inspired optimization techniques are used to handle challenges such as no existence of an end-to-end path from the source node to the destination node due to the changing network topology. These algorithms also help in preventing unnecessary energy consumption and subsequent failure of nodes due to exhaustive techniques. They give optimal solutions in less time and by consuming less energy.

The firefly algorithm mimics the nature and flashing behaviour of fireflies. Firefly algorithm has attracted attention in the last decade due to its low complexity as compared to other metaheuristic algorithms. Due to low complexity, the overall computation cost of this algorithm is also minimum. The grey wolf optimization (GWO) algorithm imitates the hunting mechanism and leadership ranking of grey wolves. The ranking of leadership in grey wolves is simulated in the form of the sets alpha, beta, delta, and omega. Additionally, there are steps to how the process of hunting/optimisation is carried out. These include hunting for prey, encircling the prey, and attacking prey, are implemented.

The paper is established as follows: Subsequent the introduction, Section 2 describes the motivation and goals of the work. Further in Section 3, the previous work has been explained. The proposed model along with all the key features has been elucidated in Section 4. The internal environment regulation unit has been elucidated in Section 4. Experimental results are discussed in Section 5. Section 6 throws light on the conclusion and future scope of the model.

II. MOTIVATION

There are limitations to how much processing can be done in a given amount of time because of physical bounds of hardware. The increasing need and demand for IoT devices by billions of users have further compounded the problem. This is the performance problem. There need to be multiple points of processing to minimise network failures. This is the availability problem. In order to avoid outages due to hardware failure, we need to run multiple instances of processing capable gateways, and be able to reroute traffic away from overloaded nodes as fast as possible. There are some cases where scaling vertically, that is maximising the number of processing devices is the right choice, but for the vast majority of smart devices, it's neither an economically procurable nor an implementation feasible choice. This is the economy problem, gaining a high enough return on investment while making the product consumer viable.

These adversities spawned the need for distributing workloads over existing infrastructure efficiently. Load balancing helps solve the performance, economy, and availability problems by providing a scaling out approach by generating the maximum possible efficiency out of the current hardware portfolio.

Load balancing is a scenario wherein the greedy approach of balancing is not sufficient. Hence we introduce the concept of fuzzy logic. Fuzzy logic is a mathematical concept wherein, instead of binary truth values (True or False), the ground truth values of any function may be any number greater than 0 and lesser than 1 (both inclusive). It's a many-valued approach to the traditional Boolean system, recognising the need for handling partial truths for problems where there isn't just a single computationally possible solution but a set of solutions which cater to different facets of the problem. These are meant to handle imprecise, incomplete or non-numerical information.

Load balancing is essentially an optimisation problem and this paper chooses nature inspired algorithms (or evolutionary algorithms) as one part of the two fold approach. Evolutionary algorithms do not have any set of assumptions for any fitness landscape. Hence they perform well under a wide variety of use cases. Simple implementations of these algorithms can solve complex problems easily. These algorithms are mostly genetic based algorithms, i.e. follow the concepts of mutation and principles of survival of the fittest. The fitter populations base their fitness on the calculation of a fitness function. The workflow of the algorithm is that it is supposed to carry the good solutions further down the genetic line, discarding the unimportant solutions and optimising the selected solutions further.

Our approach consists of combining nature inspired algorithms with fuzzy logic to originate an innovative method of load balancing for smart devices by factoring in practical intended usage aspects of the smart devices into filtering out the most efficient nodes that need balancing, and in turn efficiently balancing the data that needs processing.

III. RELATED WORK

Network optimization in the Internet of Things has gained massive attention due to a large increase in traffic from the IoT devices and things, as billions of IoT devices are expected to be pivotal in connecting global networks in the future years. Generally, the optimization of networks is defined as the technology used to improve and increase the performance of the network for various environments. A paper on network optimisation gives the guidelines of an optimization algorithm in IoT. The virtue of which can be elucidated in three major optimisation parameters. Firstly, load balancing of the data between nodes to improve network lifetime. Another requirement is an energy saving mechanism to minimise the consumption of energy by the node. Finally, a diverse principal set of devices, or the population participating in the network is

required to maintain the robustness and heterogeneity of the network.

In this work, we propose a Fuzzy Logic Based Decision System selection mechanism for correlated region selection. In a similar paper on Routing in Intrabody Nanonetworks[1], the authors argue that clustering guarantees the selection of correlated regions which contain maximum residual energy and updated data. These regions are used for data aggregation. It improves information accuracy and results in a stabilized energy consumption in the area.

An FLBDS was first proposed in 1965 [20]. Researchers have adopted them in numerous fields such as artificial intelligence and expert systems. FLBDS have an input, processing and an output stage. The mapping of inputs to membership functions is done in the input stage. Processing stage includes a set of rules, which gives an outcome for every input. The results of these rules are then combined. The combined result is converted back into a specific control output value [21]. Trapezoidal, triangular and bell curves [22] are the most commonly used membership functions. The system provides increased flexibility, low complexity and can be applied to an uncertain environment.

In the proposed approach, the devices have been clustered within spatially correlated regions. Rather than gathering data from all the devices, an optimum number of devices are chosen. Preventing data collection from all spatially correlated devices and selecting results in low energy consumption due to collection of data. The application of nature-inspired algorithms in engineering and the sciences has resulted in large communication benefits. Algorithms based on swarm intelligence like the ant colony optimization [18], grey wolf optimizer [7], firefly algorithm [9] are capable of managing complex problems with easy rules. In IOT, swarm intelligence based algorithms use the foraging practices of animals in order to solve complex optimization problems [19]. Firefly algorithm has low complexity as compared to other metaheuristic algorithms due to which it has garnered massive focus from the academic community in the previous decade [15]. Various applications have an efficient adaptation of the firefly algorithm because of its low computation time. Some examples are traveling salesman problem [17], digital image compression [19], multi-modal optimization [21] and security. The grey wolf optimizer, first proposed in [7] imitates the hierarchy and hunting behaviour of grey wolves in order to solve complex optimization problems.

It is observed that most of the above works have a single objective, and scheduling based on cost is seldom found. Therefore, we introduce a multi objective scheme on Load Balancing using Genetic Algorithms in a cloud environment to increase the resource utilization and energy efficiency, and reduce the makespan.

IV. PROPOSED MODEL

A. System Architecture

The entire scenario consists of a total number of 120 smart objects that are randomly distributed in a 250 x 250 m2 two-dimensional plane . Our network is based on the conventional cloud structure. The different devices in our network are classified into three categories to provide a variety of services in the network. These three categories are the receptive devices or sensors, IoT gateways or transmitting nodes and computational devices or base devices. Sensors lie in class 1. They receive the data and relay it to the transmitting devices. These nodes can only transmit data to a limited range. The class 2 nodes are the IOT gateways. They receive data from the receptive devices in their range. The class 3 objects are very intelligent. Their range of transmission is also high enough to forward the data packets over the internet. Of the 120 nodes, around 54 belong to class 1, 36 belong to class 2 and 30 belong to class 3. These numbers are chosen taking into account the processing power of nodes.

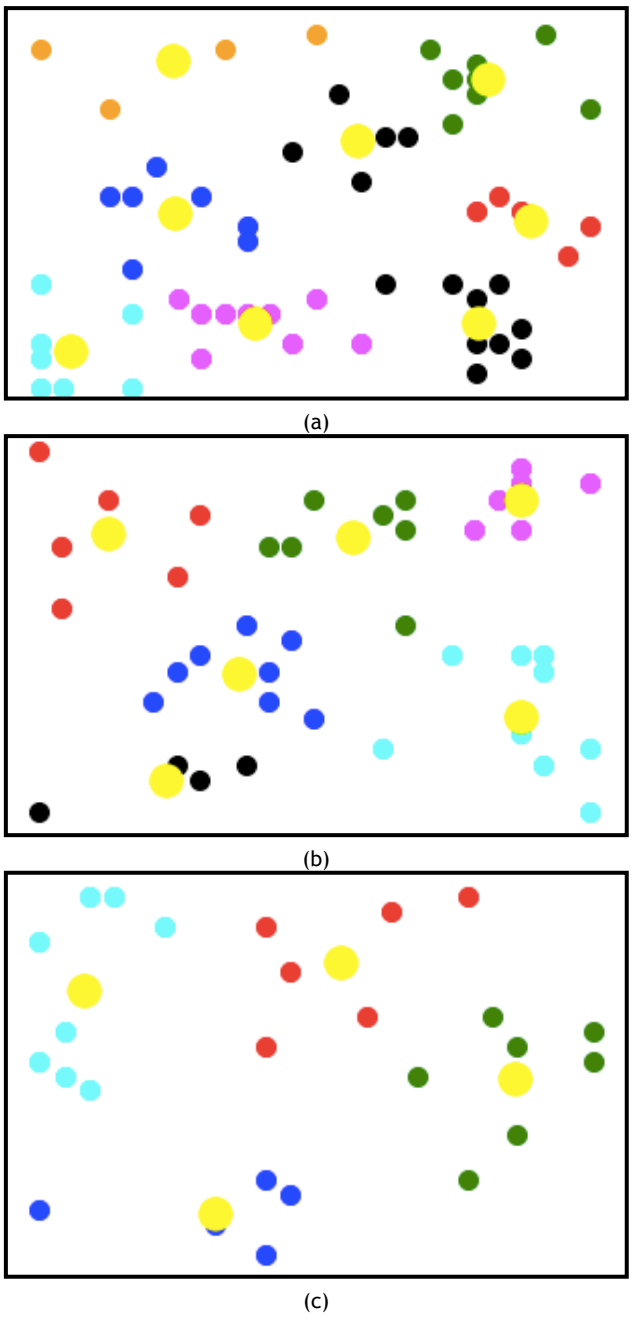

Figure 1. Clusters of different classes of nodes. (a) Clusters of Class 1 (b) Clusters of Class 2 (c) Clusters of Class 3

B.System Model

Initially the nodes are clustered and coagulated into different regions based on their class and their proximity to each other. The clustering scheme used is K-Means++ clustering. K-Means++ is an advanced version of the conventional K-Means clustering which uses a procedure to initialise the cluster centres before proceeding with the conventional K-means clustering. Since our nodes are mobile, the nodes are clustered after an interval 'x'. The value of 'x' depends on the mobility of nodes. Our nodes are not highly mobile, so we set a larger value for x.

The regions which have updated load and adequate energy are selected from the total number of clusters — customised during network setup time. Since our nodes are mobile, we recluster our nodes after a time period, x. Only the regions that have high residual energies, larger number of nodes, low mobility and minimum current load are selected using the FLDBS. For the input parameters, we have used the linguistic variables High (H), Medium (M) and Low (L). For the output parameters, we have used the linguistic variables Very High (VH) High (H), Medium (M) and Low (L) and Very Low (VL). The membership function used for fuzzification is the triangular membership function. The Mamdani Centroid Technique is used for defuzzification. Selection of clusters aids in achieving the ultimate goal of load balancing with efficient energy consumption. Now the further process is performed on these selected regions of interest. Avoiding load balancing in regions that don't have updated data helps in avoiding unnecessary data transmission and subsequently saves energy. The selection of regions is done based on their preference values.

The value of the membership function for fuzzy variables is calculated according to the graphs of membership function as visible in Figure 3. The x-axis describes the crisp values of fuzzy variables and the y-axis describes the value of the membership function.

Next, the selection of path for the data packet is performed using genetic algorithms in order to ensure reduced transmission of redundant data and prolonged network lifetime. Since the assumption is made that the nodes or sensors located in a cluster are spatially correlated and process similar kinds of data, therefore we can minimize the aggregation of redundant information by using a genetic algorithm to choose the path from selected clusters. For selecting the best path, fitness values of the paths are used. The fitness function designed makes certain the choice of the combinations of nodes for load balancing that have low mobility, highest residual energy and have updated information. The fitness function helps to increase network lifetime and increase the likelihood of efficient load balancing between similar devices. Restricting the participation of sensors in data collection can lead to an increased network lifetime and preservation of network resources that might otherwise be used in transmission of redundant data. The genetic algorithms proposed optimize the path selection for less energy consumption and decrease in redundant data transmission.

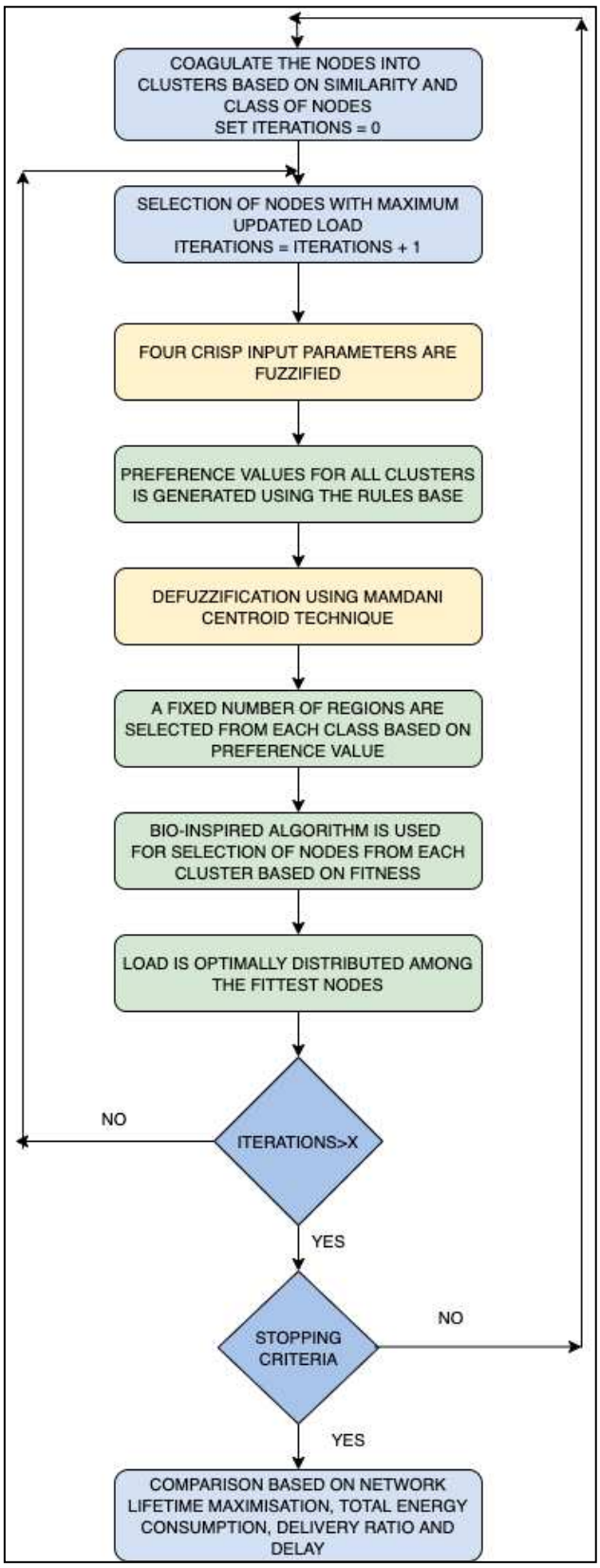

Figure 2. Flowchart of the workflow

C. Fuzzy Logic Based Decision System

The Fuzzy Logic Controller (FLC) comprises three stages, namely - input stage, processing stage and the output stage. For the input parameters, we have used the linguistic variables High (H), Medium (M) and Low (L). For the output parameters, we have used the linguistic variables Very High (VH) High (H), Medium (M) and

Low (L) and Very Low (VL). The membership function used for fuzzification is the triangular membership function. The Mamdani Centroid Technique is used for defuzzification.

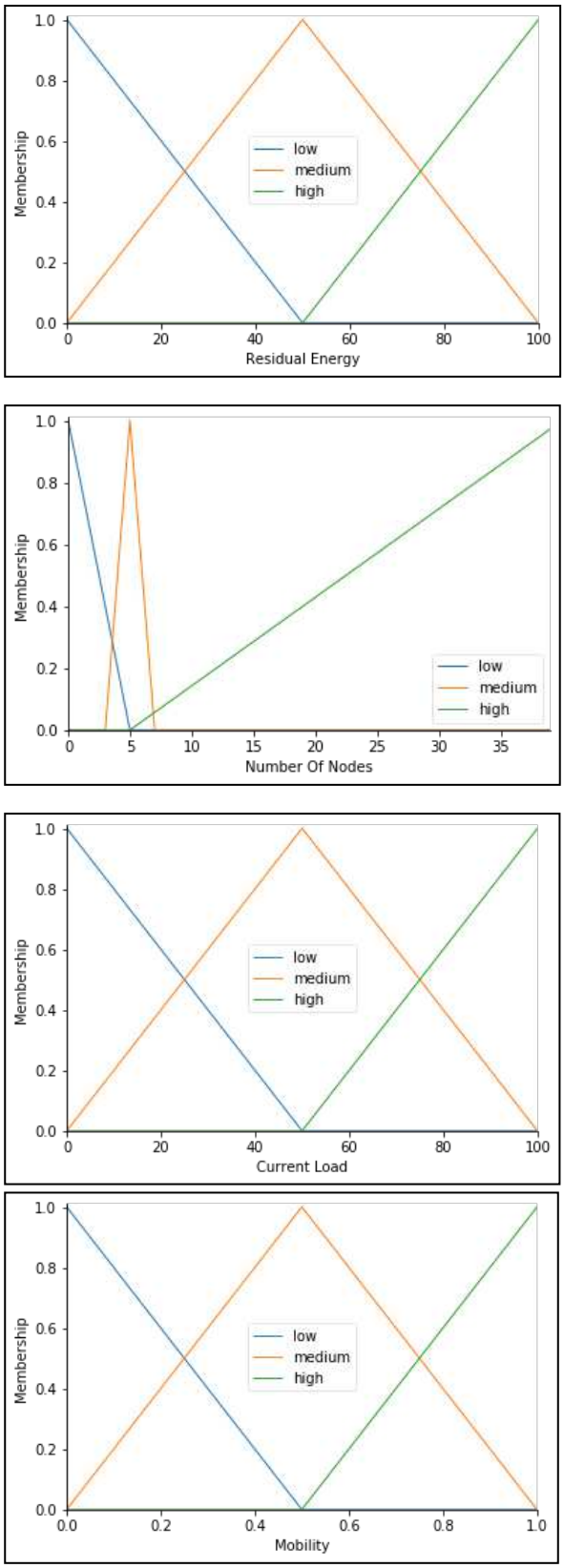

Figure 3. Input variables "Residual Energy","Number of Nodes", "Current Load" and "Mobility" used for fuzzification

The input parameters to the fuzzy logic controller are updated load, residual energy of cluster, mobility, and load handling capacity of cluster. The linguistic variables used for the input parameters are - Low (L), Medium (M), High (H). Triangular membership function is used to design the controller. The input and output variables are defined on a normalized domain of [0, 1] and are passed on to the nature inspired algorithms for selection of a cluster based on preference value. The output variable is the Preference Value of the cluster.

D. Firefly Algorithm

The Firefly Algorithm (FA) is a meta-heuristic approach, inspired by the behaviour and flashing patterns of firefly in various contexts. It is a self-adaptive and low complexity approach that helps design simplified routing and optimization solutions. Fireflies make use of their flashlight to attract other fireflies for foraging, mating, sharing food and communication. The attractiveness of a firefly is determined by its flashing pattern, along with its rate and rhythm. Given a population, the firefly algorithm can be used to find the globally optimal solution. The Firefly Algorithm is based on the following three principles: (1) All fireflies are considered to be unisex. This implies that a firefly is attracted towards another firefly irrespective of its sex. (2) Attractiveness of a firefly is directly proportional to its intensity or brightness, therefore, given two fireflies, the less brighter one will advance in the direction of the brighter one. Given two fireflies, as one moves farther from another i.e. the distance between the two fireflies increases, it appears less attractive to the first firefly. This is because, brightness of a firefly is directly proportional to its attractiveness and they decrease with an increase in distance. If no firefly brighter than the chosen firefly exists, the chosen firefly moves randomly. (3) The brightness or intensity of a firefly is ascertained by the landscape of the defined objective function. For instance, the brightness can be directly proportional to the value of the objective function for a maximization problem.

Just as brightness is used as a parameter in Firefly Algorithm, the fitness function can be defined similarly in various genetic algorithms. The steps of the firefly algorithm are defined as follows: At the start of each iteration, the initial population, initial attractiveness and solution set of the fireflies is set. The fitness function is calculated in order to evaluate the fitness of each solution. The attractiveness between fireflies is calculated. Consider 2 fireflies - i and j. Let firefly j have maximum attractiveness value. Initiate and facilitate the movement of firefly i towards firefly j. Rank the population as per the fitness values.Evaluate the current best solution. Update the global best solution if required. The attractiveness of a firefly, also called the light intensity I(r), is represented according to the following equation:

$$I(r) = I_o e^{-yr2} \quad (1)$$

$$B(r) = B_o e^{-yr2} \quad (2)$$

where 0 is the attractiveness at r = 0. The movement of a firefly from position i to j at a given time is given as

$$X_i (t + 1) = X_i (t) + B_o e^{-yr2} (Xi - Xj) + xE_i \quad (3)$$

$$d = (X_i - X_j) = m \sum k=1 (Xi - Xj)^2 \quad (4)$$

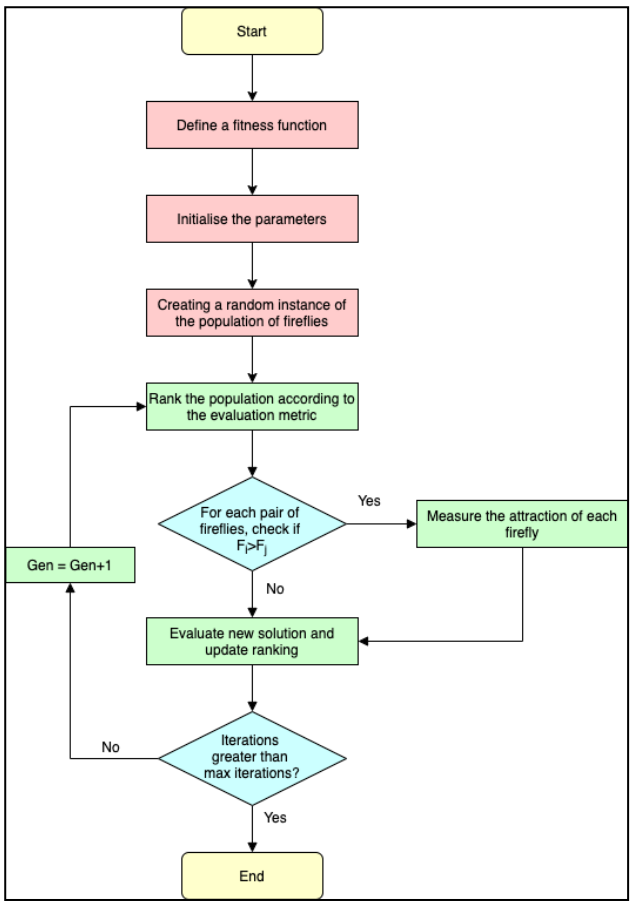

Figure 4. Flowchart of the Firefly Algorithm

The steps for the proposed firefly algorithm for optimal selection of sensors are described below:

1. **Solution Initialization:** In the firefly algorithm, each firefly represents a probable solution. Likewise, in path selection, each firefly signifies the various paths available. Each solution represents a path that the nodes could adopt going from class 1 to class 3. The solution is first initialised randomly. So multiple paths are randomly initialized and each path consists of nodes only from the top clusters that are generated using fuzzy logic. For example, if a node of class 1 with node id 7 is trying to balance its load, it will initialise multiple random solutions as: 7 → 60 → 102, 7 → 77 → 110, 7 → 80 → 115, 7 → 57 → 100 where nodes with ids 60, 77, 80 and 57 belong to class 2 and nodes with ids 102, 110, 115 and 100 belong to class 3.

2. **Fitness function calculation:** The intensity of light influences the brightness of a firefly. This value represents the firefly's fitness. The value of the fitness function is determined by three factors: residual energy of the sensor, distance from the router and the degree of mobility.

(a) Residual Energy: Selecting sensors with higher residual energy aids in extending the network lifetime. Hence, choosing nodes with higher energy present is preferred.

(b) Distance: Transmission of data across shorter distances requires low energy consumption and subsequently an increase in the network lifetime. Therefore, sensors located at the least possible distance from the router have a higher fitness value.

(c) Mobility: The degree of mobility represents how mobile the node is. It is a value between 0 and 1. Based on the above parameters, the value of the fitness function of a path is calculated by

$$\text{Fitness} = x_1(E_i) + [x_2\ (1/\ d(r, q))] + x_3\ [1/\text{mobility}] \qquad (5)$$

where x1 , x2 and x3 are used to balance the weights of three parameters and Ei represents the cumulative energy of the nodes of class 2 and class 3 in the path, mobility represents the cumulative mobility of the nodes of class 2 and class 3 in the path and d(r,q) represents the total distance of the path. For eg, if the path is 7→77→110, it represents distance from 7→77 + distance from 77 → 110.

$$x\ (0, 1) \text{ and } x1 + x2 + x3 = 1. \qquad (6)$$

3.**New Solution Updation:** After calculating the brightness of all solutions, the less bright firefly flies towards the brighter firefly. The position of the next firefly can be updated using where FFi and FFj are two fireflies, x generates random values between [0,1] and represents the randomization parameter and random function.

E.Grey Wolf Algorithm

Grey wolves are known to thrive in packs of size 6-12. They are considered as apex predators.

The leaders of the pack are called alphas or dominant wolves, they may be a male or female. The leaders make decisions regarding sleeping place, hunting, wake up time, etc. The pack is given these orders. However, sometimes the alpha follows the other members too, in a democratic manner. The whole pack acknowledges the alphas by holding their tails down. Alphas are only permitted to mate within the same pack. The alphas need not necessarily be the most powerful members of the pack, but they are the best at managing the pack. This implies that the discipline and organization of a pack is of utmost importance, a lot more than the strength of the pack.

The second level in the hierarchy is the beta wolves, which can be either male or female. These wolves are subordinates to the alphas and assist them in activities such as decision making. The betas are the best candidate to replace an alpha in the event of passing of an alpha. The beta commands the wolves that are lower in the hierarchy and plays the role of a discipliner for the pack. They give feedback to the alphas and act as their advisors. The grey wolf that ranks last in the hierarchy is the omega which acts as a scapegoat. These always have to obey all the dominant wolves and are allowed to eat last. Although the omega might not appear important, conflicts and issues have often been observed in the pack when an omega is lost. This is due to the violence and venting of frustration of the dominant wolves on the omega for satisfaction of the dominant wolves. The omegas may also act as babysitters of the pack.

Wolves that do not belong to the alpha, beta and omega categories are called the subordinate or delta wolves. This

category consists of elders, caretakers, hunters, sentinels and scouts. They command the omegas but have to obey the alphas and betas. The scouts guard the boundaries of the territory. They warn the pack in the event of any danger. Sentinels guard and protect the pack's safety. Elders are the experienced wolves that used to previously be alpha or beta. The caretakers care for the ill, weak and wounded wolves. Hunters help the alphas and betas in providing food for the pack by hunting. They track, chase and approach the prey. They pursue, encircle and harass the prey until it stops moving. Once the prey is stationary, they attack it. Algorithm and Mathematical Model: The social hierarchy and hunting behaviour of the grey wolves can be mathematically modelled as given below. Social Hierarchy The fittest solution is considered as alpha (a), the second fittest solution is called beta (b), and the third fittest solution is called delta (d). The remaining solutions are said to be omega (w). Here, the hunting or optimization is facilitated by a, b, and d wolves. The w wolves obey. Encircling Prey Grey wolves encircle their prey while hunting.

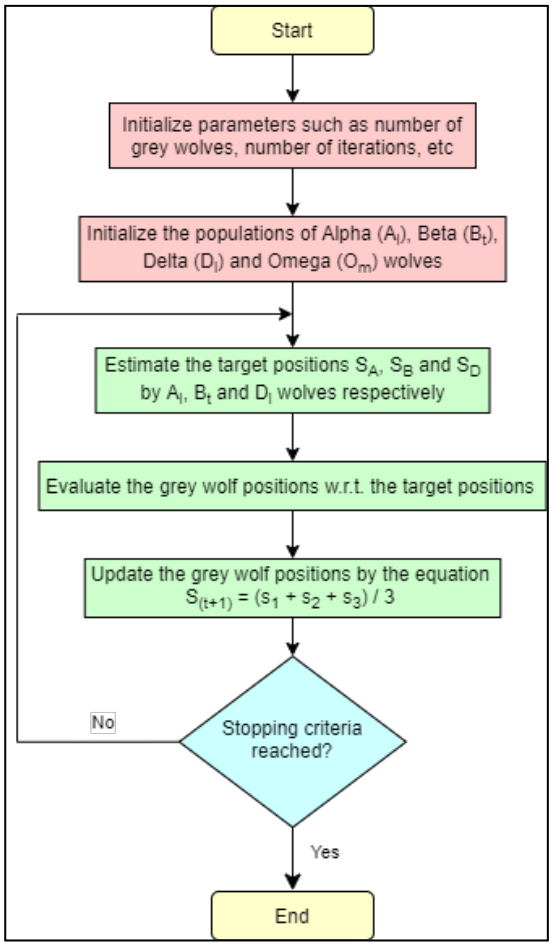

Figure 5. Flowchart of the GWO Algorithm

F. Features of the Model

- Efficient - Since our algorithm involves metaheuristics, it is computationally more efficient than brute force algorithms that perform exhaustive computations to provide optimal results.
- Scalability - The model is able to handle a large network of connected devices as the increase in number of devices won't affect the performance due to increased time complexity.
- Performance - The model will perform optimally on small as well as large architectures due to its scalable nature.
- Robustness - The system is robust in nature. However, failure of one or more components of the architecture may lead to failure of the system.
- Quick Decision Making - Our algorithm helps arrive at faster decisions due to the use of metaheuristics for optimization instead of exhaustive search mechanisms.
- Portable - The code is lightweight and hence can be easily migrated to other systems.
- Fairness of Service - There is fairness of service as heavily loaded nodes are not simply selected to get all the benefits. Rather the best path is selected for the data packet.

V. RESULT ANALYSIS

To implement the proposed architecture, Python has been used due to its high level nature, readability and portability across operating systems. Python requires relatively fewer lines of code to express a logic as compared to other languages like Java, C++ or C. The large, comprehensive set of libraries give an added advantage. In this project, we have used Python version 3.7.6. Multidimensional arrays have been processed using the NumPy library. NumPy consists of various functions that are used to perform seamless mathematical and logical operations on multidimensional arrays. For the analysis and manipulation of dataset, Pandas library is used. It consists of a combination of various programming methodologies and implementations to provide quick and tolerant data structures that make working with structured and time series data both enriching and easy. .It is an intuitive and open source library. Visualisations and graphs have been created using the Matplotlib library. It is a cross-platform library used for making two-dimensional plots from data in arrays. To implement the fuzzy functions for fuzzification and defuzzification, Skfuzzy library is used. a fuzzy logic package for Python. It includes a toolbox for many implementations, useful functions and tools of fuzzy logic algorithms for computation and projects. To simulate the networking environment and the incoming data packets on various sensor nodes, we have used a sample dataset. The dataset consists of 80,000+ entries that have been collected over a period of 24 hours. Every row gives us the timestamp at which a particular data packet enters the system, the size of the data packet and the address of the destination node by which it is sensed. The load balancing dataset was obtained from IoTanalytics.unsw.edu.au and has been used in various IEEE papers.

A. Residual Energy

Residual energy is interpreted as total energy present in the network at the end of a cycle. Initially, each of our 120 sensor nodes have 100 units of residual energy. Hence the total residual energy at the start of our simulation is 12000 units. As the simulation progresses, and the load is distributed, the residual energy of our network changes over time. We have evaluated residual energy of the network after each timestamp. Figure 6 shows that the residual energy of our network decreases as time progresses. Employing a bio-inspired algorithm in the proposed model tends to slow down the decrease in the residual energy of the network. The graph also demonstrates that both the Firefly Algorithm (FFA) and

Grey Wolf Optimizer (GWO) have a relatively similar effect on the residual energy of the network.

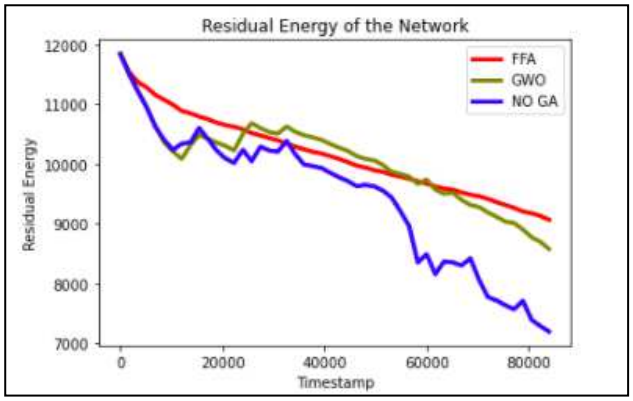

Figure 6. Residual Energy of the network in different implementations vs Timestamp

B. Number of Alive Nodes

A sensor node is said to be 'alive' when it has sufficient residual energy to participate in the load balancing process. Once the residual energy of a node decreases below a threshold level, it is no longer able to handle any load, and is said to be 'dead'. Initially, all of our 120 sensor nodes are alive. As the simulation progresses and the load is distributed, the residual energies of the individual sensor nodes start to decrease, which results in the death of some of the nodes. From Figure 7, it can be seen that when no bio-inspired algorithm is employed with the proposed model, less number of sensor nodes are still alive at the end of the simulation. The graph also depicts clearly that using the Firefly Algorithm (FFA) with the proposed model leads to a greater number of alive sensor nodes at the end of the simulation. However, using Grey Wolf Optimizer (GWO) with the proposed model does not have that much of an improvement with respect to the number of alive nodes.

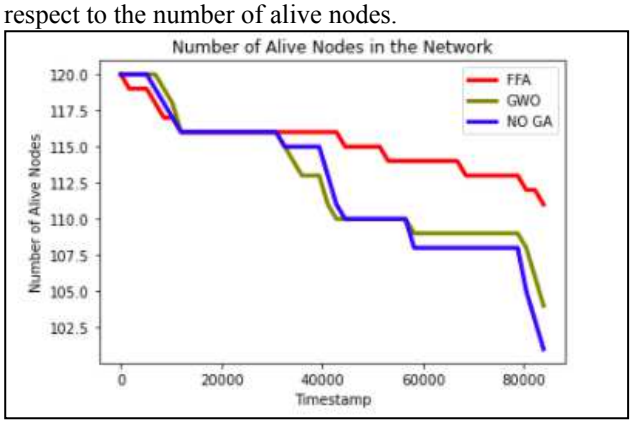

Figure 7. Number of Alive Nodes in the network in different implementations vs Timestamp

C. Packet Delivery Ratio

Packet Delivery Ratio is a measure of the successful reception of data packets across the network. A greater number of successfully received data packets is desired because it effectively leads to an increased packet delivery ratio. Figure 8 shows that when a bio-inspired algorithm is employed with the proposed model, the packet delivery ratio of the network stays at full capacity for the first half of the simulation, but decreases in the second half. When the Firefly Algorithm (FFA) is employed, the decrease in packet delivery ratio of the network is gradual, in contrast to the case when Grey Wolf Optimizer (GWO) is employed. In the case of GWO, the decrease in packet delivery ratio is more significant, but is still better than the case in which no bio-inspired algorithm is used.

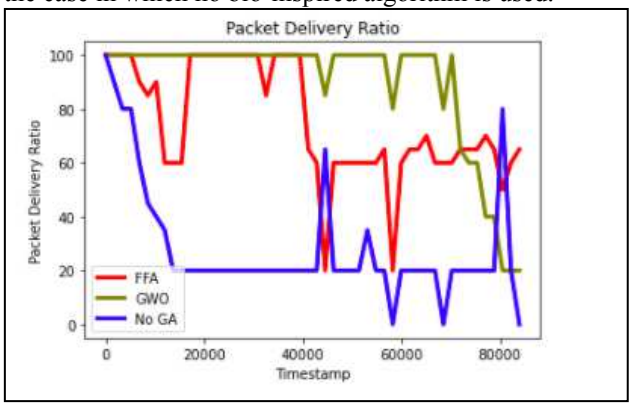

Figure 8. Packet Delivery Ration of the network in different implementations vs Timestamp

D. Average End-to-End Delay

Average end-to-end delay is defined as the total time taken by a data packet to reach from its source to its destination. It is the time that a data packet takes from its transmission till the reception. From Figure 9, it can be seen that the average end-to-end delay in the network is less when a bio-inspired algorithm is employed with the proposed model. The graph also demonstrates that the Firefly Algorithm (FFA) and the Grey Wolf Optimizer (GWO) have a relatively similar effect on the average end-to-end delay.

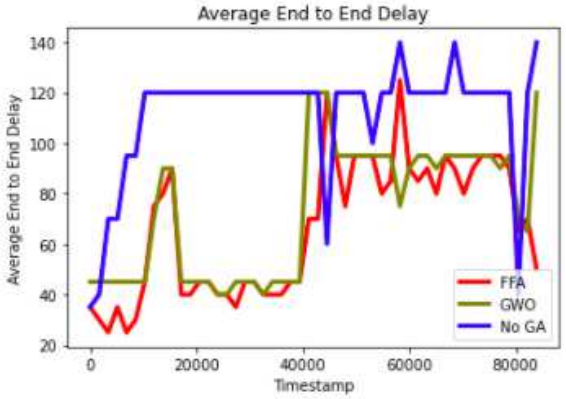

Figure 9. Average End-to-End Delay of the network in different implementations vs Timestamp

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

In our proposed approach we have used metaheuristics and FLDBS to optimize resource allocation and facilitate load balancing in IoT. Clustering the devices into nodes helped prevent redundant data from repeatedly entering the network. Using bio-inspired algorithms, we were able to arrive at optimal solutions and sub-optimal solutions in less time. Since the solutions of the path that the data packet has to be followed can lead to a combinatorial explosion in case of a larger network, our approach would prove to be much more scalable due to its heuristic nature. Additionally, since we do not exploit every single node to select the best node, we are saving energy and

subsequently improving other parameters such as the packet delivery ratio and average end-to-end delay in the network. This improvement occurs as we do not exploit every single node of the next class by considering it as a candidate to balance the load on, rather we select the nodes from the topmost clusters generated using FLDBS. These nodes are then used to create a path for the transfer of load.

B. Future Work

Until now we have run our algorithm for an IoT network with 120 devices. Further, we would like to implement and analyse our proposed approach on a more vast network using multiple bio inspired algorithms. We would also like to compare the performance of different categories of metaheuristics such as tabu search, simulated annealing, etc.

References

1. *N.N. Srinidhi, S.M. Dilip Kumar, K.R. Venugopal, Network optimizations in the Internet of Things: A review, Engineering Science and Technology, an International Journal, Volume 22, Issue 1, 2019, Pages 1-21, ISSN 2215-0986, https://doi.org/10.1016/j.jestch.2018.09.003.*

2. *Kaur G., Kaur K. (2017) An Adaptive Firefly Algorithm for Load Balancing in Cloud Computing. In: Deep K. et al. (eds) Proceedings of Sixth International Conference on Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing, vol 546. Springer, Singapore. https://doi.org/10.1007/978-981-10-3322-37*

3. *B. Shah, "Fuzzy Energy Efficient Routing for Internet of Things (IoT)," 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, 2018, pp. 320-325, doi: 10.1109/ICUFN.2018.8437033*

4. *K. Rajeswari and P. Lakshmi, "PSO Optimized Fuzzy Logic Controller for Active Suspension System," 2010 International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, 2010, pp. 278-283, doi: 10.1109/ARTCom.2010.22.*

5. *Khattak, Muhammad. (2019). Fuzzy Logic and Bio-Inspired Firefly Algorithm Based Routing Scheme in Intrabody Nanonetworks. Sensors. 19. 10.3390/s19245526.*

6. *G. Krishna Lava Kumar, Thandu Nagaraju, U. Veeresh, "FUZZY-PSO FOR CLASSIFICATION OF TRUSTED IoT DEVICES", IJAST, vol. 29, no. 9s, pp. 559-569, Apr. 2020*

7. *Florence, Paulin Shanthi, V.. (2014). A load balancing model using firefly algorithm in cloud computing. Journal of Computer Science. 10. 1156-1165. 10.3844/jcssp.2014.1156.1165*

8. *Tapale, Manisha Goudar, Rayangouda Birje, Mahantesh Patil, Rudragouda. (2020). Utility based load balancing using firefly algorithm in cloud. Journal of Data, Information and Management. 2. 10.1007/s42488-020-00022-2.*

9. *Sharma, D.K., Rodrigues, J.J.P.C., Vashishth, V. et al. RLProph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic IoT networks. Wireless Netw 26, 4319–4338 (2020). https://doi.org/10.1007/s11276-020-02331-1*

10. *Mirjalili, Seyedali Mirjalili, Seyed Lewis, Andrew. (2014). Grey Wolf Optimizer. Advances in Engineering Software. 69. 46–61. 10.1016/j.advengsoft.2013.12.007*

11. *Mohamed, Mohamed Abbas, Mustafa. (2019). Design Interval Type-2 Fuzzy Like (PID) Controller for Trajectory Tracking of Mobile Robot. International Journal of Computers, Communications Control (IJCCC). 19. 1-15. 10.33103/uot.ijccce.19.3.1*

12. *M. Padmavathi, S. M. Basha and V. V. J. R. Krishnaiah, "Load Balancing Algorithm to Reduce Make Span in Cloud Computing by Enhanced Firefly Approach," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 896-900, doi: 10.1109/ICESC48915.2020.9155662.*

13. *Chen, M.; Gonzalez, S.; Vasilakos, A.; Cao, H.; Leung, V.C. Body area networks: A survey. Mob. Netw. Appl. 2011, 16, 171–193*

14. *Zadeh, L.A. Fuzzy sets. Inf. Control 1965, 8, 338–353*

15. *Klir, G.J.; Yuan, B. Fuzzy Sets and Fuzzy Logic: Theory and Applications; Prentice-Hall: Upper Saddle River, NJ, USA, 1995; p. 563.*

16. *Tzeng, G.H.; Huang, J.J. Fuzzy Multiple Objective Decision Making; Chapman and Hall/CRC: Boca Raton, FL, USA, 2016*

17. *Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS'95, Nagoya, Japan, 4–6 October 1995; pp. 39–43.*

18. *Dorigo, M.; Birattari, M. Ant Colony Optimization; Springer: Berlin/Heidelberg, Germany, 2010.*

19. *Yang, X.S. Firefly algorithms for multimodal optimization. In Proceedings of the International Symposium on Stochastic Algorithms, Sapporo, Japan, 26–28 October 2009; pp. 169–178.*

20. *Pavai, K.; Sivagami, A.; Sridharan. Study of routing protocols in wireless sensor networks. In Proceedings of the 2009 IEEE International Conference on Advances in Computing, Control, and*

Telecommunication Technologies, Trivandrum, Kerala, 28–29 December 2009; pp. 522–525

21. *Yang, X.S.; He, X. Firefly algorithm: Recent advances and applications. arXiv 2013, arXiv:1308.3898.*

22. *Horng, M.H. Vector quantization using the firefly algorithm for image compression. Expert Syst. Appl. 2012, 39, 1078–1091.*

23. *Kumbharana, S.N.; Pandey, G.M. Solving travelling salesman problem using firefly algorithm. Int. J. Res.Sci. Adv. Technol. 2013, 2, 53–57. 24. Kazemzadeh Azad, S. Optimum design of structures using an improved firefly algorithm. Iran Univ. Sci. Technol. 2011, 1, 327–340.*

Deepak Kumar Sharma is working as an Assistant Professor in the Department of Information Technology, Netaji Subhas University of Technology (Formerly NSIT), Dwarka, New Delhi, India. He obtained his PhD in Computer Engineering from University of Delhi, India in 2016. His research interests include opportunistic networks, wireless ad hoc and sensor networks, Software Defined Networks and IoT Networks. He has over 15 years of experience in Academics. He has published various research papers in reputed international journals like ETT Wiley, IEEE Systems Journal, Computer Communication Elsevier, IJCS Wiley etc. and conferences of repute like IEEE AINA, GLOBECOM etc. He has also authored various book chapters in edited books of IET, Wiley, Springer, Elsevier etc. He is also a reviewer of various reputed journals like ETT Wiley, AIHC Springer, IJCS Wiley etc.

Jahanavi Mishra is pursuing a degree in Bachelors of Engineering in Information Technology at the Netaji Subhas University of Technology (Formerly NSIT), New Delhi, India. Her past research interests revolve around Load Optimisation, Opportunistic Networks and Computer Networks. She remains interested in researching the application of artificial intelligence and Reinforcement Learning in these categories. She is currently working as a Software Engineer at Goldman Sachs.

Aeshit Singh is currently pursuing is undergraduate Bachelor's degree in Information Technology from Netaji Subhas University of Technology (Formerly NSIT), New Delhi, India. His research interests are in the areas related to Computer Networks, Internet of Things and application of Machine Learning techniques in these areas. His specific interests are application of Deep Learning Algorithms in these research areas. He is currently working as a Software Engineer at Publicis Sapient.

Raghav Govil is pursuing a degree in Bachelors of Engineering in Information Technology at the Netaji Subhas University of Technology (Formerly NSIT), New Delhi, India. His research interests lie in Machine Learning Applications for Computer Networks and Information Extraction and Retrieval. He is currently working as a Software Engineer t Salesforce.

Dr. Krishna Kant Singh is working as Professor in CSE Department, Jain University, Bengaluru, India. He has wide teaching and research experience. Dr. Singh has acquired B.Tech, M.Tech, PGD (ML&AI) (IIIT Bangalore) and Ph.D. (IIT Roorkee) in the area of image processing and remote sensing. He has authored more than 70 research papers in Scopus and SCIE indexed journals of repute. He has also authored 25 technical books. He is also an associate editor of Journal of Intelligent & Fuzzy Systems (SCIE Indexed), IEEE ACCESS (SCIE Indexed) and Guest Editor of Open Computer Science, Complex and Intelligent System, Microprocessors and Microsystems Journal. He is also member of Editorial board of Applied Computing & Geoscience (Elsevier).

Dr. Akansha Singh is B.Tech, M.Tech and PhD in Computer Science. She received her PhD from IIT Roorkee in the area of image processing and machine learning. Currently, she is working as Associate Professor in Department of CSE, ASET, Amity university Noida. She has to her credit more than 70 research papers, 20 books and numerous conference papers. She has also national and International Patents in the field of machine learning. She has been the editor for books on emerging topics with publishers like Elsevier, Taylor and Francis, Wiley etc. Dr. Singh has served as reviewer and technical committee member for multiple conferences and journals of High Repute. She is also the Associate Editor for IEEE Access journal, Guest editor for journals like Complex & Intelligent Systems, Journal of Real Time Image processing etc. Dr. Singh has also undertaken government funded project as Principal Investigator. Her research areas include image processing, remote sensing, IoT and machine learning.