RESEARCH ARTICLE OPEN ACCESS

Optimizing Energy Efficiency and Quality of Service in Cluster-Based Iot Networks: A Comparative Analysis of Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, Lion Optimization, And Firefly Algorithms

S. Bharathi [1], Dr. D. Maruthanayagam [2]

[1] Research Scholar, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India
[2] Dean Cum Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India.

ABSTRACT

The proliferation of Internet of Things (IoT) devices has underscored the need for efficient routing strategies to enhance energy efficiency and quality of service in network communications. This paper investigates the performance of five optimization algorithms Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Lion Optimization, and Firefly Algorithms in a cluster-based IoT network architecture. We evaluate these algorithms based on key performance metrics including End-to-End Delay, Packet Delivery Ratio (PDR), Routing Overhead, Throughput, Energy Consumption, Scalability, Idle Listening Time and Latency. Using a simulation-based approach, we analyze and compare the effectiveness of each algorithm in improving routing efficiency and network performance. This paper results highlight the strengths and limitations of each algorithm, offering insights into their suitability for various IoT applications. This investigation provides a comprehensive evaluation of how these optimization techniques can be leveraged to address the challenges of energy consumption and service quality in IoT networks, paving the way for more effective and scalable IoT solutions.

Keywords: Internet of Things (IoT), Cluster-Based IoT Network , Optimization Algorithms , Genetic Algorithms (GA) , Ant Colony Optimization (ACO) , Particle Swarm Optimization (PSO), Lion Optimization ,Firefly Algorithms , Energy Efficiency and Quality of Service (QoS).

1. INTRODUCTION

The Internet of Things (IoT) represents a transformative shift in technology, connecting billions of devices across diverse domains from smart homes to industrial applications. The efficient management of these devices relies heavily on the underlying network infrastructure**, particularly in routing and data transmission.** As IoT networks expand, traditional routing strategies struggle to meet the growing demands for energy efficiency and quality of service (QoS). Addressing these challenges is crucial for sustaining the performance and scalability of IoT systems. Routing in IoT networks involves navigating complex topologies with varying node densities and energy constraints [1] [2]. **The efficiency of routing algorithms directly impacts key performance metrics such as energy consumption, latency, and packet delivery ratio**. Consequently, optimizing these algorithms is essential for ensuring that IoT networks operate efficiently and reliably [3].

In cluster-based IoT networks, where nodes are organized into **clusters to enhance communication efficiency and manageability, routing strategies must balance multiple factors**. These include minimizing energy consumption, reducing end-to-end delay, maximizing throughput, and ensuring high packet delivery ratios [4]. Traditional routing approaches often fail to adapt to dynamic network conditions and energy constraints effectively. Recent advancements in optimization techniques offer promising solutions to these challenges. Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Lion Optimization, and Firefly Algorithms have demonstrated potential in various optimization contexts. However, their comparative effectiveness in the specific context of IoT routing has not been thoroughly explored.

This paper seeks to address this gap by evaluating these algorithms in terms of their impact on QoS parameters in a cluster-based IoT network.

The primary objective of this research is to evaluate and **compare the performance of GA, ACO, PSO, Lion Optimization, and Firefly Algorithms in optimizing routing** within a cluster-based IoT network. This analysis focuses on key performance metrics, including End-to-End Delay, Packet Delivery Ratio (PDR), Routing Overhead, Throughput, Energy Consumption, Idle Listening Time, Scalability, and Latency. By analyzing these metrics, **this paper aims to identify the most effective optimization techniques for enhancing energy efficiency and overall service quality in IoT networks**.

This research makes several significant contributions to the field of IoT network optimization:

- **Comparative Analysis:** Provides a detailed comparative analysis of five advanced optimization algorithms in the context of cluster-based IoT routing.
- **Performance Metrics:** Evaluates the algorithms based on a comprehensive set of QoS parameters, offering insights into their strengths and limitations.

- **Practical Insights:** Offers practical recommendations for selecting and implementing optimization algorithms to improve IoT network performance.
- **Future Directions**: Identifies potential areas for future research and development in IoT routing optimization.

By addressing these aspects, the paper aims to advance the understanding of how different optimization techniques can be applied to enhance the efficiency and reliability of IoT networks.

II.SYSTEM MODEL

2.1. Cluster-Based IoT Networks

Cluster-based IoT networks are a prevalent architecture designed to enhance the efficiency and scalability of large-scale IoT deployments. In this architecture, the network is organized into clusters, each managed by a central node known as the **Cluster Head (CH)**. The remaining nodes within a cluster are referred to as **Member Nodes (MNs)**. This hierarchical organization simplifies the management of network resources, reduces energy consumption, and improves data aggregation and transmission efficiency [5].

The main advantages of cluster-based architectures include:

- ***Reduced Communication Overhead:*** By aggregating data at the cluster level before transmission, **cluster-based networks reduce the number of messages sent** to the base station or gateway.
- ***Energy Efficiency:*** **CHs handle data collection and routing tasks**, while MNs primarily focus on sensing and transmitting data to the CH, which can extend the battery life of individual nodes.
- ***Scalability:*** Clustering helps manage network size by **reducing the number of direct communications between nodes**, making it easier to scale the network as the number of devices increases.

Cluster formation in IoT networks is typically dynamic and involves several key steps [6]:

- ***Cluster Formation:*** Nodes determine their roles (CH or MN) based on predefined criteria, such as node energy levels, proximity, and network density. Various algorithms, such as LEACH (Low-Energy Adaptive Clustering Hierarchy), are employed to form clusters and elect CHs.
- ***Cluster Maintenance:*** **To ensure efficient network operation, clusters are periodically re-evaluated and reorganized**. This helps accommodate changes in node energy levels, mobility, or network topology.
- ***Data Aggregation and Transmission:*** CHs collect data from MNs, **aggregate it to reduce redundancy, and then forward it to the base station or gateway**. This aggregation minimizes the amount of data that needs to be transmitted over long distances, reducing energy consumption and network congestion.

In an IoT network model comprising 10 nodes labeled n1 through n10, we organize the nodes into clusters to manage communication efficiently. Specifically, the network is divided into two clusters: Cluster 1 includes nodes n1, n2, n3 (acting as the Cluster Head, or CH), n4, and n5, while Cluster 2 encompasses nodes n6, n7 (CH), n8, and n9. The sink node, n10, serves as the base station that collects data from both clusters. During network initialization, nodes are deployed, and cluster heads are selected based on criteria such as residual energy and node degree. Nodes then associate with the nearest or most suitable CH. In this model, intra-cluster communication occurs directly between nodes and their respective CHs, with nodes n1, n2, n4, and n5 communicating with CH n3, and nodes n6, n8, and n9 communicating with CH n7. For inter-cluster communication, CHs n3 and n7 aggregate data from their clusters and transmit it to the sink node n10.

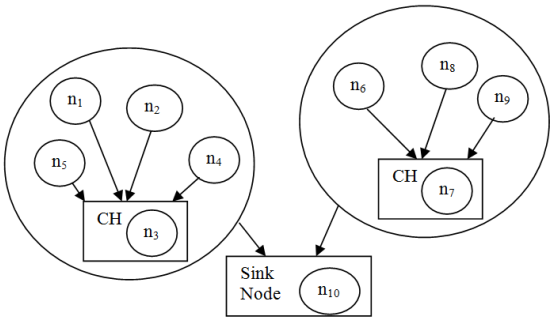

Figure 1: IoT network model for 10 nodes

2.2. Key Components of Cluster-Based IoT Networks

1. **Cluster Heads (CHs)**
   - **Role**: Act as coordinators within their respective clusters. **They collect data from Member Nodes (MNs), perform data aggregation**, and relay aggregated data to the base station or gateway.
   - **Responsibilities**: Manage intra-cluster communication, handle data aggregation, and reduce the amount of data transmitted to the base station.
2. **Member Nodes (MNs)**
   - **Role**: Collect and **transmit data to the Cluster Head**.
   - **Responsibilities**: Sense the environment or application-specific parameters, send raw data or pre-processed data to the CH.
3. **Base Station/Gateway**
   - **Role**: Acts as a central point for receiving data from multiple clusters and providing connectivity to external networks or the Internet.
   - **Responsibilities**: **Aggregate data from all clusters**, process data, and interface with external systems or applications.
4. **Communication Links**
   - **Intra-Cluster Links**: Communication between MNs and CHs within the same cluster.
   - **Inter-Cluster Links**: Communication between different clusters via CHs or directly to the base station.

2.3 Benefits of Cluster-Based Architecture

Cluster-based IoT networks offer several benefits:

- Improved Energy Efficiency: By reducing the distance that data must travel and centralizing data

aggregation at the CH, **energy consumption is significantly reduced**.

- Enhanced Network Lifetime: Effective **clustering can balance energy consumption** across the network, prolonging the overall network lifetime.
- Reduced Latency: Aggregation at the **cluster level can decrease the time required to process** and transmit data, thereby reducing latency.
- Increased Network Throughput: By minimizing direct communication among nodes and aggregating data efficiently, the **network can handle higher data rates**.

Despite the advantages, cluster-based IoT networks also face several challenges [7]:

- Cluster Head Selection: **Choosing the optimal CH is crucial for maintaining network** efficiency and prolonging node life. Inefficient CH selection can lead to uneven energy consumption and reduced network performance.
- Scalability Issues: As the number of nodes grows, maintaining optimal cluster configurations and **managing communication overhead** becomes increasingly complex.
- Data Aggregation: While **data aggregation helps reduce communication overhead**, it also introduces challenges in ensuring data accuracy and timely delivery.

2.4 Applications and Use Cases

Cluster-based IoT networks are widely used in various applications, including:

- Smart Cities: Managing sensor data for traffic control, environmental monitoring, and energy management.
- Healthcare: Monitoring patient health through wearable sensors and aggregating data for analysis.
- Industrial IoT: Collecting and analyzing data from machinery and equipment to improve operational efficiency and predictive maintenance.

In cluster-based IoT networks offer a scalable and energy-efficient approach to managing large-scale IoT deployments. However, effective cluster management and optimization are essential for addressing the inherent challenges and ensuring optimal network performance.

III. OPTIMIZATION ALGORITHMS IN IOT ROUTING

In IoT networks, **routing algorithms are crucial for determining how data is transmitted from source to destination efficiently**. Optimizations algorithms help improve various routing aspects such as energy consumption, delay, throughput, and reliability. Below, we explore several optimization algorithms and their applications to IoT routing.

3.1. Genetic Algorithms (GA) for IoT Routing

Genetic Algorithms (GA) are optimization algorithms inspired by natural selection. They work by **evolving a population of candidate solutions over several generations to find the best solution** for a given problem [8] [9] [10]. In the context of IoT routing, GA can be used to optimize routing paths, reduce energy consumption, and improve network performance.

GA Algorithm for IoT Routing

Step 1. Initialization

- ***Generate Initial Population:*** Create an initial population of candidate solutions. **Each individual (or chromosome) represents a potential routing path** through the IoT network.
  - *Chromosome Representation:* A chromosome could be represented as a sequence of nodes in the network (e.g., [N_1, N_3, N_5, and N_2]).

Step 2. Fitness Evaluation

- ***Evaluate Fitness:*** Calculate the fitness of each chromosome based on a fitness function. The fitness function evaluates how well the routing path performs based on criteria such as energy consumption, delay, and packet delivery ratio.
  - *Fitness Function :*

$$\text{Fitness}(C) = \frac{1}{\text{Cost}(C)}$$

  - *Cost Function :* **For routing, the cost function might include energy consumption** E(C), delay D(C), and routing overhead R(C):

$$COST(C) = w_E.E(C) + w_D.D(C) + w_R R(C)$$

Where w_E, w_D, and w_R are weights for energy, delay, and routing overhead, respectively.

Step 3. Selection

- ***Select Parents:*** **Choose parent chromosomes based on their fitness scores**. Chromosomes with higher fitness values are more likely to be selected.
  - *Selection Probability :* The probability p_i of selecting chromosome i can be computed as:

$$p_i = \frac{Fitness(C_i)}{\sum_{j=1}^{N} Fitness(C_j)}$$

Where N is the population size.

Step 4. Crossover

- ***Perform Crossover:*** Create offspring by combining parts of two parent chromosomes. **Crossover helps to explore new areas of the solution space**.
  - *Single-Point Crossover :* Given two parent chromosomes P1 and P2, and a crossover point c:
    Offspring 1: [$P1_1$,$P1_2$,...,$P1_c$,$P2_{c+1}$,$P2_{c+2}$,...,$P2_n$]
    Offspring 2: [$P2_1$,$P2_2$,...,$P2_c$,$P1_{c+1}$,$P1_{c+2}$,...,$P1_n$]

Step 5. Mutation

- **Apply Mutation:** Introduce random changes to offspring chromosomes to maintain diversity in the population and avoid local optima.
  - *Mutation Operation :* For a chromosome C = [N_1, N_2, ..., N_n]), a mutation might involve swapping two nodes N_i and N_j:
    $C_{mutated}$ = [N_1, ..., N_j, N_i, ..., N_n]

Step 6. Replacement

- **Replace Old Population:** Update the population by replacing some or all of the old chromosomes with new offspring.

- *Replacement Strategy :* **One common strategy is to use elitism(superiority),** where the best chromosomes from the current population are kept in the new population:

New Population=Top k Best Chromosomes+Offspring

where k is the number of elite chromosomes.

Step 7. Termination

- **Check for Termination Condition:** The **algorithm terminates when a stopping criterion is met**, such as a maximum number of generations or convergence of the fitness value.
  - *Stopping Criterion:*

If Generation≥Max Generations or Convergence Criterion is met, then stop.

Example Application: Routing Optimization in a Smart Agriculture Network

In a smart agriculture IoT network, the goal is to find optimal routing paths for sensor data to minimize energy consumption and delay. Here's how GA can be applied:

- Initialization: Generate an initial population of possible routing paths from sensor nodes to the central server.
- Fitness Evaluation: **Compute the fitness of each path** based on energy consumption (using battery consumption models), transmission delay, and routing overhead.
- Selection: **Use roulette wheel selection to choose the best paths** for crossover.
- Crossover: Apply single-point crossover to combine different routing paths to generate new potential paths.
- Mutation: **Randomly swap nodes in the routing paths** to introduce diversity.
- Replacement: Replace less fit paths with new offspring paths while retaining some of the best paths from the previous generation.
- Termination: **Continue for a predefined number of generations** or until the fitness improvement plateaus.

Genetic Algorithms provide a robust method for optimizing routing in IoT networks by iteratively evolving solutions based on a fitness function. By employing techniques such as crossover and mutation, GA explores a wide solution space and finds effective routing paths that balance multiple performance criteria.

3.2. Ant Colony Optimization (ACO) for IoT Routing

Ant Colony Optimization (ACO) is inspired by the foraging behavior of ants and uses pheromone trails to guide the search for optimal solutions [11] [12] [13]. In the context of IoT routing, **ACO can be used to find efficient routing paths that minimize energy consumption**, delay, and other metrics.

ACO Algorithm for IoT Routing

Step 1. Initialization

- **Initialize Parameters**:
- **Pheromone Matrix (τ)**: Initialize the pheromone matrix, which represents the pheromone levels on each edge of the graph. **The initial pheromone value is often set to a small constant.**

$$\tau_{ij}(t=0)=\tau_0$$

Where τ_0 is a small positive constant.

- **Heuristic Information (η)**: **Compute heuristic information**, such as inverse of the cost or distance between nodes.

$$\eta_{ij}=1/\ d_{ij}$$

Where d_{ij} is the distance or cost between nodes i and j.

- **Ants and Iterations**: Set the number of ants N_{ants} and the number of iterations T.

Step 2. Solution Construction

- **Ant Movement**: **Each ant constructs a solution (routing path)** by moving from the source node to the destination node using a probabilistic rule based on pheromone levels and heuristic information.
  - **Transition Probability**: The probability P_{ij} that an ant k moves from node iii to node j is given by:

$$P_{I,j}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{iealloweed}[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}$$

  - Where **α and β are parameters controlling the influence of pheromone** and heuristic information, respectively. The sum is taken over all allowed nodes i that can be visited next.

Step 3. Fitness Evaluation

- **Evaluate Solutions**: After all ants have constructed their solutions, **evaluate the fitness of each solution based on the objective function**, such as total energy consumption or delay.
  - **Objective Function**: For a routing path P_k , the cost C_k can be calculated as:

$$C_k=\sum_{(i,j)\epsilon P_k}(w_E.E_{ij}+w_D.D_{ij}+w_R.R_{ij})$$

where E_{ij} is the energy consumption on edge (i,j), D_{ij} is the delay on edge (i,j), R_{ij} is the routing overhead on edge (i,j), and wE , wD, wR are weights for these metrics.

Step 4. Pheromone Update

- **Local Pheromone Update**: Update the pheromone level on the edges used by ants during the construction of solutions.
  - **Local Update Rule**: $\tau_{ij}(t)=(1-\rho)\cdot\tau_{ij}(t)+\rho\cdot\tau_0$

Where, ρ is the local pheromone evaporation rate.

- **Global Pheromone Update**: After all ants have completed their tours, **update the pheromone levels globally based on the quality** of the solutions found.
  - **Global Update Rule**:

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij}$$

Where, $\Delta\tau_{ij}$ is the amount of pheromone deposited by the ants.

It is typically calculated as:

$$\Delta\tau_{ij}=\sum_{k=1}^{N_{ants}}\Delta\tau_{ij}^{k}$$

Where, Δτij is the amount of pheromone deposited by the ants. It is typically calculated as:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{C_k} & \text{if edge (i, j) is used by ant k} \\ 0 & \text{Otherwise} \end{cases}$$

Where, Q is a constant related to the amount of pheromone deposited.

Step 5. Check for Convergence

- **Termination Condition**: Determine if the algorithm should terminate based on convergence criteria or the maximum number of iterations.
  - **Stopping Criterion**: If Iteration≥T or convergence criterion is met, t hen stop.

Step 6. Solution Extraction

- **Best Solution**: **After termination, extract the best solution** (routing path) based on the lowest cost found during the iterations.

Example Application: Routing Optimization in a Smart Agriculture Network

In a smart agriculture IoT network, **ACO can be used to find optimal paths for data transmission** from sensors to a central server:

- **Initialization**: Set up pheromone values on network edges and heuristic information based on distance or cost.
- **Solution Construction**: Ants explore paths from sensor nodes to the central server, guided by pheromone levels and distance.
- **Fitness Evaluation**: Calculate the cost of each path based on energy consumption, delay, and overhead.
- **Pheromone Update**: Update pheromone levels to reinforce better paths and evaporate pheromone on less optimal paths.
- **Check for Convergence**: Continue for a predefined number of iterations or until the improvement in solution quality stops.
- **Solution Extraction**: **Select the path with the lowest cost** as the optimal routing path.

Ant Colony Optimization (ACO) provides an effective method for solving routing problems in IoT networks by simulating the foraging behavior of ants. **By iteratively updating pheromone levels and exploring different paths**, ACO can find efficient routing solutions that balance multiple performance metrics, such as energy consumption, delay and reliability.

3.3. Particle Swarm Optimization (PSO) for IoT Routing

Particle Swarm Optimization (PSO) is inspired by the **social behavior of birds and fish**. It involves a swarm of particles that explore potential solutions and adjust their positions based on their own experiences and the experiences of their neighbors[14] [15] [16]. In the context of IoT routing, **PSO can be used to optimize routing paths** to minimize metrics such as energy consumption, delay and routing overhead.

Step-by-Step PSO Algorithm for IoT Routing

Step 1. Initialization

- **Initialize Particles**: Create an initial swarm of particles where each particle represents a potential routing path in the IoT network.

  Formula:

  - **Position Vector**: Each particle's position x_i is a vector representing a routing path. For example:

    $$x_i=[N_1,N_2,...,N_n]$$

  - **Velocity Vector**: Each particle's velocity v_i represents the change in position:

    $$v_i=[v_{i1},v_{i2},...,v_{in}]$$

  - **Initialization**: Randomly initialize x_i and v_i for each particle.

Step 2. Fitness Evaluation

- **Evaluate Fitness**: **Calculate the fitness of each particle based on the objective** function. The fitness function evaluates how well the routing path performs in terms of energy consumption, delay, and routing overhead.
  - **Fitness Function**: For a routing path xi, the cost Ci can be calculated as:

    $$C_i=w_E\cdot E(x_i)+w_D\cdot D(x_i)+w_R\cdot R(x_i)$$

    Where:
    - $E(x_i)$ is the energy consumption of the path,
    - $D(x_i)$ is the delay,
    - $R(x_i)$ is the routing overhead,
    - wE , wD , and wR are weights for these metrics.

Step 3. Update Personal Best

- **Update Personal Best Position**: Each particle updates its personal best position $p_{best,i}$ if its current position is better than its previous personal best.
  - **Personal Best Update**:

    $$P_{best,i} = \begin{cases} x_i \text{ if } C_i<C_{best,i} \\ P_{best,i} \text{ Otherwise} \end{cases}$$

    Where $C_{best,i}$ is the cost of the best position found by particle.

Step 4. Update Global Best

- **Update Global Best Position**: The best position among all particles in the swarm is updated to the global best g_{best}.
  - **Global Best Update**:

    $$g_{best} = \begin{cases} P_{best,1} \text{ if} C_1 < C_{best,global} \\ g_{best} \text{ Otherwise} \end{cases}$$

    Where $C_{best,global}$ is the cost of the best position found by any particle.

Step 5. Velocity and Position Update

- **Update Velocity**: Each particle updates its velocity based on its personal best and the global best positions.
  - **Velocity Update**:

    $$v_i(t+1)=\omega\cdot v_i(t)+c_1\cdot r_1\cdot(p_{best},i-x_i(t))+c_2\cdot r_2\cdot(g_{best}-x_i(t))$$

    Where:
    - ω is the inertia weight,
    - c1 and c2 are cognitive and social coefficients,
    - r1 and r2 are random numbers between 0 and 1.
- **Update Position**: Each particle updates its position based on its velocity.
- **Position Update**:

$$x_i(t+1)=x_i(t)+v_i(t+1)$$

Ensure that the updated position is valid according to the routing constraints.

Step 6. Check for Convergence

Step 7. Termination Condition: Determine if the **algorithm should terminate based on convergence criteria** or the maximum number of iterations.

- **Stopping Criterion**:

  If Iteration≥T or convergence crit erion is met, then stop.

  Convergence can be checked by observing if the change in global best fitness or positions falls below a threshold.

Step 8. Solution Extraction

- **Best Solution**: After termination, **the global best position g_{best} represents the optimal routing path**.

Example Application: Routing Optimization in a Smart Agriculture Network

In a smart agriculture IoT network, PSO can be used to find the optimal routing paths for data transmission from various sensor nodes to a central server:

1. **Initialization**: Set up particles with random initial positions and velocities, representing possible routing paths.
2. **Fitness Evaluation**: Calculate the cost of each path based on energy consumption, delay, and routing overhead.
3. **Update Personal Best**: Each particle updates its personal best routing path if its current path is better.
4. **Update Global Best**: The swarm updates the global best routing path based on the best path found by any particle.
5. **Velocity and Position Update**: Adjust the particles velocities and positions to explore new potential routing paths.
6. **Check for Convergence**: Continue for a set number of iterations or until improvements become negligible.
7. **Solution Extraction**: Select **the global best routing path as the optimal solution** for data transmission.

Particle Swarm Optimization **(PSO) provides a powerful method for optimizing routing in IoT networks by simulating the social behavior of particles**. By iteratively updating positions and velocities, PSO explores the search space effectively to find routing paths that balance multiple performance metrics, such as energy consumption and delay.

3.4. Lion Optimization (LO) for IoT Routing

Lion Optimization (LO) is inspired by the **social hierarchy and hunting strategies of lions**. In the context of IoT routing, LO can be utilized to optimize routing paths by simulating the behaviors and interactions of lions, including their hunting tactics and social structure [17] [18] [19].

Step-by-Step LO Algorithm for IoT Routing

Step 1. Initialization

- **Initialize Parameters**:
  - **Lion Population**: Initialize a population of lions, where each lion represents a potential routing path in the IoT network.
  - **Parameters**: Set parameters such as the number of lions N_{lions}, maximum iterations T, and constants related to hunting behavior and social interaction.
  - **Lion Representation**: Each lion x_i can be represented as a sequence of nodes in the network:

    $$x_i=[N_1,N_2,...,N_n]$$

  - **Initial Fitness**: Calculate the initial fitness of each lion based on the routing path.

Step 2. Fitness Evaluation

- **Evaluate Fitness**: Compute the fitness of each lion using an objective function that considers routing metrics such as energy consumption, delay, and routing overhead.

- **Fitness Function**: For a routing path x_i, the cost C_i can be calculated as:

  $$C_i=w_E\cdot E(x_i)+w_D\cdot D(x_i)+w_R\cdot R(x_i)$$

  Where:
  - $E(x_i)$ is the energy consumption of the path,
  - $D(x_i)$ is the delay,
  - $R(x_i)$ is the routing overhead,
  - wE , wD , and wR are weights for these metrics.

Step 3. Hunting Strategy

- **Identify Best Lions**: **Sort the lions based on their fitness** and identify the top-performing lions (pride leader, sub-leaders and followers).

  Formula:
  - **Ranking**: Rank the lions based on their fitness values:

    $$Rank(i)=Sort(C_i)$$

  - **Pride Leader**: The lion with the best fitness value becomes the pride leader:

    $$x_{leader}=\arg.\min_i C_i$$

- **Hunting**: **Update the positions of the lions based on their social structure** and hunting strategy.

  Formula:
  - **Leader's Influence**: Lions update their positions towards the pride leader using:

    $$x_i(t+1)=x_i(t)+\alpha\cdot(x_{leader}-x_i(t))$$

    Where, α is a coefficient controlling the influence of the leader.

- **Sub-Leader's Influence**: Sub-leaders guide the followers towards the leader's position with some adjustments:

  $$x_i(t+1)=x_i(t)+\beta\cdot(x_{sub-leader}-x_i(t))$$

  where,**β is a coefficient controlling the influence of the sub-leader**.

- **Random Exploration**: Introduce random changes to maintain diversity:

  $$x_i(t+1)=x_i(t)+\gamma\cdot(Rand()-0.5)$$

  where γ is a coefficient controlling the amount of randomness.

Step 4. Update Pride Members

- **Adjust Positions**: **Update the positions of all lions based on the hunting strategies** and influences from the pride leader and sub-leaders.
  - **Position Update**: The position update for a lion is a combination of leader's influence, sub-leader's influence, and random exploration:

$$x_i(t+1)=_{xi}(t)+\alpha\cdot(x_{leade}r-x_i(t))+\beta\cdot(x_{sub-leader}-x_i(t))+\gamma\cdot(Rand()-0.5)$$

Step 5. Check for Convergence

- **Termination Condition**: Determine if the algorithm should stop based on convergence criteria or the maximum number of iterations.
  - **Stopping Criterion**:

    If Iteration≥T or convergence criterio n is met, then stop.Convergence can be checked by monitoring the change in the global best fitness or positions.

Step 6. Solution Extraction

- **Best Solution**: After the algorithm terminates**, the best routing path found by the pride leader** represents the optimal solution.
  - **Global Best Path**:

$$x_{best}=x_{leader}$$

where, xleader is the position of the pride leader with the lowest cost.

Example Application: Routing Optimization in a Smart Agriculture Network

In a smart agriculture IoT network, LO can be used to find the optimal routing paths for data transmission from various sensors to a central server:

1. **Initialization**: Set up lions with random initial positions representing different routing paths.
2. **Fitness Evaluation**: Calculate the cost of each routing path based on energy consumption, delay, and routing overhead.
3. **Hunting Strategy**: Update lion positions based on the pride leader's path, sub-leaders, and random exploration.
4. **Update Pride Members**: Adjust positions to converge towards optimal paths while maintaining diversity.
5. **Check for Convergence**: Continue for a predefined number of iterations or until improvements in fitness become negligible.
6. **Solution Extraction**: Select the routing path with the lowest cost as the optimal solution.

Lion Optimization **(LO) provides a nature-inspired approach to optimizing routing paths in IoT networks**. By simulating the social and hunting behaviors of lions, LO effectively explores potential solutions and converges towards optimal routing paths that balance multiple performance metrics such as energy consumption and delay.

3.5. Firefly Algorithm (FA) for IoT Routing

The Firefly Algorithm (FA) is **inspired by the flashing behavior of fireflies**. It uses the intensity of the light emitted by fireflies to guide the search for optimal solutions [20] [21] [22]. In the context of IoT routing, **FA can be used to find efficient routing paths** by optimizing performance metrics such as energy consumption, delay and throughput.

Step-by-Step FA Algorithm for IoT Routing

1. **Initialization**

Initialize Fireflies: Create an initial population of fireflies, where each firefly represents a potential routing path in the IoT network.

- **Position Vector**: Each firefly's position x_i represents a routing path:

$$x_i=[N_1,N_2,...,N_n]$$

- **Light Intensity**: Initialize the light intensity I_i of each firefly based on the fitness value. The fitness function evaluates the routing path based on metrics such as energy consumption, delay, and routing overhead.

  Fitness Function:

$$I_i=1 / C_i$$

  Where C_i is the cost of the routing path, computed as:

$$C_i=w_E\cdot E(x_i)+w_D\cdot D(x_i)+w_R\cdot R(x_i)$$

- $E(x_i)$ is the energy consumption of the path, $D(x_i)$ is the delay, $R(x_i)$ is the routing overhead, and wE, wD, wR are weights for energy consumption, delay and routing overhead, respectively.

2. **Evaluate Light Intensity**

Compute Fitness: For each firefly, **compute its light intensity based on its fitness value**. The better the fitness, the higher the light intensity.

- **Light Intensity**:

$$I_i=1/ C_i$$

3. **Movement of Fireflies**

Attractiveness: Fireflies are attracted to brighter (more fit) fireflies. Update the position of each firefly based on the attractiveness of other fireflies.

Attractiveness: The attractiveness β of firefly iii to firefly jjj is a function of their relative light intensities and the distance between them:

$$\beta_{ij} = \beta_0 . e^{-\gamma . d_{ij}^2}$$

Where, **β_0 is the attractiveness at distance 0,** γ is the light absorption coefficient, and d_{ij} is the distance between fireflies i and j.

Movement: **Update the position of each firefly based on the movement** towards brighter fireflies and a random component.

- **Position Update**:

$$x_i(t+1)=x_i(t)+\beta_{ij}\cdot(x_j(t)-x_i(t))+\alpha\cdot(Rand()-0.5)$$

Where,

- $x_j(t)$ is the position of the brighter firefly,
- α is the randomization parameter,
- Rand() is a random number between 0 and 1.

4. **Update Light Intensity**
   - **Recalculate Intensity**: After updating positions**, recalculate the light intensity** of each firefly based on its new fitness value.
     - **Recalculate Light Intensity**: $I_i= 1/ C_i$
5. **Check for Convergence**

- **Termination Condition**: Determine if the algorithm should stop based on convergence criteria or the maximum number of iterations.
  - **Stopping Criterion**:

If Iteration≥T or convergence criterion is met, then stop.Convergence can be checked by monitoring the change in the best fitness or positions.

6. **Solution Extraction**

Best Solution: After the algorithm terminates, the firefly with the highest light intensity represents the optimal routing path.

Global Best Path:

$$x_{best} = \arg . \max_i I_i$$

Where, xbest is the position of the firefly with the highest light intensity.

Example Application: Routing Optimization in a Smart Agriculture Network

In a smart agriculture IoT network, FA can be used to find the optimal routing paths for data transmission from sensors to a central server:

1. **Initialization**: Set up fireflies with random initial positions representing different routing paths.
2. **Evaluate Light Intensity**: Compute the light intensity of each firefly based on the cost of its routing path.
3. **Movement of Fireflies**: Update firefly positions based on the attractiveness of brighter fireflies and random exploration.
4. **Update Light Intensity**: **Recalculate the light intensity of each firefly** after position updates.
5. **Check for Convergence**: Continue for a predefined number of iterations or until improvements in fitness become negligible.
6. **Solution Extraction**: **Select the routing path with the highest light intensity** as the optimal solution.

The Firefly Algorithm **(FA) offers a nature-inspired approach for optimizing routing paths in IoT** networks. By simulating the flashing behavior of fireflies, FA effectively explores the solution space and converges towards optimal routing paths that balance performance metrics such as energy consumption and delay.

IV. EXPERIMENTAL RESULTS

The simulations for this work are carried out using the **NS-3 network simulator** on a Microsoft Windows 10 machine equipped with a CORE i5 processor, 8 GB of RAM, and a 2.2 GHz clock speed. Table-1 provides a comprehensive overview of the simulation parameters. The performance of various optimization algorithms Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Lion Optimization, and Firefly Algorithms is evaluated using several **Quality of Service (QoS) metrics**. These metrics include End-to-End Delay, Routing Overhead, Packet Delivery Ratio (PDR), Idle Listening Time, Energy Consumption and Throughput. **The evaluation is conducted with respect to different network sizes, specifically varying the number of nodes in the simulation**. In each simulation, all nodes in the IoT network are initially assigned a trust value of 0.5. The experiments are conducted across various node densities, including 50, 100, 150, and 200 nodes. **This setup, where the total number of mobile nodes reflects the population in the trusted model,** allows for a thorough assessment of the algorithms under diverse network conditions and densities.

Parameter Settings for the Optimization Algorithms

For the Genetic Algorithms (GA), key parameters include the population size, which typically **ranges from 50 to 200 individuals**, and the crossover rate (Pc), set between 0.7 and 0.9 to determine the likelihood of crossover between pairs of individuals. The mutation rate (Pm) usually falls between 0.01 and 0.05, influencing the probability of mutation. Selection methods such as tournament or roulette wheel selection, along with crossover methods like single-point or two-point crossover, are used to drive genetic evolution. Mutation techniques, including bit-flip and swap mutation, further refine solutions. **The number of generations for running the algorithm generally ranges from 50 to 200, with elitism often retaining 1 to 5 of the best individuals** across generations to ensure high-quality solutions.

In Ant Colony Optimization (ACO), **the number of ants typically varies from 20 to 100, impacting the exploration of the solution space**. The pheromone evaporation rate (ρ), set between 0.1 and 0.5, dictates how quickly pheromone trails dissipate. The pheromone importance factor (α) and the heuristic importance factor (β) are usually set between 1 and 2, and 2 to 5, respectively, guiding the ant's decision-making process. The algorithm employs different pheromone update methods, such as global or local updates. **The total number of iterations for ACO ranges from 50 to 200, and the amount of pheromone deposited by ants** varies based on the specific problem scale.

For Particle Swarm Optimization (PSO), **the swarm size typically ranges from 20 to 100 particles.** The inertia weight (ω), influencing the impact of previous velocities on current velocities, is usually set between 0.4 and 0.9. The cognitive coefficient (c1) and social coefficient (c2), guiding the influence of the particle's personal best and the swarm's global best positions, are typically set between 1.5 and 2.0. **Velocity limits are adjusted based on the problem's scale, while the randomness factor for exploring solutions generally ranges from 0.1 to 0.5**. The algorithm runs for a number of iterations between 50 and 200.

In Lion Optimization (LO), the number of lions ranges from 20 to 100, and the alpha lion ratio, representing the fraction of the best lions, is typically between 0.1 and 0.2. Parameters controlling the hunting behavior and social interactions of the lions are set based on problem specifics. **The number of iterations for LO generally ranges from 50 to 200,** and the random exploration factor is usually between 0.1 and 0.5, ensuring a balance between exploitation and exploration.

For the Firefly Algorithm (FA), **the number of fireflies typically ranges from 20 to 100**. The light absorption coefficient (γ), which controls the rate of light absorption, is generally set between 0.1 and 1.0. The attractiveness coefficient (β_0), representing the initial attractiveness, usually ranges from 1.0 to 2.0. The randomness parameter (α), **influencing the amount of randomness in movement, is typically set between 0.1 and 0.5**. The number of iterations for FA ranges from 50 to 200, and the distance metric used to calculate the distance between fireflies can be either Euclidean or Manhattan. The update methods focus on attraction to brighter fireflies combined with random movement. These parameter settings provide a foundational guide for tuning each optimization algorithm to achieve effective performance in various routing scenarios within IoT networks.

Table 1: Simulation Parameters

Parameter	Value
Number of nodes *h*	200 nodes
Network size *Bt* × *Ct*	500m *X 500*m
Transmission range	2500 m
Initial energy *K*0	240 J
Propagation model	Two ray ground
Number of rounds	50
Packet size	1 MB
Traffic type	CBR
MAC type	802.11
Antenna type	Omni directional antenna

4.1. End-to-End Delay

End-to-End Delay in IoT routing is the total time taken for a data packet to travel from the source node to the destination node across the network. It includes the time taken for packet transmission, propagation, queuing and processing delays.

End-to-End Delay=Transmission Delay+Propagation Delay+Queuing Delay+Processing Delay

Where,

- Transmission Delay is the time required to push all the packet's bits into the link:

Transmission Delay= (Packet Size) / (Transmission Rate)

- Propagation Delay is the time taken for a signal to propagate from the sender to the receiver:

Propagation Delay= (Distance) / (Propagation Speed)

- Queuing Delay is the time a packet spends waiting in the queue before being transmitted:

  Queuing Delay=Queue Length × Packet Arrival Rate

- Processing Delay is the time required to process the packet headers and forward the packet:

  Processing Delay=Processing Time per Packet

PSO tends to achieve lower end-to-end delay because it effectively balances exploration and exploitation during optimization. PSO's global best solution guides the swarm towards efficient paths, **reducing delay by minimizing hops and optimizing routes** based on current network conditions. Its ability to adapt quickly to changing conditions helps in finding low-latency paths efficiently. ACO can achieve good results in end-to-end delay by effectively leveraging pheromone trails to find short paths. However, its performance can be impacted by the pheromone evaporation rate and the exploration-exploitation trade-off. While it generally finds efficient routes, **it might require more iteration to converge to an optimal solution compared to PSO**. FA performs well in optimizing end-to-end delay by guiding fireflies towards brighter solutions (shorter delays). However, **the performance can be limited by the randomness** factor and the balance between attraction and random movement, which can lead to slower convergence and higher delays compared to PSO. **GA can be less efficient in reducing end-to-end delay due to its reliance on crossover** and mutation operations, which may lead to slower convergence. While GA explores a diverse set of solutions, it might take longer to find optimal paths compared to PSO, impacting the end-to-end delay. **LO may have higher end-to-end delays** due to its complex behavior of hunting and social interactions among lions. This LO algorithm's convergence to optimal paths can be slower, and the exploration of the **solution space may not be efficient like PSO, leading to potentially higher delays.**

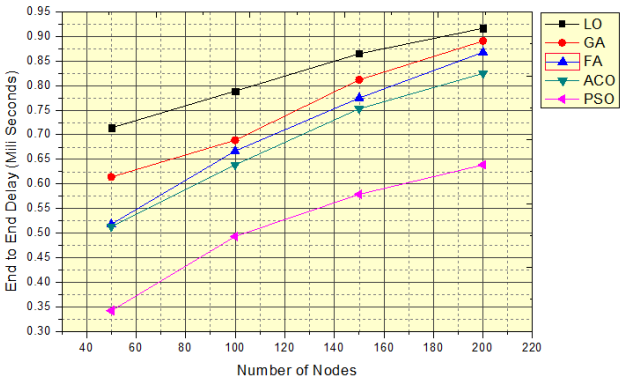

Figure 2: End to End Delay Vs Number of nodes

In conclude Figure 2 Shown, PSO is generally the best algorithm for minimizing end-to-end delay in IoT routing due to its efficient balance between exploration and exploitation, quick adaptation to network conditions, and effective convergence to optimal routes. ACO and FA also provide good results but may exhibit slower convergence or higher delays due to their respective pheromone or attractiveness-based strategies. GA and LO, while effective, often result in longer convergence times and higher end-to-end delays due to their operational complexities and exploration mechanisms.

4.2. Packet Delivery Ratio (PDR)

Packet Delivery Ratio **(PDR) is a key performance metric in IoT routing** that measures the effectiveness of a routing algorithm in delivering packets from the source to the destination**. It is defined as the ratio of the number of successfully received packets at the destination** to the number of packets sent by the source.

PDR = (Number of Packets Received) / (Number of Packets Sent) x 100%

PSO often excels in packet delivery ratio **due to its effective global search capabilities**. By optimizing paths based on swarm intelligence, **PSO can find routes that minimize packet loss** and improve overall delivery success. The algorithm's ability to adapt quickly to network changes and avoid congested or faulty paths contributes to higher PDR.

ACO is effective in improving packet delivery ratio **by utilizing pheromone trails to guide ants** towards optimal paths. The pheromone-based approach helps in identifying reliable routes, which can lead to a high PDR. **However, ACO may require more iteration to converge, and pheromone decay** can affect the delivery ratio if not properly managed. FA can achieve good packet delivery ratios by guiding fireflies towards brighter (more optimal) solutions. **But, its performance can be influenced by the randomness parameter** and light absorption coefficient, which can sometimes lead to suboptimal paths or increased packet loss. GA can be less efficient in terms of packet delivery ratio due to the potential for slower convergence and the use of crossover and mutation operations. While GA explores a broad solution space, **it may not always focus on optimizing packet delivery, leading to lower PDR in some cases**. LO may exhibit lower packet delivery ratios because **its complex behavioral strategies** can lead to longer convergence times and less effective routing paths. The lion's hunting and social interactions **might not always lead to optimal routing solutions, impacting packet delivery**.

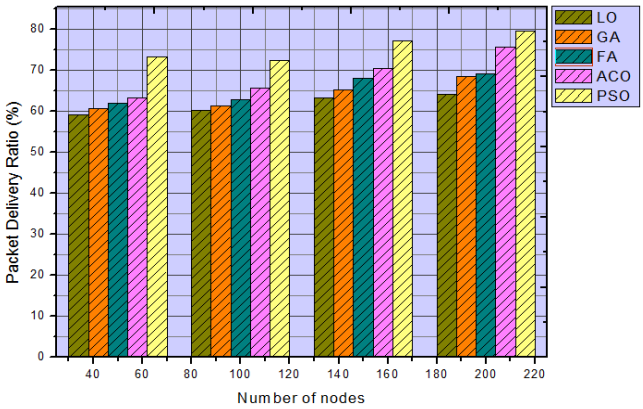

Figure 3: Packet Delivery Ratio Vs Number of nodes

Figure 3 Shown, PSO generally provides the best Packet Delivery Ratio (PDR) in IoT routing due to its efficient search capabilities, adaptability to dynamic network conditions, and ability to identify reliable paths. ACO also performs well but may be slower to converge and can be affected by pheromone management issues. FA offers good performance but may be influenced by its randomness factors. GA and LO, while useful, often show lower PDR due to slower convergence and less effective optimization of routing paths.

4.3. Routing Overhead

Routing Overhead refers to the additional network resources consumed by routing protocols beyond the actual data transmission. It includes the control messages, packet headers, and any other protocol-specific data required to establish and maintain routes**. Lower routing overhead is desirable as it signifies more efficient use of network resources.**

Routing Overhead = (Total Control Packets Sent) /(Total Data Packets Sent) x 100

PSO typically exhibits lower routing overhead due to its efficient search mechanism. PSO uses a swarm of particles to explore the solution space, with minimal communication between particles compared to some other algorithms. This leads to reduced control message exchange and overhead, as the focus is primarily on optimizing the routing paths directly without extensive route maintenance. **ACO can have higher routing overhead due to the continuous exchange of pheromone information** and the need for ants to frequently communicate their discovered routes. The pheromone update process and route discovery involve numerous control packets, which contribute to increase routing overhead, especially in dynamic or large networks**. FA may also exhibit higher routing overhead compared to PSO** due to the communication required among fireflies to share information about light intensity (quality of solutions). The need to exchange information about fitness levels and update positions contributes to additional overhead, although it is generally less than ACO. **GA can lead to moderate to high routing overhead because of the frequent generation** of control messages for crossover and mutation operations. The need for maintaining a population of potential solutions and performing genetic operations on them can result in considerable control packet exchanges. **LO tends to have higher routing overhead due to its complex hunting** and social interaction strategies among lions. The frequent updates and interactions for positioning and hunting can increase the number of control messages exchanged, thereby rising the routing overhead.

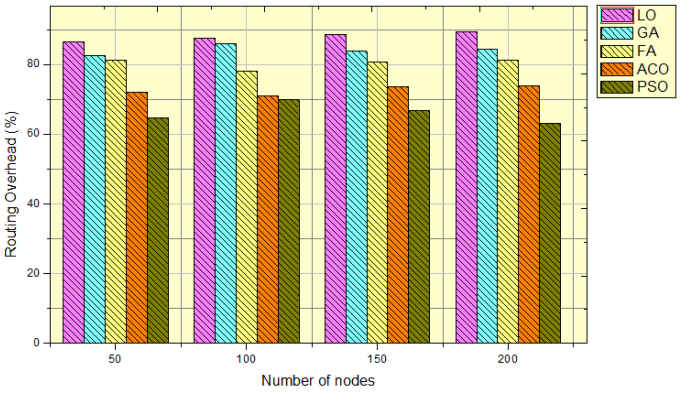

Figure 4: Routing Overhead Vs Number of nodes

Figure 4 Shown, PSO is generally the best for minimizing routing overhead in IoT routing due to its efficient particle-based communication, which reduces the need for excessive control messages. ACO tends to have higher routing overhead because of its extensive pheromone communication and route discovery processes. FA also

incurs some overhead due to information sharing among fireflies but generally less than ACO. GA can result in moderate to high overhead due to genetic operations, and LO typically shows higher overhead due to its complex behavioral strategies.

4.4. Throughput

Throughput is a measure of the rate at which packets are successfully delivered to the destination over a network. It is typically expressed in bits per second (bps) or packets per second (pps). Higher throughput indicates that more data is being transmitted successfully, which is crucial for efficient network performance.

Throughput = (Total Data Received) /(Total Time Taken)

Where, Total Data Received is the total amount of data successfully received at the destination. Total Time Taken is the total time taken for the data to be transmitted and received.

PSO generally offers high throughput due to its efficient optimization of routing paths. By balancing exploration and exploitation, PSO can identify routes that maximize data transmission efficiency and minimize delays. Its global search capabilities allow it to optimize paths that facilitate high data rates and efficient use of network resources. ACO can also achieve good throughput by utilizing pheromone trails to discover optimal paths for data transmission. The algorithm's ability to find efficient routes based on pheromone feedback generally supports high throughput. **However, the effectiveness can be influenced by pheromone evaporation rates** and the exploration-exploitation trade-off, which may sometimes limit throughput. FA provides good throughput by directing fireflies towards brighter solutions, which typically correspond to efficient routes. However, the performance in terms of throughput can be **variable depending on the parameters used, such as the attractiveness factor and randomness**. If these parameters are not well-tuned, throughput may not be as high as that achieved with PSO. **GA can achieve moderate throughput** by exploring a broad range of solutions through crossover and mutation. However, the convergence process might be slower compared to PSO, which can affect the optimization of routing paths and, consequently, throughput. Additionally, the genetic operations can sometimes lead to suboptimal routing paths, impacting the overall throughput. **LO often exhibits lower throughput compared to PSO due to its complex behavioral strategies**, which can lead to slower convergence and less effective path optimization. The hunting and social interactions of lions might not always result in the most efficient routing paths, potentially reducing throughput.

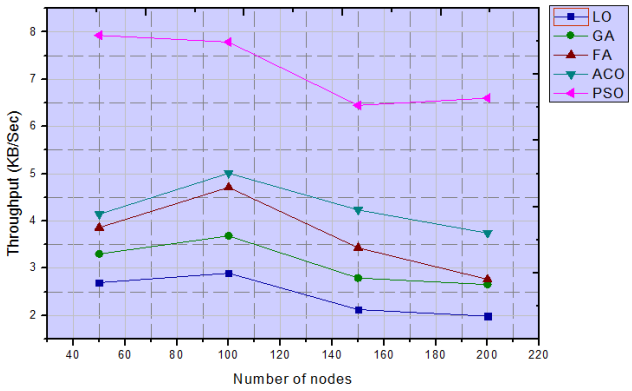

Figure 5: Throughput Vs Number of nodes

Figure 5 Shown, PSO is typically the best algorithm for achieving high throughput in IoT routing due to its effective balance between exploration and exploitation, which enables it to find and optimize high-data-rate paths efficiently. ACO also supports good throughput by discovering efficient routes based on pheromone feedback, though its performance can be influenced by pheromone management. FA can deliver good throughput but is dependent on the tuning of its parameters. GA and LO generally result in lower throughput compared to PSO due to slower convergence and less efficient path optimization.

4.5. Energy Consumption

Energy Consumption refers to the amount of energy used by nodes in a network to transmit, receive, and process data. In IoT routing, minimizing energy consumption is crucial for prolonging the network's operational lifetime, especially given the limited energy resources of IoT devices.

Energy Consumption=Energy for Transmission+Energy for Reception+Energy for Processing

Where:

- **Energy for Transmission** is the energy used to send data packets:
  E_{trans}=Packet Size×Transmission Energy per Bit
- **Energy for Reception** is the energy used to receive data packets:
  E_{rec}=Packet Size×Reception Energy per Bit
- **Energy for Processing** is the energy used for processing and routing decisions:
  E_{proc}=Processing Time×Processing Energy per Unit Time

PSO tends to have lower energy consumption because it minimizes the number of control messages and optimizes routing paths effectively. By efficiently exploring the solution space and finding optimal paths with fewer hops, PSO reduces the overall energy used for data transmission and processing. Its adaptive nature ensures that energy is used efficiently throughout the network. **ACO can result in higher energy consumption due to the frequent updates of pheromone trails** and the numerous control packets required for route discovery and maintenance. The constant exchange of pheromone information and route adjustments can lead to increased energy usage, especially in large or dynamic networks. **FA may incur moderate energy consumption due to the need for fireflies to exchange**

information about their fitness (brightness). Although this exchange is less intensive than in ACO, it can still contribute to additional energy usage. The algorithm's randomness and attraction mechanisms can sometimes result in suboptimal routing paths, affecting energy efficiency. **GA typically leads to higher energy consumption due to the genetic operations like crossover and mutation**, which require significant control and communication among nodes. The process of maintaining and evolving a population of solutions can result in substantial energy expenditure for transmitting and processing genetic data. **LO often exhibits higher energy consumption due to its complex behavior involving hunting and social interactions among lions**. The frequent updates and interactions required for position adjustments and hunting strategies can lead to increased energy usage for data transmission and processing.

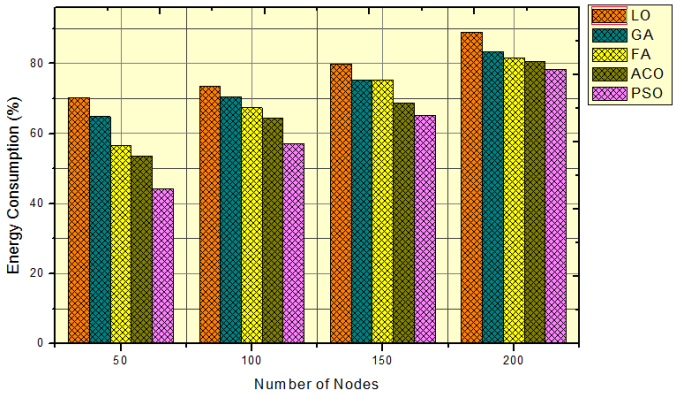

Figure 6: Energy Consumption Vs Number of nodes

Figure 6 Shown, PSO is generally the best for minimizing energy consumption in IoT routing due to its efficient optimization of routing paths and low control message overhead. ACO can result in higher energy usage because of the extensive pheromone exchange and route updates. FA typically has moderate energy consumption due to information sharing among fireflies. GA and LO usually have higher energy consumption due to the significant amount of control messages and complex behavioral strategies, leading to increased energy usage in routing and processing.

4.6. Latency

Latency refers to the time delay experienced in the network from the moment a data packet is sent by the source node until it is received by the destination node. In IoT routing, minimizing latency is critical for ensuring timely delivery of data and improving the responsiveness of the network.

Latency=Transmission Time+Propagation Time+Queueing Time+Processing Time

Where,

- Transmission Time is the time required to push all the packet's bits into the link:
  T_{trans}= (Packet Size)/ (Transmission Rate)
- Propagation Time is the time taken for the signal to travel from sender to receiver:
  T_{prop}= (Distance) / (Propagation Speed)
- Queueing Time is the time a packet spends in a queue before being transmitted:
  T_{queue} = Average Queue Length x Average Service Time
- Processing Time is the time required to process the packet at intermediate nodes:
  T_{proc}=Processing Time per Packet × Number of Nodes

PSO is often the most effective at minimizing latency due to its efficient optimization process. By quickly converging to optimal or near-optimal solutions, PSO can identify paths that reduce the number of hops and transmission delays, thereby lowering the overall latency. The algorithm's ability to rapidly adapt and optimize paths in real-time helps maintain low latency in dynamic IoT networks. **ACO can exhibit higher latency due to the time required for pheromone updates and route discoveries.** The algorithm's process of exploring and updating paths through pheromone trails involves multiple iterations, which can introduce delays. As a result, ACO might have longer latency compared to PSO, especially in larger or more dynamic networks where pheromone management becomes more complex. **FA typically shows moderate latency.** The algorithm's use of attractiveness and light intensity to guide fireflies towards optimal solutions can introduce delays due to the need for frequent updates and fitness evaluations. Although it generally performs well, its effectiveness in reducing latency depends on the parameter tuning and the speed of convergence. **GA** can result **in higher latency due to the time-consuming processes of crossover, mutation, and fitness evaluation**. The population-based approach requires multiple generations to converge to an optimal solution, which can introduce delays. The complexity of genetic operations may lead to increased latency in routing decisions**. LO often experiences the highest latency among the algorithms due to its complex behavioral strategies**, including hunting and social interactions. The frequent updates and interactions required for position adjustments and hunting strategies contribute to increased latency, as these interactions can be computationally intensive and time-consuming.

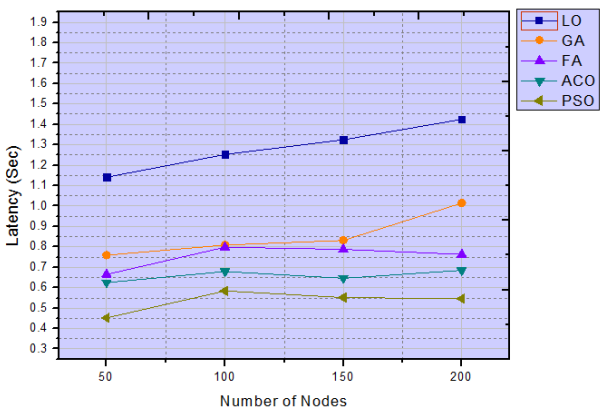

Figure 7: Latency Vs Number of nodes

Figure 7 Shown, PSO is typically the best algorithm for minimizing latency in IoT routing due to its rapid convergence and efficient optimization of routing paths. ACO tends to have higher latency due to the iterative pheromone update process and route discovery overhead. FA can show moderate latency, influenced by parameter tuning and the fitness evaluation process. GA generally results in higher latency due to the evolutionary processes and multiple generations required for convergence. LO often has the highest latency due to its complex interaction and behavioral strategies, which can introduce significant delays in routing.

4.7. Idle Listening Time

Idle Listening Time refers to the period during which an **IoT node remains active and listens for incoming messages, but no data is being transmitted or received**. This time is considered a waste of energy because the node's radio is on without any productive communication. **Minimizing Idle Listening Time is crucial for enhancing energy efficiency in IoT networks**, as it helps to extend the lifespan of battery-powered nodes. To quantify Idle Listening Time, you can use the following formula:

Idle Listening Time=Total Time Node is Active−Time Node is Transmitting or Receiving

In a more formal analysis, if T_{active} is the total time the node is in an active state, $T_{transmit}$ is the time spent transmitting data, and $T_{receive}$ is the time spent receiving data, then:

$$Idle\ Listening\ Time = T_{active} - T_{transmit} + T_{receive}$$

PSO can optimize routing paths to reduce Idle Listening Time by finding the most energy-efficient routes. Its ability to explore a wide solution space and converge to an optimal solution quickly makes it effective for minimizing energy consumption. **ACO can optimize routing by finding efficient paths and reducing energy waste.** However, it may require more iteration to converge to a solution compared to PSO, potentially resulting in higher Idle Listening Time during the search phase. **FA can be effective in reducing Idle Listening Time by finding optimal paths.** Its search capability is generally good, but it may not always converge as quickly as PSO, leading to potentially higher energy consumption during the search process. GA can be effective in optimizing routing paths but may involve **higher computational complexity and longer convergence times compared to PSO**, potentially resulting in higher Idle Listening Time. LO can provide competitive results in optimizing routing paths. However, **its convergence time and computational complexity can vary**, potentially affecting Idle Listening Time compared to PSO.

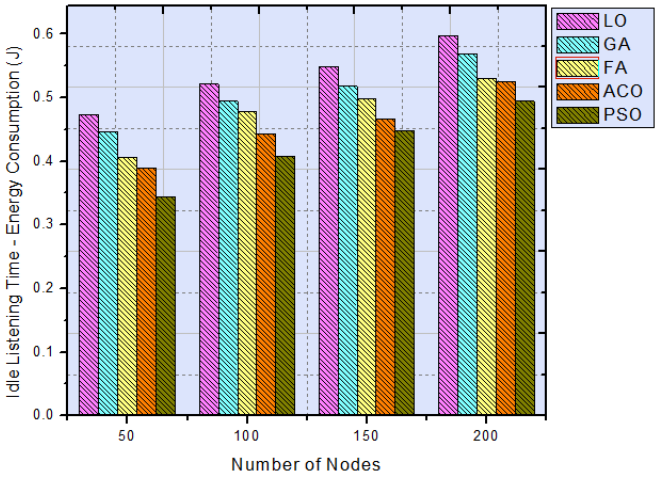

Figure 8: Idle Listening Time Vs Number of nodes

Figure 8 Shown, PSO is favored for its fast convergence and efficient exploration of the solution space, leading to lower Idle Listening Time and better overall energy efficiency in IoT routing.ACO, Effective but may take more iteration to converge, potentially leading to higher Idle Listening Time during the search phase. FA, Good search capability but may not converge as quickly as PSO, which can impact energy efficiency.GA, Effective but often involves higher computational complexity and longer convergence times, potentially resulting in higher Idle Listening Time. LO, Can be competitive but may have variable convergence times and complexity, affecting Idle Listening Time. For optimizing energy efficiency in IoT routing, PSO is generally the most effective algorithm due to its balance of exploration and exploitation, leading to lower Idle Listening Time and better overall energy performance compared to the other algorithms.

4.8. Scalability

Scalability refers to the ability of a routing algorithm to handle an increasing number of nodes or network size efficiently. An algorithm is considered scalable if its performance remains effective as the network grows in terms of nodes, network size, or traffic load. Scalability is often measured by assessing performance metrics like throughput, delay, and overhead as the number of nodes increases. One way to quantify scalability is:

Scalability Index = (Performance Metric)/ (Network Size or Number of Nodes)

Where, Performance Metric could be *throughput, delay, or overhead*. Network Size or Number of Nodes is the total number of nodes in the network. Measures the average energy consumed by each node as the network size increases.

Average Energy Consumption per Node= (Total Energy Consumed) / N

PSO is generally the most scalable algorithm among the listed options. Its particle-based approach efficiently explores the solution space and adapts to increasing network sizes with minimal increase in computational complexity.

The swarm intelligence model allows **PSO to handle larger networks without a significant degradation in performance**, as it primarily relies on local and global best solutions rather than exhaustive search or complex interactions. **ACO can face scalability challenges because the pheromone update process** and route discovery require significant communication overhead, which increases with the network size. As the number of nodes grows, the pheromone trails and the number of ants need to be managed carefully to avoid excessive computation and memory usage, which can impact scalability. **FA shows moderate scalability.** While it efficiently directs fireflies towards optimal solutions, the algorithm's performance can be affected by the need to maintain and update information about the fitness (brightness) of multiple fireflies. As the network size increases, the interaction among fireflies and the updating process can become computationally intensive, this may impact scalability. **GA generally exhibits lower scalability due to the computational complexity** involved in managing and evolving a large population of solutions**.** The processes of crossover, mutation, and fitness evaluation can become computationally expensive and resource-intensive as the number of nodes increases, leading to slower performance and scalability issues. **LO typically shows the lowest scalability among the algorithms**. Its complex behavioral strategies, such as hunting and social interactions, involve frequent updates and communication among lions. As the network size increases, these interactions can lead to significant computational and communication overhead, affecting scalability.

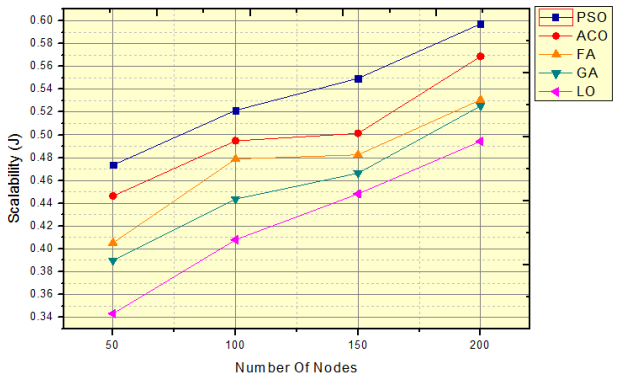

Figure 9: Scalability Vs Number of nodes

Figure 9 Shown, PSO is the best algorithm for scalability in IoT routing due to its efficient exploration and minimal computational overhead, which allows it to handle larger networks effectively. ACO faces scalability challenges due to the increasing complexity of pheromone management and route discovery with network size. FA shows moderate scalability but can become computationally intensive with larger networks. GA and LO are generally experience lower scalability due to their high computational and communication overhead, which also increases with the number of nodes and network size.

V. CONCLUSION

In this research work, we examined several optimization algorithms Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Firefly Algorithm (FA), Genetic Algorithms (GA) and Lion Optimization (LO) for improving energy efficiency and Quality of Service (QoS) parameters in IoT routing. Each algorithm was analyzed based on its performance in terms of end-to-end delay, packet delivery ratio (PDR), routing overhead, throughput, energy consumption, scalability and latency. Particle Swarm Optimization **(PSO) emerged as the most effective algorithm for IoT routing due to its ability to minimize latency, maximize throughput, and reduce energy consumption efficiently**. Its swarm intelligence approach allows it to quickly converge to optimal or near-optimal routing solutions, making it highly suitable for dynamic and large-scale IoT networks. Ant Colony Optimization (ACO), while effective in discovering efficient paths, showed higher latency and energy consumption due to the pheromone update mechanism and route discovery overhead. Its performance is heavily influenced by the management of pheromone trails and network size. Firefly Algorithm (FA), though generally effective, demonstrated moderate scalability and latency. Its performance can be improved with better parameter tuning and optimization of the firefly attraction mechanism. Genetic Algorithms (GA) exhibited higher latency and energy consumption due to the complexity of genetic operations and the need for multiple generations to converge. While GA provides a diverse set of solutions, its computational overhead limits its effectiveness in large-scale networks. Lion Optimization (LO) faced challenges in scalability and latency due to its complex behavioral strategies. The frequent updates and interactions among lions led to increased computational overhead and delays in routing decisions.

Future research should focus on addressing the limitations identified in this study and exploring the following areas:

- **Algorithm Enhancements:** Investigate improvements and **hybridization of the existing algorithms to combine the strengths of multiple techniques.** For example, combining PSO with other algorithms like DNN or RNN could enhance performance metrics such as scalability and energy efficiency.
- **Dynamic Network Environments:** Develop and test algorithms in highly dynamic and heterogeneous IoT environments where nodes frequently join, leave, or move. This will help in **assessing the algorithms' adaptability and robustness** in real-world scenarios.
- **Energy-Efficient Mechanisms:** Explore advanced energy-efficient techniques and mechanisms within the optimization algorithms to **further reduce energy consumption**, especially in resource-constrained IoT devices.

VI.REFERENCES

[1]. A. Almazroi, "A Survey on Internet of Things (IoT) Architecture, Protocols and Applications,"

IEEE Access, vol. 9, pp. 46321-46338, 2021. DOI: 10.1109/ACCESS.2021.3064942.

[2]. S. K. Sharma and S. A. R. R. Sharma, "Internet of Things (IoT) for Smart City: A Survey," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6852-6867, Aug. 2020. DOI: 10.1109/JIOT.2019.2968720.

[3]. N. Kumar, M. N. B. Shankar, and R. G. Rajan, "A Comprehensive Survey on IoT Networks and Services: Architectures, Protocols, and Future Directions," IEEE Transactions on Network and Service Management, vol. 16, no. 2, pp. 567-589, Jun. 2019. DOI: 10.1109/TNSM.2019.2917462.

[4]. R. K. Gupta, V. S. S. M. Krishna, and A. Kumar, "Routing Protocols for Internet of Things (IoT): A Survey," IEEE Access, vol. 8, pp. 48477-48499, 2020. DOI: 10.1109/ACCESS.2020.2974028.

[5]. M. F. A. S. Sheikh and H. A. Javed, "A Survey on Cluster-Based Routing Protocols for Wireless Sensor Networks and IoT," IEEE Access, vol. 8, pp. 22928-22950, 2020. DOI: 10.1109/ACCESS.2020.2970551

[6]. M. Zhang and H. Xu, "Cluster-Based Routing Protocols for Internet of Things: A Review," IEEE Access, vol. 7, pp. 21665-21681, 2019. DOI: 10.1109/ACCESS.2019.2898044.

[7]. R. Shankar, P. R. Kumar, and A. Kumar, "Efficient Cluster-Based Routing Protocols for IoT Networks: A Survey," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4645-4660, Jun. 2019. DOI: 10.1109/JIOT.2018.2889841.

[8]. J. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, 1975. ISBN: 0472084605.

[9]. D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, Reading, MA, 1989. ISBN: 0201157675.

[10]. Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Springer, Berlin, 1996. ISBN: 0387947631.

[11]. M. Dorigo and T. Stützle, "Ant Colony Optimization," MIT Press, Cambridge, MA, 2004. ISBN: 0262033580.

[12]. S. C. H. Yang and K. K. Goh, "A Review of Ant Colony Optimization Algorithms for Routing in Communication Networks," IEEE Access, vol. 6, pp. 75267-75282, 2018. DOI: 10.1109/ACCESS.2018.2880556.

[13]. T. Stützle and M. Dorigo, "A Short Tutorial on Ant Colony Optimization," Proceedings of the International Workshop on Ant Algorithms, pp. 1-12, 2001. DOI: 10.1007/3-540-44577-6_1.

[14]. J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, Dec. 1995. DOI: 10.1109/ICNN.1995.488968.

[15]. Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 69-73, May 1998. DOI: 10.1109/ICEC.1998.699146.

[16]. A. P. Engelbrecht, "Computational Intelligence: An Introduction," IEEE Transactions on Evolutionary Computation, vol. 7, no. 4, pp. 368-372, Aug. 2003. DOI: 10.1109/TEVC.2003.814440.

[17]. A. M. J. Khan, T. T. Khan, and W. Khan, "Lion Optimization Algorithm: A New Metaheuristic Algorithm for Solving Optimization Problems," IEEE Access, vol. 8, pp. 72199-72212, 2020. DOI: 10.1109/ACCESS.2020.2988935.

[18]. A. G. N. Ali and R. A. Saeed, "Lion Algorithm for Optimization: Theory and Applications," Proceedings of the International Conference on Computational Intelligence and Networks, pp. 80-85, Dec. 2015. DOI: 10.1109/CINE.2015.24.

[19]. H. Faris, A. A. J. Ahmed, and M. J. K. El-Raouf, "Lion Algorithm for Network Optimization: A Review and New Directions," IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 647-658, Jul. 2021. DOI: 10.1109/TETC.2020.3035265.

[20]. X.-S. Yang, "Firefly Algorithms for Multimodal Optimization," Stochastic Algorithms: Foundations and Applications, vol. 5792, pp. 169-178, 2009. DOI: 10.1007/978-3-642-00497-0_13.

[21]. X.-S. Yang, "A New Metaheuristic Firefly Algorithm," Proceedings of the International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2009), pp. 1-8, 2009. DOI: 10.1007/978-3-642-04675-9_1.

[22]. R. K. Gupta and S. A. Khan, "Firefly Algorithm-Based Routing for Energy-Efficient Wireless Sensor Networks," IEEE Access, vol. 9, pp. 39617-39628, 2021. DOI: 10.1109/ACCESS.2021.3064548.

ABOUT THE AUTHORS

S.Bharathi received her **M.Phil** Degree from Periyar University, Salem in the Year 2018. She has received her **M.Sc** Degree from Periyar University, Salem in the year 2016. She is pursuing her **Ph.D** Degree (Full Time) in Sri Vijay Vidyalaya College of Arts and Science, Nallampalli, Dharmapuri, Tamilnadu, India. Her Current research of interests includes Internet of Things, Cloud Computing, Network Security and Cryptography.

Dr.D.Maruthanayagam received his **Ph.D** Degree from Manonmaniam Sundaranar University, Tirunelveli in the year 2014. He received his **M.Phil** Degree from Bharathidasan University, Trichy in the year 2005. He received his **M.C.A** Degree from Madras University, Chennai in the year 2000. He is working as **Dean cum Professor**, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts &

Science, Dharmapuri, Tamilnadu, India. He has above **23 years** of experience in academic field. He has published **8 books**, more than **60 papers** in International Journals and **35 papers** in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.