Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.035389
Article

Levy Flight Firefly Based Efficient Resource Allocation for Fog Environment

Anu* **and Anita Singhrova**

Deenbandhu Chhotu Ram University of Science and Technology, Murthal, 131039, Haryana, India
*Corresponding Author: Anu. Email: aujlan77anu@gmail.com
Received: 18 August 2022; Accepted: 22 December 2022

Abstract: Fog computing is an emergent and powerful computing paradigm to serve latency-sensitive applications by executing internet of things (IoT) applications in the proximity of the network. Fog computing offers computational and storage services between cloud and terminal devices. However, an efficient resource allocation to execute the IoT applications in a fog environment is still challenging due to limited resource availability and low delay requirement of services. A large number of heterogeneous shareable resources makes fog computing a complex environment. In the sight of these issues, this paper has proposed an efficient levy flight firefly-based resource allocation technique. The levy flight algorithm is a metaheuristic algorithm. It offers high efficiency and success rate because of its longer step length and fast convergence rate. Thus, it treats global optimization problems more efficiently and naturally. A system framework for fog computing is presented, followed by the proposed resource allocation scheme in the fog computing environment. Experimental evaluation and comparison with the firefly algorithm (FA), particle swarm optimization (PSO), genetic algorithm (GA) and hybrid algorithm using GA and PSO (GAPSO) have been conducted to validate the effectiveness and efficiency of the proposed algorithm. Simulation results show that the proposed algorithm performs efficient resource allocation and improves the quality of service (QoS). The proposed algorithm reduces average waiting time, average execution time, average turnaround time, processing cost and energy consumption and increases resource utilization and task success rate compared to FA, GAPSO, PSO and GA.

Keywords: Fog computing; resource allocation; firefly; IoT; cloud

1 Introduction

An enormous amount of data is generated with the exponential growth of the networks. Data is being forwarded to the cloud for further processing in traditional networking [1]. Transmitting the massive amount of computing data generated by a number of internet of things (IoT) devices can cause extensive delay, increased response time, impaired service experience and bandwidth bottleneck

of network if cloud computing mode is espoused for processing [2,3]. These problems can be solved by introducing an intermediate layer between the cloud and IoT devices for processing the data. Therefore, a key technology, termed fog computing, was developed, which works as an intermediary between the cloud and the edge of the network [4]. Fog computing is an expansion of cloud computing that puts storage and computational capabilities nearer to the IoT devices so that network congestion, latency, and response time can be minimized [5]. Therefore, fog computing is also named "cloud to the ground" that enables the cloud capabilities near to end-user [6].

The fog layer consists of several geographically distributed highly virtualized fog servers that bridges IoT users and the cloud [3]. Each fog server is analogous to the lightweight cloud datacenter equipping on-board storage, computation and communication facility so that cloud-based services can be moved nearer to edge devices resulting in low latency, reduced network traffic, less power consumption and minimized operational cost [7]. Local devices with sufficient processing, storage and networking power to compute latency-sensitive tasks, such as gateways, routers, controllers, embedded servers etc., can function as fog nodes in a fog environment [4]. Placing resources close to the edge of the network helps end users' requests reach the computing node in a minimum time to optimize the processing time of tasks. Small, critical and latency-sensitive tasks are prioritized for processing in the fog computing layer due to the restricted power of fog nodes. However, when fog computing resources are insufficient to process any task, the cloud helps to meet the expanding demands for large-scale computation-insensitive applications. Although fog computing offers numerous benefits, this is still a very challenging paradigm, and researchers face many difficulties regarding resource allocation. The issue of resource allocation in fog computing is very dissimilar to cloud computing [8]. Cloud datacenters are much more robust in performance than fog nodes, whereas fog nodes have limited capacity-constrained resources to compute massive amounts of data. Therefore, fog nodes do not match the processing and storage capabilities of dedicated cloud servers, and if these limited fog resources are not allocated effectively, then it affects the quality of service (QoS). Thus, satisfying the IoT user's computing requirements and effective and optimal allocation of resources has become a challenge for researchers. Effective resource allocation seeks enormous importance while providing fog computing services to IoT users.

1.1 Motivation

Significant challenges in fog computing are to provide faster execution, lesser delay and efficient resource allocation so as to fulfil QoS requirements and improve responsiveness in the highly unpredictable, dynamic and heterogenous environment, which cannot be resolved by traditional resource allocation methods. Hence, to achieve an optimal solution resource allocation problem is expounded by using a metaheuristic technique. This has motivated to propose a novel method using a metaheuristic approach to address the resource allocation problem in fog computing.

1.2 Contribution

The paper focuses on the efficient and optimal redundant allocation of resources in fog computing. Resource allocation is a concept to determine the best available resources for user requests in the fog environment so that all resources can be used effectively and efficiently [8]. Resource allocation problem is an NP-hard problem, so approximation, heuristic and meta-heuristic algorithms are best

suited to solve such problems in preference to exhaustive search algorithms [8]. The key contributions of this paper are summarized underneath:

- The Levy flight-based firefly algorithm (LFFA) is proposed for efficient resource allocation.
- The proposed algorithm allocates the resources in a fog environment in an efficient manner so that resource utilization is optimized along with minimized delay and waiting time.
- The proposed approach is assessed and contrasted with other metaheuristic resource allocation algorithms. The simulation results validate the effectiveness of the proposed algorithm in terms of turnaround time, resource utilization, execution time, processing cost, waiting time, task success rate and energy consumption.

1.3 Organization of the Paper

The remaining sections of this paper are systematized as follows: Section 2 presents a literature review in which various techniques for resource allocation are reviewed. The system architecture is presented in Section 3. The problem is formulated in Section 4. Section 5 presents the explanation for the proposed algorithm. Simulation results are discussed in Section 6, followed by the conclusion in Section 7.

2 Literature Review

The resource allocation problem became more challenging in a fog environment as compared to cloud computing. Many accomplished studies investigated several resource allocation methods and frameworks. A resource allocation policy based upon priced time petri nets (PTPNs) to select substantial resources autonomously from a set of pre-allocated resources was proposed in [9]. For the purpose of encouraging profit awareness in a fog environment, a two-level hierarchical real-time resource allocation model was expanded to guarantee the service provider minimal revenue [10].

Task priority-based resource allocation (TPRA) algorithm was a distributed fog resource allocation algorithm which was introduced to assign the computational tasks to fog resources in an efficient manner so that the conflict of required resources for requesting tasks can be solved [11]. In order to deal with the heterogeneity of fog nodes, diverse types of tasks, and several transmission paths among fog nodes, an optimization problem with the objective function of minimum task delay was established [12]. Allocation of these capacity-limited heterogeneous resources to several competing tasks, a framework based on market equilibrium was proposed, which provided an appropriate resource bundle to every service so that high resource utilization was achieved [13].

A fog-enabled resource management technique (ROUTER) was designed to provide efficient services to IoT devices and to optimize the response time, energy consumption, latency and network traffic by using particle swarm optimization (PSO) [14]. PSO with a graphics processing unit (GPU) is used for efficient offloading in mobile edge computing to decrease energy consumption and delay [15]. For efficient resource utilization in forecasting the mobile traffic for software-defined mobile networks (SDMN), a nature-inspired whale optimization Algorithm (WOA) was developed [16]. An efficient resource allocation (ERA) method and an efficient architecture were designed for resource allocation in a fog environment. This architecture was designed using virtualization to solve the issue of fault tolerance and resource underutilization and overutilization [17].

A peer-to-peer (P2P) based architecture along with service-oriented architecture (SOA) was designed for achieving high-quality computations and for improving the energy efficiency of networks; a decision-based resource allocation algorithm was proposed [18]. In a dynamic resource prediction and allocation scheme, fog nodes were ranked by using TOPSIS so that the most suitable fog node was identified for the incoming tasks. Logistic regression, a machine learning approach, simultaneously calculated the load of every fog node so that result was forwarded to the fog broker for future decisions [19]. In order to achieve low latency processing requirements for the requests generated by resource-constrained IoT devices, a latency minimum offloading technique for optimized offloading decisions and effective resource allocation of fog nodes was proposed, which effectively alleviated the problem of resource limitation and improved QoS [20]. An efficient resource allocation algorithm was presented using a decision tree learning technique based upon service size, virtual machines (VM) capacity and completion time to manage user requests to balance workload between fog and cloud computing environment [21].

To achieve the balance between transmission delay and power consumption, a workload allocation problem was formulated, which further decomposed into subproblems to achieve minimum power consumption and constrained delay [22]. A firefly algorithm-based framework for efficient resource allocation and load balancing for fair resource sharing was proposed in [23]. A metaheuristic-based framework to deal with the heterogeneity of resources and tasks in a fog computing environment using three metaheuristic techniques: PSO, binary particle swarm optimization (BPSO) and BAT algorithms were presented to preserve the tradeoff between the makespan and energy consumption [24].

In a novel hybrid bio-inspired algorithm, two metaheuristic algorithms modified particle swarm algorithm (MPSO) and the modified cat swarm algorithm (MCSO), were combined to manage the resources in a fog environment and to minimize the response time and maximize the resource utilization [25]. An energy-efficient algorithm was proposed to allocate the best suitable fog resources to offloaded tasks so that computation cost was minimized [26]. Issues related to bandwidth, latency, resource utilization and execution time were resolved with an efficient load-balancing scheduling algorithm [27]. To maximize resource utilization, an optimized resource allocation [28] and an efficient hyper-heuristic resource allocation algorithm were proposed. It performed heuristics using machine learning techniques so that a better solution for workflow scheduling was achieved [29].

3 System Architecture

The system architecture for the fog environment, also depicted in Fig. 1, consists of three layers: lower layer, intermediate layer and top layer. The lower layer, also termed as a terminal layer, consists of the IoT devices which generate several application requests to the fog servers.

Fog servers in the intermediate layer, termed the fog layer, are responsible for receiving end-user requests, providing services and resources to requests, delivering the response to IoT users and delegating services to other fog or cloud servers if required [7]. Requests that cannot be processed at the fog layer owing to a high resource requirement are directed to cloud datacenters, which exist in the top layer, also called the cloud layer.

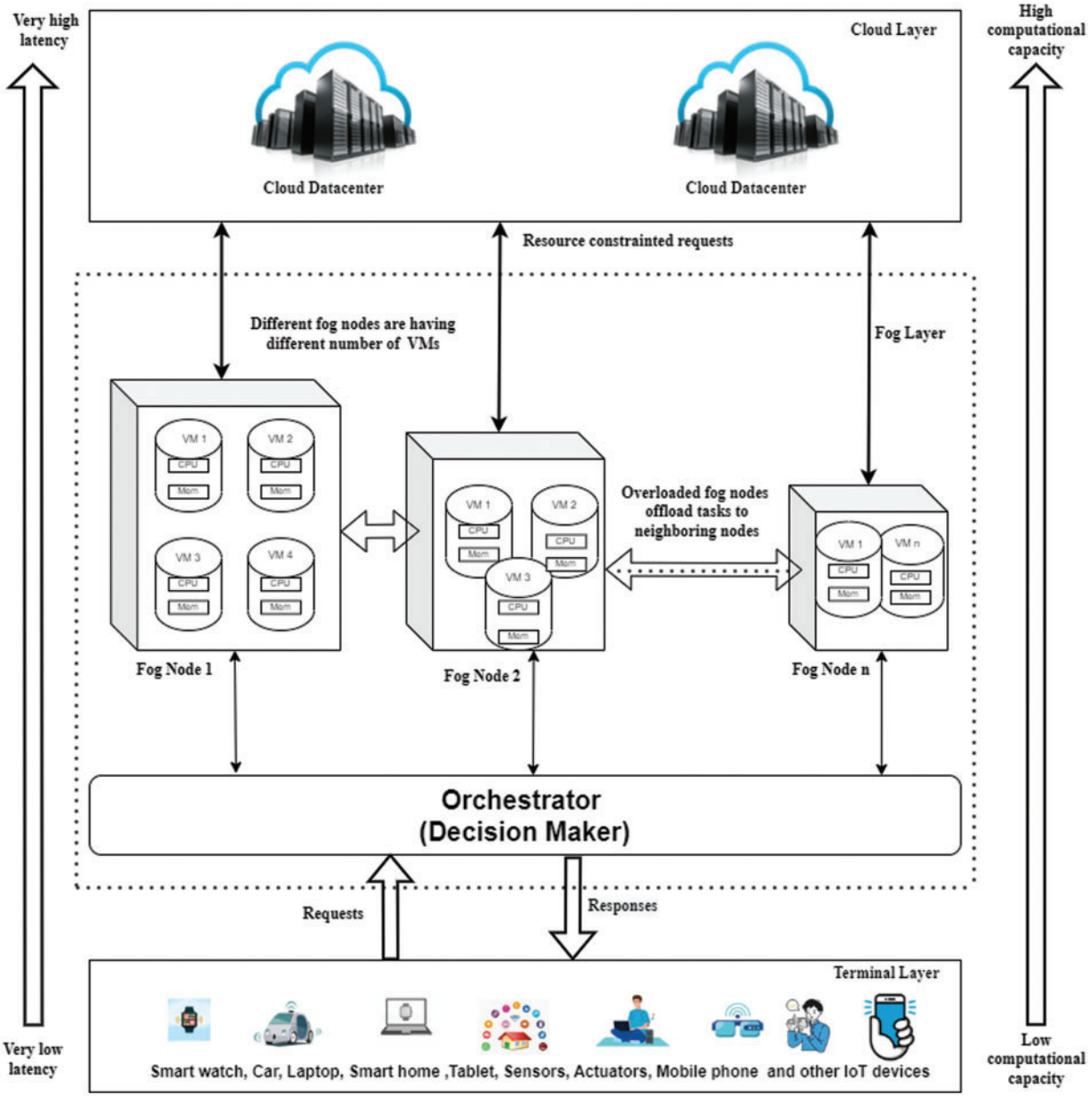

Figure 1: System architecture

4 Problem Formulation

Resources can be of different types, such as memory, network bandwidth, processor and storage. The main objective is to propose an efficient resource allocation in a fog environment using a multi-objective metaheuristic optimization algorithm. Several researchers have presented various algorithms for resource allocation, like genetic algorithm (GA), PSO, hybridization of GA and PSO (GAPSO) etc., but they are not much effective. The proposed algorithm improves and optimizes the QoS parameters while allocating resources. The main aim is to enhance the throughput of fog computing

systems and improve QoS parameters by minimizing the processing time and cost for the tasks and maximizing resource utilization.

4.1 Objective Function

The tasks generated by IoT devices are executed on fog nodes and cloud datacenters. A task is described as a well-defined process which corresponds to a service request made by an end user. A set of 'x' tasks is submitted so that resources are allocated for the execution of that job. Tasks are represented by the task set $\{T_1, T_2, \ldots, T_x\}$. T_p represents an individual task, where 'p' varies from 1 to x. Suppose there are 'y' fog nodes, $\{F_1, F_2 \ldots, F_y\}$ in the fog environment and each fog node serves a set of tasks. An individual fog node is represented using F_q, where 'q' ranges from 1 to y. Every fog node is a set of limited capacity VMs containing computation, memory and storage capabilities such as (processing unit, memory unit and storage unit). A fog node 'q' can have 'k' VMs, $\{VM_{q,1}, VM_{q,2}, VM_{q,3}, \ldots VM_{q,k}\}$. Each VM has a set of resources denoted as $\{R^1_{q,k}, R^2_{q,k}, R^3_{q,k} \ldots, R^z_{q,k}\}$, Where $R^z_{q,k}$ represents z^{th} resource of k^{th} VM in q^{th} fog node. Where z ε {CPU, storage and memory} i.e., 'z' is the resource type of a VM.

4.1.1 Resource Utilization

The resource capacity of the fog node is quantified by the total number of physical resources available. This study is focused on optimizing CPU and memory utilization. The capacity of fog device F_q resources can be represented as $\Delta_{Fq} = \left(\Delta^c_{Fq}, \Delta^{mm}_{Fq}\right)$ where Δ^c_{Fq} represents maximum CPU capacity and Δ^{mm}_{Fq} represents maximum memory of F_q. The maximum workload (WL_{Fq}) of a q^{th} fog device within a time interval $t = (t_1 - t_2)$ is defined using Eq. (1).

$$WL_{Fq}(t) = \alpha \sum_{t=t_1}^{t_2} \Delta^c_{Fq}(t) + (1-\alpha) \sum_{t=t_1}^{t_2} \Delta^{mm}_{Fq}(t) \tag{1}$$

where $\alpha \varepsilon [0, 1]$ and $\Delta^c_{Fq}, \Delta^{mm}_{Fq}$ lies between 0 to 100

Every task T_p requests a set of resources such as (R^c_{Tp}, R^{mm}_{Tp}) so that task is processed successfully.

The workload (WL_{Tp}) for task T_p can be defined using Eq. (2).

$$WL_{Tp}(t) = \alpha \sum_{t=t_1}^{t_2} R^c_{Tp}(t) + (1-\alpha) \sum_{t=t_1}^{t_2} R^{mm}_{Tp}(t) \tag{2}$$

Each task must be assigned to an appropriate fog device for processing in such a way that the maximum workload of a fog device F_q should always be more than the workload consumed by a task T_p i.e.,

$$WL_{Tp}(t) \leq WL_{Fq}(t), \forall (p, q, t) \tag{3}$$

Resource utilization (RU^{Tp}_{Fq}) of a fog device F_q to process task T_p is given as:

$$RU^{Tp}_{Fq} = WL_{Tp}(t) / WL_{Fq}(t) * 100, \forall (p, q) \tag{4}$$

4.1.2 Processing Cost

The processing cost of a task T_p depends upon the capacity of allocated computing resources like CPU and memory. The processing time ($\mathcal{P}T^{Fq}_{Tp}$) of a requested task T_p depends upon its size (S_{Tp}) and

capacity of CPU (CPU_{Tp}^{Fq}) of allocated computing device F_q.

$$\mathcal{P}T_{Tp}^{Fq} = S_{Tp} / CPU_{Tp}^{Fq},\ \forall\ (p,\ q) \tag{5}$$

Thus, the CPU computing cost $\left(\mathcal{P}\theta_{TpFq}^{c}\right)$ of task T_p for using CPU of F_q is defined as :

$$\mathcal{P}\theta_{TpFq}^{c} = \left(\mathcal{P}T_{Tp}^{Fq} * \mathcal{P}\theta_{Fq}^{c}\right) / t_1,\ \forall (p,\ q) \tag{6}$$

where, $\mathcal{P}\theta_{Fq}^{c}$ is CPU computing cost of allocated device for unit time interval t_1.

Task T_p requires some amount of memory of computing device F_q till completion of processing. Thus, the memory cost ($\mathcal{P}\theta_{TpFq}^{mm}$) for task T_p can be defined as:

$$\mathcal{P}\theta_{TpFq}^{mm} = \left(\mathcal{P}T_{Tp}^{Fq} * \mathcal{P}\theta_{Fq}^{mm}\right) / t_1,\ \forall (p,\ q) \tag{7}$$

where the memory cost of an assigned computing device for unit time interval t_1 is given as $\mathcal{P}\theta_{Fq}^{mm}$.

As a result, the processing cost for task T_p allocated to computing device F_q is defined as follows:

$$\mathcal{P}\theta_{TpFq} = \mathcal{P}\theta_{TpFq}^{c} + \mathcal{P}\theta_{TpFq}^{\mathrm{mm}},\ \forall (p,\ q) \tag{8}$$

4.1.3 Turnaround Time

Turnaround time (TAT) is the total time taken to complete a task or fulfil the user's request. It encompasses the execution time (ET) and waiting time (WT).

$$\text{Turnaround Time } (TAT_{Fq}^{Tp}) = ET_{Fq}^{Tp} + WT_{Tp} \tag{9}$$

where, TAT_{Fq}^{Tp} is the turnaround time of task T_p allocated to the F_q.

Execution time (ET_{Fq}^{Tp}) of task T_p allocated to the F_q is the most prominent parameter for measuring the efficiency and performance of the fog device, which is given as:

$$\text{Execution Time } \left(ET_{Fq}^{Tp}\right) = S_{Tp}/CPU_{mips} \tag{10}$$

where, S_{Tp} is the total size of task T_p and CPU_{mips} is the average processing power of a CPU requires to execute task T_p. Waiting time $\left(WT_{Tp}\right)$ of task T_p is the time between the reception of the task by the fog device and starting time of execution.

$$\text{Waiting Time } \left(WT_{Tp}\right) = \text{Execution Start Time} - \text{Task Reception Time} \tag{11}$$

The main objective of the proposed algorithm is to allocate the resources to tasks in an efficient manner so that resource utilization can be maximized and total processing cost and turnaround time of task can be minimized.

$$\text{Maximize } RU_{Fq}^{Tp} \tag{12}$$

$$\text{Minimize } \mathcal{P}\theta_{TpFq},\ TAT_{Fq}^{Tp}$$

Subject to constraints

(i) The workload consumed by a task T_p should always be less than the maximum workload of a fog device F_q i.e., $\sum_{Tp=1}^{x} WL_{Tp}(t) \leq WL_{Fq}(t)$

(ii) The requested amount of CPU by a task should always be less than the maximum capacity of CPU of F_q i.e., $\sum_{F_q=1}^{y} R_{F_q}^{c}(t) \leq \Delta_{F_q}^{c}(t)$

(iii) The requested amount of memory by a task should always be less than the maximum capacity of the memory of F_q i.e., $\sum_{F_q=1}^{y} R_{F_q}^{mm}(t) \leq \Delta_{F_q}^{mm}(t)$

(iv) Resource utilization should always be less than 100 percent i.e., $\sum_{F_q=1}^{y} RU_{F_q} \leq 100$

5 Proposed Algorithm

The process of resource allocation for a requested task is done in an optimized and efficient way by using a proposed levy flight-based firefly algorithm [30,31].

5.1 Existing Firefly Algorithm

The firefly algorithm is a metaheuristic bio-inspired algorithm based on fireflies flashing and attractiveness behaviour, which provides two significant benefits over other algorithms. The automatic subdivision of population and control over randomness to speed up the convergence makes it highly suitable to deal with resource optimization problem. In this algorithm, solutions are generated randomly, and those solutions are considered fireflies. Then the brightness of each firefly will be calculated depending on the objective function. This algorithm works on the principle of attractiveness, i.e., each firefly is being attracted towards a brighter firefly, and in case no firefly is brighter, then the movement of the firefly will be random. This algorithm will notify the best available resources for tasks [32]. If any task does not find any suitable resource, then resources will be allocated randomly. The FA involves the following steps:

a. *Initialization:* Initialize the fireflies, number of iterations, initial population of fireflies and attractiveness function.
b. *Fitness Function:* The quality of solutions can be evaluated by calculating the fitness function.
c. *Attractiveness Calculation:* The attractiveness of each firefly is calculated with respect to other fireflies. The fitness of each firefly is directly proportional to its attractiveness value.
d. *Movement Towards Brighter Firefly:* The attractiveness or brightness of each of firefly is calculated and compared with other fireflies. During every iteration, the fittest firefly is selected, and the firefly moves towards the optimal solution, i.e., brighter firefly. The fewer bright firefly will always attract towards the brighter fireflies.

$$if\ \left(Attr_j > Attr_i\right)\ then\ move\ firefly\ 'i'\ towards\ firefly\ 'j' \tag{13}$$

The movement of i^{th} firefly at position u_i moving to a more attractive j^{th} firefly at position u_j is given as:

$$u_i(t+1) = u_i(t) + \beta_0 e^{-\gamma r^2}\left(u_i - u_j\right) + \alpha\varepsilon_i \tag{14}$$

where β_0 is the attractiveness value, and it is directly relative to the fitness value. $\beta_0 e^{-\gamma r^2}$ is the attractiveness value between two fireflies u_i and u_j. Attractiveness (β_0) varies according to the distance r between two fireflies. The distance r between i^{th} and j^{th} fireflies is given by Eq. (15).

$$r_{ij} = \sqrt{\sum \left(u_i - u_j\right)^2} \tag{15}$$

$\alpha\varepsilon_i$ is a randomization parameter. If $\beta_0 = 0$, then firefly moves randomly based on Eq. (16).

$$u_i(t+1) = u_i(t) + \alpha\varepsilon_i \tag{16}$$

The attractiveness of the firefly at the new position is compared with the attractiveness at the old position. If a new position generates a higher attractiveness value, then the firefly moves towards that position. Otherwise, it will remain at its current position. The firefly will move randomly in case no brighter firefly is found. After the attainment of the termination criterion, it will give the best and optimal solution.

e. Ranking: Global best solution is being identified by sorting the fireflies population according to their fitness value.

5.2 Proposed Levy Flight Firefly Algorithm (LFFA)

By combining the characteristics of levy flight with three idealized rules, a new algorithm LFFA is proposed. Several studies have demonstrated that many animals, birds as well as insects follow the characteristics of levy flight which is a random flight procedure having variable step length [33]. In a levy flight, the length between two consecutive direction changes and the span of flight are drawn from a probability distribution. Numerical studies analyzed that the levy flight algorithm is superior to classic algorithms in case of efficiency and success rate. The typical characteristics of levy flights are further applied to several optimization problems to enhance the capability of algorithms.

In LFFA, the firefly 'i' is moved towards a brighter firefly 'j' using Eq. (17).

$$u_i(t+1) = u_i(t) + \beta_0 e^{-\gamma r^2}\left(u_i - u_j\right) + \alpha\ sign\left[rand - \frac{1}{2}\right] \oplus Levy\ (\lambda) \tag{17}$$

In Eq. (17), the first term depicts the current position, the second term is for attraction, and the third term is randomization due to levy flight having α as the randomization parameter. The meaning of product $\oplus$ is entry-wise multiplications. The sign [rand-1/2] is a random direction where rand ε [0, 1]. Random step length is taken from levy distribution which has infinite variance having an infinite mean. In this modified version of the firefly, random walk is via levy flight which is more efficient for exploring the search space because of its longer step length. LFFA convergence is faster than other metaheuristic algorithms, so it treats global optimization problems more efficiently and naturally. LFFA never get stuck in local optima because it has the ability to find local optima as well as global optima simultaneously in a very effective manner, so the search space is enlarged and more uniform, as well as strengthens the movement ability of fireflies.

Levy flight follows Levy Distribution (LD) which is difficult to achieve. There are various methods to achieve the step size samples, but the Mantegna algorithm [34] provides a direct and efficient approach. The mathematical expression to simulate flight path using the Mantegna algorithm is presented as:

$$Levy\ (\lambda) = \frac{g}{|h|^{\frac{1}{\Upsilon}}} \tag{18}$$

Here Υ is a real parameter, g and h follow the normal distribution (N).

$$\mathrm{g} \approx \mathrm{N}\left(0, \sigma_g^2\right) and\ \mathrm{h} \approx \mathrm{N}\left(0, \sigma_h^2\right) \tag{19}$$

where $\sigma_h = 1$ and σ_g can be calculated as:

$$\sigma_g = \left\{ \frac{\Gamma(1+\Upsilon)\sin\left(\frac{\Upsilon}{2}\pi\right)}{\Upsilon\Gamma\left(1+\frac{\Upsilon}{2}\right)2^{\Upsilon-\frac{1}{2}}} \right\}^{\frac{1}{\Upsilon}}, \tag{20}$$

where Γ is a gamma function, $\Upsilon = 1.5$ [34].

LFFA is a metaheuristic technique used to allocate efficient resources to tasks generated by IoT users. In the proposed algorithm, firstly, VMs are sorted on the basis of their speed then tasks are assigned to the sorted list of VMs. The fireflies are the set of schedules. A schedule is a mapping of task over a VM. Each firefly has its own position and fitness value. The proposed algorithm finds a suitable VM with sufficient resources, providing minimum cost and processing time for every real-time activity. Hence, three types of QoS parameters to find the best available VM i.e., execution time, processing cost and resource utilization, were introduced to find the best available VM. The proposed multi-objective fitness function, F(x), using three QoS parameters, is defined in Eq. (21).

$$\text{Fitness Function, } F(x) = \alpha_1 * ET_{F_q}^{T_p} + \alpha_2 * \mathcal{P}\theta_{T_pF_q} + \alpha_3 * \frac{1}{RU_{F_q}^{T_p}} \tag{21}$$

The pseudocode of LFFA is given in Fig. 2.

Input: List of Schedules. A schedule is a mapping of tasks over VM.
Output: Best available VMs allocated to tasks.

```
1)  Define Objective Function f(x) for all fireflies.
2)  Initialize all parameters, define initial position of firefly, fitness and brightness of each firefly.
3)  While (t< MaxGeneration) or (Stop Criterion)
4)        for all fireflies do
5)              Compute Fitness Function, F(x) as given in Eq. (21)
6)        end for all fireflies
7)        for each firefly_i  do
8)              for each firefly_j  do
9)                    compute the attractiveness of firefly_j in view of firefly_i
10)             end for firefly_j
11)       end for firefly_i
12)       for each firefly_i  do
13)             for each firefly_j  do
14)                   Find the maximum attractiveness and its position
15)                   move firefly_j towards firefly_i according to Eqs. (17-20)
16)             end for firefly_j
17)       end for firefly_i
18) Rank fireflies and find the best and optimal firefly
19) end while stop criterion reached
20) end of algorithm
```

Figure 2: Pseudocode for proposed LFFA

The optimization of fireflies is dependent upon their respective fitness values. By using the fitness function given in Eq. (21), the fittest VM is selected for upcoming tasks. The fittest VM is one that executes the tasks within minimum time and cost with maximum resource utilization. The fitness value of each firefly is evaluated, and a firefly with a minimum fitness value will be treated as a brighter firefly. Unassigned tasks check for brighter firefly and move toward brighter firefly using Eq. (17). If no brighter firefly is found unassigned tasks will move randomly using Eqs. (18)–(20). The continuous LFFA cannot be applied directly to solve discrete optimization problems such as resource allocation. Modified metaheuristic optimization techniques are used for discrete problems. In this

study, the smallest position value (SPV) rule proposed by Tasgetiren et al. [35] is being used so that continuous optimization techniques can be applied to solve all discrete computational problems.

6 Simulation and Results

The PureEdgeSim simulator is used to implement and evaluate the proposed algorithm. PureEdgeSim is a simulation toolkit used to simulate the various resource management techniques in cloud, edge and fog environments and evaluate the performance of these environments in terms of resource utilization, delays, energy consumption, network utilization, task success rate etc. This section discusses the simulator and simulation parameters. The simulation results show that LFFA shows promising results in terms of resource utilization, execution time, waiting time, turnaround time, processing cost and energy consumption in comparison to other resource allocation techniques. Table 1 lists the simulation parameters that are employed to run the simulation and capture the results of the performance metrics. The simulation parameters for LFFA are shown in Table 2 [30,34,36].

Table 1: Simulation parameters

Parameter	Range of values
Number of edge devices	10–500
Total VMs in fog layer	2–50
MIPS	2000
RAM	4 GB
Bandwidth	300 mbps
VM policy	Space shared
Storage	20 GB
Number of tasks	10–2000
Task length	100–2000 MIPS

Table 2: LFFA parameters

Parameter	Values
Number of fireflies	10
α	[0, 1]
β	1.0
γ	1.0
Υ	1.5
Γ	0.05
Number of iterations	10
$\alpha_1, \alpha_2, \alpha_3$	0.5, 0.2, 0.3

The number of tasks increases as the edge devices increase, and resources allocated to tasks are not the same in all iterations. Resources are allocated to the tasks depending upon their availability and quality of service. The QoS of the resources depends upon various parameters such as availability,

energy consumption, remaining battery power, execution delay etc. In the proposed algorithm, efficient resources are chosen to provide better performance in a fog environment. As set up for evaluation, this study considers different applications, mainly augmented reality, smart home, heavy computer applications and e-health applications, where each request has different resource and delay requirements. Each of these applications has different resource and delay requirements. Every algorithm is executed for 10 simulation runs, and then after 10 runs mean value is considered for every parameter, which is represented using PV in Eq. (22). The performance improvement rate ($\mathcal{P}l$) of the proposed algorithm is shown in Tables 3–9. The $\mathcal{P}l$ is used to evaluate the rate of improvement for the m^{th} algorithm w.r.t. n^{th} algorithm, which is calculated using Eq. (22).

$$(\mathcal{P}l)\ (\%) = \left(\frac{\sum_m PV - \sum_n PV}{\sum_n PV}\right) * 100 \tag{22}$$

Table 3: Performance improvement rate of average waiting time

	Average waiting time (in sec)					$\mathcal{P}l$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{P}l$ over FA	$\mathcal{P}l$ over GAPSO	$\mathcal{P}l$ over PSO	$\mathcal{P}l$ over GA
10	0.024	0.028	0.029	0.036	0.047	−13.21	−16.18	−32.31	−48.30
50	0.031	0.038	0.039	0.047	0.059	−17.94	−20.87	−33.83	−47.64
100	0.032	0.039	0.042	0.052	0.066	−19.23	−25.18	−39.77	−52.34
150	0.039	0.044	0.048	0.062	0.079	−12.50	−19.96	−37.60	−51.20
200	0.050	0.053	0.056	0.080	0.087	−5.32	−11.07	−37.71	−42.82
250	0.057	0.061	0.080	0.090	0.100	−6.87	−28.79	−36.71	−43.16
300	0.073	0.083	0.099	0.120	0.142	−12.11	−26.26	−39.12	−48.42

Table 4: Performance improvement rate of average execution time

	Average execution time (in sec)					$\mathcal{P}l$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{P}l$ over FA	$\mathcal{P}l$ over GAPSO	$\mathcal{P}l$ over PSO	$\mathcal{P}l$ over GA
10	0.423	0.450	0.475	0.531	0.580	−5.91	−10.79	−20.31	−27.07
50	0.479	0.498	0.529	0.584	0.642	−3.66	−9.36	−17.84	−25.34
100	0.533	0.541	0.610	0.660	0.719	−1.55	−12.72	−19.31	−25.91
150	0.587	0.617	0.684	0.768	0.846	−4.74	−14.19	−23.54	−30.55
200	0.641	0.685	0.760	0.812	0.872	−6.39	−15.61	−21.05	−26.46
250	0.682	0.747	0.803	0.876	0.949	−8.79	−15.16	−22.16	−28.18
300	0.738	0.801	0.879	1.033	1.150	−7.87	−16.05	−28.56	−35.83

Table 5: Performance improvement rate of average turnaround time

	Average turnaround time (in sec)					$\mathcal{P}l$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{P}l$ over firefly	$\mathcal{P}l$ over GAPSO	$\mathcal{P}l$ over PSO	$\mathcal{P}l$ over GA
10	0.448	0.478	0.503	0.567	0.627	−6.34	−11.10	−21.07	−28.66
50	0.511	0.536	0.568	0.631	0.702	−4.67	−10.15	−19.03	−27.23
100	0.564	0.580	0.652	0.712	0.785	−2.74	−13.52	−20.81	−28.13
150	0.626	0.661	0.733	0.830	0.925	−5.25	−14.57	−24.58	−32.31
200	0.691	0.738	0.816	0.892	0.959	−6.32	−15.30	−22.55	−27.95
250	0.739	0.808	0.883	0.966	1.049	−8.65	−16.39	−23.51	−29.61
300	0.811	0.884	0.979	1.153	1.292	−8.27	−17.09	−29.66	−37.21

Table 6: Performance improvement rate of resource utilization

Resource utilization (in percentage)						$\mathcal{PI}$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{PI}$ over firefly	$\mathcal{PI}$ over GAPSO	$\mathcal{PI}$ over PSO	$\mathcal{PI}$ over GA
10	0.094	0.088	0.081	0.068	0.064	7.04	16.42	38.68	47.57
50	0.104	0.098	0.092	0.078	0.074	6.21	13.48	33.85	40.23
100	0.152	0.139	0.127	0.110	0.105	9.44	19.42	38.47	44.67
150	0.184	0.170	0.158	0.149	0.122	8.23	16.51	23.29	50.86
200	0.245	0.210	0.209	0.178	0.167	16.43	17.10	37.67	46.67
250	0.313	0.300	0.278	0.227	0.218	4.33	12.72	38.15	43.47
300	0.357	0.338	0.318	0.265	0.241	5.41	12.11	34.63	47.86

Table 7: Performance improvement rate of processing cost

Processing cost (in term of time units)						$\mathcal{PI}$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{PI}$ over firefly	$\mathcal{PI}$ over GAPSO	$\mathcal{PI}$ over PSO	$\mathcal{PI}$ over GA
10	2.230	2.491	2.922	3.240	4.120	−10.48	−23.68	−31.17	−45.87
50	3.420	3.822	4.150	5.320	5.900	−10.52	−17.59	−35.71	−42.03
100	4.060	5.201	5.670	6.020	7.130	−21.94	−28.40	−32.56	−43.06
150	5.710	6.003	6.290	8.470	9.560	−4.88	−9.22	−32.59	−40.27
200	6.630	7.441	8.270	9.560	11.210	−10.90	−19.83	−30.65	−40.86
250	7.720	8.650	9.460	10.980	12.090	−10.75	−18.39	−29.69	−36.15
300	8.260	8.995	10.110	12.060	13.870	−8.17	−18.30	−31.51	−40.45

Table 8: Performance improvement rate of task success rate

Task success rate (in percentage)						$\mathcal{PI}$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{PI}$ over firefly	$\mathcal{PI}$ over GAPSO	$\mathcal{PI}$ over PSO	$\mathcal{PI}$ over GA
10	98.00	97.00	92.00	90.00	90.00	1.03	6.52	8.89	8.89
50	97.77	96.00	90.20	88.50	88.00	1.84	8.39	10.47	11.10
100	94.69	93.01	90.00	87.12	86.00	1.81	5.21	8.69	10.10
150	92.50	91.20	89.67	86.54	82.25	1.43	3.16	6.89	12.46
200	92.10	90.00	89.12	85.23	81.65	2.33	3.34	8.06	12.80
250	91.98	89.31	87.23	85.00	80.23	2.99	5.45	8.21	14.65
300	91.83	89.00	86.12	84.98	80.02	3.18	6.63	8.06	14.76

Table 9: Performance improvement rate of energy consumption

Energy consumption (in Wh)						$\mathcal{PI}$ w.r.t. LFFA (in percentage)			
Edge devices	LFFA	FA	GAPSO	PSO	GA	$\mathcal{PI}$ over firefly	$\mathcal{PI}$ over GAPSO	$\mathcal{PI}$ over PSO	$\mathcal{PI}$ over GA
10	2.233	2.372	2.413	3.088	3.224	−5.87	−7.47	−27.69	−30.75
50	3.456	3.790	4.000	4.470	4.840	−8.82	−13.61	−22.69	−28.60
100	4.762	4.813	5.165	5.821	6.357	−1.06	−7.79	−18.19	−25.08
150	6.058	6.210	6.454	6.919	8.086	−2.44	−6.14	−12.44	−25.08
200	7.003	7.412	7.802	8.056	9.121	−5.51	−10.23	−13.07	−23.22
250	8.142	8.773	8.892	9.774	10.659	−7.20	−8.44	−16.70	−23.62
300	8.198	8.902	9.023	10.129	11.125	−7.91	−9.14	−19.06	−26.31

6.1 Average Waiting Time

Time taken by a task in the waiting queue to get the best available resource is known as waiting time. The average waiting time of 'x' tasks is defined in Eq. (23), and evaluated values for different algorithms are presented in Fig. 3.

$$Average\ waiting\ time = \sum_{i=1}^{x} (Turnaround\ time - burst\ time)/x \tag{23}$$

In comparison to other algorithms average waiting time of the proposed algorithm is significantly less. This is due to the fact that resources which have the capacity to serve the incoming requests are only included in search space, and resources which are ineligible to process the request are kept out of search space. Hence waiting time gets minimized. The convergence rate of the firefly algorithm is better than other metaheuristic algorithms, so the waiting time is less than other algorithms. The lesser value is desired in case of average waiting time, execution time, turnaround time, processing cost and energy consumption. The negative values in Tables 3–5, 7 and 9 depict the improvement, as LFFA minimizes these parameters w.r.t FA [32], GAPSO [37], PSO [24] and GA [38].

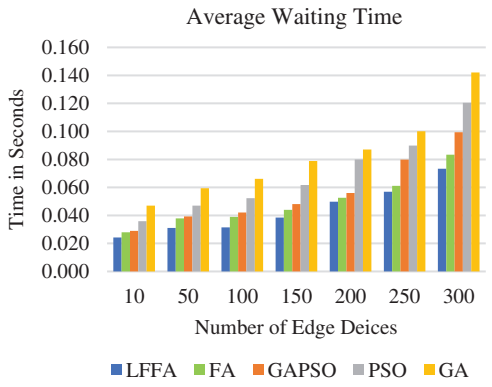

Figure 3: Average waiting time

6.2 Execution Time

Execution time is the amount of time taken by a task for its execution, and it is evaluated using Eq. (10). The result of execution time for different resource allocation algorithms with different numbers of edge devices is shown in Fig. 4. The number of edge devices considered in this experiment range from 10 to 300. The simulation results revealed that the LFFA outperforms other resource allocation algorithms. GA algorithm has the worst execution time compared to other algorithms. The LFFA leads to minimized delay resulting in minimized execution time, further improving the response time.

6.3 Average Turnaround Time

Turnaround time is evaluated by adding the execution time and waiting time and is defined using Eq. (9). For the proposed algorithms, execution time and waiting time are better than other algorithms, and so is turnaround time. The average turnaround time is shown in Fig. 5.

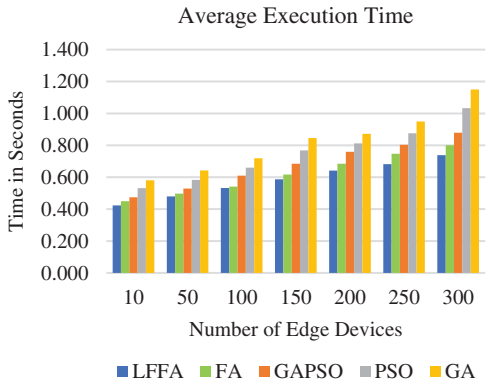

Figure 4: Average execution time

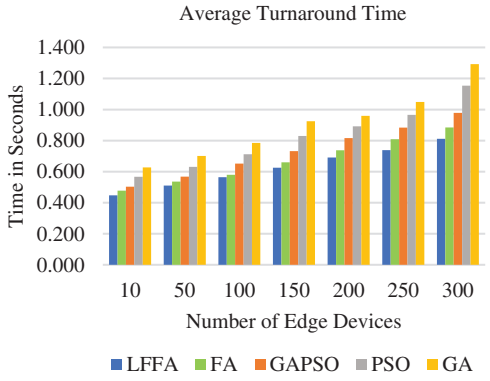

Figure 5: Average turnaround time

6.4 Average Resource Utilization

The main objective of the proposed algorithm is to maximize resource utilization. In the proposed algorithm, the fitness value is determined in such a way that resource utilization is maximized, which is evaluated using Eq. (4). For any resource to become a brighter firefly, it must fulfil the fitness function condition. After analysis, it has been inferred that the resource utilization ratio has increased for the proposed algorithm as the number of edge devices has increased. The same is shown in Fig. 6.

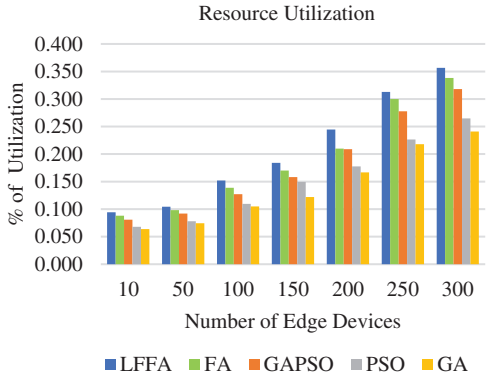

Figure 6: Resource utilization

6.5 Processing Cost

The processing cost of a task depends upon the capacity of allocated computing resources like CPU and memory, as defined in Eq. (8). Full data processing at fog nodes results in higher costs and forwarding all data to the cloud increases delay. The main objective of the proposed algorithm is to find the balance between processing cost and other QoS parameters. Fig. 7 shows that the proposed algorithm supersedes all other algorithms and performs computations with minimal cost.

After evaluation of Pl for all parameters, it has been observed that the proposed algorithm shows better results than FA, GAPSO, GA and PSO algorithm.

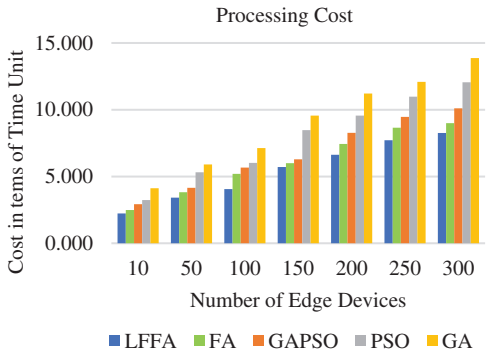

Figure 7: Processing cost

7 Analysis

For the optimization algorithm, QoS parameters, namely task success rate and energy consumption, have been considered and are discussed in this section.

$$QoS = f\,(Task\ Success\ Rate,\ Energy\ Consumption)$$

7.1 Task Success Rate

Analysis of successfully executed tasks by using different resource allocation algorithms is shown in Fig. 8. It has been observed that the percentage of successfully executed tasks is better when LFFA is used than other metaheuristic algorithms.

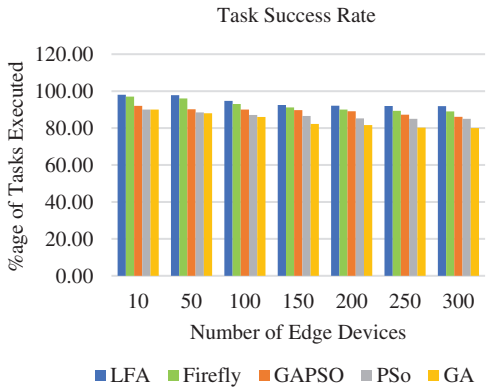

Figure 8: Task success rate

In the firefly algorithm, the resource that matches the requirements of upcoming tasks is considered a brighter one and gets allocated to tasks so that the failure ratio is minimized and the task success rate is increased as compared to other algorithms. Total tasks successfully executed can be evaluated using the Eq. (24).

$$Task\ Success\ Rate\ \ (TSR) = \frac{Total\ number\ of\ \ successfully\ completed\ tasks}{Total\ generated\ tasks} \tag{24}$$

7.2 Energy Consumption

The energy consumption problem is solved by a minimization problem of task execution time and a maximization problem of resource utilization. Fig. 9 displays the variations in energy consumption of several algorithms. According to the observations, GA consumes more energy as the number of devices increases, while LFFA is showing better results among all comparative algorithms. The energy consumption of a resource can be calculated using Eq. (25).

$$EC = RU_t * (EC_{max} - EC_{min}) + EC_{min} \tag{25}$$

where EC is the total energy consumed by a resource, RU_t is the resource utilization at a time 't', EC_{max} is the maximum energy consumption by a resource at the highest load, and EC_{min} is the minimum energy consumption when the resource is in idle mode.

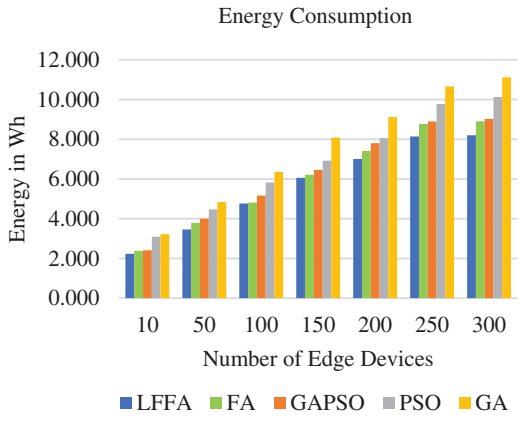

Figure 9: Energy consumption

7.3 Qualitative Analysis

The algorithms were compared quantitatively in earlier sections. This work also presents some qualitative differences among these algorithms, contrasting the basic behaviour of algorithms in Table 10.

With regard to time complexity, 'n' represents population size, 't' is the number of iterations, 'd' is the dimension size, 'g' is the size of generations and 'm' is the size of individuals. It is clearly observed that the time complexity of LFFA is equivalent to FA and better than GAPSO and GA. The time complexity of PSO also depends on the dimension of search space, which is equal to the number of components within a particle. For small dimensions, the time complexity is almost similar to LFFA, but for the higher dimensions, the complexity of PSO will be equal to or greater than $O(n^2t)$, which is more than the complexity of LFFA. So, the resultant time complexity for the algorithms under comparison is GA > GAPSO ≥ PSO ≥ FA ≃ LFFA. Further, the time complexity is

directly proportional to computational complexity. It has been demonstrated in Table 10 that LFFA outperforms other comparative algorithms qualitatively.

Table 10: Qualitative comparison of LFFA w.r.t different metaheuristic algorithms

Sr. No.	Qualitative parameters	LFFA	FA	GAPSO	PSO	GA
1.	Ability to find an optimal solution without local search	High	Low	Moderate	Low	Low
2.	Effect of population size on computational time	Linear	Linear	Exponential	Linear	Exponential
3.	Premature convergence	No	Sometime	Yes	Yes	Yes
4.	Stuck in local optima	No	Sometime	Sometime	Yes	Yes
5.	Methodology	Swarm intelligence with levy distribution	Swarm intelligence	Swarm intelligence with evolution strategy	Swarm intelligence	Natural selection and evolution strategy
6.	Time complexity	O(ntlog(n))	O(ntlog(n))	$O(n^2t)$	O(ndt)	O(g(nmt))

Table 11: Comparative analysis of LFFA w.r.t different resource allocation algorithms

Parameters	FA	GAPSO	PSO	GA	Remarks
Average waiting time	−12.5	−21.18	−36.72	−47.69	Improvement
Average execution time	−5.56	−13.41	−21.82	−28.48	Improvement
Average turnaround time	−6.03	−14.02	−23.03	−30.16	Improvement
Resource utilization	+8.15	+15.39	+34.96	+45.90	Improvement
Processing cost	−11.09	−19.34	−31.98	−41.24	Improvement
Task success rate	+2.09	+5.53	+8.47	+12.11	Improvement
Energy consumption	−5.54	−8.97	−18.55	−26.09	Improvement

8 Conclusion

The main aim of this study is to implement effective and efficient utilization of available resources and to reduce the latency and delay so that IoT applications can be enacted nearby the edge of the network. This study is focused on the resource allocation problem in the area of fog computing. The proposed algorithm supports attaining efficient resource allocation and better QoS. Then it is compared with FA, GAPSO, GA and PSO, and a comparative analysis is presented in Table 11, which shows that LFFA is better for allocating resources to the tasks.

The proposed algorithm reduces average waiting time by 12.5%, 21.18%, 36.72%, 47.69%, average execution time by 5.56%, 13.41%, 21.82%, 28.48%, average turnaround time by 6.03%, 14.02%, 23.03%, 30.16%, processing cost by 11.09%, 19.34%, 31.98%, 41.24% and energy consumption by 5.54%, 8.97%, 18.55%, 26.09% w.r.t FA, GAPSO, PSO and GA but also increases resource utilization by 8.15%, 15.39%, 34.96%, 45.90% and Task success rate by 2.09%, 5.53%, 8.47%, 12.11% in comparison to other algorithms. The negative values depict the improvement for average waiting time, average execution time, average turnaround time, processing cost and energy consumption, but in the case of resource utilization and task success rate improvement is depicted by positive values. The quantitative and qualitative analysis exhibits that the proposed LFFA performs better compared to other metaheuristic algorithms.

Acknowledgement: Authors also acknowledge the department of Computer Science and Engineering of Deenbandhu Chhotu Ram University of Science and Technology for providing various facilities in the research Lab.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[3] S. El Kafhali and K. Salah, "Efficient and dynamic scaling of fog nodes for IoT devices," *The Journal of Supercomputing*, vol. 73, no. 12, pp. 5261–5284, 2017.

[4] R. Mahmud, R. Kotagiri and R. Buyya, "Fog computing: A taxonomy, survey and future directions," In: B. Di Martino, K. -C. Li, L. T. Yang and A. Esposito (Eds.), *Internet of Everything*, Singapore: Springer Singapore, pp. 103–130, 2018.

[5] P. Hu, S. Dhelim, H. Ning and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[6] A. A. A. Gad-Elrab and A. Y. Noaman, "A two-tier bipartite graph task allocation approach based on fuzzy clustering in cloud–fog environment," *Future Generation Computer Systems*, vol. 103, pp. 79–90, 2020.

[7] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei *et al.,* "Fog computing: Focusing on mobile users at the edge," ArXiv150201815 Cs, Mar. 2016, Accessed: Apr. 11, 2021. [Online]. Available: http://arxiv.org/abs/1502.01815

[8] M. Ghobaei-Arani, A. Souri and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *Journal of Grid Computing*, vol. 18, pp. 1–42, 2019.

[9] L. Ni, J. Zhang, C. Jiang, C. Yan and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[10] V. Kochar and A. Sarkar, "Real time resource allocation on a dynamic two-level symbiotic fog architecture," in *2016 Sixth Int. Symp. on Embedded Computing and System Design (ISED)*, Patna, India, pp. 49–55, Dec. 2016.

[11] H. Tran-Dang and D. -S. Kim, "Task priority-based resource allocation algorithm for task offloading in fog-enabled IoT systems," in *2021 Int. Conf. on Information Networking (ICOIN)*, Jeju Island, Korea (South), pp. 674–679, Jan. 2021.

[12] C. Yin, T. Li, X. Qu and S. Yuan, "An optimization method for resource allocation in fog computing," in *2020 Int. Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, Rhodes, Greece, pp. 821–828, Nov. 2020.

[13] D. T. Nguyen, L. B. Le and V. K. Bhargava, "A market-based framework for multi-resource allocation in fog computing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1151–1164, 2019.

[14] S. S. Gill, P. Garraghan and R. Buyya, "ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices," *Journal of Systems and Software*, vol. 154, pp. 125–138, 2019.

[15] M. H. Mousa and M. K. Hussein, "Efficient UAV-based MEC using GPU-based PSO and voronoi diagrams," *Computer Modelling in Engineering & Sciences*, vol. 133, no. 2, pp. 413–434, 2022.

[16] Anupriya and A. Singhrova, "Enhanced whale optimization-based traffic forecasting for SDMN based traffic," *ICT Express*, vol. 7, no. 2, pp. 143–151, 2021.
[17] S. Agarwal, S. Yadav and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *International Journal of Information Engineering and Electronic Business*, vol. 8, no. 1, pp. 48–61, 2016.
[18] A. Basu, S. Mistry, S. Maity and S. Dutta, "A novel energy aware resource allocation algorithm into a P2P based fog computing environment," In: C. Badica, P. Liatsis, L. Kharb and D. Chahal (Eds.), *Information, Communication and Computing Technology*, vol. 1170, Singapore: Springer Singapore, pp. 88–97, 2020.
[19] H. Bashir, S. Lee and K. H. Kim, "Resource allocation through logistic regression and multicriteria decision making method in IoT fog computing," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 2, pp. e3824, 2019.
[20] Q. Wang and S. Chen, "Latency-minimum offloading decision and resource allocation for fog-enabled internet of things networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 12, pp. e3880, 2020.
[21] A. A. Alsaffar, H. P. Pham, C. -S. Hong, E. -N. Huh and M. Aazam, "An architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing," *Mobile Information Systems*, vol. 2016, pp. 1–15, 2016.
[22] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.
[23] K. Hassan, N. Javaid, F. Zafar, S. Rehman, M. Zahid *et al.,* "A cloud fog-based framework for efficient resource allocation using firefly algorithm," In: L. Barolli, F. -Y. Leu, T. Enokido and H. -C. Chen (Eds.), *Advances on Broadband and Wireless Computing, Communication and Applications*, vol. 25, Cham: Springer International Publishing, pp. 431–443, 2019.
[24] S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo and E. Dutkiewicz, "Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4497–4506, 2018.
[25] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan *et al.,* "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing," *IEEE Access*, vol. 7, pp. 115760–115773, 2019.
[26] Q. Li, J. Zhao, Y. Gong and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for internet of everything," *China Communications*, vol. 16, no. 3, pp. 32–41, 2019.
[27] M. Verma, N. Bhardwaj and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *International Journal of Information Technology and Computer Science*, vol. 8, no. 4, pp. 1–10, 2016.
[28] J. Arravinth and D. Manjula, "Multi-agent with multi objective-based optimized resource allocation on inter-cloud," *Intelligent Automation and Soft Computing*, vol. 34, no. 1, pp. 133–147, 2022.
[29] S. Kabirzadeh, D. Rahbari and M. Nickray, "A hyper heuristic algorithm for scheduling of fog networks," in *21st Conf. of Open Innovations Association (FRUCT)*, Helsinki, pp. 148–155, Nov. 2017.
[30] X. -S. Yang, "Firefly algorithm, levy flights and global optimization," arXiv, Mar. 07, 2010. Accessed: May 21, 2022. [Online]. Available: http://arxiv.org/abs/1003.1464
[31] I. Fister, I. Fister, X. -S. Yang and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.
[32] Anu and A. Singhrova, "Optimal healthcare resource allocation in covid scenario using firefly algorithm," *International Journal of Engineering Trends and Technology*, vol. 70, no. 5, pp. 240–250, 2022.
[33] P. Barthelemy, J. Bertolotti and D. S. Wiersma, "A lévy flight for light," *Nature*, vol. 453, no. 7194, pp. 495–498, 2008.
[34] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes," *Physical Review E*, vol. 49, no. 5, pp. 4677–4683, 1994.

[35] M. F. Tasgetiren, Y. -C. Liang, M. Sevkli and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.
[36] S. Abedi, M. Ghobaei-Arani, E. Khorami and M. Mojarad, "Dynamic resource allocation using improved firefly optimization algorithm in cloud environment," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. e2055394(2601–2627), 2022.
[37] V. Yadav, B. V. Natesha and R. M. R. Guddeti, "GA-PSO: Service allocation in fog computing environment using hybrid bio-inspired algorithm," in *TENCON 2019–2019 IEEE Region 10 Conf. (TENCON)*, Kochi, India, pp. 1280–1285, Oct. 2019.
[38] C. Canali and R. Lancellotti, "GASP: Genetic algorithms for service placement in fog computing systems," *Algorithms*, vol. 12, no. 10, pp. 201, 2019.