Cat Swarm Optimization with Lévy Flight for Link Load Balancing in SDN

Kwaku Kwarteng[1], Kwame O. Gyasi[1], Justice O. Agyemang[1], Kwame Agyekum[1], Kingsford Kwakye[1], Ellis M. Sani[1], Emmanuel A. Ampomah[1], and Kusi A. Bonsu[2]

[1]*Kwame Nkrumah University of Science and Technology, Kumasi, Ghana,*
[2]*Sunyani Technical University, Sunyani, Ghana*

https://doi.org/10.26636/jtit.2025.1.1773

Abstract — Efficient network communications with optimal network path selection play a key role in the modern world. Conventional path selection algorithms often face numerous challenges resulting from their limited scope of application. This research proposes a modified swarm intelligence approach, known as cat swarm optimization (CSO) with Lévy flight that is used for network link load balancing and routing optimization. CSO's quick convergence capabilities are suitable for rapid response applications; however, the approach is prone to getting stuck in local optima. Lévy flight enhances search efficiency, thus aiding in escaping local optima. CSO with Lévy flight (CSO-LF) outperforms original CSO and PSO algorithms in terms of solution quality and robustness across various benchmarks. The proposed method has been evaluated in software defined networks (SDN) with nine benchmark functions assessed. CSO-LF achieved the best scores in both the best and worst positions. When used in SDN for link load balancing, CSO-LF demonstrated lower latency and higher throughput than CSO, and lower latency and higher throughput than PSO in a fat tree topology.

Keywords — cat swarm optimization, Lévy flight, load balancing, software-defined networks

1. Introduction

Optimization plays a critical role across numerous scientific areas, as it allows to improve various algorithms of the conventional, evolutionary and nature-inspired metaheuristic variety, thus striving to solve more complex challenges. In recent years, nature-inspired algorithms have gained popularity, especially when it comes to addressing non-linear optimization tasks [1].

Software-defined networks (SDNs) allow to address some of the inherent challenges of modern network management by separating the control and data planes. This separation allows for a centralized view of the network, which facilitates a more effective implementation [2], [3]. Although SDNs alleviate some limitations in scalability and management of traditional network structures, they also introduce new challenges, especially in achieving balanced network loads. Many existing methods face difficulties in reaching a global optimum and handling non-linear dynamics (Fig. 1).

To address these issues, metaheuristic techniques, particularly those inspired by natural phenomena, such as swarm intelligence, have emerged as a promising solution. Moreover,

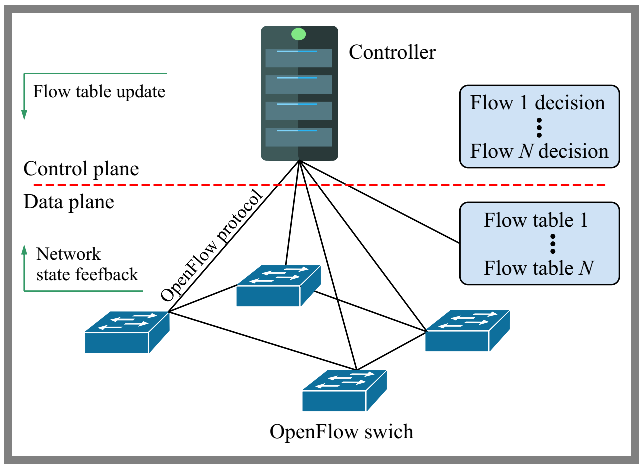

Fig. 1. Architecture of an SDN-based system.

techniques such as anomalous diffusion and Lévy flights offer the potential for improved optimization by enabling larger movements through solution spaces [1].

Bearing in mind the context defined above, this research is devoted to advancing SDN technology by utilizing cat swarm optimization with Lévy flight (CSOLF) methods. The primary objective is to improve load balancing and path optimization within SDNs. This method seeks to address the complexities of modern computer networks, promoting better scalability, efficient resource management, and improved network performance.

The contributions of this work are listed below.

- improvement of the global search capability of the original CSO algorithm by applying Lévy flight,
- implementation of the proposed algorithm in the SDN controller for link load balancing,
- comparison of throughput- and latency-related metrics of the proposed method with the original parameters of cat swarm optimization (CSO) and particle swarm optimization (PSO) algorithms.

PSO, inspired by the behavior of bird or fish swarms [4], was initially introduced as a metaheuristic algorithm for the optimization of functions. Since then, it has been applied to various optimization problems, including dynamic systems [5], pattern matching [6], traveling salesman problem [7], scheduling, and vehicle routing [8]. The CSO algorithm based on cat behavior [9] operates in two modes, seeking and tracing, for exploration and exploitation, respectively. CSO has found applications in function optimization [10], task allocation [11], and workflow scheduling [12].

The remaining parts of this paper are structured as follows. Section 2 discusses related works. Section 3 presents the concepts of Lévy flight and CSO. Section 4 provides a comprehensive explanation and describes the implementation of the proposed algorithm. Section 5 presents the experimental analysis performed in various scenarios. Finally, concluding remarks are presented in Section 6.

2. Related Works

Swarm intelligence and other algorithms are often used in optimization and computational techniques, especially in software-defined networks (SDN) [13]. Leveraging these algorithms, SDN offers an optimal framework for scalable and programmable networks. In complex network management, these algorithms address an extensive solution space and multiple targets. Integrating heuristics with SDN improves load balancing [14]. However, studies in the literature have demonstrated that integrating Lévy flight with metaheuristic algorithms significantly enhances their efficiency, both in SDN and in other applications.

Kolodziejczyk et. al. [15] proposed particle swarm optimization (PSO) methods based on the characteristics of the Lévy distribution. This involves employing the Lévy distribution to start the swarm and using the Lévy flight as a scalar inertia coefficient. Bousmaha *et al.* [16] introduced a training technique that combines PSO with multiverse optimization using Lévy flight. This approach helps prevent premature convergence and achieves a better balance between exploration and exploitation. To address poor performance of quantum-behaved PSO (QPSO) in high-dimensional problems, paper [17] incorporated Lévy flight and straight flight (SF) strategies into QPSO. This method showed strong performance in solving engineering design optimization challenges.

Ant colony optimization (ACO) was improved in [18], based on the Lévy distribution applied to the candidate selection process, and took advantage of the Lévy flight, which increased searching speed and also ensured a better exploration of search space. In article [19], a greedy Lévy ACO was proposed, combining epsilon greedy and Lévy flight strategies to tackle complex combinatorial optimization problems. This method is developed in max-min ACO and is applied to solve the traveling salesman problem. Paper [20] introduced a hybrid max-min ant system (HMMAS) that incorporates the Lévy flight strategy to address the limitations of the traditional approach. HMMAS improves diversity by dynamically adjusting its parameters.

Verma et. al. [21] proposed modified chicken swarm optimization (MCSO) that addresses local optima and early convergence issues in basic CSO by incorporating Lévy flight as a random feature. This enables MCSO to navigate cases where conventional methods may fail to find neighboring solutions. Through experiments on various benchmark functions and pressure vessel design problems, MCSO demonstrated efficient performance, with faster convergence to global optima. Statistical analysis and comparisons with other optimization methods confirmed the effectiveness of MCSO in various problem domains. Article [22] introduced an improved artificial bee colony (ABC) algorithm incorporating Lévy flight (LABC) to improve the exploitation capability of the ABC algorithm for estimations performed in the 3-p distribution. Compared to other metaheuristic algorithms, the results demonstrated that LABC provided more precise maximum likelihood (ML) estimations. The authors of [23] developed a gray wolf optimization algorithm with dynamic adjustment of the inertia weight and a Lévy flight strategy. In the early stages of iteration, Lévy increase the probability of enhancing global search capabilities and boosts population diversity.

To recapitulate, the integration of Lévy into various metaheuristic algorithms has resulted in improvements in solving complex optimization problems. These techniques improved the balance between exploration and exploitation, improved convergence rates, and provided robust solutions to a wide class of computational challenges. However, there are still several optimization algorithms that have not yet been integrated with Lévy flight. Table 1 summarizes the findings obtained from the literature.

3. CSO and Lévy Flight

3.1. Original CSO Algorithm

CSO is a metaheuristic algorithm that mimics the behavior of cats [9]. It alternates between the seeking and tracing phases, representing local and global search for optimal solution. The cat that records the best solution will be kept in memory once the cats have been divided into these two phases, from which new positions and fitness functions will be evaluated. These processes are repeated, until the termination criteria are met [24].

In the search phase, the cat rests or observes and searches for the best solution by slightly adjusting its position and evaluating the results to avoid rapid changes, ensuring a more thorough exploration of the solution space [25].

The position of each cat is adjusted using the following equation:

$$X_{cn} = (1 \pm \text{SRD} \times R) \times X_c \;, \tag{1}$$

where X_c denotes the existing position of the cat, X_{cn} denotes the updated position of the cat, SRD denotes the search range of the selected dimension, and R denotes a random number within the range $0, \ldots, 1$. The X_{cn} parameter determines the direction of the cat's movement.

The fitness solution (FS) measure assesses the effectiveness of each candidate solution or cat in addressing the optimization problem. The FS function assigns a value that reflects how close a solution is to the optimal result, guiding the algorithm's exploration (seeking phase) and exploitation (tracing phase) behaviors. Higher fitness values lead cats to promising regions

Tab. 1. Summary of the literature review.

No.	Topic	Metrics	Solution	Limitation
[15]	PSO and Lévy flight integration	Average performance, standard deviation	Modified particle swarm optimization	Overhead
[16]	Automatic selection of hidden neurons and weights in neural networks for data classification using hybrid PSO multi-verse optimization based on Lévy flight	Standard momentum back propagation and adaptive learning rate	PSO with multi-verse optimization using Lévy flight	Scability issues
[17]	Quantum PSO with optimal guided Lévy and straight flight for solving optimization problems	Friedman rank test	PSO with straight flight and Lévy flight	High computational cost
[18]	An ant colony optimization (ACO) with Lévy flight	Throughput	Elman neural network for network optimization	Slow convergence
[19]	Improving ant colony optimization algorithm with epsilon greedy and Lévy flight	Travelling salesman and related problems	Ant colony optimization algorithm with epsilon greedy and Lévy flight	Complex
[20]	A hybrid max-min ant system by Lévy flight and opposition-based learning	Travelling salesman and related problems	Ant colony optimization with Lévy flight and opposition-based learning	Potential convergence instability
[21]	Lévy's flight guided modified chicken swarm optimization	Win-tie-loss, Bonferroni-Dunn post-hoc, and Wilcoxon tests	Modified chicken swarm optimization to solve early convergence problem of chicken swarm optimization	Single objective problem scenario
[22]	Artificial bee colony with Lévy flights for parameter estimation of 3-p Weibull distribution	Machine learning estimation tests	Artificial bee colony with Lévy flights	Limited scalability on high dimensional problems
[23]	Grey wolf optimization algorithm based on dynamically adjusting inertial weight and Lévy flight strategy	Standard test functions	Grey wolf optimization with Lévy flight	Difficulty in balancing inertia weight adjustments

of the solution space, allowing the algorithm to iteratively converge on the best possible outcome [25].

In the second tracing phase, the cat is hunting its prey in the tracing phase. The position and velocity of the prey are used by the cat to calculate its movement speed and direction after finding the prey while resting in the search phase [25]. The equation for velocity of CSO's cat k in dimension d is:

$$v_{k,d} = v_{k,d} + r_1 \times c_1(X_{\text{best},d} - X_{k,d}) \,, \tag{2}$$

where $v_{k,d}$ denotes the velocity of the cat k, $X_{best,d}$ denotes the best position of the cat, $X_{k,d}$ denotes the position of the k-th cat, c_1 is a constant and r_1 denotes a random number between 0 and 1.

With this velocity, the cat traverses the M-dimensional decision space and reports each new position. The new position is determined by the following formula:

$$X_{k,d,new} = X_{k,d,old} + v_{k,d} \,, \tag{3}$$

where $X_{k,d,new}$ represents the new position of the k-th cat, and $X_{k,d,old}$ represents the present position of the k-th cat.

Completion of the algorithm is determined based on the achievement of termination conditions [26] which include the number of iterations, progress, and time.

3.2. Lévy Flight

Lévy flight is a class of non-Gaussian random walks whose step length is drawn from the Lévy distribution, often in terms of a simple power law equation [27]. The simple power law equation is given as:

$$L \sim |s|^{-\mu}, \quad 0 < \mu \leqslant 2 \,, \tag{4}$$

where L is the length of the step, s is the step and μ is the Lévy exponent, controlling the scale of the steps.

In practice, a Lévy flight step can be modeled as:

$$L = \frac{\text{rand}()}{|\,\text{rand}()\,|^{\frac{1}{\mu}}} \,,$$

where $\text{rand}()$ generates normally distributed random numbers and μ controls the step size.

Tab. 2. Group of test functions used in numerical evaluations [29].

Function	Expression	Dim	Search range
Ackley	$f_1 = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$	2	$-5 \leqslant x \leqslant 5$
Drop wave	$f_2 = -\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$	2	$-5.12 \leqslant x \leqslant 5.12$
Bukin	$f_3 = 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$	2	$-15 \leqslant x_1 \leqslant -5,$ $-3 \leqslant x_2 \leqslant 3$
Three hump camel	$f_4 = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	2	$-5 \leqslant x \leqslant 5$
Matyas	$f_5 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	$-10 \leqslant x \leqslant 10$
Bohachevsky	$f_6 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	5	$-100 \leqslant x \leqslant 100$
Sphere	$f_7 = \sum_{i=1}^{d} x_i^2$	2	$-5.12 \leqslant x \leqslant 5.12$
Sum squares	$f_8 = \sum_{i=1}^{d} ix_i^2$	2	$-10 \leqslant x \leqslant 10$
Sum of different powers	$f_9 = \sum_{i=1}^{d} \lvert x_i\rvert^{i+1}$	2	$-1 \leqslant x \leqslant 1$

This equation models the mixture of small, local searches (short steps) and long-range exploration (large jumps) characteristic of Lévy flight, making it useful for global optimization in metaheuristics. Lévy flights are more efficient than Brownian random walks when it comes to exploring large-scale search spaces. There are many reasons to explain this efficiency, one of which stems from the fact that the variance of Lévy flights increases much faster than the linear relationship of Brownian random walks [28].

4. Proposed Method

Despite the many CSO variants available in the literature, the problems of premature convergence and generating inefficient results continue to persist. The Lévy flight method is used to solve these problems and enables CSO to generate more efficient results. This method ensures that the CSO, which is unable to perform the global search well, will enjoy better search efficiency and will not be trapped in local minima. In the CSO-LF method, the Gaussian random walks used in the tracing mode are replaced with Lévy flight.

4.1. CSO-LF Algorithm

The CSO-LF follows a process similar to the original CSO in the seeking phase. However, the movement that occurs in the tracing phase is where the novelty of the algorithm is shown.

In the tracing phase, the new position of the cat is calculated by:

$$X_{k,d,\text{new}} = \left[\alpha \cdot L(\beta)\right] X_{k,d,\text{old}} + v_{k,d} \,, \tag{5}$$

where α is the scaling parameter that determines the step size in the Lévy flight and $L(\beta)$ denotes a random number vector obtained using Lévy distribution with the β exponent.

In Eq. (5), the movement taking place in the tracing phase, which is originally powered by the Brownian random walk, is replaced with the Lévy flight. In the Lévy flight, the step size follows a probability distribution, which allows for occasional long jumps [27]. The scaling parameter affects the frequency with which longer jumps occur, and thus, it is a very important parameter in ensuring the overall performance of the algorithm. In this paper, the scaling parameter has been fine-tuned to ensure optimal search efficiency.

The algorithm is terminated based on achieving the termination conditions. In this study, is defined by a predetermined number of iterations. The algorithm continues until the last iteration has been completed. The pseudocode of the proposed solution is given as Algorithm 1.

4.2. Test Functions

A selection of unimodal and multimodal test functions [29] was used to perform a numerical evaluation for the suggested approach (as shown in Tab. 2). The functions were chosen for their diverse shapes and scaling ability. f_1, f_2 and f_3 are functions with many local minima, f_4 is a valley-shaped function, f_5 is a plate-shaped function and f_6, f_7, f_8 and f_9 are bowl-shaped functions.

The initial population was generated from a uniform distribution across the search space, proportional to its specified boundaries. Table 2 provides the range limits for all functions. The maximum iteration count was set to 100 for a population size of 50.

The numerical evaluations were carried out in five trials, with each trial producing a set of optimization results. For each function, three key metrics were recorded: the best, the worst and the average best solutions in the five trials. Here, the “worst” outcome represents the highest (least optimal) function value obtained in any of the five tests, indicating the algorithm’s most unfavorable performance. This metric provides information on variability and resilience under less favorable conditions. These metrics were then ranked using a standard competition ranking system, where lower scores in the specific categories indicate more effective algorithms.

Algorithm 1 Pseudo-code for cat swarm optimization with Lévy flight (CSO-LF)

```
1: Initialize parameters
2: $\alpha$ = scaling parameter (step size in Lévy flight)
3: $\beta$ = Lévy distribution exponent
4: Number of cats (agents), iterations, dimensions
5: Initialize positions $X[k][d]$ and velocities $v[k][d]$ for each cat
6: while stopping condition is not met do
7:     for each cat $k$ do
8:         if in the seek phase then
9:             Perform the seek phase
               (according to the standard CSO)
10:        else in the tracing phase
11:            for each dimension $d$ of the cat $k$ do
12:                Generate a random vector $L(\beta)$
13:                Compute a new position:
14:                $X[k][d]_{\text{new}} = (\alpha \cdot L(\beta)) \cdot$
                   $X[k][d]_{\text{old}} + v[k][d]$
15:            end for
16:            Update the cat position $X[k]$ with $X[k]_{\text{new}}$
17:        end if
18:    end for
19:    Update velocities $v[k][d]$ based on the new position
20:    Evaluate the fitness of each cat’s new position
21:    Identify and update the best position found so far
22: end while
23: Output the best position found and its fitness value
```

Tab. 3. Notations used.

Notation	Description
G	Topology of the given network
V	Set of all switches
E	Set of all links that connect the switches together
$\phi_{s,d}$	Set of all feasible paths for switch pair v_s and v_d
I	Initial population
$F(k)$	Fitness function of the k-th cat
$P_{utilization_k}$	Time it takes for a packet to be transmitted on the k-th path
$P_{congestion_k}$	Available bandwidth on the k-th path

4.3. Network Modelling

The proposed CSO-LF is modeled as a load balancing problem that was equated to find the path with the least cost in the SDN environment whose parameters are shown in Tab. 3. From the data plane perspective, the topology of the network can be modeled by:

$$G = (V, E) , \tag{6}$$

where V is a set of switches, given by $V = v_1, \ldots, v_n$ with $n = |N|$ and E is a set of links that connect the switches to each other, given by $E = e_1, \ldots, e_m$ with $m = |E|$.

The algorithm seeks to minimize path utilization $P_{utilization}$ and path congestion $P_{congestion}$ of a given topology. Based on the condition above, the objective function is designed, namely:

$$\text{Fitness function} = \min(P_{utilization} \times P_{congestion}) , \tag{7}$$

In CSO-LF, the SDN controller first generates, randomly, the initial population, i.e. a set of cats. The position of each cat, which is a possible and available path, consists of a set of switches that can connect the sender side and the receiver side. This is defined by the following:

$$C = \left\{ c_k \mid c_k \in \phi_{(s,d)}, (s,d) \in V \right\}, \ \forall k, \quad k = 1, \ldots, K, \tag{8}$$

where $\phi_{(s,d)}$ is a set of all feasible paths for each pair v_s and v_d.

Tab. 4. Network topologies used in evaluation.

Mesh topology	Fat tree topology
10 switches, 10 hosts	20 switches, 16 hosts
20 switches, 20 hosts	45 switches, 54 hosts
30 switches, 30 hosts	80 switches, 124 hosts
40 switches, 40 hosts	125 switches, 250 hosts
50 switches, 50 hosts	180 switches, 432 hosts

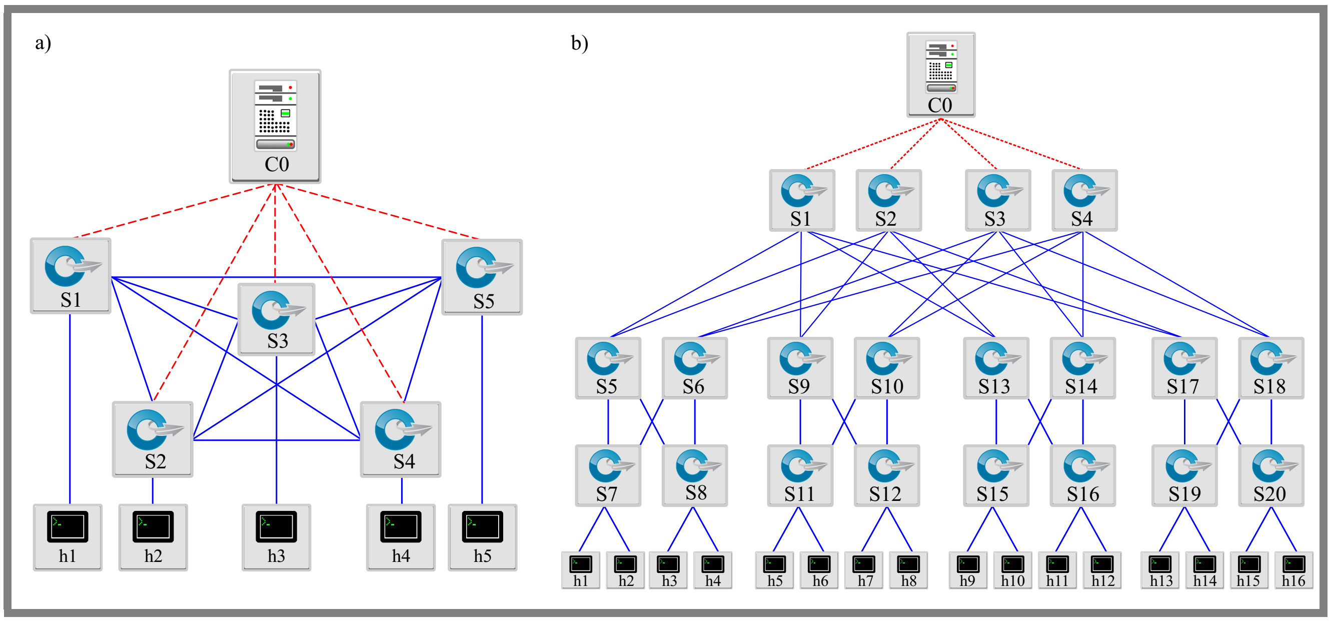

Fig. 2. Test topologies: a) mesh and b) fat tree.

Each path is made up of switches V_k and links E_k, where $V_k = \{v_i | v_i \in c_k\}$ is the set of switches in cluster c_k and $E_k = \{e_{i,j} | e_{i,j} \in c_k\}$ is the set of links in cluster c_k.

After initialization, the fitness value is calculated for each initialization condition. In this paper, a fitness function corresponding to path utilization and congestion is taken into consideration.

Let us assume that the generated initial population is I, such that $I = p_1, p_2, \ldots, p_k$, the fitness function of cat k is given as:

$$F(k) = \min \left(P_{utilization_k} \times P_{congestion_k}, \quad k \in C \right), \tag{9}$$

where $P_{utilization_k}$ defines how long a packet awaits to be transmitted on the k-th path denoted by:

$$P_{utilization_k} = \frac{\lambda_k}{\mu_k}, \tag{10}$$

This provides a good cost metric, since it is low for low loads and goes to infinity for very high loads. $P_{congestion_k}$ defines the available bandwidth of the path and is inversely proportional to its bandwidth, denoted by:

$$P_{congestion_k} = \frac{1}{P_{bandwidth_k}}. \tag{11}$$

The CSO-LF algorithm uses a swarm of cats to perform a search for the possible path, with each cat representing a candidate path. The algorithm iteratively updates the position of each cat based on a Lévy flight step. The new position is chosen from the list of possible paths based on the fitness of the new position. If the fitness of the new position is greater than the fitness of the current position, the cat moves to the new location. This process continues until the assumed number of iterations is reached.

4.4. Implementing CSO-LF Generated Paths in SDN

In SDN environments, after the CSO-LF algorithm has determined the best path for network traffic, the selected path must be implemented as flow entries in the switches controlled by the SDN controller. This process involves several steps:

- **Path encoding and flow rule generation**. The path selected by the CSO-LF algorithm needs to be encoded into a series of flow rules. Each flow rule specifies how the network traffic that satisfies certain criteria should be treated. Typically, these flow rules include such information as source and destination addresses, ports, and quality-of-service requirements.
- **Flow table update**. The SDN controller processes the received flow rules and updates the flow tables of the relevant switches. Flow tables are used by switches to determine how to forward network traffic. The SDN controller installs the new flow entries in the switches along the selected path. If necessary, it may also remove any old flow entries associated with the previous path.
- **Flow table matching**. When network traffic arrives at a switch, the switch examines its flow table to determine how to handle the traffic. The switch matches the incoming packets with the installed flow rules. If a match is found, the switch takes the specified action, such as forwarding the traffic along the path defined by the flow rule.
- **Packet forwarding**. In the next step, network traffic is forwarded by the switches along the path defined by the flow rules. Switches ensure that traffic follows the chosen path, and can also perform such actions as quality of service (QoS) shaping, security checks, and other functions specified by the flow rules.
- **Dynamic path adaptation**. In SDN environments, network conditions and requirements may change. Therefore, the SDN controller continuously monitors the network and can

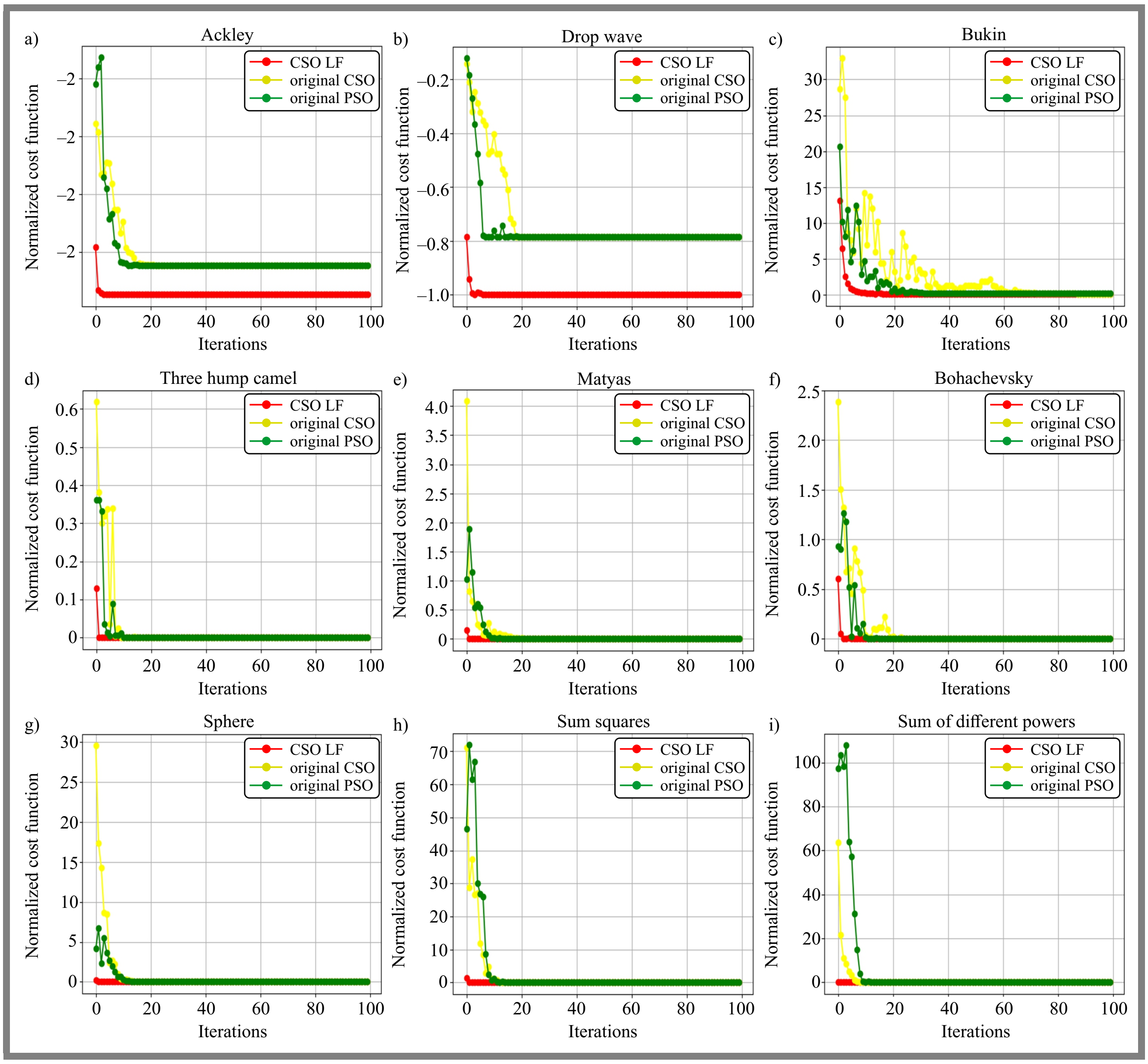

Fig. 3. Convergence plots for the three algorithms for an initial population of 50 agents in 100 iterations.

dynamically adapt flow entries if needed. If the controller detects changes in the network, it can recalculate paths (using the CSO-LF algorithm), generate new flow rules, and update switches to reflect these changes.

By following these steps, the selected path generated by the CSO-LF algorithm is implemented in the SDN network. This dynamic and software-driven approach to path selection and flow rule management is one of the key advantages of SDN, as it facilitates efficient traffic engineering and offers adaptability in response to changing network conditions.

4.5. Configuration and Setup

The software environment used for the CSO-LF experiment is the Python 3.8.10 programming language. Other parameter settings of the algorithm include: SMP = 2, CDC = 1, SRD = 0.1, SPC = false, MR = 0.67. The parameters of the Lévy component include $\beta = 1.5$ and $\alpha = 0.2$. SDN simulations were conducted in the Oracle VM VirtualBox 6.1 hypervisor with a Linux Ubuntu (64-bit) operating system, Mininet emulator, RYU controller, and an OpenFlow communication protocol.

Two different network topologies of different sizes were used to test the proposed method: mesh and fat tree topology. These topologies were chosen for their different behaviors and logical arrangements.

The mesh topology provides high redundancy and fault tolerance, while the tree topology offers hierarchical organizational design, fault tolerance, and high-level scalability.

Figure 2 shows the different topologies connected to a centralized controller, while Tab. 4 shows different topology sizes used to test the proposed algorithm.

4.6. Performance Evaluation Method

In this research, latency and network throughput are considered to be key performance metrics. Throughput represents the rate at which data is successfully transferred from a source to a destination over a communication channel [28]. The equation to calculate throughput is:

$$\text{Throughput} = \frac{\text{Amount of data transferred}}{\text{Time taken}}. \tag{12}$$

Latency, also known as delay, refers to the time it takes a data packet to travel from its source to its destination. It includes various components, such as transmission delay, propagation delay, queueing delay, and processing delay. The equation for calculating latency depends on the specific components involved in the process, but may be simplified as:

$$\begin{aligned}\text{Latency} = \text{Transmission delay} + \text{Propagation delay}\\ + \text{Queuing delay} + \text{Processing delay}\end{aligned}. \tag{13}$$

5. Results and Discussions

In this section, the experimental results of the proposed methodology are analyzed and its performance is evaluated.

5.1. CSO-LF

Three algorithms (i.e., CSO, PSO, and CSO-LF) are taken through the optimization process for the chosen test functions. In Tab. 5, the results of numerical experiments are displayed.

The results indicate that CSO-LF consistently achieves the highest precision across all the test functions, especially in Ackley, three-hump camel, Bohachevsky, sphere, sum squares, and sum of different powers. It excels in locating positions closest to the global minima with low scores, showcasing superior convergence. In general, CSO-LF emerges as the most accurate algorithm. This comparison suggests that CSO-LF is the most effective solution for optimization tasks that require a high degree of accuracy.

Figure 3 presents a collection of graphs comparing the convergence properties of all three algorithms. The plots depict the best optimization outcome after five trials.

The proposed approach, which converges within a comparatively smaller number of iterations, outperforms the other two algorithms (CSO and PSO), according to the convergence plots.

5.2. CSO-LF in SDN

The performance of CSO-LF in SDN is evaluated using latency and throughput metrics across two different topologies. The best-performing algorithm should have the lowest latency and the highest throughput. In Fig. 4a, it was observed that CSO-LF generally underperformed in terms of latency, but Fig. 4b shows it outperformed the original CSO as far as the throughput measure is concerned.

In Figs. 4c–d, one may see that CSO-LF outperformed CSO in terms of latency and throughput, while Fig. 4a shows that the proposed method had the highest latency. This is because

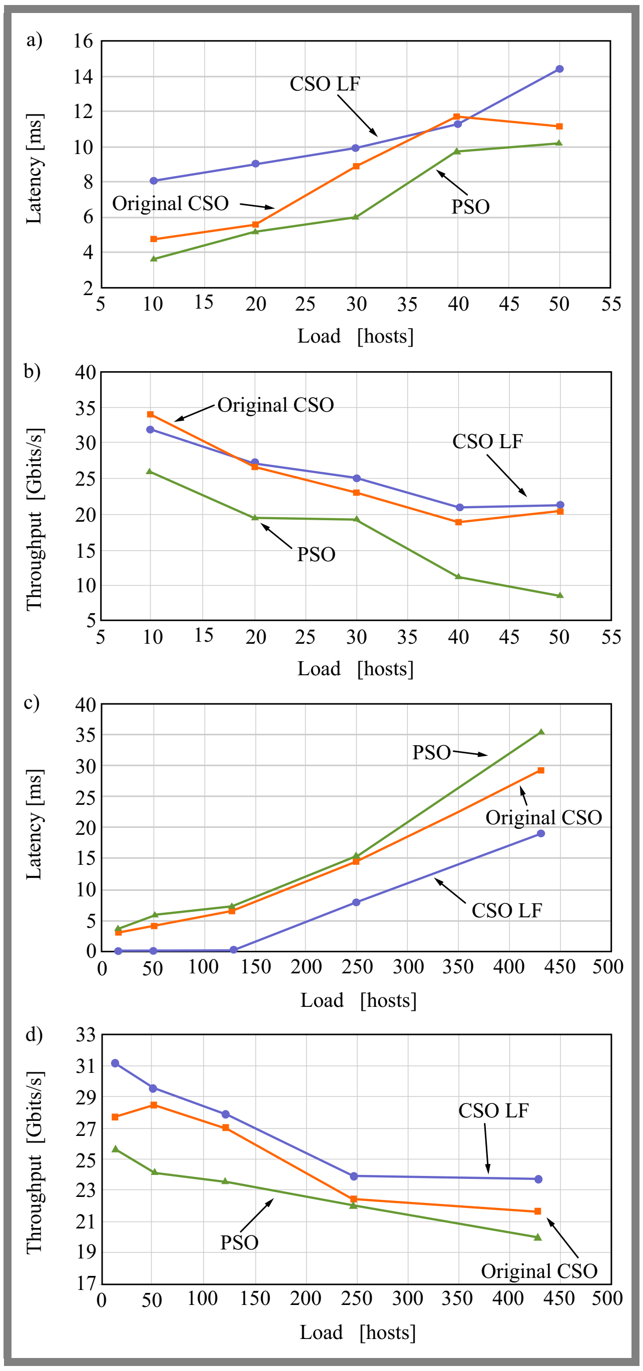

Fig. 4. Simulated latency and throughput results of the three algorithms for: a-b) mesh topology and c-d) fat tree topology.

in a small-scale network, the distances between nodes are relatively short. However, Lévy flights have occasional long jumps, which may cause a search to go beyond adjacent nodes and take a considerable amount of time to locate the optimal path for transmission of a packet – hence the higher latency. Furthermore, PSO recorded the lowest latency, highlighting its potential use case in relatively smaller networks.

From Fig. 4b, it was also observed that the proposed method had a lower throughput varying between 10 and 20 hosts, but its value became higher as the number of hosts increased. Again, Lévy flight steps take more time to converge to an optimal solution in smaller networks due to the initial explo-

Tab. 5. Comparison between CSO-LF, CSO, and PSO based on their positions and scores.

Function	Algorithm	Best score	Best position	Worst score	Worst position
Ackley	CSO-LF	–9.46	[2.6E–08, 6.9E–13]	3.58	[–8.19, –3.67]
	CSO	–8.46	[6.28, 6.28]	3.49	[–7.52, –2.99]
	PSO	–8.46	[6.28, 6.28]	3.31	[–3.96, 8.13]
Drop wave	CSO-LF	–1.00	[–7.07E–15, –9.58E–13]	–0.01	[–0.25, 0.66]
	CSO	–0.79	[0.18, 0.67]	–0.05	[0.60, 4.74]
	PSO	–0.79	[–0.09, 0.06]	–0.01	[3.73, –7.32]
Bukin	CSO-LF	0.10	[6.43E–19, –3.45E–20]	64.00	[–0.11, 0.41]
	CSO	0.24	[9.99, 0.99]	235.6	[7.66, –4.96]
	PSO	0.20	[10.00, 1.00]	46.98	[15.60, 2.21]
Three hump camel	CSO-LF	7.76E–34	[–2.76E–19, –2.78E–17]	140 779	[–9.83, 5.79]
	CSO	9.30E–08	[1.02E–04, –3.24E–04]	148 414	[9.92, –1.84]
	PSO	1.06E–25	[2.47E–13, –1.32E–13]	18 325	[7.07, –3.21]
Matyas	CSO-LF	1.06E–31	[–2.36E–21, 6.38E–16]	0.22	[–0.09, –0.05]
	CSO	4.27E–05	[–0.01, –0.01]	123.87	[8.68, –9.85]
	PSO	3.83E–18	[4.03E–09, 2.73E–09]	3.26	[–4.62, –2.87]
Bohachevsky	CSO-LF	0.0	[–2.80E–27, –2.05E–28]	0.60	[–0.56, –0.07]
	CSO	1.23E–10	[2.23E–06, –1.24E–06]	171.16	[7.65, 7.48]
	PSO	0.0	[–1.36E–09, 1.23E–09]	59.59	[2.19, –5.20]
Sphere	CSO-LF	1.11E–24	[1.57E–14, –1.05E–12]	2.36	[0.93, –0.61]
	CSO	7.73E–06	[–0.01, 0.01]	121.58	[4.77, –5.93]
	PSO	2.23E–17	[2.06E–09, 1.99E–10]	82.70	[3.04, 1.69]
Sum squares	CSO-LF	1.69E–20	[4.33E–11, 1.29E–15]	976.26	[–1.72, 8.39]
	CSO	2.71E–06	[–0.01, 0.01]	703.72	[1.32, 7.91]
	PSO	1.42E–17	[–2.87E–10, 5.55E–10]	323.38	[0.19, 1.11]

Tab. 6. Average performance for mesh and fat tree topologies.

Topology	Latency [ms]			Throughput [Gbps]		
	CSO-LF	CSO	PSO	CSO-LF	CSO	PSO
Mesh	9.50	8.37	6.82	25.26	22.50	16.85
Fat tree	5.48	11.51	13.16	27.27	23.40	23.02

Tab. 7. Percentage difference in performance between CSO-LF and CSO.

Topology	Latency	Throughput
Mesh	–32.84%	39.94%
Fat tree	82.40%	16.90%

ration phase, which can result in lower throughput during this convergence period. In larger networks, the algorithm has more opportunities to explore and discover better paths, and

Tab. 8. Percentage difference in performance between CSO-LF and PSO.

Topology	Latency	Throughput
Mesh	–12.65%	11.56%
Fat tree	70.98%	15.28%

the convergence phase may be shorter relative to the size of the network, leading to larger throughput values.

Tables 7–8 show the percentage differences in performance between CSO-LF and two other optimization algorithms, CSO and PSO, across two network topologies.

In the mesh topology, CSO-LF's latency is 12.65% higher compared to CSO, indicating that CSO performs better in terms of latency in this topology. In terms of throughput, CSO-LF shows an 11.56% improvement over CSO, suggesting that CSO-LF achieves higher throughput in the mesh topology. In the fat tree topology, the percentage differences are more substantial. CSO-LF exhibits a significantly lower latency (70.98%) compared to CSO, indicating a substantial improvement in latency performance. For throughput, CSO-LF also shows a 15.28% improvement over CSO in the fat tree topology, although the difference is not as significant as in the case of latency.

In the mesh topology, CSO-LF shows a latency that is 32.84% higher compared to PSO, indicating that PSO performs better in terms of latency in this network topology. However, in terms of throughput, CSO-LF demonstrates a substantial improvement (39.94%) over PSO, suggesting that CSO-LF achieves significantly higher throughput in the mesh topology compared to PSO. In the fat tree topology, CSO-LF again exhibits a significant improvement over PSO, with a difference of 82.40% in terms latency and 16.90% in terms of throughput. This indicates that CSO-LF outperforms PSO by a large margin, both in terms of latency and throughput, in the fat tree topology.

In general, the latency increased and the throughput decreased with increasing network size. However, CSO-LF generally shows superior performance compared to CSO and PSO in terms of both latency and throughput. The analysis reveals that the proposed method exhibits lower latency, resulting in faster response and reduced data transmission delays. Additionally, it achieves higher throughput, enabling more efficient data transfer and improved network capacity utilization.

6. Conclusions

This research introduces CSO-LF as a prospective solution for tackling optimization problems, particularly in SDN link load balancing. While CSO demonstrates rapid convergence, making it ideal for applications requiring quick responses, its vulnerability to become stuck in local optima poses a limitation. CSO-LF addresses this issue by integrating the Lévy flight technique, thus augmenting search optimality and offering improved performance in navigating complex optimization landscapes.

The proposed method was evaluated on nine popular functions. The numerical results showed that CSO-LF achieved the best scores in terms of the best and worst positions. When implemented in SDN for link load balancing, CSO-LF recorded a lower latency and a throughput that was 15.28% higher compared to CSO, and latency that was 82.40% lower when compared to PSO in the fat tree topology. Its versatility suggests potential applications in controller placement, virtual network mapping, flow entry optimization, and signal processing.

Future research efforts should consider investigating the scaling parameter, which plays a pivotal role in determining Lévy flight step sizes. Furthermore, CSO-LF's application to solve various networking optimization challenges beyond load balancing should be explored as well, and comparative studies with other algorithms should be conducted for broader validation. Evaluation of its scalability and performance in even larger and more complex network environments (such as multicontroller scenarios) is crucial for real-world use.

References

[1] *Nature-Inspired Algorithms and Applied Optimization*, ed. by X.-S. Yang, Springer, Cham, 341 p., 2018 (https://doi.org/10.1007/978-3-319-67669-2).

[2] P. Goransson, C. Black, and T. Culver, *Software Defined Networks, A Comprehensive Approach*, 2nd ed., Elsevier, Cambridge, 2017 (ISBN: 9780128045794).

[3] H. Qi and K. Li, *Software Defined Networking Applications in Distributed Datacenters*, Springer, Cham, 76 p., 2016 (https://doi.org/10.1007/978-3-319-33135-5).

[4] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization", *Proc. of the IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995 (https://doi.org/10.1109/ICNN.1995.488968).

[5] X. Hu and R.C. Eberhart, "Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems", *Proc. of the 2002 Congress on Evolutionary Computation. CEC'02*, pp. 1666–1670, 2002 (https://doi.org/10.1109/CEC.2002.1004492).

[6] M. Kong, P. Tian, and Y. Kao, "A New Ant Colony Optimization Algorithm for the Multidimensional Knapsack Problem", *Computers & Operations Research*, vol. 35, pp. 2672–2863, 2008 (https://doi.org/10.1016/j.cor.2006.12.029).

[7] V. Pureza and P.M. Franca, "Vehicle Routing Problems via Tabu Search Metaheuristic", Centre De Recherche Sur Les Transports Publication, pp. 142–149, 1991.

[8] H. Keller, U. Pferschy, and D. Pisinger, *Knapsack Problem*, Springer, Berlin, 568 p., 2003 (https://doi.org/10.1007/978-3-540-24777-7).

[9] S.C. Chu, P.W. Tsai, and J.S. Pan, "Cat Swarm Optimization", *PRICAI 2006: Trends in Artificial Intelligence*, pp. 854–858, 2006 (https://doi.org/10.1007/978-3-540-36668-3_94).

[10] O. Bozorg-Haddad, *Advanced Optimization by Nature-Inspired Algorithms*, Springer, Singapore, 174 p., 2018 (https://doi.org/10.1007/978-981-10-5221-7).

[11] I. Boussaid, J. Lepagnot, and P. Siarry, "A Survey on Optimization Metaheuristics", *Information Sciences*, vol. 237, pp. 82–117, 2013 (https://doi.org/10.1016/j.ins.2013.02.041).

[12] M. Andresen *et al.*, "Simulated Annealing and Genetic Algorithms for Minimizing Mean Flow Time in an Open Shop", *Mathematical and Computer Modelling*, vol. 48, pp. 1279–1293, 2008 (https://doi.org/10.1016/j.mcm.2008.01.002).

[13] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 87–98, 2014 (https://doi.org/10.1145/2602204.2602219).

[14] M. Hamdan *et al.*, "A Comprehensive Survey of Load Balancing Techniques on Software-defined Network", *Journal of Network and Computer Applications*, vol. 174, 2021 (https://doi.org/10.1016/j.jnca.2020.102856).

[15] J. Kolodziejczyk and Y. Tarasenko, "Particle Swarm Optimization and Lévy Flight Integration", *Procedia Computer Science*, vol. 192, pp. 4658–4671, 2021 (https://doi.org/10.1016/j.procs.2021.09.244).

[16] R. Bousmaha, R.M. Hamou, and A. Amine, "Automatic Selection of Hidden Neurons and Weights in Neural Networks for Data Classification Using Hybrid Particle Swarm Optimization, Multi-verse Optimization Based on Lévy Flight", *Evolutionary Intelligence*, vol. 15, pp. 1695–1714, 2022 (https://doi.org/10.1007/s12065021-00579-w).

[17] X. Liu, G.-G. Wang, and L. Wang, "LSFQPSO: Quantum Particle Swarm Optimization with Optimal Guided Lévy Flight and Straight Flight for Solving Optimization Problems", *Engineering with Computers*, vol. 38, pp. 4651–4682, 2022 (https://doi.org/10.1007/s00366-021-01497-2).

[18] Y. Liu and B. Cao, "A Novel Ant Colony Optimization Algorithm with Lévy Flight", *IEEE Access*, vol. 8, pp. 67205–67213, 2020 (https://doi.org/10.1109/ACCESS.2020.2985498).

[19] Y. Liu, B. Cao, and H. Li, "Improving Ant Colony Optimization Algorithm with Epsilon Greedy and Lévy Flight", Complex and

Intelligent Systems, vol. 7, pp. 1711–1722, 2021 (https://doi.org/10.1007/s40747-020-00138-3).

[20] Z. Zhang, Z. Xu, S. Luan, and X. Li, "A Hybrid Max-min Ant System by Lévy Flight and Opposition-based Learning", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 35, 2021 (https://doi.org/10.1142/S0218001421510137).

[21] S. Verma, S.P. Sahu, and T.P. Sahu, "MCSO: Lévy's Flight Guided Modified Chicken Swarm Optimization", *IETE Journal of Research*, vol. 70, 2024 (https://doi.org/10.1080/03772063.2023.2194265).

[22] A. Yonar and N.Y. Pehlivan, "Artificial Bee Colony with Lévy Flights for Parameter Estimation of 3-p Weibull Distribution", *Iranian Journal of Science and Technology*, vol. 44, pp. 851–864, 2020 (https://doi.org/10.1007/s40995-020-00886-4).

[23] Y. Chen, J. Xi, H. Wang, and X. Liu, "Grey Wolf Optimization Algorithm Based on Dynamically Adjusting Inertial Weight and Lévy Flight Strategy", *Evolutionary Intelligence*, vol. 16, pp. 917–927, 2023 (https://doi.org/10.1007/s12065-022-00705-2).

[24] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier, Waltham, 222 p., 2014 (https://doi.org/10.1016/C2013-0-01368-0).

[25] X.-S. Yang, *Nature-inspired Algorithms and Applied Optimization*, Springer, Cham, 341 p., 2018 (https://doi.org/10.1007/978-3-319-67669-2).

[26] A.O. Jefia, S.I. Popoola, and A.A. Atayero, "Software-defined Networking: Current Trends, Challenges, and Future Directions", *Proc. of the International Conference on Industrial Engineering and Operations Management*, pp. 1677–1685, 2018 (https://doi.org/10.46254/NA03.20180435).

[27] A. Chechkin, R. Metzler, J. Klafter, and V.Y. Gonchar, "Introduction to the Theory of Lévy Flights", in: *Anomalous Transport: Foundations and Applications*, pp. 129–162, 2008 (https://doi.org/10.1002/9783527622979.ch5).

[28] A.A. Al-Temeemy, J.W. Spencer, and J.F. Ralph, "Lévy Flights for Improved Ladar Scanning", *2010 IEEE International Conference on Imaging Systems and Techniques*, Thessaloniki, Greece, 2010 (https://doi.org/10.1109/IST.2010.5548519).

[29] S. Surjanovic and D. Bingham, "Optimization Test Problems", *Virtual Library of Simulation Experiments: Test Functions and Datasets*, SFU [Online] (https://www.sfu.ca/~ssurjano/optimization.html).

Kwaku Kwarteng, M.Phil.
Department of Telecommunication Engineering
https://orcid.org/0009-0009-4416-8335
E-mail: akyeampongkwaku30@gmail.com
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Kwame O. Gyasi, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0000-0002-4923-4452
E-mail: kotenggyasi@knust.edu.gh
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Justice O. Agyemang, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0000-0002-9949-3823
E-mail: justice.agyemang@knust.edu.gh
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Kwame Agyekum, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0000-0002-7935-9950
E-mail: kooagyekum@knust.edu.gh
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Kingsford Kwakye, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0009-0006-0900-1685
E-mail: ksobengkwakye@knust.edu.gh
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Ellis M. Sani, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0000-0001-9344-2075
E-mail: smellis.coe@knust.edu.gh
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Emmanuel A. Ampomah, Ph.D.
Department of Telecommunication Engineering
https://orcid.org/0000-0001-6498-4000
E-mail: eaffume@gmail.com
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
https://www.knust.edu.gh

Kusi A. Bonsu, Ph.D.
Department of Electrical & Electronic Engineering
https://orcid.org/0000-0002-5474-5206
E-mail: kusiankrah@stu.edu.gh
Sunyani Technical University, Sunyani, Ghana
https://stu.edu.gh