Mesopotamian journal of Big Data
Vol. **(2025)**, 2025, **pp**. 295–312
DOI: https://doi.org/10.58496/MJBD/2025/019 , ISSN: 2958-6453
https://mesopotamian.press/journals/index.php/BigData

Research Article

Enhancing Throughput in a Network Function Virtualization Environment via the Manta Ray Foraging Optimization Algorithm

Sanaa S. Alwan [1,*], Asia Ali Salman [1]

[1] *College of Computer Science, University of Technology, Baghdad, Iraq*

ARTICLEINFO

Article History

Received 01 Jul 2025
Revised 31 Aug 2025
Accepted 09 Sep 2025
Published 26 Oct 2025

Keywords

Network function virtualization (NFV)

Manta ray foraging optimization (MRFO)

Swarm Intelligence

Virtual Network Functions (VNFs)

ABSTRACT

Network function virtualization (NFV) has emerged as a transformative paradigm in which traditional hardware appliances are replaced with virtual network functions (VNFs) running on commodity hardware. While NFV offers scalability and flexibility, it faces major challenges in sustaining high throughput and minimizing resource overhead under dynamic traffic conditions. In particular, flooding algorithm-based request propagation often leads to excessive redundancy, congestion, and resource waste. To address this limitation, this study applies Manta ray foraging optimization (MRFO), a swarm intelligence algorithm inspired by natural foraging behaviors, to optimize packet routing and resource allocation in NFV environments. The research employs a Barabási–Albert (BA) scale-free topology model to simulate realistic NFV infrastructures. The performance is evaluated by comparing the conventional flooding algorithm with MRFO-based routing under varying network sizes and time-to-live (TTL) values. The key metrics include throughput, packet delay, CPU and memory utilization, and the success rate. The simulation results demonstrate that MRFO consistently outperforms flooding in medium- and large-scale networks, achieving up to 53.6% improvement in throughput, reduced average delay (2.6 s → 1.7 s), and more balanced resource utilization across NFVs. However, in small-scale networks with limited routing paths, MRFO introduces computational overhead that reduces performance compared with the flooding algorithm. These findings highlight the significance of swarm intelligence for NFV optimization, showing that MRFO is best suited for scalable, dynamic infrastructures such as the cloud, Internet of Things (IoT), and 5G edge networks. This study contributes a novel integration of MRFO with flooding algorithm-based propagation, offering new insights into adaptive and resource-aware NFV optimization strategies.

1. INTRODUCTION

Communication systems can now be designed using NFV, a transformative technology that enhances modern networks. Instead of hardware appliances, abstract functions are executed through software on commercial off-the-shelf (COTS) hardware. The paradigm is scalable, flexible and cost-efficient in cloud (IoT) and 5G edge networks [1][2]. Nevertheless, no matter how beneficial NFV can be, there are lingering problems associated with achieving and sustaining system throughput and the optimal use of resources under dynamic traffic conditions. One of the major problems is the mechanism of packet propagation: as they are propagated traditionally, by the flooding algorithm (overspreading), they induce unnecessary redundancy that can lead to congestion, the consumption of CPU/memory resources, and a consequent throughput decline [3]. These difficulties are especially inapplicable to large-scale NFV deployments where the uncontrolled spread of requests may rapidly exhaust the system.

Scientists have examined several optimization methods to address these problems. Swarm intelligence and metaheuristic algorithms, including particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], the whale optimization algorithm (WOA) [6], and artificial bee colony (ABC) [7], have been deployed for the placement, scheduling, and load balancing of VNFs. The most recent solutions apply reinforcement learning (RL) [8] and federated learning (FL) [9] to adaptive NFV management. Although these methods offer benefits, they have restraints, which include premature convergence, high computational overheads, and limited adaptability to a highly dynamic NFV environment.

*Corresponding author. Email: cs.24.25@grad.uotechnology.edu.iq

The research gap is the low coverage of swarm intelligence algorithms as a method of optimizing throughput in flood algorithm-based NFV propagation. Specifically, the MRFO algorithm, which has demonstrated good results in energy systems, among others, has not been used systematically in an NFV context where redundant packets, as well as resource-congested situations, are critical areas.

This study fills this gap by employing MRFO in NFV packet propagation to increase throughput and decrease resource overhead. The significant contributions of this work are as follows:

1. Modelling NFV topologies via the BA scale-free network model to reflect realistic infrastructure.
2. Analysing the limitations of the flooding algorithm with TTL and its impact on throughput and resource consumption.
3. MRFO is applied as a swarm intelligence algorithm to optimize NFV routing and load balancing via a cost function that is based on CPU, memory, and bandwidth utilization.
4. MRFO is compared with the flooding algorithm across small, medium, and large NFV networks to evaluate scalability and adaptability.
5. Development of a Python-based simulation framework with a graphical interface enables real-time visualization and parameter control.

The value of this study is that MRFO can have a significant positive effect on the performance of NFV, which is scalable and dynamic, with a possible performance improvement of up to 53.6% in throughput, less packet delay, and more balanced resource utilization. The results of this study have potential practical applications in cloud, IoT, and 5G edge computing systems, where the ability to adjust and optimize NFV is paramount. This paper is organized into the following sections. Section two examines previous related studies. Section three details the methodology, standards and algorithms used to improve performance. Section four presents the simulation results and evaluates the standards. The paper concludes with a summary of the findings.

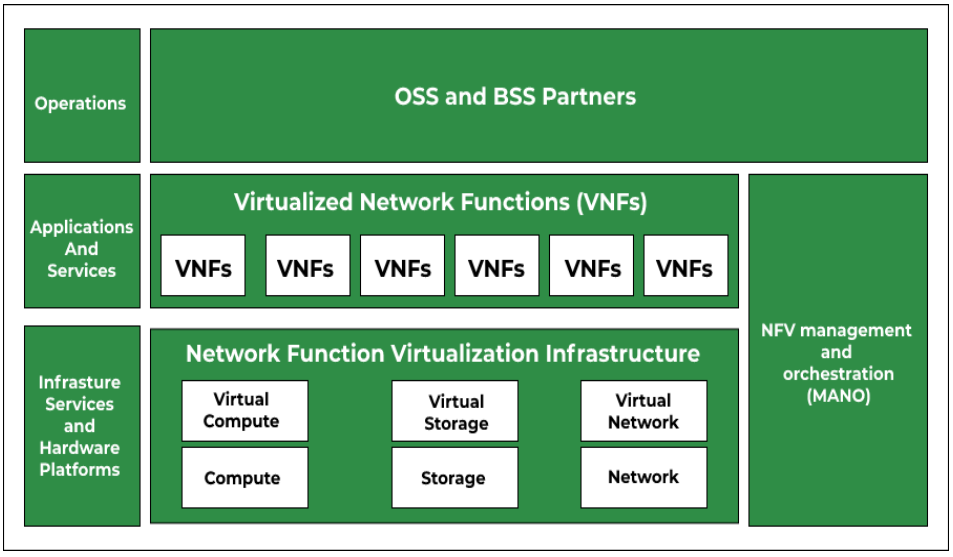

Fig. 1. NFV architecture based on reference no. [12]

2. RELATED WORKS

NFV optimization of system throughput is one of the research topics in modern networks because of the current demands on scalable, intelligent, and programmable network infrastructures, both in the cloud and in 5G and edge networks.

Early efforts to optimize NFV performance relied heavily on heuristic and greedy techniques to perform static NFV placement and reduce energy consumption. For example, Poobalan and Sangeetha [1] presented a hybrid optimization framework that integrates virtual machine scaling and optimal switching strategies to enhance the load distribution and minimize energy consumption within cloud data centers. Mohanty and Sahoo [13] investigated the application of metaheuristic algorithms for the capacitated controller placement problem in software-defined networks (SDNs), with an emphasis on failure resilience. Jun and Li [2] proposed the MG-CS algorithm, which combines microgenetic search with cuckoo search to improve the load distribution; however, the method suffers from high convergence time.

PSO was also investigated by Chen et al. [3], who demonstrated good results in resource-aware NFV placement, although it was prone to premature convergence in larger and more dynamic networks. Ajil and Kumar [4] applied ACO to optimize network load and routing paths, whereas Gupta and Singh [8] developed a WOA-based scheduler that improved throughput–latency trade-offs. Mishra and Gupta [7] proposed DRABC-LB, a dynamic (ABC) algorithm for cloud load balancing, which achieved gains in throughput but lacked mechanisms for real-time failure responsiveness.

In addition to metaheuristics, more intelligent methods using artificial intelligence (AI) and machine learning (ML) have emerged to predict workloads and make adaptive placement decisions. For example, Chen and Zhang [10] applied (RL) in conjunction with the marine predator algorithm to dynamically optimize task scheduling. Reddy et al. [11] established an evolutionary model that combines clustering with deep neural networks (DNNs) and Markov random fields (MRFs) to improve the accuracy of workload prediction and distribution. A study by Mohanty and Sahoo [12] discussed FL as a way to obtain privacy-sensitive distributed scheduling of NFV platforms; nevertheless, they reported that synchronization overhead and model drift were serious issues in dynamic workload settings.

Swarm intelligence (SI) was identified as a particularly suitable optimization approach for NFV because of its lightweight and decentralized decision-making. Poobalan et al. [16] introduced a hybrid VM-scaling and load distribution model based on swarm intelligence principles, offering improved system efficiency but requiring manual tuning for different deployment settings. Suriya and Sumithra [15] applied cooperative swarm-based spectrum sensing techniques to enhance adaptive handovers, improving NFV responsiveness under varying conditions. Bunkham et al. [22] reviewed the marine predator algorithm (MPA), noting its potential for adaptive routing and load minimization in discrete optimization problems.

A relatively new and underutilized algorithm is MRFO, which is inspired by the natural foraging behaviors of manta rays and includes encircling, chain foraging, and cyclone foraging strategies. Hassan et al. [23] demonstrated the effectiveness of MRFO in hybrid energy systems, where it achieved superior convergence and resource efficiency. Despite its success in other domains, MRFO has not been extensively explored in the context of NFV, particularly in scenarios involving flooding algorithm-based packet propagation.

In addition to optimization, resilience and fault-tolerant architectures are necessary to achieve service continuity in NFV, especially where VNFs crash or relocate when there are traffic surges. Adiel et al. [5] proposed a self-organized failure-detection mechanism that uses autonomous agents on sensor networks, which is a decentralized and self-healing mechanism that is well aligned with the NFV requirements. Similarly, Al-Karkhi and Fasli [6] designed an agent-based recovery model of virtual organizations, which uses randomized routing algorithms to alleviate node failures and to allocate tasks optimally dynamically. Moreover, decentralized trust and permission-management frameworks, such as the one discussed by Rana et al. [24] with its own blockchain-based certificate system deployed in Hyperledger Fabric, highlight the critical role of distributed technologies in designing safe and scalable NFV systems. Table 1 shows a comparison study that summarizes the related works.

TABLE I COMPARATIVE SUMMARY OF RELATED WORK

Technique	Authors	Year	Advantages	Limitations	Key Metrics
Hybrid Heuristic (VM scaling + switching)	Poobalan & Sangeetha [1]	2021	Improving load distribution, reducing energy	Static, lacks adaptability	Energy use, load
MG-CS (Microgenetic + Cuckoo Search)	Jun & Li [2]	2023	Better load distribution	High convergence time	Throughput
PSO	Chen et al. [3]	2022	Resource-aware placement	Premature convergence	CPU, memory
ACO	Ajil & Kumar [4]	2023	Optimizing routing paths	High computation cost	Throughput, delay
WOA	Gupta & Singh [8]	2024	Balancing latency and throughput	Weak in failure handling	Throughput, latency
DRABC-LB (ABC)	Mishra & Gupta [7]	2024	Gains in throughput	No real-time failure response	Throughput
RL + Marine Predators	Chen & Zhang [10]	2023	Adaptive, dynamic	High training overhead	Completion time
Evolutionary + DNN/MRF	Reddy et al. [11]	2023	Accurate workload prediction	Complex, resource-heavy	Prediction accuracy
Federated Learning	Mohanty & Sahoo [12]	2024	Privacy-preserving	Sync overhead, model drift	Scheduling accuracy
SI hybrid load model	Poobalan et al. [16]	2024	Efficiency gains	Needs manual tuning	Load efficiency
Swarm spectrum sensing	Suriya & Sumithra [15]	2024	Improving handovers	Limited to spectrum mgmt.	NFV responsiveness

Marine Predators (review)	Bunkham et al. [22]	2024	Adaptive routing potential	Still theoretical	Routing efficiency
MRFO	Hassan et al. [23]	2024	Strong convergence, resource efficient	Not applied to NFV flooding	Convergence, efficiency

Unlike prior studies, which focused on placement, scheduling or load balancing, this study stands out because it explores the coalescence of the MRFO method and flooding algorithm-based NFV request dissemination. To address throughput degradation, redundancy, and wasting of resources caused by the unaltered flooding algorithm, this research provides a lightweight and adaptive, resource-conscious framework for the optimization of NFV implementations in cloud, IoT, and 5G environments.

3. METHODOLOGY

This section describes the methodology and framework used to maximize the throughput of the system in NFV scenarios, as shown in Fig. 1. First, a scale-free Barabási–Albert model simulates the topology of an extensible service. Second, the propagation of packets is subject to specific (TTL) limits to restrict the lifetime of each request/message. Finally, routing switches between normal flooding and the MRFO algorithm, all on a Python-based simulator that simulates dynamic traffic and resource states. As a result, these optimizations can be quantified as performance improvements in NFV on various infrastructures.

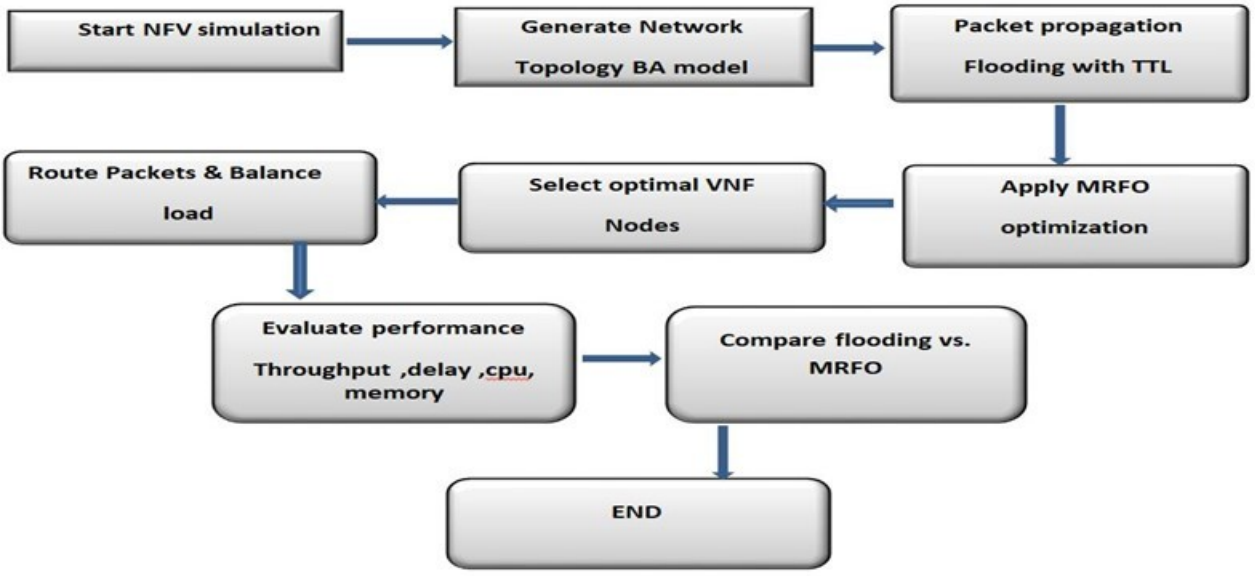

Fig. 2 Packet propagation via the flooding algorithm with time to live value (TTL)

3.1 Network Topology Formation Using the Barabási–Albert Model

The main ingredient in modelling NFV is the creation of a feasible topology-like network. The BA model is used in this paper to generate the virtual infrastructure since it can generate a scale-free network, which is quite similar to real internet and cloud computing networks. The BA model is based on the principle that preferentially attached nodes with a high degree have a better chance of attracting new edges.

Mathematically, the probability *P(i)* that a new node connects to an existing node *i* is given by Eq. (1):

$$P(i) = \frac{k_i}{\sum_{j \in v} kj} \qquad (1)$$

where *ki* is the degree of node *i* and *v* is the set of existing nodes. This mechanism leads to the emergence of hub nodes, which are essential for understanding the load distribution and bottleneck formation in NFV systems.

In the simulation, each node is instantiated as a router, switch, firewall, or load balancer, with heterogeneous configurations in terms of CPU cores, memory size, and bandwidth. These characteristics are assigned randomly within defined bounds to

reflect deployment variability in real-world VNFs. This paper is organized into the following sections. Section two examines previous related studies. Section three details the methodology, standards and algorithms used to improve performance. Section four presents the simulation results and evaluates the standards. The paper concludes with a summary of the findings.

2 depicts the suggested created system of NFV with workflow steps. Fig .3 Figure 2 depicts the logical roles of the network elements—routers, switches, firewalls, and load balancers. Fig. 4 shows the corresponding traffic flows generated under the baseline flooding algorithm scheme. Together, these illustrations clarify how VNFs are chained and where bottlenecks may emerge, motivating the MRFO-based optimization described in the next section.

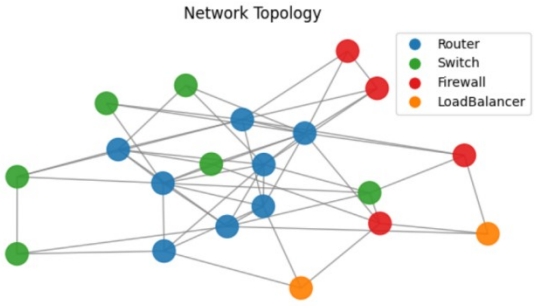

Fig. 3 Network topology of 20 NFVs

The environment, thereby, consists of different numbers of NFVs, such as routers, switches, and load balancers, and the client request depends on them to accomplish the tasks.

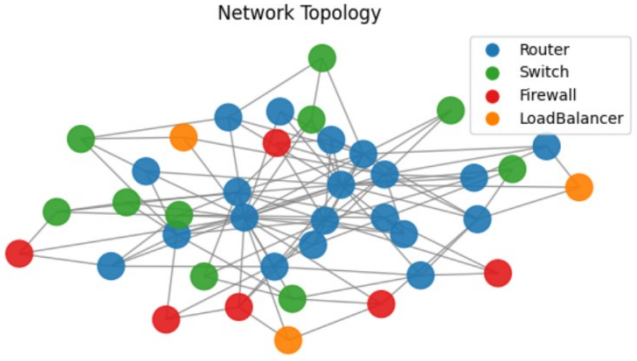

Fig. 4 Network topology of 39 NFVs

3.2 Packet Propagation via the flooding Algorithm

After the network is established, packet propagation is handled via a flooding algorithm, which ensures maximum reachability by having each node forward packet to all neighbors.

To prevent looping and minimize the sender's waiting time to receive a response, the TTL field is added to each packet and decremented at each hop. Packets are discarded when TTL reaches zero, and the sender is informed that the message has failed. Otherwise, the message may reach the destination and provide the sender with the requested information.

The flood algorithm is used for its reliability, especially in unknown or dynamic networks. However, it:

- Doing so does not avoid redundancy and may cause congestion to avoid each packet maintaining a history of visited nodes. If a node detects its ID in the packet's path history, it will drop the packet; this ensures that no cyclic paths are followed.
- Local dilemmas are avoided since forwarding is not decision-based.
- This does not guarantee global optimality, such as for the shortest path.

In this context, TTL serves as a control variable that modulates both network reachability and resource saturation. The following table summarizes the observed impact of varying TTL values on node performance metrics. Each client can

generate one or multiple service requests during the simulation, each possibly requesting a specific device (router, switch, firewall, or load balancer) with particular CPU/memory specifications. Client requests are formatted as follows:

(client_id, target_node_id, request_type, TTL), on the basis of the simulation results.

Table II shows how different TTL values affect average requests, CPU usage, and memory usage across NFV nodes.

The table demonstrates a trade-off: increasing the TTL allows the NFVs to handle more requests and improves the overall throughput of the system, but it also increases the demand for computational resources (CPU and memory). The optimal TTL value depends on the specific application's needs, balancing the desire for high throughput (more requests processed) with the available hardware resources.

TABLE II. IMPACT OF TTL ON THE RESOURCE CONSUMPTION OF NFVS

TTL	Avg. Requests per NFVs	Avg. CPU Cost (%)	Avg. Memory Cost (%)
2	8.5	15.2	11.6
3	21.4	37.8	32.5
5	39.6	69.1	61.4

3.3 Flooding Algorithm

Flooding is a fundamental data dissemination technique in computer networks. It involves transmitting a packet to all neighboring nodes, which then propagate the packet throughout the network. In this way, the information can be delivered to every node in the network without using complicated routing algorithms. The use of flooding is common in routing protocols, mainly in mobile ad hoc networks and sensor networks; this is because flooding protocols are simple and robust.

Within the scope of NFV, flooding, as represented in Algorithm (1), can be used to disseminate (SFC) requests or updates to all the devices in the virtualized infrastructure. An SFC specifies a set of VNFs, such as firewalls and load balancers, over which all network traffic must flow to meet service demands. Although flooding can guarantee that these SFC messages will be received by every possible processing node, it has the drawbacks of incurring a large amount of bandwidth overhead, and it also creates a high possibility of network congestion caused by unnecessary and multiple forwardings of messages. To overcome these problems, superior flooding algorithm mechanisms, which include the controlled flooding algorithm, the probabilistic flooding algorithm and the directional flooding algorithm, which attempt to minimize redundant messages sent to a node without jeopardizing delivery efficiency, are frequently used.

Algorithm 1 Flooding Algorithm

Input: Packet (client_id, node_id, request_type, TTL), Network Graph, Visited_Nodes

Output: Successful delivery to an eligible NFV node or packet discard

BEGIN

- Initialize queue Q ← [source_node]
- While Q is not empty do
    a. current_node ← Q.pop()
    b. If current_node ∉ Visited_Nodes then
        i. Add current_node to Visited_Nodes
        ii. Check if current_node matches the request type and has sufficient CPU/memory
            • If yes, process request and return SUCCESS
            • If no, proceed
        iii. Decrement TTL ← TTL - 1
        iv. If TTL > 0
            • Append all neighbors of current_node to Q (excluding previously visited)

End While

Return FAILURE (TTL expired or no suitable node found)

END

3.4 The Role of Swarm Intelligence in NFV Optimization

Traditional routing mechanisms in NFV environments, such as shortest-path or random walk strategies, are typically static and do not account for real-time resource availability [26]. In contrast, swarm intelligence algorithms are inherently dynamic and decentralized, making them highly suitable for complex, virtualized systems. These algorithms emulate the collective behavior of biological agents (e.g., ants, wolves, and manta rays), allowing adaptive decision-making in response to environmental changes.

The algorithm employed in the developed simulation framework is the MRFO algorithm. This algorithm is implemented in the routing layer and is called when forwarding packets to choose the best NFV node to process them, depending on current network conditions.

3.5 Manta ray foraging optimization (MRFO)

The MRFO algorithm mimics the foraging behavior of manta rays and includes three main search strategies: encircling prey, chain foraging, and cyclone foraging. Each iteration uses all the strategies, and each candidate solution in the MRFO population represents a potential NFV node to use in routing packets. The appropriateness of every node is analysed through a cost function that encompasses existing resource utilization. Fig. 5 shows the behavior of the manta ray in its search for food, as it determines its location and makes it a pivot point. Each individual swims back and forth around the pivot point and jumps to a new location, and then each individual updates its location to find the best location.

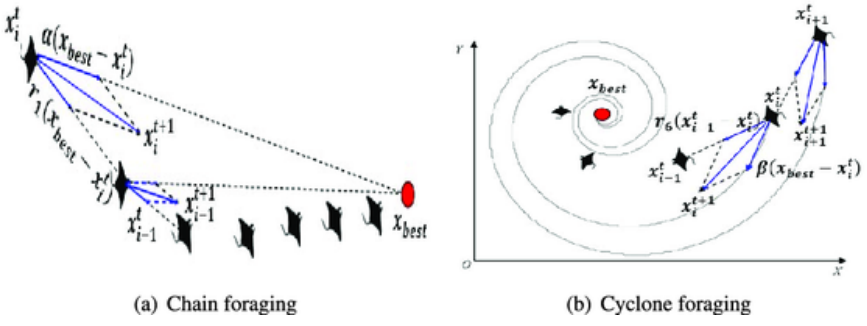

Fig. 5 Manta ray foraging optimization

First, the framework proposes the use of a flooding algorithm as a starting point of dissemination, according to which each node transmits received packets to all the neighbors until the TTL parameter reaches a preset limit. Even though such a method ensures reliability, it introduces considerable redundancy, congestion and resource overhead, thus impairing throughput.

To address these inefficiencies, the MRFO algorithm is integrated at the forwarding decision stage of the flooding algorithm. Instead of blindly forwarding packets to all neighboring nodes, MRFO evaluates the candidate neighbors on the basis of a multiobjective cost function that considers Eq. (2):

$$Cost(n) = w_1.CPU_n + w_2.Memory_n + w_3.Bandwidth_n \quad (2)$$

where:

- CPU_n is the current CPU utilization of node n.
- $Memory_n$ is the memory load of node n.
- $Bandwidth_n$ is the available link bandwidth for node n.
- w1, w2, w3 are weights reflecting the relative importance of each metric, set empirically (0.5, 0.3, 0.2, respectively, distribution [22]).

Table III defines the MRFO parameters, including adaptive factors, random coefficients, iteration limits, and population size. These parameters have been used to accomplish this work.

TABLE III. SEARCH STRATEGY PARAMETERS

Parameter	Description
A	Adaptive encircling factor: A=2(1−t/T)
β(t)	Cyclone damping factor
C1, C2, C3	Random coefficients for chain foraging
T	Total number of iterations
Population Size	Number of candidate nodes (default: 50)

MRFO then applies its three natural foraging behaviors to optimize the selection of the next hop:

- Chain Foraging: This ensures cooperative search among candidate nodes.
- Cyclone Foraging: This strategy diversifies the search to avoid local optima.
- Encircling Prey: balances exploitation by converging on the most resource-efficient node.

This mechanism replaces the broadcast step of the flooding algorithm with an adaptive node selection process, allowing packets to be forwarded only to the most resource-efficient and least congested neighbor. As a result, the number of redundant transmissions is reduced, throughput is improved, and CPU/memory resources are better balanced across the NFV infrastructure.

The integration is illustrated in Algorithm 2, where the flooding algorithm first propagates packets, but the forwarding decision is optimized via MRFO before routing continues.

Algorithm 2 MRFO

Input: List of NFV nodes, Packet size, Population size, Maximum iterations
Output: Best node index for routing

BEGIN
- **Initialize** a list of random candidate positions (nodes)
- **Evaluate** the cost of each candidate using CPU, memory, and bandwidth
- **Find the best position** with the lowest cost (best node thus far)
- **Repeat for max_iter times**:
    - For each candidate:
        - Update its position using a *chain foraging* formula
        - If the new cost is better, update the best position
- **Apply somersault foraging**:
    - For each candidate:
        - Adjust its position toward the best known position
        - Keep the position if cost improves

Return the index of the final best node
END

3.6 Client Requests and NFV Environment

In this work, the clients can be scalable in number and different types of requests. Clients can request a complete network environment, such as switches and routers, which load balcers with memory and CPU special characteristics. Other types of clients can ask for a single device, a router with a CPU, with specific criteria that may be higher than those of other types of NFV environments. The number of clients is an adjustable value. At the beginning of the simulation, we used 20 clients and then increased it to 60 to test scalability and study the simulation results.

Format of client requests: Message Message (client_id, node_id_in_NFV, request_type, TTL)

where

- client_id: Unique identifier for each client.
- node_id_in_NFV: Targeted node within the NFV infrastructure.

- request_type: Type of service or resource requested.
- TTL: Time-to-live, which determines the lifespan of the request packet in the network.

If the request is accepted by one of the NFV nodes, then it will take a specific amount of time to guarantee the delivery of the task.

If the received node is unable to process the request, then it sends it to the nearest neighbor until the search process finds a match with one of the NFV nodes or the TTL is equal to zero, which means that the task fails.

If a node matches the request type but is currently overloaded (high CPU/memory), the MRFO algorithm dynamically redirects the packet toward the next best NFV node via real-time network metrics.

The algorithm iteratively updates candidate positions (i.e., node selections) via these strategies, gradually converging toward an optimal solution that minimizes the cost function. This process ensures that the most resource-efficient node is selected for packet routing at each decision point.

4. SIMULATION RESULTS AND EVALUATION METRICS

The simulation operates in two phases. In phase 1, packets are routed via a nonoptimized method, allowing for reference measurements. In phase 2, the MRFO algorithm is employed to guide packet routing. The key performance metrics are logged as follows:

- Throughput (bps): Total data successfully transmitted per second as in Eq. (3).

$$Throughput = \frac{\sum_{i=1}^{N} D_i}{T} \quad (3)$$

where D is the total size of successfully transmitted packets/data (in bits or bytes), N is the total number of successfully received packets, and T is the total simulation or transmission time (in seconds).

- Packet Delay: Average time taken for packets to reach their destination.
- Resource Utilization: CPU, memory, and bandwidth usage across all nodes.
- Packet success rate: ratio of successfully processed packets to total packets.

Table IV provides average performance metrics (throughput, delay, CPU cost, and memory cost), demonstrating MRFO's efficiency over the flood algorithm.

TABLE IV. PERFORMANCE COMPARISON BETWEEN THE FLOODING ALGORITHM AND MRFO

Metric	Flooding algorithm	MRFO-Based Routing
Throughput (bps)	1443.20	2191.47
Average Delay (s)	2.6	1.7
CPU Cost (%)	58.2	42.3
Memory Cost (%)	49.1	34.5

The results presented herein clarify how MRFO dynamically reallocates packet flows, thereby achieving higher CPU- and memory-utilization efficiency while sustaining overall throughput across heterogeneous NFV infrastructures. In both experiments, the network size and TTL values are equal to (). Additionally, the data in this table strongly suggest that the MRFO-based algorithm is superior to the traditional flooding algorithm. It not only improves network performance (higher throughput, lower delay) but also has a lower demand for system resources (CPU and memory). High resource consumption and lower performance of the flooding algorithm

rithm are expected due to the nature of the algorithm, which continuously sends packets everywhere, leading to considerable wasted time and congestion. The MRFO-based approach, by contrast, is more intelligent and targeted.

4.1 Visualization and Interface Integration

A graphical user interface (GUI) was created in Python to support the interactive simulation setup and real-time visualization. It can adjust the following parameters: TTL, the number of nodes, and the swarm algorithm.

Accordingly, the system combines realistic network modelling, controlled flooding-based propagation and bioinspired optimization to improve NFV system performance. The BA model is topologically realistic, the flooding algorithm with TTL can be used in the analysis of reachability, and routing can be performed via MRFO in an adaptive resource-aware decision process. The empirical outcomes confirm the efficiency of this strategy, which shows a tremendous increase in throughput, latency and load balancing between the virtualized nodes.

Simulations were conducted to compare the flood algorithm with the MRFO algorithm under different TTL settings in the NFV environment. Two configurations were evaluated: (i) the default shortest-path routing baseline and (ii) MRFO-based swarm optimization. System performance was assessed in terms of throughput and packet-processing time to quantify the impact of the swarm algorithm on overall NFV throughput. The simulation environment consisted of multiple network nodes with different types of network functions, including firewalls and load balancers. Each node has varying levels of CPU cores and memory resources.

The experimental evaluation was performed in two extensive simulation runs wherein the performance of default routing was compared with that of the MRFO swarm optimization algorithm in the NFV scenario. Both runs involved both routing methods to compare them from a direct perspective.

Common parameters for both runs:

Table V lists simulation setup values such as the population size and maximum number of iterations. In detail, the optimization algorithm used in this study was configured to run with a population of 50 individuals and would stop after 100 iterations. These values are crucial for researchers to replicate experiments and obtain results.

TABLE V. ALGORITHM PARAMETERS

Parameter	Value
Population Size	50
Max Iterations	100

Table VI lists the network setup values, such as the bandwidth, number of edges per node, TTL, and packet interval.

These parameters define a moderately connected network environment with varying bandwidths. The TTL of 5 hops is for experimental purposes and can be changed to higher or lower values, showing that the simulation is focused on a small-scale to large-scale or highly efficient routing environment. The packet interval of 1.5 seconds indicates a low-to-moderate traffic load, which is useful for testing routing protocols without immediately overwhelming the network.

TABLE VI. NETWORK CHARACTERISTICS

Parameter	Value
Base Bandwidth	100 Mbps
Max Bandwidth	1000 Mbps
Edges per Node	3
TTL	5 hops
Packet Interval	1.5 seconds

Run #1: Small Network Configuration

In the network infrastructure configuration, these parameters can be adjusted according to the available environment request.

Table VII describes the composition of the small network with the number of (routers, switches, firewalls, load balancers, clients, and simulation time). The values are set at the beginning of the simulation and can be set to any value selected by the user through the GUI of the suggested system.

TABLE VII. **NETWORK INFRASTRUCTURE (RUN #1)**

Component	Count
Routers	8
Switches	6
Firewalls	4
Load Balancers	2
Clients	20
Simulation Time	30 seconds

Table VII compares the flood algorithm and MRFO in terms of throughput, packet success, CPU, memory, and bandwidth utilization. The flood algorithm is simple and has low resource utilization at the individual node level but provides poor performance (low throughput). Its low resource usage is a side effect of its inefficiency. The MRFO-optimized algorithm provides dramatically superior performance (53.6% higher throughput) when the complex routing logic is centralized on dedicated load balancers. This approach requires a much higher resource commitment (CPU, memory, and bandwidth) on those specific nodes, but a more efficient and high-performing network is achieved.

TABLE VIII. **RESULTS OF RUN #1**

Metric	Flooding Algorithm	MRFO-Optimized Routing
Throughput	937.07 bps	1439.47 bps (**53.6% improvement**)
Total Packets	49	49
Successful Packets	49 (100% success rate)	49 (100% success rate)
CPU Utilization	Highest: 25.03% (Node 9)	94.22–95.90% (Load Balancers: Nodes 18–19)
Memory Utilization	Generally, below 1.5%	3.18–10.46% (Load Balancers: Nodes 18–19)
Bandwidth Utilization	Below 0.01%	0.46–0.93% (Load Balancers: Nodes 18–19)

Figure 6 shows an image captured during a simulation. The value of the links is the cost between each pair of nodes. It shows the traffic flows through the network, highlighting the performance and resource consumption of different paths and nodes. The figure correlates with the data presented in the tables above.

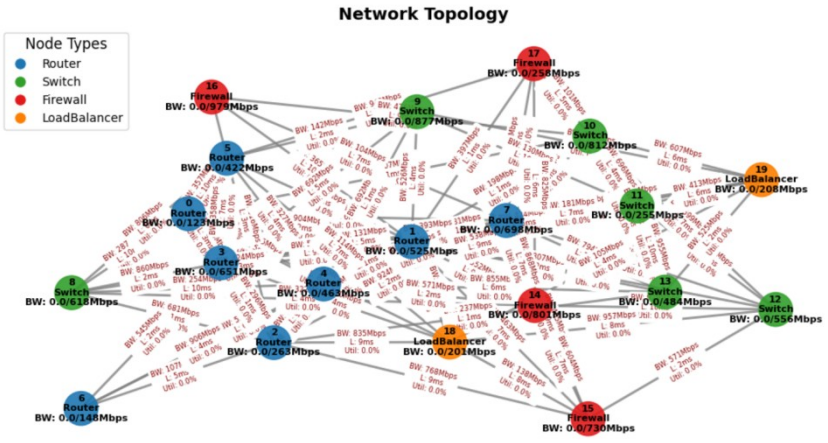

Fig. 6. Network topology of 20 NFVs

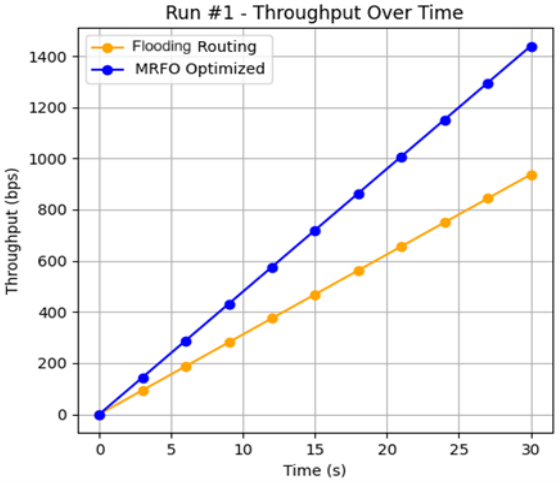

Fig. 7. Throughput Over Time of Run #1

In Figure 7, the steeper slope and higher final value of the blue line prove that the MRFO algorithm is more effective at processing and delivering data over time. This improved performance comes from its ability to find better paths, avoiding the inefficiencies and congestion caused by the flooding algorithm's random broadcasting of packets.

Run #2: Larger network configuration

Table IX defines the large-scale network setup for the second simulation run.

TABLE IX. **NETWORK INFRASTRUCTURE (RUN #2)**

Component	Count
Routers	20
Switches	10
Firewalls	6
Load Balancers	3
Clients	60
Simulation Time	100 seconds

Table X defines the large-scale network setup for the second simulation run. The results clearly demonstrate the shift from an inefficient, distributed approach (Flooding) to a more intelligent, centralized, and high-performance approach (MRFO-Optimized) in a modern network. MRFO achieves significantly higher throughput by centralizing complex routing work onto a few dedicated, high-resource nodes. The flood algorithm, while successfully delivering packets, is shown to be inefficient in utilizing resources, leading multiple nodes to 100% CPU utilization without providing a performance benefit.

TABLE X. **RESULTS OF RUN #2**

Metric	Flooding Algorithm	MRFO-Optimized Routing
Throughput	1986.4 bps	2677.2 bps (**34.8% improvement**)
Total Packets	284	284
Successful Packets	284 (100% success rate)	284 (100% success rate)
CPU Utilization	100% (Nodes 6, 17, 28, 36, 38)	99.47–100% (Load Balancers: Nodes 36–38)
Memory Utilization	Up to 3.44%	13.39–14.50% (Load Balancers: Nodes 36–38)
Bandwidth Utilization	Up to 0.52%	2.23–3.16% (Load Balancers: Nodes 36–38)

Figure 8 shows an image captured during a simulation. The values of the links are the costs between each pair of nodes. It shows the traffic flows through the network, highlighting the performance and resource consumption of different paths and nodes. The figure correlates with the data presented in Table x in run 2.

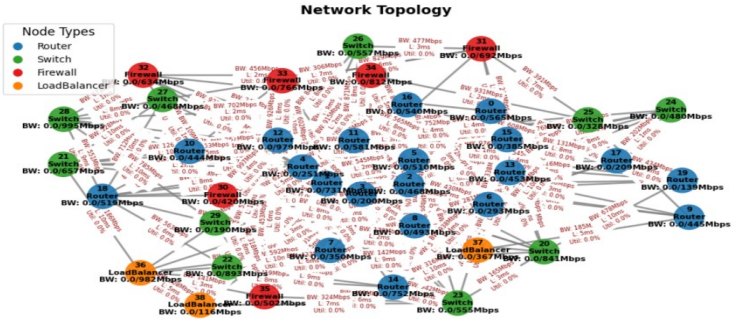

Fig. 8 Network topology of 39 NFVs

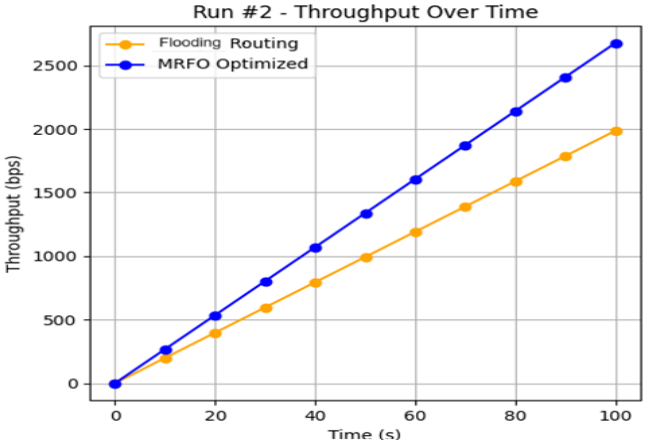

Fig. 9. Throughput Over Time of Run #2

Figure 9 visually reinforces the findings from the previous tables. The results indicate that MRFO-optimized routing consistently and significantly outperforms the flooding algorithm. The performance advantage of the MRFO algorithm is not just a one-time gain; it is a persistent benefit that grows over the course of the simulation.

Run #3: Basic network configuration

Table XI details the small 5-node network used in the third simulation. To test both of the applied algorithms.

TABLE XI. NETWORK INFRASTRUCTURE (RUN #3)

Component	Count
Routers	2
Switches	1
Firewalls	1
Load Balancers	1
Clients	20
Simulation Time	100 seconds

Table XII provides results showing that MRFO underperformed compared with flooding in small networks. The throughput of the flooding algorithm is higher than that of the MRFO optimization method. However, small networks are not the ones most commonly used once.

Despite the lower throughput number, it could still be considered superior depending on the application. For real-time applications where every packet is critical (e.g., video conferencing, financial transactions), a 100% success rate is far more valuable than a slightly higher data rate with packet loss. Notably, Figure 10 shows the network with an NFV equal to 5.

TABLE XII. RESULTS OF RUN #3

Metric	Flooding Algorithm	MRFO-Optimized Routing
Throughput	539.4 bps	459.20 bps
Total Packets	184	135
Successful Packets	180	135
CPU Utilization	100%	98.47–100%
Memory Utilization	Up to 1.11%	8.02–9.63%
Bandwidth Utilization	Up to 0.31%	1.87–1.95%

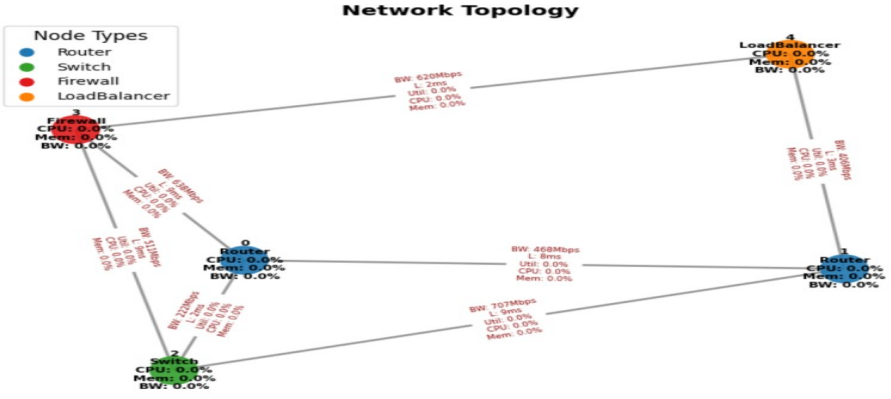

Fig. 10 Network Topology of 5 NFVs

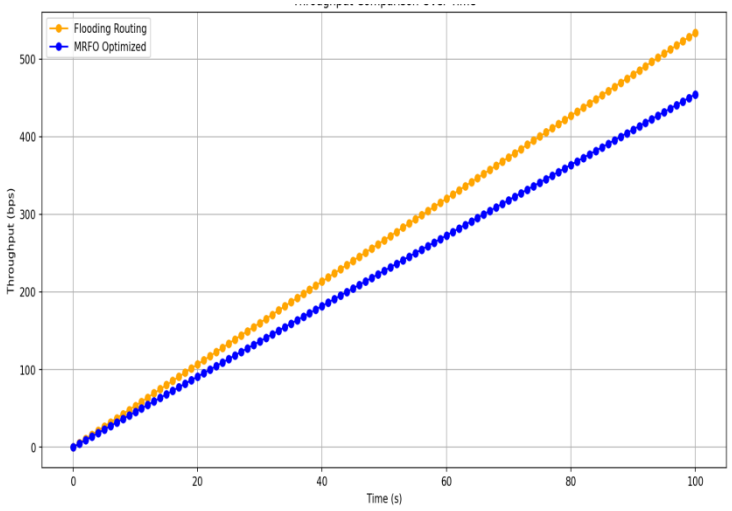

Fig. 11. Throughput Over Time of Run #3

Table XIII consolidates throughput results from all three runs, showing MRFO's improvements in Runs #1 and #2 but a decline in Run #3. In two out of three runs (Runs #1 and #2), the MRFO algorithm provides a substantial and consistent improvement in network throughput over the simple flood algorithm. In Run #3, lower throughput is associated with zero packet loss and a significantly lower overall packet transmission count, as shown in Figure 11.

TABLE XIII. **THROUGHPUT PERFORMANCE SUMMARY**

Metric	Run #1 (Flooding)	Run #1 (MRFO)	Run #2 (Flooding)	Run #2 (MRFO)	Run #3 (Flooding)	Run #3 (MRFO)
Throughput (bps)	937.07	1439.47	1986.4	2677.2	539.4	459.20
Improvement	-	+53.6%	-	+34.8%	+15.72%	

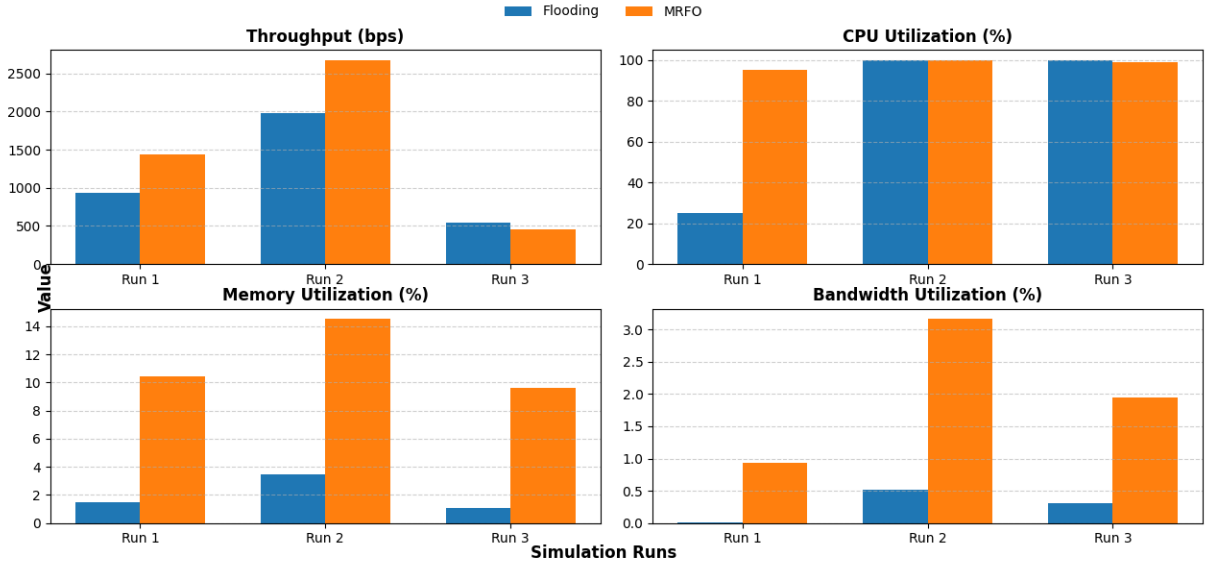

Fig. 12. Metrics Comparison

The study of simulations in three various network setups offers critical insights into the outcomes of the MRFO-optimized routing algorithm in comparison to those of the conventional flooding algorithm, as shown in Table XIII and Figure 12.

General NFV System Analysis

In Run #1, which represents a small-scale yet moderately complex NFV network (8 routers, 6 switches, 4 firewalls, 2 load balancers, and 20 clients), MRFO achieves a 53.6% improvement in throughput compared with the flood algorithm. This increase is due primarily to how MRFO handles packet generation and routing. In this scenario, each client generates packets at regular intervals, and MRFO dynamically computes optimal paths on the basis of real-time network metrics (e.g., available bandwidth, node CPU/memory usage, and hop count). The presence of multiple NFV components provides routing flexibility, allowing MRFO to intelligently distribute packets through underutilized paths. By avoiding congestion and minimizing redundant transmissions, MRFO ensures that a greater number of packets are delivered per second, resulting in a significant throughput gain. In contrast, the flood algorithm sends packets across all possible paths, leading to unnecessary traffic and resource waste.

In Run #2, which involves a larger and more complex NFV environment (20 routers, 10 switches, 6 firewalls, 3 load balancers, and 60 clients), MRFO maintained a 34.8% throughput improvement. Although the relative gain was lower than that in Run #1, the absolute throughput was much higher because of the increased scale and intensity of traffic. With 60 clients simultaneously generating packets, the total traffic load was significantly greater. The MRFO algorithm efficiently handles this increased demand by leveraging an expanded number of routing paths and NFV resources. The algorithm continuously adjusts its routing decisions to prevent congestion, using available CPU and bandwidth at the load balancers (nodes 36–38) nearly to full capacity (CPU ~100%, memory up to 14.5%). This intelligent traffic management ensures that MRFO maintains high delivery rates and throughput even in high-density environments, confirming its robustness and scalability.

In contrast, in Run #3, which simulated a small NFV topology with a limited number of nodes and connections, the performance of MRFO was poorer than that of the flooding algorithm. This can be explained by the inherent characteristics of MRFO. As a population-based met heuristic, MRFO requires a sufficiently large and diverse search space to explore candidate solutions effectively. In small networks, where the number of available forwarding nodes and paths is highly constrained, the MRFO search and optimization overheads become disproportionate relative to the problem size. Consequently, the algorithm introduces additional latency without providing meaningful optimization benefits, leading to reduced throughput and higher delays than those of the simple flooding algorithm.

This highlights a limitation of MRFO: it is less effective in small, static NFV environments, where the network size and routing options are minimal, and simpler techniques such as flooding algorithms or greedy heuristics are more efficient.

Therefore, MRFO is most beneficial in dynamic, scalable environments with numerous NFV elements and clients. In such contexts, the combination of active packet generation, network awareness, and adaptive routing enables MRFO to significantly improve throughput and resource utilization. However, in static or minimal networks, the optimization overhead outweighs the benefits, making simpler protocols such as the flood algorithm more efficient.

5. CONCLUSION

This study introduces an optimization framework that integrates MRFO with flooding algorithm-based request propagation in NFVs. The proposed method was evaluated against a conventional flooding algorithm under varying network sizes via throughput, delay, CPU, memory utilization, and the success rate as performance metrics.

The results showed that MRFO achieves substantial improvements in medium- and large-scale NFV environments, delivering 34–53% higher throughput, reduced delays, and more balanced resource utilization. These benefits stem from MRFO's adaptive swarm-based decision-making, which reduces redundant packet transmissions and ensures more efficient forwarding than does blind flooding algorithms.

However, in small-scale networks (Run #3), MRFO performed worse than did the flooding algorithm. This limitation arises because MRFO's optimization overhead outweighs its benefits when only a few forwarding paths are available. In such constrained environments, simple techniques such as the flooding algorithm remain more effective.

The findings confirm that MRFO offers three major benefits for NFV systems: scalability to large topologies, adaptability to dynamic workloads, and efficient resource utilization. These qualities make it highly suitable for 5G, IoT, and cloud NFV infrastructures, where demand fluctuations and large-scale deployments are common.

Future research could focus on enhancing MRFO through dynamic parameter tuning, designing an energy-aware cost function to support sustainable NFV, and validating the approach in a real 5G testbed to confirm its practical applicability.

Conflicts of Interest
The authors declare no conflicts of interest.

Funding
None

Acknowledgment
We would like to thank all the reviewers for the helpful comments and suggestions they gave us while we were planning and making this research. Their generosity with their time is very much appreciated.

References
[1] P. Poobalan and S. Sangeetha, "Optimal switching and load distribution strategy for energy minimization and performance optimization of cloud data center," *J. Eng. Res.*, vol. 2021, pp. 1–10, 2021, doi: 10.36909/jer.13245.
[2] Z. Jun and Y. Li, "MG-CS: Micro-Genetic and Cuckoo Search Algorithms for Load-Balancing and Power Minimization in Cloud Computing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 11, pp. 1–15, 2023, doi: 10.14569/IJACSA.2023.01411117.
[3] H. Chen, J. Rodrigues, and F. Xia, "Guest editorial: special section on distributed intelligence over Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1–3, 2022, doi: 10.1109/TII.2022.3162306.
[4] Ajil and E. S. Kumar, "A comprehensive study of LB technique in cloud infrastructure," *SN Comput. Sci.*, vol. 4, no. 10, pp. 1–12, 2023, doi: 10.1007/s42979-022-01588-x.
[5] N. G. Adiel, A. A. Salman, and N. F. Hassan, *Node failure in self-organized sensor networks*. Berlin, Germany: De Gruyter, 2025. [Online]. Available: https://nbn-resolving.de/urn:nbn:de:101:1-2503220539079.074354154737. doi: 10.1515/eng-2024-0083.
[6] A. A. S. Al-Karkhi and M. Fasli, "Virtual Organizations for Resource Allocation under Random Failure in a Network of Agents," in *Proc. 2nd Sci. Conf. Comput. Sci. (SCCS)*, Baghdad, Iraq, 2019, pp. 6–11, doi: 10.1109/SCCS.2019.8852598.
[7] R. Mishra and M. Gupta, "DRABC-LB: A novel resource-aware load balancing algorithm based on dynamic artificial bee colony for dynamic resource allocation in cloud," *SN Comput. Sci.*, vol. 5, no. 35, pp. 1–14, 2024, doi: 10.1007/s42979-023-02570-x.
[8] S. Gupta and R. S. Singh, "User-defined weight-based multiobjective task scheduling in cloud using whale optimization algorithm," *Simul. Model. Pract. Theory*, vol. 132, pp. 1–18, 2024, doi: 10.1016/j.simpat.2024.102915.
[9] V. Sharma, R. Beniwal, and V. Kumar, "Multilevel trust-based secure and optimal IoT-WSN routing for environmental monitoring applications," *J. Supercomput.*, vol. 80, pp. 1–20, 2024, doi: 10.1007/s11227-023-05875-z.
[10] D. Chen and Y. Zhang, "Diversity-aware marine predators algorithm for task scheduling in cloud computing," *Entropy*, vol. 25, no. 2, p. 285, 2023, doi: 10.3390/e25020285.
[11] H. Reddy, A. Lathigara, and R. Aluvalu, "Clustering-based EO with MRF technique for effective load balancing in cloud computing," *Int. J. Pervasive Comput. Commun.*, vol. 19, no. 2, pp. 1–18, 2023, doi: 10.1108/IJPCC-01-2023-0022.
[12] GeeksforGeeks, "Network functions virtualization." [Online]. Available: https://www.geeksforgeeks.org/network-functions-virtualization.
[13] S. Mohanty and B. Sahoo, "Metaheuristic algorithms for capacitated controller placement in software-defined networks considering failure resilience," *Concurrency Comput. Pract. Exp.*, vol. 36, no. 3, pp. 1–15, 2024, doi: 10.1002/cpe.8254.
[14] P. Poobalan, S. Sangeetha, and P. Shanthakumar, "Performance optimization and energy minimization of cloud data center using optimal switching and load distribution model," *Sustain. Comput. Inform. Syst.*, vol. 40, pp. 1–10, 2024, doi: 10.1016/j.suscom.2024.101013.
[15] S. Singh and R. Kumar, "Energy-efficient optimization with threshold-based workflow scheduling and virtual machine consolidation in cloud environment," *Wireless Pers. Commun.*, vol. 131, pp. 1–22, 2023, doi: 10.1007/s11277-022-10049-w.
[16] M. Suriya and M. G. Sumithra, "Efficient swarm intelligent optimization techniques using cooperative spectrum sensing for terrestrial handovers," *Wireless Netw.*, vol. 30, pp. 1–18, 2024, doi: 10.1007/s11276-024-03856-5.

[17] P. Poobalan, P. Shanthakumar, and M. R. Joel, "Hybrid optimization-enabled VM scaling based load distribution and optimal switching strategy in cloud data center," *Wireless Netw.*, vol. 30, pp. 1–12, 2024, doi: 10.1007/s11276-023-03532-0.

[18] K. Ramya and S. Ayothi, "Hybrid Prairie Dog and Beluga Whale optimization algorithm for multiobjective load balanced-task scheduling in cloud computing environments," *China Commun.*, vol. 21, no. 2, pp. 1–15, 2024, doi: 10.23919/JCC.2023.0097.
[19] R. Aron and A. Abraham, "Resource scheduling methods for cloud computing environment: The role of meta-heuristics and artificial intelligence," *Eng. Appl. Artif. Intell.*, vol. 115, pp. 1–20, 2022, doi: 10.1016/j.engappai.2022.105345.
[20] R. Rathinam, P. Sivakumar, and S. Sigamani, "SJFO: Sail Jelly Fish Optimization enabled VM migration with DRNN-based prediction for load balancing in cloud computing," *Neural Syst.*, vol. 34, no. 4, pp. 1–15, 2024, doi: 10.1080/0954898X.2024.2359609.
[21] M. Rahmani, J. Tanveer, and F. S. Gharehchopogh, "A novel offloading strategy for multiuser optimization in blockchain-enabled Mobile Edge Computing networks for improved IoT performance," *Comput. Electr. Eng.*, vol. 113, pp. 1–18, 2024, doi: 10.1016/j.compeleceng.2024.109514.
[22] S. Tripathi and S. Tripathi, "An efficient workflow scheduling using genetically modified golden jackal optimization with recurrent autoencoder in cloud computing," *Int. J. Netw. Manage.*, vol. 35, no. 2, pp. 1–18, 2025, doi: 10.1002/nem.2318.
[23] N. Bunkham, C. Pongpattanasiri, et al., "Marine Predators Algorithm for discrete optimization problems: A review of the state-of-the-art," *Naresuan Univ. Eng. J.*, vol. 19, no. 3, pp. 1–12, 2024.
[24] M. H. Hassan, S. Kamel, A. Alateeq, and A. Alassaf, "Optimal power flow in hybrid Wind-PV-V2G systems with dynamic load demand using a hybrid MRFO-AHA algorithm," *IEEE Trans. Sustain. Energy*, vol. 15, no. 4, pp. 1–12, 2024, doi: 10.1109/ACCESS.2024.3496123.
[25] L. A. Rana, F. Ghani, A. A. Salman, and A. B. Khudhair, "Blockchain-based student certificate management and system sharing using Hyperledger Fabric platform," *Period. Eng. Nat. Sci.*, vol. 10, no. 2, 2023, doi: 10.21533/pen.v10i2.2839.
[26] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2016, doi: 10.1109/COMST.2015.2477041.
[27] T. O. Alani and A. M. Al-Sadi, "Survey of optimizing dynamic virtual local area network algorithm for software-defined wide area network," *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 21, no. 1, pp. 77–87, Feb. 2023, doi: 10.12928/TELKOMNIKA.v21i1.24249.
[28] A. S. Salim and A. M. Al-Sadi, "A survey and classification of software defined network based campus networks," *AIP Conf. Proc.*, vol. 3169, no. 1, art. no. 030037, 2025, doi: 10.1063/5.0255522.
[29] M. Khamees, A. Al-Saadi, and R. J. Al-Bahadili, "An investigation of using traffic load in SDN based load balancing," *Iraqi J. Comput. Commun. Control Syst. Eng.*, vol. 20, no. 3, p. 6, Jul. 2020, doi: 10.33103/uot.ijccce.20.3.6.