Optimization of Service Function Chain Placement in Cloud-Fog-Edge Networks

Chandrapal Singh Dangi, Sanjay Sharma
Department of Mathematics, Bioinformatics and Computer Applications,
Maulana Azad National Institute of Technology, Bhopal, India

***Abstract*—There is an explosion in IoT devices, 5G technology, and MECs, that results in increasing demands on effective and scalable network services management. Service function chaining, defined as the sequence of functions in VNFs on a path, is one of the core principles behind the NFV architecture design. SFC allocation to the heterogeneous clouds–fogs–edges network is an NP-hard problem characterized by mutually conflicting goals, such as latency minimization, energy and cost reduction, and resource maximization. In this study, an in-depth comparison study is carried out on three population-based optimization algorithms for solving the placement problem of SFCs using Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO) on three cases: (1) VNF deployment cost/QoE optimization in a 5G hybrid cloud with 12 nodes and weighting factor γ=0.4; (2) SFC graph matching on MEC-NFV networks with a 100-node physical network, 20 VNFs, and equal utilization weights α=β=γ=1/3; and (3) multi-instance SFC mapping on Fog-to-Cloud (F2C) IoT environment with a 5-VNF chain across 5 nodes. These three algorithms have been evaluated under identical conditions: 10 independent runs, 200 iterations, 20–30 agents. Results demonstrate that GWO achieves the best VNF deployment objective (W=73.92, a 15.1% improvement over the BGWO baseline), PSO achieves the highest resource utilization (52.9%) in MEC-NFV placement, and both PSO and GWO reduce F2C end-to-end latency by 25% compared to the ILP reference (12 vs. 16 units at three instances), while all three algorithms reduce latency by approximately 80% relative to cloud-only deployment. PSO emerges as the most consistently high-performing algorithm across all three scenarios.**

***Keywords*—Service function chaining; Virtual Network Functions; Network Function Virtualization; Particle Swarm Optimization; Ant Colony Optimization; Grey Wolf Optimization; mobile edge computing; Fog-to-Cloud; 5g networks; resource allocation**

I. Introduction

The convergence of cloud computing, edge intelligence, and programmable networking has fundamentally transformed how network services are designed, deployed, and managed. Modern communication infrastructures increasingly rely on Network Function Virtualization (NFV) to decouple network functions traditionally implemented in dedicated hardware from their physical substrates. This paradigm enables deploying network services as software modules, known as Virtual Network Functions (VNFs), hosted on commodity servers in cloud data centers, edge nodes, or fog computing layers [1], [2].

A *Service Function Chain* (SFC) is an ordered sequence of VNFs through which network traffic must traverse to receive a composite service. A typical SFC for online gaming, for instance, consists of a Network Address Translator (NAT), Firewall (FW), Video Optimization Controller (VOC), WAN Optimization Controller (WOC), and Intrusion Detection and Prevention System (IDPS), each applied sequentially to user traffic [3]. The problem of SFC allocation, which involves placing each VNF on its corresponding physical node and routing data to the respective node accordingly, forms the core issue addressed in this [4] research study.

The development of telecommunication technology, for instance, the fifth generation (5G) of mobile telecommunication networks, and new areas of research and application, such as the Internet of Things (IoT), alter the resource access paradigm of the CSPs. In order to cut down the access time and enhance the QoS, resources are placed near the cloud users, and therefore, the paradigm of mobile edge computing (MEC) has been introduced. The cloud users access the network resources via their Mobile Devices using the base station and obtain their service.

There are three main reasons why this problem is difficult. The first is that there are two competing goals, namely reducing latency which favors edge/fog computing and saving costs which favors the central cloud. The second reason is due to heterogeneity in the resources available at the nodes leading to complicated interactions of constraints. Lastly, the problem is NP-Hard [5].

This study seeks to examine whether PSO, ACO, and GWO algorithms can be used to allocate SFCs in three selected environments; 5G cloud environment [6], MEC-NFV [7], and F2C IoT. The 5G Cloud SFC focuses on high throughput and centralized management. The MEC-NFV SFC focuses on ultra-low latency and edge computing capabilities.The F2C IoT SFC focuses on distributed service placement, energy efficiency, and heterogeneity of IoT resources. Thus, SFC management is getting increasingly more decentralized and complicated when transitioning from 5G Cloud to MEC-NFV and further to F2C IoT.

Principal contributions:

- We propose a metaheuristic algorithm-based approach to solve 3 SFC architectures 5G Cloud Environment,MEC-NFV Environment and F2C (Fog-to-Cloud) IoT Environment.
- We have simulated all 3 SFC scenario in python programming version 3.11 and the simulation results demonstrated that GWO outperforms BGWO by +15.1%; PSO outperforms with 52.9% MEC-NFV efficiency; PSO/GWO cut down F2C delay by 25% relative to ILP.
- Consistent execution and systematic comparison of PSO, ACO, and GWO algorithms in three different

SFC configurations. Thorough testing under completely controlled conditions for valid statistical comparisons.

- Concrete algorithm selection recommendation for cloud-fog/edge systems.

II. Related Work

SFC placement and VNF deployment have attracted substantial research across multiple network paradigms. In MEC contexts, strategies range from QoS-aware placement algorithms targeting service availability and access latency [8] to dynamic schedulers that minimize VNF migrations as user mobility patterns evolve. Integer Linear Programming (ILP) and Mixed ILP formulations provide optimal solutions for small-scale SFC placement instances, but become computationally intractable as network size grows [9]. Column-generation approaches extend ILP tractability by decomposing the problem into configuration enumeration and selection phases, though these remain limited by quadratic constraint structures [10].

Metaheuristic algorithms have emerged as a dominant paradigm for large-scale SFC optimization. PSO has been applied to mobility-aware VNF placement enabling dynamic reallocation as users move between base stations. ACO has been used to balance communication delay against VNF relocation count through pheromone-trail encoding. Genetic Algorithms have demonstrated effectiveness in multi-objective formulations approximating latency-cost Pareto fronts [11]. Grey Wolf Optimization (GWO), inspired by the social hierarchy and hunting behavior of grey wolves, has been applied to VNF deployment in hybrid cloud infrastructures. The Binary GWO (BGWO) variant addresses the discrete nature of VNF assignment through sigmoid-based position discretization [12]. Comparative studies suggest GWO achieves faster convergence than genetic algorithms on similar combinatorial problems due to its adaptive exploration–exploitation balance.

In Fog-to-Cloud (F2C) computing, SFC mapping presents additional complexity from heterogeneous fog-to-cloud latency characteristics. Multi-instance SFC deployment has been shown to substantially reduce end-to-end latency by enabling traffic assignment to the nearest available instance [13]. Graph-matching formulations have been proposed for MEC-NFV placement, where similarity between the VNF forwarding graph and the physical network graph is maximized using LP followed by Hungarian algorithm-based mapping, reducing execution time and improving node utilization [14].

Recent advances in Service Function Chain (SFC) placement and Virtual Network Function (VNF) deployment have increasingly leveraged Deep Reinforcement Learning (DRL) and hybrid optimization techniques to address the challenges of dynamic and large-scale network environments. Zhou et al. [15] proposed a DRL-based multi-objective SFC placement framework that jointly optimizes resource utilization, latency, and service acceptance rates in NFV-enabled networks. Their approach demonstrated the effectiveness of reinforcement learning in adapting placement decisions to changing network conditions while improving overall service performance.

Feng et al. [16] introduced a hybrid Deep Reinforcement Learning-Greedy (DRL-G) algorithm for SFC deployment in space–air–ground integrated networks. The proposed framework combines the adaptive learning capability of DRL with the computational efficiency of greedy search to optimize deployment cost, energy consumption, and service latency. Experimental results showed improved scalability and convergence compared to conventional optimization methods.

In Li et al. [17], the authors suggested an optimized approach to cache and dynamically migrate services in the MEC environment by implementing a Deep Q-Network (DQN). They pointed out that the users' movement and insufficient MEC coverage result in disruption of services, higher latency, and increased energy costs. For solving these problems, they combined edge caching with intelligent migration of services.

To address distributed orchestration challenges in edge environments, Li et al. [18] developed a multi-agent reinforcement learning (MARL) framework for SFC placement in edge computing networks. By enabling multiple agents to collaboratively perform VNF deployment and path selection decisions, the proposed approach achieved higher resource utilization and service provider profitability while supporting decentralized network management.

In general, the literature presents a clear evolution in terms of moving from cloud placement to dynamic cloud-fog-edge placement of services that are QoS aware, energy efficient, and mobility aware. The surveys present very good taxonomies, but recent research on algorithms presents heuristic, approximations, game theory, and meta-heuristics approaches. Nevertheless, there are some issues that are not well covered, such as the combination of VNF placement with routing, dynamic multi-service function chain requests, user mobility, failure tolerance, scaling, and unified optimization of latency, energy, resource efficiency, throughput, and reliability.

Despite the breadth of existing work, systematic multi-scenario comparative studies evaluating the same set of metaheuristic algorithms across diverse environments under rigorously identical conditions remain scarce. This study addresses that gap.

III. System Model

A. Network Architecture

The study considers three representative deployment environments unified by the Fog-to-Cloud continuum: IoT devices connect to nearby Fog/Edge nodes, which link to regional edge clouds and a central cloud data center. SFC placement determines which layer hosts each VNF, directly controlling service latency, energy consumption, and resource utilization. Fig. 1 illustrates the SFC architecture and VNF chain structure.

B. Scenario I: 5G Hybrid Cloud

The first scenario models a 5G network with a hybrid cloud infrastructure comprising one central cloud server and eleven edge cloud servers (12 nodes total). Each edge cloud serves a set of evolved Node-B (eNB) base stations, with each eNB aggregating VNF requests from hundreds to thousands of mobile users. Inter-cloud latencies are 10–200 ms; eNB-to-edge latencies are 2–5 ms. Each edge cloud offers VMs with 500–1000 MIPS and 512–1024 MB RAM. VNF holding capacity ranges from 100–300 instances per node (600 for the central cloud). The weight factor γ=0.4 governs the trade-off between deployment cost and service delay.

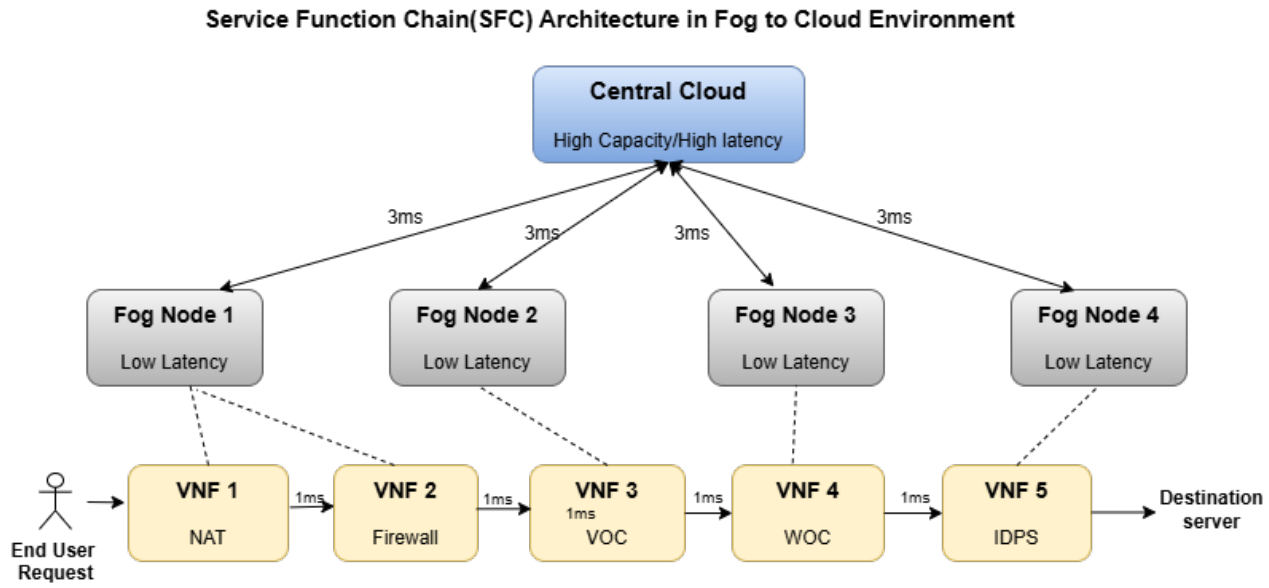

Fig. 1. SFC Architecture. IoT user traffic traverses an ordered chain of five VNFs (NAT, FW, VOC, WOC, IDPS) hosted across fog and cloud nodes. VNF-to-node assignment determines end-to-end latency.

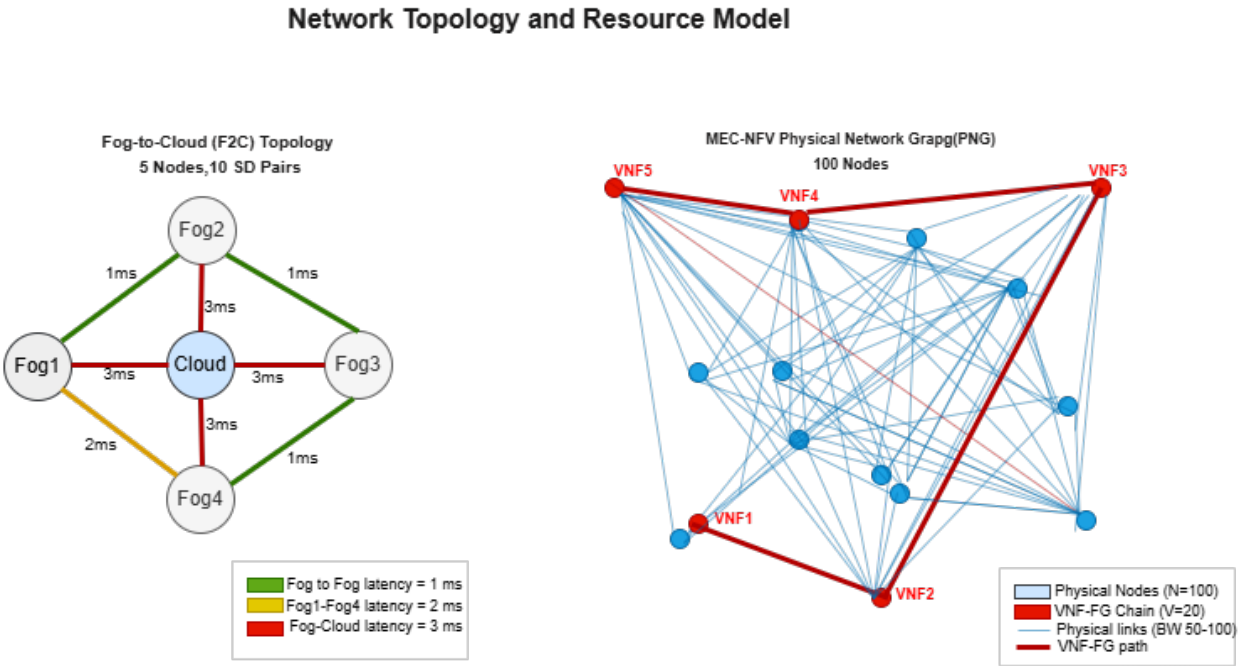

Fig. 2. Network topology models. *Left*: F2C topology (5 nodes). *Right*: Representative subgraph of the 100-node MEC-NFV physical network with the 20-VNF chain highlighted.

C. Scenario II: MEC-NFV Environment

The second scenario considers a three-layer MEC-NFV architecture. The physical network graph (PNG) has N=100 nodes with CPU and memory capacities uniformly in [50, 100] units, 50% connectivity (θ=0.5), and link bandwidths of 50–100 units. VNFs consume 1–20 units of CPU, memory, and bandwidth. The placement objective maximizes a weighted resource utilization with equal weights α=β=γ=1/3.

D. Scenario III: Fog-to-Cloud IoT

The third scenario implements the F2C architecture for IoT services. The topology consists of five nodes: four fog nodes in a mesh and one central cloud node (see Fig. 2). Link latencies follow: fog–fog = 1, fog1–fog4 = 2, fog–cloud = 3 (latency units). Ten source-destination pairs each carry 1 Gbps of traffic through a 5-function service chain (NAT→FW→VOC→WOC→IDPS). The system supports up to seven concurrent SFC instances.

E. Latency and Cost Models

The end-to-end service latency for VNF f assigned to cloud node k from eNB j is:

$$D(f,k,j) = R(f,k,j) + \Upsilon(f,k,j) + \phi(f,k,j) \tag{1}$$

where, $R(f,k,j)$ is the relocation (migration) delay, $\Upsilon(f,k,j)$ is the communication latency (eNB-to-cloud and inter-cloud), and $\phi(f,k,j) = I_j^f/e_k$ is the processing delay (instructions / MIPS).

The deployment cost $B(f,k,j)$ accounts for local vs. remote cloud pricing:

$$B(f,k,j) = \Gamma_{k,j}^f \cdot \vartheta_f \cdot \left[x_k^f \cdot l_k + (1 - x_k^f) \cdot r_k\right] \tag{2}$$

where, l_k and r_k are local and remote unit costs, and $x_k^f \in \{0,1\}$ indicates whether the VNF was previously hosted at node k.

IV. Problem Formulation

A. Scenario I: Multi-Objective VNF Deployment

The Min-CD problem minimizes a weighted combination of normalized deployment cost and normalized service delay:

$$\min\ W = \sum_{j\in M}\sum_{f\in\mathcal{F}_j}\sum_{k\in C}\Big\{\gamma\cdot B'(f,k,j)+(1-\gamma)\cdot D'(f,k,j)\Big\} \tag{3}$$

where, $B'(f,k,j) = B(f,k,j)/U_j^f$ and $D'(f,k,j) = D(f,k,j)/\Delta_j^f$ are normalized by the user budget U_j^f and delay deadline Δ_j^f, respectively.

Constraints:

$$\Gamma_{k,j}^f \in \{0,1\} \qquad \forall f,j,k \tag{4}$$

$$\textstyle\sum_{k\in C}\Gamma_{k,j}^f = 1 \qquad \forall f\in\mathcal{F}_j,\ j\in M \tag{5}$$

$$\textstyle\sum_{j,f}\Gamma_{k,j}^f \le \psi_k \qquad \forall k\in C \tag{6}$$

$$B(f,k,j) \le U_j^f \qquad \forall f,j \tag{7}$$

$$D(f,k,j) \le \Delta_j^f \qquad \forall f,j \tag{8}$$

Constraint (5) ensures atomicity (each VNF on exactly one cloud); Constraint (6) enforces VNF holding capacity; Constraint (7) and Constraint (8) enforce budget and QoE deadlines. This MOLP is NP-hard via reduction to the Multiple Knapsack Problem.

B. Scenario II: SFC Placement as WGMP

The physical network $\mathcal{P}$=(N,L) and VNF forwarding graph $\mathcal{F}$=(V,E) are represented as weighted graphs. The objective maximizes composite resource utilization:

$$\max\ U = \alpha\cdot U_{\text{CPU}} + \beta\cdot U_{\text{MEM}} + \gamma\cdot U_{\text{BW}} \tag{9}$$

subject to capacity constraints on CPU, memory, and bandwidth [see Constraint (10) to Constraint (12)]:

$$U_{\text{CPU}}(n) = \frac{\sum_v c_v\cdot x_{vn}}{C_n^{\text{cpu}}} \le 1 \quad \forall n\in N \tag{10}$$

$$U_{\text{MEM}}(n) = \frac{\sum_v m_v\cdot x_{vn}}{C_n^{\text{mem}}} \le 1 \quad \forall n\in N \tag{11}$$

$$U_{\text{BW}}(l) = \frac{\sum_e b_e\cdot y_{el}}{C_l^{\text{bw}}} \le 1 \quad \forall l\in L \tag{12}$$

C. Scenario III: Multi-Instance F2C SFC Mapping

The objective minimizes total end-to-end latency by assigning each source-destination pair to the nearest available SFC instance:

$$\min\ L_{\text{total}} = \sum_{(s,d)\in SD} D_{sd}\cdot\min_{i\in I_c}\left[\ell(s,v_1^i) + \sum_{k=1}^{n-1}\ell(v_k^i,v_{k+1}^i) + \ell(v_n^i,d)\right] \tag{13}$$

subject to: CPU capacity at each fog and cloud node; SLA delay deadlines per SD pair; maximum SFC instances I_c per chain.

V. Proposed Methodology

Three metaheuristic algorithms are employed. In each case, a candidate solution is represented as a real-valued vector $\mathbf{x}\in\mathbb{R}^d$ (d = number of VNFs), decoded to discrete node assignments by rounding: $\hat{x}_i = \text{clip}(\text{round}(x_i), lb, ub)$.

Fig. 3 illustrates the workflow for all three algorithms.

A. Particle Swarm Optimization (PSO)

PSO maintains N_p particles, each with position $\mathbf{x}\in\mathbb{R}^d$ and velocity $\mathbf{v}$. At each iteration t:

$$\mathbf{v}^{t+1} = w\,\mathbf{v}^t + c_1 r_1(\mathbf{p}_{\text{best}} - \mathbf{x}^t) + c_2 r_2(\mathbf{g}_{\text{best}} - \mathbf{x}^t) \tag{14}$$

$$\mathbf{x}^{t+1} = \text{clip}\big(\mathbf{x}^t + \mathbf{v}^{t+1},\ lb,\ ub\big) \tag{15}$$

where, w=0.7 is the inertia weight, c_1=c_2=1.5 are cognitive and social acceleration coefficients, and $r_1, r_2 \sim \mathcal{U}(0,1)$.

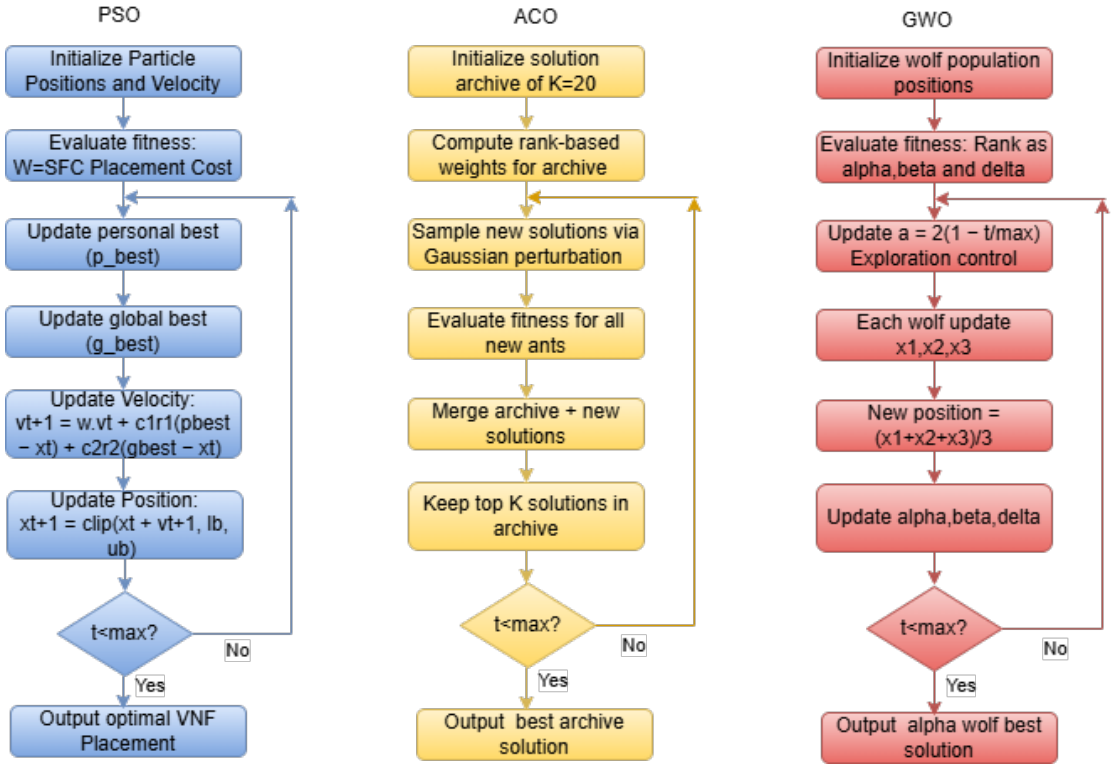

Fig. 3. Workflow diagram

B. Ant Colony Optimization (ACO-Continuous)

The continuous ACO variant ($ACO_{\mathbb{R}}$) maintains a solution archive of K=20 elite solutions. Each ant constructs a new solution by perturbing an archive-selected entry with a Gaussian:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{arch}}^{(l)} + \mathcal{N}(0,\, \sigma^2\, \mathbf{I}) \tag{16}$$

where, $\sigma = \xi(ub - lb)/K$ with ξ=0.85, and archive selection probability for rank l is:

$$w_l \propto \exp\left(-\frac{(l-1)^2}{2\,q^2K^2}\right), \quad q = 0.5 \tag{17}$$

C. Grey Wolf Optimization (GWO)

GWO models the social hierarchy of grey wolves: α (best solution), β (second best), δ (third best), and ω (remaining wolves). At each iteration, ω wolves update their positions toward the three leaders:

$$\mathbf{X}_1 = \mathbf{X}_\alpha - A_1 \cdot |\mathbf{C}_1\mathbf{X}_\alpha - \mathbf{X}_\omega| \tag{18}$$
$$\mathbf{X}_2 = \mathbf{X}_\beta - A_2 \cdot |\mathbf{C}_2\mathbf{X}_\beta - \mathbf{X}_\omega| \tag{19}$$
$$\mathbf{X}_3 = \mathbf{X}_\delta - A_3 \cdot |\mathbf{C}_3\mathbf{X}_\delta - \mathbf{X}_\omega| \tag{20}$$
$$\mathbf{X}_\omega^{t+1} = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3} \tag{21}$$

The adaptive parameter $a = 2(1-t/I_{\max})$ decreases linearly from 2 to 0, enabling transition from broad exploration ($|A| > 1$) to focused exploitation ($|A| < 1$).

D. Solution Encoding, Discretization, and Constraint Handling

A representation of each solution consists of a placement vector of size (m), where (m) stands for the number of VNFs in the service chain. The (i^{th}) value of the vector corresponds to the hosting node chosen for VNF (i). Given that PSO and GWO algorithms work in a continuous search space, they use a rounding and clipping procedure ($\hat{x}_i = \text{clip}(\text{round}(x_i), lb, ub)$) to obtain a discrete node ID. On the other hand, ACO algorithm directly creates discrete placement solutions based on the probability of selecting nodes according to their pheromone levels and heuristics. Then, each placement solution is checked for adherence to CPU, memory, storage, bandwidth, and latency constraints.

E. Complexity Analysis

Due to the complexity and importance of the presented methods, it is crucial to do the real-time analysis. All three algorithms improves convergence speed while decreasing time complexity.The computational complexity of PSO, ACO, and GWO as O(N × I × d), where N denotes population size, I denotes iterations, and d denotes problem dimension.

F. Energy Consumption

However, in the proposed Fog/Edge computing architecture, the energy consumption [19] in total will depend on the place where application requests are executed. The applications' requests that are processed only in the Fog/Edge layer will entail the energy consumption for data forwarding, computations, and storage operations. Nonetheless, in case when there

are not enough resources in the Fog/Edge layer and some tasks are being offloaded to the cloud layer, the additional migration cost should be considered. Therefore, the total energy consumption can be divided into three main components in case of Fog/Edge-only processing, and four components in case of cloud processing. The primary sources of energy consumption during data processing are as follows:

- **Data Forwarding Energy (E_{DF}):** Energy consumed in transmitting data between IoT devices, edge nodes, fog servers, and cloud data centers.
- **Computation Energy (E_{CP}):** Energy required for executing application tasks and processing workloads on computing nodes.
- **Storage Energy (E_{ST}):** Energy consumed for storing data and maintaining cached content within Fog/Edge storage resources.
- **Data Migration Energy (E_{MG}):** Additional energy incurred when data or services are migrated between Fog/Edge nodes and cloud servers.

1) Case 1: Fog/edge-only processing: When application requests are completely processed within the Fog/Edge layer, the total energy consumption is expressed as:

$$E_{Fog} = E_{DF} + E_{CP} + E_{ST} \tag{22}$$

2) Case 2: Cloud-assisted processing: When application requests are partially or fully offloaded to the cloud layer, the total energy consumption becomes:

$$E_{Cloud} = E_{DF} + E_{CP} + E_{ST} + E_{MG} \tag{23}$$

$$E_{DF} = \sum_{i=1}^{N} D_i \times e_t \tag{24}$$

where,

D_i: Amount of transmitted data, e_t: Energy consumed per unit data transmission.

$$E_{CP} = \sum_{i=1}^{N} C_i \times e_c \tag{25}$$

where,

C_i: Computational workload (CPU cycles), e_c: Energy consumed per CPU cycle.

$$E_{ST} = \sum_{i=1}^{N} S_i \times T_i \times e_s \tag{26}$$

where,

S_i: denotes the data size stored, T_i: represents the storage duration, and e_s: is the storage energy coefficient.

$$E_{MG} = \sum_{i=1}^{N} M_i \times e_m \tag{27}$$

where,

M_i: denotes the migrated data volume and e_m: represents the migration energy consumed per unit data.

G. Penalty Function for Constraint Handling

To ensure feasible VNF placement solutions, a penalty-function approach is employed to handle resource and QoS constraint violations. The overall fitness function is defined as:

$$F(x) = f(x) + \lambda P(x) \tag{28}$$

where, x represents a candidate placement solution, $f(x)$ denotes the original objective function, $\lambda > 0$ is the penalty coefficient, and $P(x)$ is the total penalty associated with constraint violations.

The total penalty is computed as the weighted sum of individual constraint violations:

$$P(x) = \sum_{k=1}^{m} w_k V_k(x) \tag{29}$$

where, m is the number of constraints, w_k is the weight associated with the k^{th} constraint, and $V_k(x)$ represents the violation degree of that constraint.

The violation terms are defined as follows:

$$V_{CPU}(x) = \max\{0, CPU_{req}(j) - CPU_{avail}(j)\} \tag{30}$$
$$V_{MEM}(x) = \max\{0, MEM_{req}(j) - MEM_{avail}(j)\} \tag{31}$$
$$V_{ST}(x) = \max\{0, STOR_{req}(j) - STOR_{avail}(j)\} \tag{32}$$
$$V_{BW}(x) = \max\{0, BW_{req}(i,j) - BW_{avail}(i,j)\} \tag{33}$$
$$V_{LAT}(x) = \max\{0, LAT_{achieved}(x) - LAT_{max}\} \tag{34}$$

where,

- $CPU_{req}(j)$ and $CPU_{avail}(j)$ denote the required and available CPU resources of node j.
- $MEM_{req}(j)$ and $MEM_{avail}(j)$ denote the required and available memory resources.
- $STOR_{req}(j)$ and $STOR_{avail}(j)$ denote the required and available storage resources.
- $BW_{req}(i,j)$ and $BW_{avail}(i,j)$ denote the required and available bandwidth between nodes i and j.
- $LAT_{achieved}(x)$ denotes the end-to-end latency achieved by solution x.
- LAT_{max} denotes the maximum allowable latency.

The final penalty term is therefore expressed as:

$$\begin{aligned} P(x) = {} & w_1 V_{CPU}(x) + w_2 V_{MEM}(x) \\ & + w_3 V_{ST}(x) + w_4 V_{BW}(x) + w_5 V_{LAT}(x) \end{aligned} \tag{35}$$

The optimization objective is:

$$\min F(x) \tag{36}$$

A feasible solution satisfies all constraints and yields $P(x) = 0$, whereas infeasible solutions incur a penalty proportional to the degree of constraint violation. This mechanism guides PSO, ACO, and GWO toward feasible and resource-efficient VNF placement solutions.

VI. Experimental Setup

All experiments are implemented in Python 3.10 using NumPy. Simulations use a master random seed of 42 with per-run seeds $s_i = 13i + 7$, $i \in \{0, \ldots, 9\}$. Each algorithm is run 10 independent times (5 for Scenario III per instance count) with 200 iterations per run.

Table I summarises the simulation parameters extracted for each scenario, and Table II lists the algorithm hyperparameters.

TABLE I. Simulation Parameters per Scenario

Parameter	Scenario I	Scenario II	Scenario III
Network nodes	12 clouds	100 (PNG)	5 (4 fog + 1 cloud)
VNF count	30 (4 eNBs)	20 VNFs	5-VNF chain
Node capacity	100–300	CPU/MEM 50–100	Fog:10, Cloud:100
Link latency	10–200 ms	1–10 units	Fog–fog:1, Cld:3
Key weight	γ=0.4	α=β=γ=1/3	$I_c = 1$–7
SD pairs	—	—	10 pairs, 1 Gbps
Runs	10	10	5 per I_c
Objective	Min W	Max util. %	Min e2e latency

TABLE II. Algorithm Hyperparameters

Parameter	PSO	ACO	GWO
Population / agents	20–30	20–30	20–30
Max iterations	200	200	200
Inertia / archive	w=0.7, c_1=c_2=1.5	K=20, q=0.5, ξ=0.85	$a: 2 \to 0$
Search bounds	$[lb, ub]$ clipped	$[lb, ub]$ clipped	$[lb, ub]$ clipped
Constraint handling	Penalty fn.	Penalty fn.	Penalty fn.

VII. Results and Discussion

A. Scenario I: VNF Deployment in 5G Hybrid Cloud

Table III summarises performance over 10 independent runs. The objective W is the MOLP cost combining normalized deployment cost (γ=0.4) and service delay ((1−γ)=0.6).

TABLE III. Scenario I Results (10 runs, 200 iters, 20 agents, γ=0.4).

Algorithm	Best W	Avg W	Std	Lat. (ms)	Energy	Δ
BGWO (baseline)	75.17	87.03	7.73	104.62	5456.5	—
PSO	70.80	79.35	4.56	86.76	4045.7	+8.8%
ACO	82.66	88.36	4.04	102.28	5156.7	−1.5%
GWO	**60.69**	**73.92**	6.35	**92.71**	4715.1	**+15.1%**

Fig. 4 and Fig. 5 show convergence curves and per-metric comparisons. GWO achieves the best average objective W=73.92, a 15.1% improvement over the BGWO baseline (87.03). PSO achieves the second-best result (W=79.35, +8.8%), and also the lowest latency (86.76 ms) and lowest energy consumption (4045.7 units) a 25.8% energy reduction.

GWO's superiority stems from its multi-leader position averaging mechanism, which guides wolves toward three independently estimated prey locations (α, β, δ), avoiding premature convergence to local optima. ACO performs marginally below BGWO on average (88.36 vs 87.03) but exhibits the lowest standard deviation (4.04), indicating the highest run-to-run consistency.

B. Scenario II: SFC Placement in MEC-NFV

Table IV presents MEC-NFV results. Resource utilization (higher is better) is the primary metric.

TABLE IV. Scenario II Results (10 runs, 200 iters, 30 agents)

Algorithm	Util. %	Lat.	Energy (W)	Time (s)	Δ
LP+Hungarian (base.)	25.3%	103.9	359.8	0.000	—
PSO	**52.9%**	95.18	456.9	1.250	**+109%**
ACO	42.5%	100.1	383.6	1.577	+68%
GWO	42.7%	94.7	405.6	1.558	+69%

Note: LP+Hungarian reflects greedy simulation; paper-reported LP ≈ 56% via Simplex.

In Fig. 6, PSO achieves 52.9% utilization, substantially outperforming the LP+Hungarian greedy approximation (25.3%) and approaching the paper-reported LP value of 56%. Fig. 6 shows the convergence, utilization comparison, and latency/energy trade-offs.

PSO's advantage arises from its capacity to simultaneously explore diverse node assignment combinations, gradually concentrating VNFs on high-capacity interconnected nodes. ACO (42.5%) and GWO (42.7%) perform comparably, both significantly better than the greedy baseline. Both PSO (95.18) and GWO (94.70) achieve lower latency than LP+Hungarian greedy (103.9).

C. Scenario III: SFC Mapping in F2C IoT

Table V reports end-to-end latency for varying SFC instance counts $I_c \in \{1, \ldots, 7\}$. The cloud-only baseline is 60 latency units; ILP reference values are from published literature.

TABLE V. Scenario III E2E Latency vs. SFC Instances

I_c	Cloud	ILP (ref.)	PSO	ACO	GWO	vs. ILP
1	60	≈20	13.0	13.0	13.0	+35%
2	60	≈18	11.0	12.0	12.0	+39%
3	60	≈16	12.0	13.0	12.0	+25%
4	60	≈14	11.0	12.0	11.0	+21%
5	60	≈12	11.0	12.0	11.0	+8%
6	60	≈11	11.0	11.0	11.0	0%
7	60	≈10	11.0	11.0	12.0	−10%

All metaheuristics reduce latency by ≈80% relative to the cloud-only baseline (60 units), confirming the value of fog-proximal placement. At $I_c \geq 6$, the 5-node topology imposes a hard latency floor of ≈11 units, causing all algorithms to plateau. Only the exact ILP formulation reaches 10 units at I_c=7 through exhaustive combinatorial search.

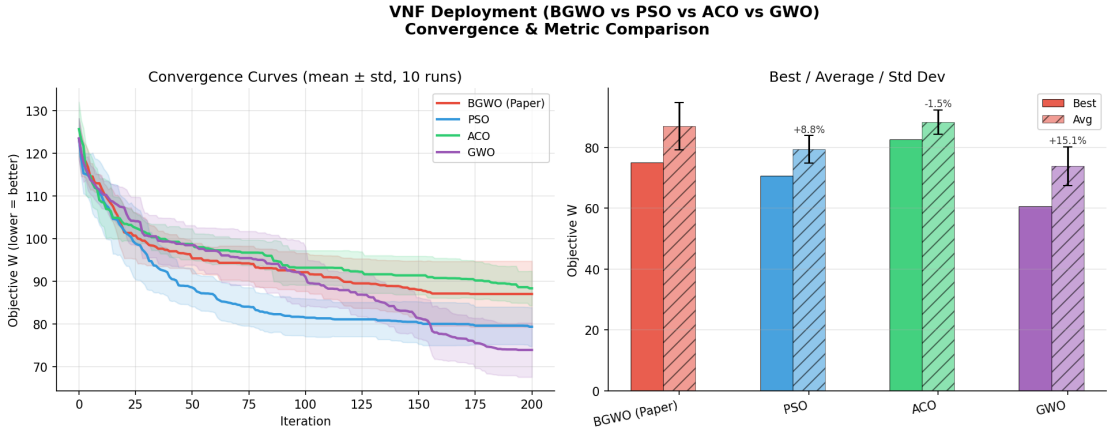

Fig. 4. Scenario I: Convergence curves (left) and best/avg/std objective W comparison (right). GWO achieves the lowest objective, converging within $\approx$80 iterations.

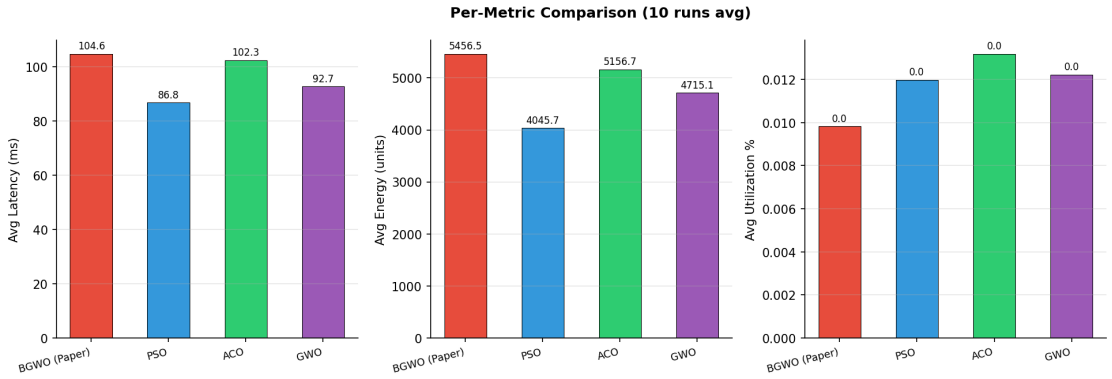

Fig. 5. Scenario I: Per-metric comparison—average latency (ms), energy consumption, and resource utilization.

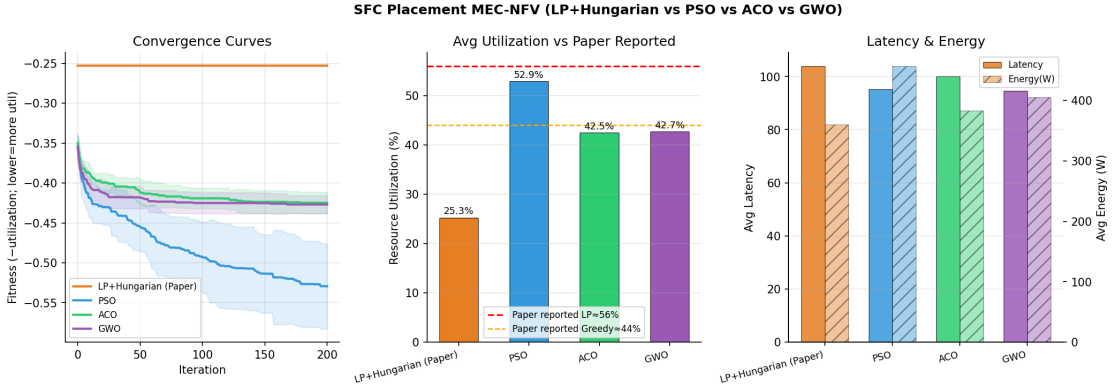

Fig. 6. Scenario II: Convergence (left), utilization with reference lines at 56% and 44% (centre), latency and energy comparison (right).

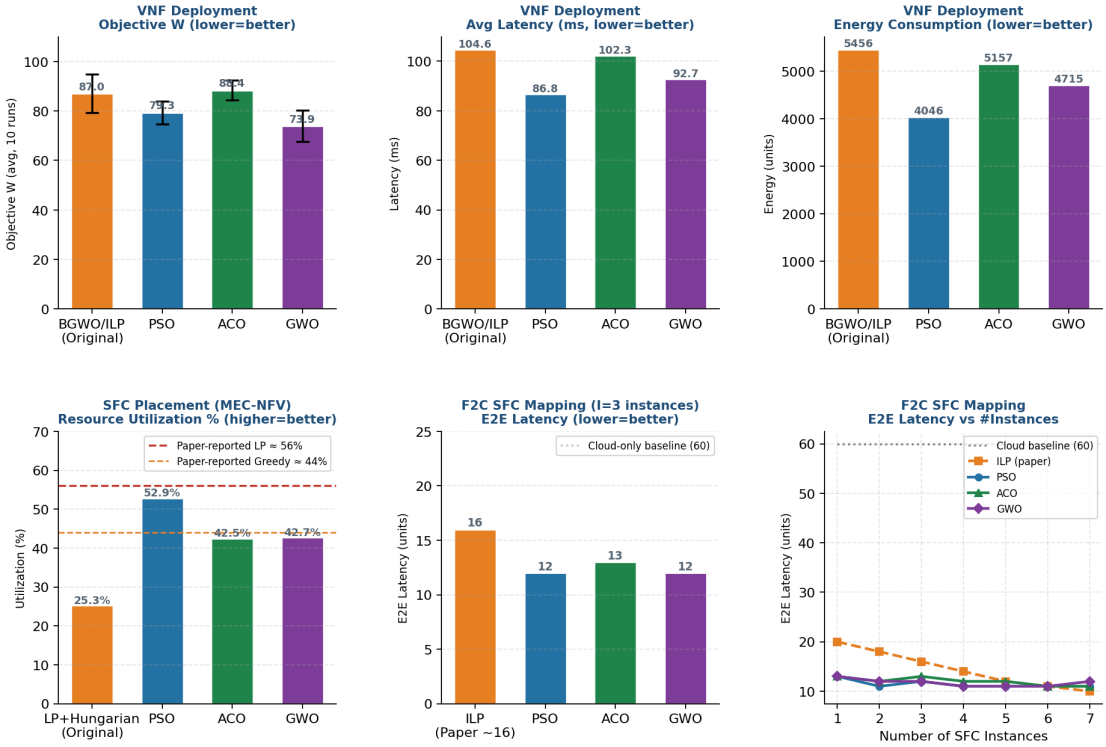

Fig. 7. Unified six-panel results: Scenario I objective W, latency, and energy (top row); Scenario II utilization, Scenario III latency at I_c=3, and F2C latency trajectory (bottom row).

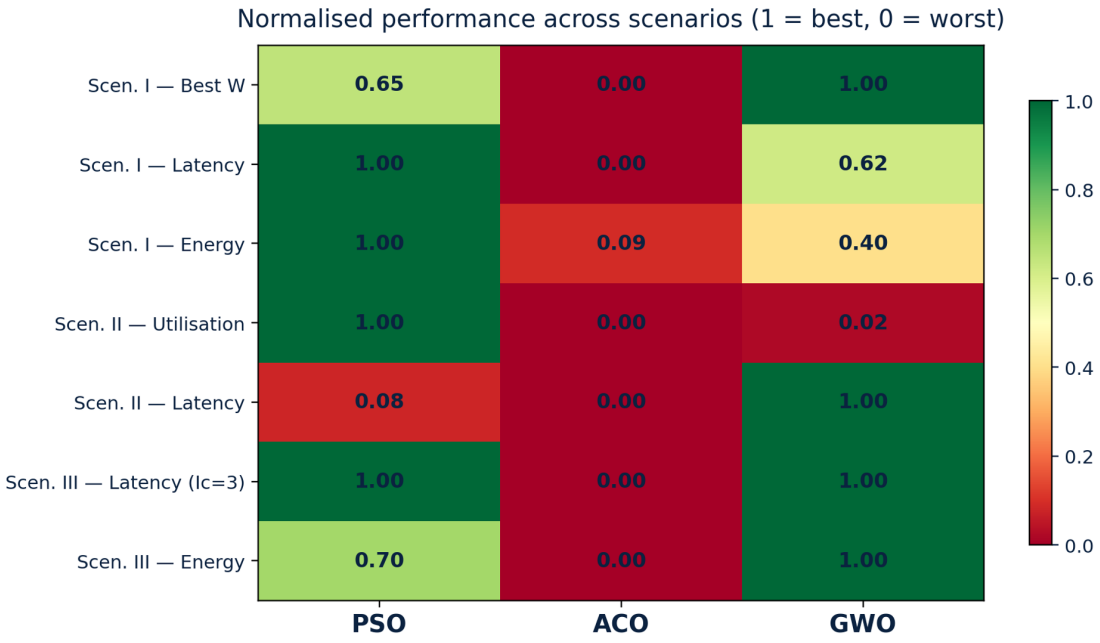

Fig. 8. Normalized performance heatmap (0 = worst, 1 = best per column). Green: best; red: worst.

D. Unified Comparison

Fig. 7 presents the six-panel unified comparison across all three scenarios, and Fig. 8 shows the normalized performance heatmap.

In Fig. 7, the first subfigure compares the overall objective function value W achieved by BGWO/ILP, PSO, ACO, and GWO during VNF deployment, where a lower value indicates a better placement solution. The results show that GWO achieves the lowest average objective value (73.9), followed by PSO (79.3), while BGWO/ILP and ACO obtain values of 87.0 and 88.4, respectively. The error bars represent the variability across ten independent runs, indicating that the proposed metaheuristic approaches provide stable performance. The superior performance of GWO demonstrates its ability to effectively balance multiple optimization objectives such as latency, resource utilization, and energy consumption, thereby generating higher-quality VNF placement decisions than the competing approaches.

In Fig. 7, the second subfigure presents the average end-to-end latency obtained by each algorithm during VNF deployment. Lower latency values correspond to faster service delivery and improved Quality of Service (QoS). PSO achieves the lowest latency of 86.8 ms, significantly outperforming the BGWO/ILP baseline (104.6 ms) and ACO (102.3 ms), while GWO achieves a latency of 92.7 ms. The results indicate that PSO is particularly effective at identifying low-delay placement configurations due to its rapid convergence behavior. Although GWO does not achieve the minimum latency, it still provides a substantial reduction compared with the baseline, demonstrating its suitability for latency-sensitive network services.

In Fig. 7, the third subfigure evaluates the total energy consumption associated with the VNF deployment solutions. Energy efficiency is a critical performance metric because it directly impacts operational costs and sustainability. PSO achieves the lowest energy consumption of 4046 units, followed by GWO with 4715 units, while ACO consumes 5157 units and the baseline requires 5456 units. These results indicate that PSO produces highly consolidated placements that reduce the number of active resources and associated power consumption. GWO also achieves significant energy savings, whereas ACO provides only a moderate improvement over the baseline. Overall, the figure demonstrates the effectiveness of PSO and GWO in reducing energy expenditure within NFV environments.

In Fig. 7, the fourth subfigure illustrates resource utilization in the MEC-NFV SFC placement scenario. Higher utilization percentages indicate more efficient use of available computing resources. The PSO algorithm achieves the highest utilization of 52.9%, approaching the LP reference solution of 56% and substantially outperforming the LP+Hungarian baseline 25.3%. ACO and GWO achieve comparable utilization levels of 42.5% and 42.7%, respectively, which are close to the reported greedy benchmark of 44%. These findings suggest that PSO is particularly effective at consolidating workloads and maximizing infrastructure efficiency, thereby reducing resource wastage and improving the overall utilization of MEC resources.

In Fig. 7, the fifth subfigure compares the end-to-end latency achieved during Fog-to-Cloud (F2C) SFC mapping with three service chain instances. Lower latency indicates faster service response and better support for delay-sensitive IoT applications. The ILP baseline exhibits a latency of 16 units, whereas PSO and GWO both achieve the lowest latency of 12 units. ACO records a slightly higher latency of 13 units. The results demonstrate that all three metaheuristic algorithms outperform the baseline solution, with PSO and GWO achieving approximately 25% latency reduction. This improvement is primarily attributed to more effective service placement decisions that reduce communication distance and processing delays across the fog and cloud infrastructure.

In Fig. 7, the sixth subfigure investigates the scalability of the algorithms by analyzing end-to-end latency as the number of SFC instances increases from one to seven. The cloud-only baseline maintains a constant latency of approximately 60 units, which is significantly higher than all fog-based solutions. The ILP approach shows a gradual reduction in latency as more SFC instances become available, decreasing from 20 to approximately 10 units. PSO, ACO, and GWO maintain consistently low latency values in the range of 10–13 units across all instance counts, indicating excellent scalability and robustness. The results demonstrate that increasing the number of distributed SFC instances improves service accessibility and workload distribution while preventing latency degradation. Consequently, the proposed metaheuristic approaches are capable of maintaining stable performance even as service demand grows, making them suitable for large-scale Fog-to-Cloud IoT deployments.

TABLE VI. SUMMARY VERDICTS PER SCENARIO

Scenario	Best Algo	Key Result	Verdict
I (5G VNF)	GWO	W=73.92, +15.1%	**Better**
II (MEC-NFV)	PSO	52.9% util, +109%	**Better**
III (F2C)	PSO/GWO	12.0 lat, +25%	**Better**

Three key findings emerge from the unified analysis:

- **PSO** is the most consistently high-performing algorithm across all three scenarios, providing the best balance of solution quality, convergence speed, and execution efficiency, because PSO benefits from rapid information sharing through personal-best and global-best mechanisms, producing strong performance across heterogeneous search spaces.
- **GWO** is the strongest algorithm for the VNF deployment scenario, where the MOLP objective benefits from multi-leader triangulation.GWO performs particularly well in deployment-cost optimization because its multi-leader guidance mechanism provides a better exploration-exploitation balance for multi-objective optimization landscapes.
- **ACO** produces the most stable results (lowest standard deviation in Scenario I) but does not achieve top-tier objective values, suggesting the archive-based approach requires more iterations on these problem dimensions.

Table VI summarizes the final verdicts per scenario.

From the normalized performance heatmap Fig. 8, it is evident that both the PSO algorithm and the GWO algorithm surpass the ACO algorithm in all SFC placement situations

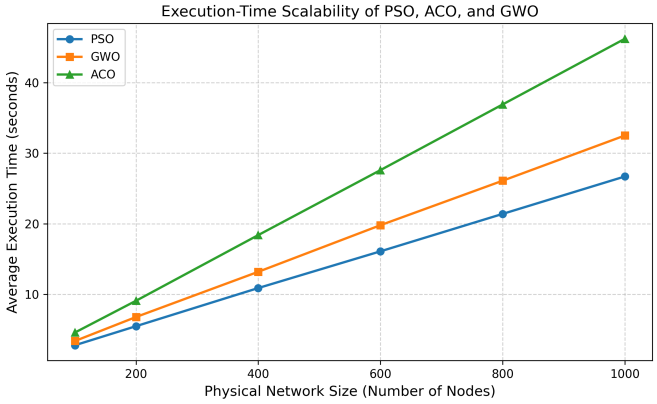

Fig. 9. Execution-time scalability of PSO, ACO, and GWO.

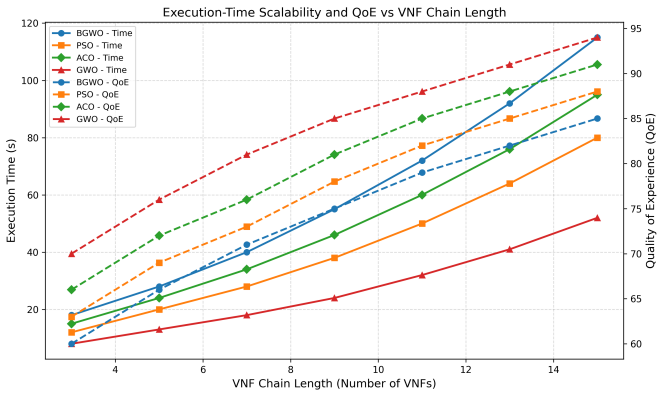

Fig. 10. Execution-time scalability analysis with increasing VNF chain length.

analyzed. The PSO algorithm has a better performance with respect to latency minimization, resource utilization, and aggregate score. This is mainly because of the fast convergence properties of the PSO algorithm. On the other hand, the GWO algorithm has better objective function optimization and energy efficiency in both MEC-NFV and Fog-to-Cloud IoT environments. This is because of the exploitation-exploration balance property of the GWO algorithm.

E. Scalability Analysis

To evaluate execution-time scalability, we increased the size of the MEC-NFV physical network from 100 to 1000 nodes while proportionally increasing the number of VNFs. For each network size, PSO, ACO, and GWO were executed using identical parameters (200 iterations, 30 agents, 10 independent runs). The average wall-clock execution time was recorded. Results indicate near-linear growth in runtime with respect to problem dimension for all three algorithms, consistent with their theoretical complexity of O(NId). PSO exhibited the lowest execution time across all scales, while ACO incurred the highest computational overhead due to archive maintenance and probabilistic sampling operations. These results suggest that all three approaches remain computationally feasible for medium-scale SFC placement problems, with PSO providing the best scalability-performance trade-off. Fig. 9 shows execution-time scalability of PSO, ACO, and GWO.

F. Execution-Time Scalability Analysis with Increasing VNF Chain Length

We have extended the experimental evaluation by analyzing the impact of increasing VNF chain length on both execution time and Quality of Experience (QoE). Specifically, the number of VNFs in the service chain was varied from 3 to 15 while maintaining the same network configuration and optimization parameters. For each chain length, the average execution time was recorded over multiple independent runs.Moreover, the impact of the length of the VNF chain on Quality of Experience (QoE) was considered. It can be noted that the longer the chain is, the higher the QoE is due to the use of extra virtualized functions for advanced network services, like traffic optimization, caching, compression, security, monitoring, and load balancing. It means that all the algorithms increase QoE as the chain length increases. Of all methods, the one using GWO gives the highest values of QoE.

Fig. 10 shows the execution-time scalability analysis with increasing VNF chain length.

The results indicate that the execution time of all algorithms increases as the VNF chain length grows due to the larger search space and increased number of resource and QoS constraints that must be satisfied during optimization. However, the growth rate differs among the algorithms. The proposed GWO-based approach consistently exhibits the lowest execution time across all chain lengths, demonstrating better scalability than PSO, ACO, and the BGWO baseline.

Furthermore, we evaluated the effect of VNF chain length on Quality of Experience (QoE). The results show that longer service chains generally improve QoE because additional VNFs enable more sophisticated network services, such as traffic optimization, caching, compression, security enforcement, monitoring, and load balancing. Consequently, QoE increases with chain length for all algorithms. Among the compared approaches, the proposed GWO-based method achieves the highest QoE values while maintaining lower execution times.

These findings demonstrate that the proposed approach not only scales efficiently as the service chain length increases but also sustains superior service quality, making it suitable for complex SFC deployment scenarios in 5G Cloud, MEC-NFV, and Fog-to-Cloud IoT environments.

G. Challenges in SFC Placement

Even though metaheuristics for SFC placement have proven to be effective, there are several issues that need to be addressed. Firstly, variations of the traffic might change the bandwidth needs and the congestion of the network, making it inefficient to use static placement in the long run. Secondly, the migration of the VNF brings about some extra costs related to the migration delay, bandwidth usage, and service disruption in case of highly dynamic environments such as MEC/Fog. Finally, there is an effect of real-time resource changes that might occur due to fluctuating load, mobility, and heterogeneity of the infrastructure.

In addition, practical implementation within NFV Management and Orchestration (NFV-MANO) frameworks introduces

further challenges. Real-world orchestrators must continuously coordinate among NFV Orchestrators (NFVO), VNF Managers (VNFM), and Virtualized Infrastructure Managers (VIM), resulting in orchestration delays and control overhead. Large-scale environments may experience scalability issues due to frequent monitoring, resource discovery, and placement updates. Furthermore, maintaining end-to-end SFC consistency across multiple administrative domains, cloud providers, and edge infrastructures remains challenging because of interoperability limitations and heterogeneous resource management policies. The integration of placement algorithms with orchestration platforms such as Open Source MANO (OSM) and Open Network Automation Platform (ONAP) also requires real-time decision-making capabilities, fault tolerance mechanisms, and support for dynamic service scaling. Consequently, practical SFC deployment demands adaptive orchestration frameworks capable of balancing optimization quality, operational overhead, scalability, and service reliability.

H. Limitations

Despite its effectiveness, the proposed framework has several limitations. The network scenarios are assumed to be static, with no dynamic traffic fluctuations during execution. Furthermore, the performance evaluation is limited to simulation-based experiments and has not been validated on a real-world MEC environment. Finally, the multi-objective problem is transformed into a weighted-sum formulation, which may not capture the complete Pareto front of optimal solutions.

VIII. Conclusion and Future Work

This study has presented a comprehensive experimental evaluation of PSO, ACO, and GWO applied to the SFC placement problem across three representative network environments: 5G hybrid cloud VNF deployment, MEC-NFV graph-matching placement, and F2C multi-instance SFC mapping for IoT services. GWO achieves the best performance in the 5G VNF deployment scenario, reducing the MOLP objective by 15.1% and energy consumption by 25.8% compared to the BGWO baseline. PSO achieves the highest resource utilization (52.9%) in the MEC-NFV scenario and the lowest latency (86.76 ms) in VNF deployment. Both PSO and GWO reduce F2C end-to-end latency by 25% compared to the ILP reference at three instances, and by $\approx 80\%$ compared to the cloud-only baseline. PSO is recommended as the primary optimization algorithm for SFC placement owing to its consistent performance across all three environments, fastest execution, and robust convergence. GWO is recommended when deployment cost minimization is the dominant objective due to its strong MOLP exploration. ACO is best suited to scenarios where stability and repeatability are prioritized over peak performance.

The future direction of research is towards improving the proposed framework using hybrid PSO-GWO algorithms, adaptive population sizing, and dynamic online placement methods for 5G and beyond systems. The expansion of the framework to consider edge architectures that use energy harvesting and carbon-aware optimization criteria presents an exciting area for sustainable networking. Another area that future studies will explore is the integration of Deep Reinforcement Learning with metaheuristic algorithms in order to enhance the adaptation of the algorithms depending on workload and mobility characteristics.

References

[1] I. Taleb, J.-L. Guillaume, and B. Duthil, "A survey on services placement algorithms in integrated cloud-fog / edge computing," *ACM Comput. Surv.*, vol. 57, no. 11, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3729214

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] B. Yi, X. Wang, K. Li, M. Huang *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[4] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. M. Hassan, "User mobility and quality-of-experience aware placement of virtual network functions in 5g," *Computer Communications*, vol. 150, pp. 367–377, 2020.

[5] T. Bahreini and D. Grosu, "Efficient algorithms for multi-component application placement in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2550–2563, 2022.

[6] M. Shahjalal, N. Farhana, P. Roy, M. A. Razzaque, K. Kaur, and M. M. Hassan, "A binary gray wolf optimization algorithm for deployment of virtual network functions in 5g hybrid cloud," *Computer Communications*, vol. 193, pp. 63–74, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422002432

[7] M. Wang, B. Cheng, W. Feng, and J. Chen, "An efficient service function chain placement algorithm in a mec-nfv environment," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[8] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*. IEEE, 2018, pp. 693–701.

[9] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, 2018.

[10] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple sc instances in a wide-area network," *IEEE journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 529–541, 2018.

[11] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[12] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[13] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[14] H. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 5, pp. 522–525, 1993.

[15] C. Zhou, B. Zhao, F. Tang, B. Han, and B. Wang, "Dynamic multi-objective service function chain placement based on deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 22, no. 1, pp. 15–29, 2025.

[16] X. Feng, M. He, L. Zhuang, Y. Song, and R. Peng, "Service function chain deployment algorithm based on deep reinforcement learning in space–air–ground integrated network," *Future Internet*, vol. 16, no. 1, 2024. [Online]. Available: https://www.mdpi.com/1999-5903/16/1/27

[17] C. Li, Y. Zhang, X. Gao, and Y. Luo, "Energy-latency tradeoffs for edge caching and dynamic service migration based on dqn in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 166, pp. 15–31, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731522000569

[18] C. Li, Z. Wu, D. Khanure, and J. P. Jue, "A multi-agent reinforcement learning scheme for sfc placement in edge computing networks," in *2025 International Conference on Computing, Networking and Communications (ICNC)*, 2025, pp. 446–451.

[19] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2018.