GVNF-P: An Energy-Aware Genetic Algorithm for VNF Placement in B5G Networks

Matheus Pantoja, Albert Santos, Reyso Teixeira, Rafael Vieira, Diego Cardoso

***Abstract*— This paper presents GVNF-P, a genetic-algorithm framework that minimises energy consumption while placing Virtual Network Functions (VNFs) for service function chains in beyond-5G (B5G) cloud networks. The algorithm encodes Service Function Chain (SFC) order within chromosomes, leverages a server-centred power model that separates static and dynamic costs, and employs problem-aware crossover and mutation operators to guide the search through feasible solutions. Experiments on heterogeneous B5G topologies show that GVNF-P achieves energy savings within 3% of an integer linear programming optimum while reducing computation time by more than an order of magnitude—dropping from 4.5 minutes to just 0.12 minutes—and keeping memory usage low under realistic traffic and resource load conditions.**

***Keywords*— Energy efficiency, B5G networks, NFV, VNF placement, Genetic algorithm.**

I. Introduction

Mobile traffic grows exponentially, driven by the proliferation of connected devices and increasingly demanding applications. In this setting, Network Function Virtualization (NFV) replaces costly, inflexible proprietary middleboxes with software-based Virtual Network Functions (VNFs) such as firewalls (FW), Network Address Translation (NAT) and Intrusion Detection Systems (IDS), all running on commodity servers [1]–[3]. By decoupling function logic from dedicated hardware, NFV lowers capital expenditure, accelerates automation and equips networks for the service diversity expected in beyond-5G (B5G) environments.

Each flow must still traverse a fixed sequence of VNFs—the Service Function Chain (SFC)—for example FW $\rightarrow$ NAT $\rightarrow$ IDS. When instances are scattered arbitrarily, some servers are activated unnecessarily, while others remain under-utilised. Such waste compounds an already serious picture: the Information and Communication Technology(ICT) sector draws roughly 109 GW worldwide [4]. In carrier-scale B5G clouds, even small placement inefficiencies elevate operational costs and the carbon footprint [5].

The crux of the problem lies in deciding how many instances of each VNF to create and on which servers to deploy them, under CPU and memory constraints and the order enforced by the SFC. The decision space is combinatorial: as the number of VNFs, servers and chains increases, it grows explosively. Naïve strategies that simply disperse functions or replicate instances to *provide headroom* switch on idle hardware, burn energy needlessly and diminish local traffic consolidation—an issue intensified by heterogeneous loads and topologies in B5G networks.

To address this bottleneck, we propose a Genetic Algorithm (GA) placement framework that encodes the SFC order directly into the chromosome, limiting the search to feasible solutions. The algorithm uses a power model distinguishing static (idle) and dynamic (load-dependent) energy consumption, promoting consolidation of VNFs on the minimum number of active servers. Experiments on heterogeneous B5G topologies show that the method achieves energy efficiency close to an Integer Linear Programming (ILP) reference [5], while drastically reducing computational cost. Key features include: a compact chromosome encoding preserving SFC order without repair; a unified objective function accounting for static and dynamic power via a server-centric model; near-optimal energy efficiency within a few percentage points of ILP with significantly lower runtime; and a scalable, modular framework validated across diverse B5G scenarios.

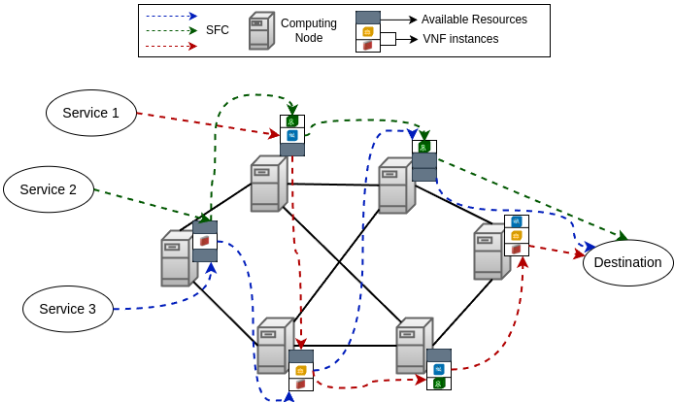

Fig. 1. A multiple source updating system enabled by NFV. Three SFCs (depicted in three different colors) are integrated into the network.

The remainder of this article is organised as follows. Section II reviews energy-aware NFV-placement research; Section III presents the power model and problem formulation; Section IV details the proposed GA; Section V describes the simulation environment and discusses results; Finally, Section VI concludes this work.

II. Related Work

Several studies have addressed the VNF placement problem from diverse angles, such as latency minimization, load balancing, energy efficiency, and scalability, using approaches

Matheus Pantoja, UFPA, Belém-PA, e-mail: matheus.pantoja@itec.ufpa.br; Albert Santos, UFPA, Belém-PA, e-mail: albert.santos@itec.ufpa.br; Reyso Teixeira, UFPA, Belém-PA, e-mail: reyso.teixeira@itec.ufpa.br; Rafael Vieira, UEAP, Macapá-AP, e-mail: rafael.vieira@ueap.edu.br; Diego Cardoso, UFPA, Belém-PA, e-mail: diego@ufpa.br

ranging from exact models to heuristics and machine learning, each with trade-offs in performance and computational cost.

In [6], a genetic algorithm was proposed to handle heterogeneous service demands by preserving VNF order within chains. While effective in improving acceptance ratios, it does not consider energy consumption. The method in [7] used a Deep Q-Network to adapt placements dynamically, improving convergence and resource use, but also overlooked energy efficiency. The ILP formulation in [8] achieved optimal latency-aware placements, yet proved computationally infeasible at scale. Conversely, the algorithm in [4] introduced energy-aware decisions that significantly reduced power use, although it lacked the flexibility of heuristic solutions.

In contrast, our approach integrates an SFC-compliant genetic algorithm with a unified power model, enabling near-optimal energy efficiency across servers, switches, and links, while maintaining low computational cost and high scalability.

III. Problem Definition

The VNF placement problem involves deciding the number and location of VNF instances and routing each demand through its SFC. The solution must respect server CPU capacities, VNF processing limits, link bandwidth, end-to-end delay requirements, and the function order without forming cycles.

The objective is to minimise total energy consumption, accounting for switch idle and active-port costs, server idle and dynamic CPU usage, and link transmission power. A solution is feasible if it satisfies all constraints and optimal if it minimises energy usage. The problem is strongly NP-Hard, even under simplified conditions [5].

A. System Model

The physical substrate is abstracted as a bidirectional graph $G = (N, L)$, where each vertex $i \in N$ integrates a packet-switching fabric with a general-purpose server that can host one or more VNF instances. Every node exposes an aggregate budget of CPU, memory, and storage; a VNF placed on the node consumes a slice of those resources and triggers two power components: a static cost for keeping the physical machine powered on and a dynamic cost that scales with the number of active VNF instances. Edges $(i, j) \in L$ share a uniform link-capacity budget that limits the sum of traffic routed over them and incur an additional power cost only when at least one flow is present.

Traffic is represented by a set D of service demands. Each demand $d = \langle o_d, t_d, B_d, \mathcal{C}_d \rangle$ specifies an origin node o_d, a destination node t_d, a required bandwidth B_d, and an ordered service-function chain $\mathcal{C}_d = (v_1, \ldots, v_m)$ that must be traversed between source and destination.

TABLE I

Principal sets, parameters and variables.

Symbol	Description	Type
	Sets	
N	Set of all nodes	–
L	Set of all links	–
F	Set of all VNFs	–
D	Set of all demands	–
	Parameters	
C_i^r	Capacity of resource $r \in \{\text{CPU}\}$ on node i	–
C_f^r	Resource r required by one instance of VNF f	–
$B_{(i,j)}$	Bandwidth of link (i, j)	–
B_d	Bandwidth requested by demand d	–
$\mathcal{C}_d$	Ordered VNF chain required by demand d	–
B_f	Processing capacity of one VNF–f instance	–
D_f	Processing delay introduced by VNF f	–
$D_{(i,j)}$	Propagation delay on link (i, j)	–
D_d	End-to-end delay bound for demand d	–
P_{sm}, P_{mm}	Idle / peak server power	–
P_{ss}, P_p	Idle switch power / per-active-port power	–
	Decision variables	
x_i	1 if the server at node i is powered on	bin.
y_i	1 if the switch part of node i is powered on	bin.
$l_{(i,j)}$	1 if link (i, j) carries any traffic	bin.
$z_{i,f}$	Number of VNF–f instances deployed on node i	$\mathbb{Z}_{\geq 0}$
$u_{i,f,d}$	1 if demand d is processed by VNF f on node i	bin.
$w^{f \to g}_{(i,j),d}$	1 if segment $(f \to g)$ of demand d is routed over link (i, j)	bin.

B. Constraints

C1 — VNF visitation:

$$\sum_{i \in N} u_{i,f,d} = 1, \qquad \forall d \in D,\ \forall f \in \mathcal{C}_d. \tag{1}$$

Equation (1) forces each demand to be processed exactly once by every VNF that appears in its service-function chain, selecting one—and only one—physical node to perform each step.

C2 — Usage / placement consistency:

$$u_{i,f,d} \leq z_{i,f}, \qquad \forall i \in N,\ \forall f \in F,\ \forall d \in D. \tag{2}$$

Whenever a flow claims that VNF f handles it at node i, inequality (2) guarantees that at least one instance of f is really deployed there.

C3 — Node CPU capacity:

$$\sum_{f \in F} C_f^{\text{CPU}}\, z_{i,f} \leq C_i^{\text{CPU}}, \qquad \forall i \in N. \tag{3}$$

The cumulative CPU demand of all VNF instances placed on a node may not exceed that node's hardware limit.

C4 — Per-VNF throughput capacity:

$$\sum_{d \in D} B_d\, u_{i,f,d} \leq B_f\, z_{i,f}, \qquad \forall i \in N,\ \forall f \in F. \tag{4}$$

Traffic forwarded through a given VNF on a node is bounded by the aggregate processing rate provided by the instances of that VNF hosted there.

C5 — Link capacity:

$$\sum_{d \in D} \sum_{(f,g) \in \mathcal{C}_d} B_d\, w^{f \to g}_{(i,j),d} \leq B_{(i,j)}, \qquad \forall (i, j) \in L. \tag{5}$$

The left-hand summation collects the bandwidth of all virtual segments routed over link (i, j); it must not exceed the physical bandwidth of the link.

C6 — Flow conservation:

$$\sum_{j:(i,j)\in L} w^{f\to g}_{(i,j),d} - \sum_{j:(j,i)\in L} w^{f\to g}_{(j,i),d} = u_{i,f,d} - u_{i,g,d}, \quad (6)$$

for every demand d, each consecutive pair (f, g) in its chain, and every node i. Incoming and outgoing units of the virtual segment balance at every intermediate node, while a positive/negative right-hand term injects (absorbs) the flow where the segment originates (terminates).

C7 — Origin and destination anchoring:

$$u_{o_d,f_{\text{first}},d} = 1, \qquad u_{t_d,f_{\text{last}},d} = 1, \quad \forall d \in D. \quad (7)$$

Equation (7) pins the first function of the chain to the physical source node o_d and the last function to the physical destination t_d, thereby ensuring that traffic enters and leaves the substrate at the correct locations.

C8 — Loop avoidance: single egress:

$$\sum_{j\in N}\sum_{f\in F}\sum_{g\in F} w^{f\to g}_{(i,j),d} \le 1, \quad \forall i \in N,\ \forall d \in D. \quad (8)$$

A demand may leave a node at most once, which removes the possibility of routing cycles and reduces unnecessary switch activations.

C9 — Loop avoidance: single ingress:

$$\sum_{j\in N}\sum_{f\in F}\sum_{g\in F} w^{f\to g}_{(j,i),d} \le 1, \quad \forall i \in N,\ \forall d \in D. \quad (9)$$

Symmetric to C8, a demand can enter a node only once.

C10 — End-to-end delay:

$$\sum_{i\in N}\sum_{f\in F} D_f\, u_{i,f,d} + \sum_{(i,j)\in L}\sum_{(f,g)\in \mathcal{C}_d} D_{(i,j)}\, w^{f\to g}_{(i,j),d} \le D_d, \quad \forall d \in D. \quad (10)$$

Equation (10) ensures that, for each demand d, the total latency—computed as the sum of the processing delays D_f of all VNFs in its chain and the propagation delays $D_{(i,j)}$ over the chosen physical links—does not exceed the per-demand SLA D_d.

C11–C13 — Activation implications:

$$w^{f\to g}_{(i,j),d} \le l_{(i,j)}, \qquad \forall (i,j) \in L,\ d, f, g, \quad (11a)$$

$$l_{(i,j)} \le y_i, \qquad \forall (i,j) \in L, \quad (11b)$$

$$z_{i,f} \le x_i, \qquad \forall i \in N,\ f \in F. \quad (11c)$$

Because all variables are binary, the simple inequalities in (11) suffice to capture the logical relations: a link is active only if some flow traverses it; a switch is powered on when at least one incident link is active; and a server is powered on when it hosts at least one real VNF instance.

C. Energy Objective

Server power consumption:

$$P_{\text{srv}} = P_{sm}\sum_{i\in N} x_i + \big(P_{mm} - P_{sm}\big)\sum_{i\in N}\sum_{f\in F} \frac{C^{\text{CPU}}_f}{C^{\text{CPU}}_i}\, z_{i,f}. \quad (12)$$

The first term represents the static idle power of every powered-on physical machine; the second term is a linear approximation of the dynamic component proportional to CPU load contributed by the hosted VNFs.

Switching-fabric power consumption:

$$P_{\text{net}} = P_{ss}\sum_{i\in N} y_i + 2P_p \sum_{(i,j)\in L} l_{(i,j)}. \quad (13)$$

Each active switch incurs an idle cost P_{ss}; every active link lights two ports, hence the factor 2 multiplying P_p.

Global objective:

$$\min\ P_{\text{tot}} = P_{\text{srv}} + P_{\text{net}}. \quad (14)$$

Minimising P_{tot} simultaneously drives server consolidation and path shortening, delivering an energy-efficient placement-and-routing solution while all constraints (1)–(11) safeguard capacity, consistency and quality-of-service requirements.

IV. Proposed Method: GVNF-P Algorithm

Building upon the mathematical formulation of Section III-B, we designed **GVNF-P**—a Genetic Algorithm whose evolutionary components are tailored to the subtleties of VNF placement. Whereas traditional GAs treat chromosomes as abstract bit strings, GVNF-P integrates knowledge of the SFC, heterogeneous server capacities, and the unified power-consumption model, guiding the search almost exclusively through feasible—and energy-efficient—regions of the solution space.

A. Chromosome Encoding

Let $\mathcal{F} = \{f_1, \ldots, f_m\}$ denote the ordered set of network functions in the SFC, and $\mathcal{N}$ the set of physical nodes. A chromosome is the ordered vector

$$\mathbf{c} = [n_1, n_2, \ldots, n_m], \qquad n_k \in \mathcal{N},$$

where gene n_k specifies the host of function f_k. This mapping is *bijective*: decoding $\mathbf{c} \mapsto Z$ yields a unique allocation matrix, and no chromosome can violate the SFC order.

a) Seeding Strategy: The initial population mixes:

(i) *guided* individuals that place the most CPU-intensive functions on the most energy-efficient servers; and
(ii) purely random individuals for broad exploration.

B. Fitness Evaluation

For a chromosome $\mathbf{c}$, the fitness is defined as follows:

$$\text{fit}(\mathbf{c}) = P_{\text{srv}}(\mathbf{c}) + P_{\text{net}}(\mathbf{c}) \quad \text{if all constraints are satisfied;}$$

$$\text{fit}(\mathbf{c}) = \mathcal{M} \quad \text{otherwise,}$$

with $\mathcal{M} \gg \max P_{\text{tot}}$. Constraints are tested—CPU, per-VNF throughput, activation consistency, energy—in that order, aborting at the first violation to save computation.

TABLE II
CHARACTERISTICS OF THE SERVICE TYPES AND THEIR RESPECTIVE SFCS, BANDWIDTH, ALLOWABLE DELAYS, AND TRAFFIC PROPORTIONS

Service Type	Web Service	VoIP	Video Streaming	Online Gaming
VNF Chain	NAT–FW–TM–WOC–IDPS	NAT–FW–TM–FW–NAT	NAT–FW–TM–VOC–IDPS	NAT–FW–VOC–WOC–IDPS
Bandwidth	100 kbps	64 kbps	4 Mbps	50 kbps
Delay	500 ms	100 ms	100 ms	60 ms
% Traffic	18.2 %	11.8 %	69.9 %	0.1 %

C. Problem-Aware Operators

The GVNF-P algorithm employs problem-aware evolutionary operators. **Selection** is performed through a deterministic tournament of size k, ensuring competitive pressure among individuals. The **crossover** operator adopts a *power-balanced one-point* strategy, in which a cut point q is drawn uniformly at random, and parent chromosomes exchange their suffixes only if the resulting offspring respect the CPU constraints of all servers. **Mutation** is implemented as a *capacity-guided reset*: each gene is subject to mutation with probability p_{m}, selecting a new node in proportion to its available CPU resources.

a) Robustness mechanisms: GVNF-P embeds two safeguards directly in the evolutionary loop of Algorithm 1: (i) *elitism*, which copies the best μ individuals unchanged into the next generation; and (ii) an *adaptive mutation schedule*, which increases p_{m} by Δ whenever the global best solution stalls for T_{stall} generations.

Algorithm 1 GVNF-P: Energy-Centric GA for VNF Placement (compact)

Require: POP, GEN, k, p_c, p_m^0, Δ, T_{stall}, μ, $\mathcal{M}$
$P \leftarrow$ INITPOPULATION(POP); EVALFITNESS(P)
$c^* \leftarrow \arg\min_{c\in P} \mathrm{fit}(c)$; $p_m \leftarrow p_m^0$; stall$\leftarrow 0$
for $g = 1 \rightarrow$ GEN **do**
  $P' \leftarrow$ ELITE(P, μ)
  while $|P'| <$ POP **do**
    Select p_1, p_2 by TOURNAMENT(P, k)
    $(o_1, o_2) \leftarrow$ CROSSOVER(p_1, p_2, p_c)
    MUTATE(o_1, p_m); MUTATE(o_2, p_m)
    EVALFITNESS($\{o_1, o_2\}$)
    $P' \leftarrow P' \cup \{o_1, o_2\}$
  end while
  $P \leftarrow P'$
  if $\min_{c\in P} \mathrm{fit}(c) < \mathrm{fit}(c^*)$ **then**
    $c^* \leftarrow \arg\min_{c\in P} \mathrm{fit}(c)$; stall$\leftarrow 0$; $p_m \leftarrow p_m^0$
  else if (++stall = T_{stall}) **then**
    $p_m\Delta$; stall$\leftarrow 0$
  end if
end for
return c^* =0

Every operator in GVNF-P pursues energy efficiency—the search starts, mates, and mutates with power in mind. Elitism guarantees steady progress, and adaptive mutation revives diversity whenever improvement stalls, keeping the algorithm both focused and exploratory.

V. SOLUTION EVALUATION METHODOLOGY

This section compares GVNF-P against two benchmarks: Random, which assigns VNFs to nodes randomly, and ILP, which provides the optimal solution. The comparison focuses on energy consumption. The ILP formulation was implemented in Pyomo and solved with CPLEX version 22.1.1. The GA was implemented in Python 3.8.17 and all run on an Intel® Core™ i7-7700 @ 3.60GHz with 16GB RAM, Linux Mint. Experiments were repeated 10 times with varying workloads, and averages are reported.

Service flows include video streaming (VS), web services (WS), voice over IP (VoIP), and online gaming (OG), each requiring specific functions and bandwidth (Table II).

Tests considered six VNF types: NAT, Firewall, Traffic Monitor, WAN Optimization Controller, Video Optimization Controller, and Intrusion Detection System. Demands were generated and assigned to services according to traffic percentages (Table II). Each VNF required 4 CPU cores and 200 Mb/s processing capacity [9].

The simulations were conducted on the DFN topology from SNDLib, comprising 10 nodes and 45 links [10]. Power parameters followed [9]: each switch consumed 130W, active links added 10W, and servers consumed 175W when idle and up to 250W at full load. Each node included a 16-core server. Five scenarios were evaluated, varying the number of demands per node from 5 to 9, to analyse GVNF-P's performance under different workloads. For each case, GA hyperparameters (*pop*, *tor*, *gen*, *mut*, *cross*) were automatically tuned using Optuna (15 trials) and then applied across all evaluated scenarios, as shown in Table III.

TABLE III
GVNF-P – OPTIMAL HYPERPARAMETERS

Dem.	POP_SIZE	TOUR	GEN	MUT	CROSS
5	60	3	50	0.17	0.6
6	80	2	40	0.26	0.85
7	70	3	50	0.12	0.5
8	50	3	30	0.08	0.95
9	40	3	70	0.22	0.9

VI. RESULTS

Figure 2 compares the mean energy consumption (in watts) of ILP, GVNF-P and the random baseline as the number of demands per node increases from 5 to 9. ILP rises smoothly from 1 380 W to 1 750 W; the values in parentheses—(30 W at 5 demands/node and 25 W at 9 demands/node)—denote the standard deviation of each mean. GVNF-P closely follows ILP, deviating by at most 3 %—from 1 420 W (20 W) to 1 770 W (18 W). In contrast, the random heuristic consumes substantially more power, from 1 530 W (80 W) to 1 895 W (45 W), and exhibits the greatest variance, underscoring GVNF-P's near-optimal energy efficiency.

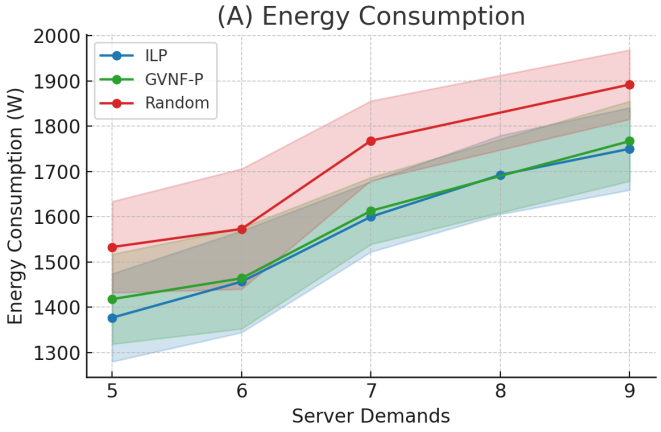

Fig. 2. Average Energy Consumption

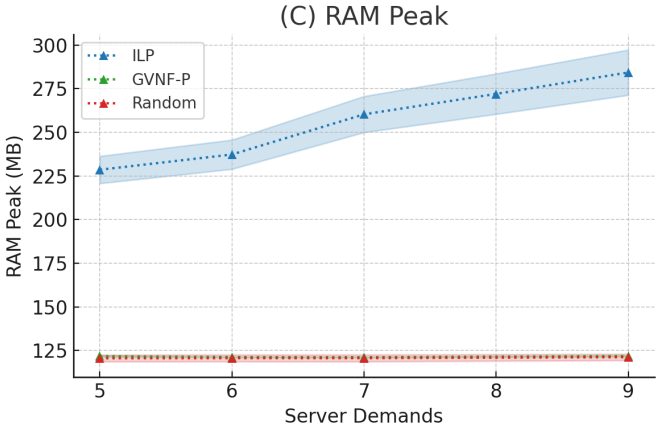

Fig. 4. Peak RAM Usage

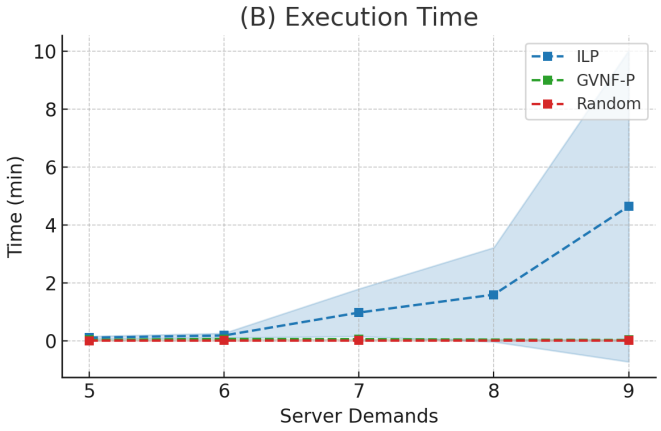

Fig. 3. Average Execution Time per Method

Figure 3 shows the mean solution time (in minutes) on the DFN topology as demands per node grow from 5 to 9. ILP's runtime climbs from 0.10 min to 4.70 min; the numbers in parentheses—(0.02 min at 5 demands/node and 0.45 min at 9 demands/node)—indicate the standard deviation around each average. GVNF-P remains consistently efficient, varying only from 0.12 min (0.03 min) to 0.18 min (0.05 min). The random method is essentially instantaneous (0.02 min, 0.005 min) but offers no energy-efficiency guarantees.

Figure 4 plots the maximum RAM consumption (in megabytes) observed as demands per node increase from 5 to 9. ILP's footprint grows from 230 MB to 285 MB; the parentheses—(8 MB at 5 demands/node and 6 MB at 9 demands/node)—report the standard deviation of each measurement. Both GVNF-P and the random assignment hold steady around 120–125 MB, with minimal variation (1 MB), highlighting the genetic algorithm's very low memory overhead.

VII. Conclusion

This work presented GVNF-P, a genetic algorithm tailored for energy-efficient VNF placement in B5G networks. By encoding SFC order in chromosomes and using a unified power model, GVNF-P closely matched the energy performance of an ILP baseline while reducing execution time by over an order of magnitude and maintaining low memory usage. Hyperparameter tuning with Optuna ensured robust configurations, and simulations on the DFN topology demonstrated scalability under increasing demand.

Future work includes: (i) exploring alternative metaheuristics, such as particle swarm optimization, to compare convergence dynamics; (ii) testing under higher traffic loads (above ten demands per node); and (iii) extending the model to a multi-objective formulation that jointly minimises energy and latency.

References

[1] J. Sun, Y. Zhang, F. Liu, H. Wang, X. Xu, and Y. Li, "A survey on the placement of virtual network functions," *Journal of Network and Computer Applications*, vol. 202, p. 103361, 2022.

[2] Z. Sharif, M. B. Jasser, K.-L. A. Yau, and A. Amphawan, "Advancements and challenges in latency-optimized joint sfc placement and routing: a comprehensive review and future perspectives," *International Journal of System Assurance Engineering and Management*, pp. 1–34, 2025.

[3] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 1–30, 2021.

[4] X. Zhang, Z. Xu, L. Fan, S. Yu, and Y. Qu, "Near-optimal energy-efficient algorithm for virtual network function placement," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 553–567, 2019.

[5] J. Li, X. Qi, J. Li, Z. Su, Y. Su, and L. Liu, "Fault diagnosis in the network function virtualization: A survey, taxonomy, and future directions," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 121–19 142, 2024.

[6] A. Serra, F. Paganelli, A. Brogi, and P. Cappanera, "A genetic algorithm for placing vnf chains with multiple flavours," in *2024 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2024, pp. 1–6.

[7] R. Mohamed, M. Avgeris, A. Leivadeas, and I. Lambadaris, "Optimizing resource fragmentation in virtual network function placement using deep reinforcement learning," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[8] A. Shirol, M. Vijayalakshmi, S. Dyavappanavar, and R. Shrinidhi, "Optimal placement of vnfs and service function chaining in an edge computing environment," in *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*. IEEE, 2024, pp. 741–748.

[9] A. Varasteh, M. De Andrade, C. M. Machuca, L. Wosinska, and W. Kellerer, "Power-aware virtual network function placement and routing using an abstraction technique," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[10] Zuse Institute Berlin, "Sndlib: The sndlib network optimization library," http://sndlib.zib.de/home.action, 2024, accessed: 2024-05-24.