

GEOVA: A Genetic Algorithm for Energy-Efficient VNF Placement in O-RAN Networks

Matheus Pantoja, Miguel Santos, Albert Santos, Rafael Vieira, Diego Cardoso

Abstract—This paper proposes GEOVA (*Genetic Energy-aware O-RAN VNF Allocator*), a batch Genetic Algorithm for the joint placement of access functions — virtualized Distributed Unit (vDU) and virtualized Centralized Unit (vCU) — and service chains in Open Radio Access Network (O-RAN) deployments. The algorithm minimizes the power consumption of the computational nodes hosting the VNFs subject to Service Level Agreement (SLA) constraints on latency, capacity, and isolation. Each chromosome jointly represents the placement of all scenario demands, which the genetic operators exploit to pack more admitted demands onto the infrastructure already switched on. The evaluation considers two topologies (POLSKA, 12 nodes; POPS, 27 nodes) under different load intensities. Under low load, GEOVA admits up to 13.9 percentage points more demands than the baselines while drawing up to 16.9% less power per accepted demand; under saturation it keeps both advantages on the tighter topology (+2.5 percentage points, $p < 0.001$) and matches the best baseline in acceptance on the larger one at lower power. The gains therefore do not come from rejecting demand.

Keywords—Open RAN, VNF placement, Genetic Algorithm, GRASP, service function chains.

I. INTRODUCTION

Fifth-generation (5G) mobile networks and their evolutions must support services with highly heterogeneous requirements, including high throughput, low latency, and massive device connectivity. To address this diversity, the Open Radio Access Network (O-RAN) architecture introduces functional disaggregation, virtualization, and open interfaces between components such as the Open Radio Unit (O-RU), the Open Distributed Unit (O-DU), and the Open Centralized Unit (O-CU) [1], [2]. This shift reduces dependence on proprietary, vendor-locked equipment and allows access network functions to be deployed as software over general-purpose computing nodes, supporting multi-vendor interoperability [2].

This architectural flexibility, however, increases the complexity of network planning. The choice of where to instantiate virtual functions directly impacts Central Processing Unit (CPU) usage, energy consumption, bandwidth utilization, and the propagation delay accumulated along the transport paths between consecutive functions. In addition, different functional splits between DU and CU impose distinct requirements on the fronthaul and midhaul [3]. As a result, inadequate placement decisions can waste resources or render delay-sensitive demands infeasible.

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

The challenge is further amplified when each demand, in addition to traversing the O-RAN access chain, must be processed by a sequence of service functions. In environments based on Network Function Virtualization (NFV), such services are modeled as Service Function Chains (SFCs), which must be placed while satisfying ordering, capacity, and end-to-end delay constraints [4]. A request may therefore need to traverse vDU, vCU, and functions such as Network Address Translation (NAT), firewall, or traffic inspection, jointly combining placement and routing decisions on the same infrastructure.

The resulting problem is combinatorial and known to be NP-hard [5], as it requires selecting nodes, function instances, and paths under multiple coupled constraints. Exact formulations are valuable for rigorously defining the problem, but tend to lose scalability as the problem size grows — e.g., with the numbers of demands, functions, and candidate paths [6]. In this context, heuristics and metaheuristics are suitable alternatives for producing good-quality solutions within computational time compatible with the timescales of planning and reconfiguration tasks performed offline by the Service Management and Orchestration (SMO) layer [2].

This paper proposes GEOVA, a Genetic Algorithm (GA) for virtual function placement in a hierarchical O-RAN infrastructure, considering an access chain with vDU/vCU and SFCs associated with the requested services. The model minimizes the power consumption of the infrastructure and treats end-to-end delay as an SLA constraint, jointly with the remaining capacity and isolation constraints on access functions. The proposal is compared against GRASP and a greedy strategy. Section II reviews related work; Section III presents the problem formulation; Section IV details the proposed genetic algorithm; Section V presents the evaluation scenario and methodology; Section VI discusses the experimental results; and Section VII concludes the paper.

II. RELATED WORK

Virtual Network Function (VNF) placement in O-RAN has been addressed through exact models, heuristics, and learning-based methods. For the access chain, [5] formulates the joint DU/CU placement and routing problem in Xhaul networks as an NP-hard Mixed Integer Linear Program (MILP) and proposes decomposition-based heuristics, while [6] reduces the combinatorial complexity by restricting routing to a set of predefined paths. Both works focus exclusively on the DU/CU chain and do not consider downstream service chains.

More recent works incorporate energy-aware objectives or learning-based search. [7] employs Integer Linear Program-

TABLE I
MAIN SETS, PARAMETERS, AND DECISION VARIABLES.

Sets	Description	
N	Set of computational nodes	
E	Set of physical links	
F	Set of virtual functions	
D	Set of demands	
$\mathcal{K}_d$	Stages of the chain of demand d	
$\mathcal{H}_d$	Hops between elements of chain d	
Parameters	Description	
c_n	CPU capacity of node n	
b_{ij}	Data rate capacity of link (i, j)	
ℓ_{ij}	Propagation delay of link (i, j)	
λ_d	Data rate required by demand d	
Δ_d	End-to-end delay budget of demand d	
c_f	CPU required by function f	
q_f	Capacity of one instance of function f	
δ_f	Processing delay of function f	
p_n^{idle}, p_n^{mc}	Idle and maximum power of node n	
$\Delta^{FH}, \Delta^{\Delta}$	Fronthaul and midhaul delay budgets	
$\beta_{d,n}$	1 if $n = s_d$; 0 otherwise (ingress indicator)	
M	Big positive number	
Variables	Description	Type
a_d	1 if demand d is accepted	Binary
$x_{d,k,n}$	1 if stage k of d is placed on node n	Binary
$r_{d,h,ij}$	1 if hop h of d uses link (i, j)	Binary
u_n	1 if node n is active	Binary
$m_{n,f}$	Instances of function f on node n	Integer
η_n	CPU load allocated to node n	Continuous

ming (ILP) and a graph-based heuristic to jointly minimize delay and energy consumption in Cloud-Fog RAN, with emphasis on dynamic functional splitting of DU/CU; [8] applies deep reinforcement learning to the joint placement of users and CU/DU; and [9] adopts the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for multi-objective placement of O-RAN data centers, focused on facility location rather than on function chains.

To our knowledge, the joint placement of vDU/vCU and downstream SFCs under a power-minimization formulation evaluated by an evolutionary metaheuristic has not been reported.

III. PROBLEM FORMULATION

The proposed formulation casts the VNF placement problem in O-RAN as an MILP. For each demand, the model jointly decides (i) whether the demand is admitted, (ii) on which node each functional stage is hosted, and (iii) which physical paths carry the traffic between consecutive stages. These decisions are coupled through processing-capacity, link data-rate, end-to-end delay, and access-chain isolation constraints. Table I summarizes the sets, parameters, and variables used in the formulation.

A. Infrastructure and Demands

The physical infrastructure is modeled as a bidirected graph $G = (N, E)$, where N is the set of computational nodes and E is the set of arcs; each physical link between two nodes appears in E as a pair of opposing arcs. Each node $n \in N$ has processing capacity c_n , hierarchical layer $\tau_n \in \{\text{far-edge}, \text{edge}, \text{core}\}$, idle power p_n^{idle} , and maximum power p_n^{max} . Each link $(i, j) \in E$ provides data-rate capacity b_{ij} and propagation delay ℓ_{ij} .

Let F be the set of virtual functions. Each function $f \in F$ is characterized by a CPU demand per traffic unit c_f , a per-instance capacity q_f , a processing delay δ_f , a set of eligible layers Γ_f , and an instance limit $\bar{m}_{n,f}$ at each node n .

Let D be the set of demands. Each demand $d \in D$ is specified by an ingress node s_d , a data rate λ_d , an end-to-end delay budget Δ_d , and a Service Function Chain (SFC) $S_d = (f_{d,1}, \dots, f_{d,K_d^S})$. Before reaching its SFC, every demand must traverse the O-RAN access chain composed of vDU and vCU. The complete chain of demand d is therefore

$$\mathcal{C}_d = (\text{vDU}, \text{vCU}, f_{d,1}, \dots, f_{d,K_d^S}). \quad (1)$$

The total number of stages in $\mathcal{C}_d$ is $K_d = |\mathcal{C}_d|$, indexed by $\mathcal{K}_d = \{1, \dots, K_d\}$, and $g_{d,k}$ denotes the function executed at stage k . The set of inter-stage hops is $\mathcal{H}_d = \{0, 1, \dots, K_d-1\}$: hop 0 carries traffic from the ingress node s_d to the vDU, and each subsequent hop carries traffic between two consecutive stages of the chain.

B. Constraints

An accepted demand must have each of its stages hosted on exactly one node, while a rejected one occupies nothing:

$$\sum_{n \in N} x_{d,k,n} = a_d, \quad \forall d \in D, \forall k \in \mathcal{K}_d. \quad (2)$$

A stage may only go to a node that is switched on and whose O-RAN layer supports its function, with $e_{f,n} = 1$ iff $\tau_n \in \Gamma_f$:

$$x_{d,k,n} \leq e_{g_{d,k},n}, \quad x_{d,k,n} \leq u_n, \quad \forall d, \forall k, \forall n. \quad (3)$$

Traffic must then reach the stages that process it. On hop 0, flow conservation carries the demand from its ingress s_d to stage 1, where $N^+(n)$ and $N^-(n)$ denote the successors and predecessors of n :

$$\sum_{j \in N^+(n)} r_{d,0,nj} - \sum_{i \in N^-(n)} r_{d,0,in} = a_d \beta_{d,n} - x_{d,1,n}, \quad \forall n \in N. \quad (4)$$

The same balance links every pair of consecutive stages, for $h \in \{1, \dots, K_d-1\}$:

$$\sum_{j \in N^+(n)} r_{d,h,nj} - \sum_{i \in N^-(n)} r_{d,h,in} = x_{d,h,n} - x_{d,h+1,n}, \quad \forall n \in N. \quad (5)$$

A node can host no more load than its processor supports, and only while it is active:

$$\eta_n = \sum_d \sum_k \lambda_d c_{g_{d,k}} x_{d,k,n}, \quad \eta_n \leq c_n u_n, \quad \forall n \in N. \quad (6)$$

Each function also has finite per-instance throughput, so the load it receives on a node bounds how many instances that node must run:

$$\sum_d \sum_{k: g_{d,k}=f} \lambda_d x_{d,k,n} \leq q_f m_{n,f}, \quad m_{n,f} \leq \bar{m}_{n,f} u_n, \quad \forall n, \forall f. \quad (7)$$

Links are likewise finite, and a rejected demand routes nothing:

$$\sum_d \sum_h \lambda_d r_{d,h,ij} \leq b_{ij}, \quad r_{d,h,ij} \leq a_d, \quad \forall (i, j) \in E. \quad (8)$$

The access chain carries the tightest budgets of the model: hops 0 and 1 must fit the fronthaul and midhaul limits, the Big-M term releasing them for a rejected demand:

$$\sum_{(i,j)} \ell_{ij} r_{d,0,ij} \leq \Delta^{FH} + M(1 - a_d), \quad \forall d, \quad (9)$$

$$\sum_{(i,j)} \ell_{ij} r_{d,1,ij} \leq \Delta^{MH} + M(1 - a_d), \quad \forall d. \quad (10)$$

Finally, the end-to-end delay accumulates propagation and processing along the whole chain, and must fit the demand's SLA whenever it is admitted:

$$L_d = \sum_{h,(i,j)} \ell_{ij} r_{d,h,ij} + \sum_{k,n} \delta_{g_d,k} x_{d,k,n}, \quad \forall d, \quad (11)$$

$$L_d \leq \Delta_d + M(1 - a_d), \quad \forall d. \quad (12)$$

C. Objective Function

The power consumption of node n combines an activation cost and a component proportional to its CPU load:

$$P_n = p_n^{idle} u_n + (p_n^{max} - p_n^{idle}) \frac{\eta_n}{c_n}, \quad \forall n \in N. \quad (13)$$

Minimizing power alone would be solved by rejecting every demand and powering all nodes down, so the model maximizes admission first and uses power only to separate solutions that admit equally many demands:

$$\min Z = -W \sum_{d \in D} a_d + C_E, \quad C_E = \frac{\sum_{n \in N} P_n}{\sum_{n \in N} p_n^{max}}. \quad (14)$$

As $C_E \in [0, 1]$, any $W > 1$ makes one further admitted demand outweigh the largest attainable power saving, so (14) realizes the lexicographic order $(\sum_d a_d, -C_E)$. The metric C_E is dimensionless, enabling comparisons across instances of different sizes; energy over a planning window follows by multiplying by its duration.

IV. GEOVA

GEOVA is a Genetic Algorithm (GA) [10] with a *batch* chromosome encoding: each individual jointly represents the SFC placement of all scenario demands. This lets the operators (i) reconsider early placements that would block later demands, improving acceptance, and (ii) raise the utilization of the nodes already switched on, amortizing the idle power p_n^{idle} in (13) over more admitted demands. The vDU/vCU access chain is not encoded: Eqs. (9)–(10) leave few feasible placements, so a greedy routine resolves them at each evaluation and the search focuses on the larger SFC space.

A. Representation and Fitness Function

Let $D = \{d_1, \dots, d_{|D|}\}$ be the ordered set of demands and S_{d_i} the service SFC of d_i . A chromosome is a tuple of per-demand blocks $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(|D|)})$, where each block $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_{|S_{d_i}|}^{(i)}) \in N^{|S_{d_i}|}$ and each gene $x_j^{(i)}$ identifies the candidate computational node to host the j -th stage of the SFC of demand d_i . The total chromosome length is therefore $\sum_i |S_{d_i}|$.

Algorithm 1 GEOVA — Batch Genetic Algorithm for energy-efficient SFC placement

Input: demands D , graph G , parameters $(\mu, G_{\max}, p_m, k_t, \varepsilon)$

Output: per-demand allocation

```

1:  $P \leftarrow \text{INITHYBRIDPOP}(D, G, \mu)$ 
2: evaluate  $\Phi(\mathbf{X})$  for each  $\mathbf{X} \in P$ 
3:  $\mathbf{X}^* \leftarrow \arg \max_{\mathbf{X} \in P} \Phi(\mathbf{X})$ 
4: for  $g = 1$  to  $G_{\max}$  do
5:    $P' \leftarrow \varepsilon$  best individuals of  $P$  ▷ elitism
6:   while  $|P'| < \mu$  do
7:      $p_1 \leftarrow \text{TOURNAMENT}(P, k_t)$ ;  $p_2 \leftarrow \text{TOURNAMENT}(P, k_t)$ 
8:      $(c_1, c_2) \leftarrow \text{BLOCKCROSSOVER}(p_1, p_2)$ 
9:      $\text{MUTATE}(c_1, p_m)$ ;  $\text{MUTATE}(c_2, p_m)$ 
10:     $P' \leftarrow P' \cup \{c_1, c_2\}$ 
11:   end while
12:    $P \leftarrow P'$ ; evaluate  $\Phi$  on  $P$ 
13:   if  $\max_{\mathbf{X} \in P} \Phi(\mathbf{X}) > \Phi(\mathbf{X}^*)$  then
14:      $\mathbf{X}^* \leftarrow \arg \max_{\mathbf{X} \in P} \Phi(\mathbf{X})$ 
15:   end if
16: end for
17: return decoded allocation from  $\mathbf{X}^*$ 
    
```

Fitness mirrors the objective in (14): the decoder may reject a demand because sequential construction makes it locally infeasible, so individuals are ranked by admission first and by power only under ties. Formally, fitness is the *lexicographic* pair

$$\Phi(\mathbf{X}) = (n_{ac}(\mathbf{X}), -C_E(\mathbf{X})), \quad (15)$$

where $n_{ac}(\mathbf{X})$ is the number of accepted demands and $C_E(\mathbf{X})$ the normalized consumption defined in (14). Thus $\Phi(\mathbf{X}_1) > \Phi(\mathbf{X}_2)$ holds when $n_{ac}(\mathbf{X}_1) > n_{ac}(\mathbf{X}_2)$, or, under ties, when $C_E(\mathbf{X}_1) < C_E(\mathbf{X}_2)$.

Decoding allocates demands sequentially on an initially empty state. For each d_i , the procedure greedily places the access chain on a temporary state, then iterates over the SFC stages reserving CPU (6), bandwidth on the shortest path (8), and latency compatible with Δ_{d_i} (12). Any violation discards the temporary state and rejects the demand ($a_{d_i} = 0$); otherwise it is committed ($a_{d_i} = 1$). After all demands, $n_{ac}(\mathbf{X})$ and $C_E(\mathbf{X})$ are computed.

B. Hybrid Initialization

The initial population of size μ combines four strategies for diversity: one pure greedy individual; $\approx 40\%$ *consolidators* that prefer already-active nodes (biasing toward low power); $\approx 30\%$ randomized greedy individuals (uniform sampling over the k best candidates per stage); and the remaining slots filled by greedy variants under aggressive mutation ($p_m = 0.25$). All strategies enforce layer eligibility and local constraints during construction.

C. Evolutionary Operators

Selection uses tournaments of size k_t . *Block crossover* inherits each per-demand block of the offspring from one

parent with probability 0.5, preserving entire SFC subsolutions. *Mutation* replaces each gene $x_j^{(i)}$ (with probability p_m) by a node drawn uniformly from the topologically closest neighbors of $x_{j-1}^{(i)}$ that are eligible for $g_{d_i,j}$, favoring low-latency perturbations. *Elitism* of ε individuals preserves the best chromosomes across generations.

V. EVALUATION SCENARIO

A. Topologies and Computational Resources

The POLSKA topology [12] comprises 12 nodes (2 *core*, 5 *edge*, 5 *far-edge*) and 18 bidirectional links. The POPS topology, derived from the RNP backbone [13], spans the Brazilian territory with 27 nodes (2 *core*, 7 *edge*, 18 *far-edge*) and 45 bidirectional links, providing greater topological slack.

All links operate at 400 Gbps; propagation delay follows the Haversine formula with tortuosity $1.3 \times$ [14], plus a +2 ms penalty on links incident to core nodes. We adopt $\Delta^{FH} = 0.25$ ms and a per-vCU aggregate bandwidth of ~ 16 Gbps; Δ^{MH} is not binding for the chain lengths evaluated and is left unconstrained. Table II reports hardware profiles per layer, with $p_n^{idle} = 0.6 p_n^{max}$.

TABLE II

COMPUTATIONAL PARAMETERS PER LAYER.

Parameter	Far-Edge	Edge	Core
CPU (c_n , cores)	40	128	512
p_n^{max} (W)	300	800	1500
vDU limit ($\bar{m}_{n,vDU}$)	8	20	0
Eligible service functions	no	yes	yes

B. Services and Demand Generation

Each service, described in Table III, is characterized by its SFC (CPU as a superscript), throughput λ_d , and SLA Δ_d . Every demand is preceded by the O-RAN access chain — vDU (4 cores, $\delta_f = 0.5$ ms) and vCU (1 core, $\delta_f = 0.25$ ms); all remaining VNFs have $\delta_f = 10$ ms. In each scenario, the service of each demand is sampled according to the *Proportion* column [14], and the ingress node s_d is drawn uniformly among *far-edge* and *edge* nodes.

C. Experimental Protocol and Metrics

For each pair (topology, $|D|$), where $|D| \in \{60, 70, 80, 90, 100\}$ is the number of demands per scenario, we generate 30 independent scenarios (seeds 0–29). GEOVA is compared against GREEDY (purely constructive) and GRASP [11] (constructive + restricted randomization + local search), sharing feasibility and power routines so that only the decision core differs. GEOVA parameters (Table IV) are used unchanged for both topologies and all loads. Varying $\mu \in \{20, 30, 50\}$, $G_{max} \in \{15, 25, 40\}$ and $p_m \in \{0.04, 0.08, 0.15\}$ at $|D| = 80$ shifts acceptance by at most 1.8 pp and unit power by 1.9 W, and G_{max} changes neither — the search converges before generation 15. No exact optimum is used as a reference: the MILP does not scale to these instances, and instances small enough to solve exactly stay uncongested, where every algorithm admits all

TABLE III
SERVICES AND CORRESPONDING SFCs (CPU^{cores}).

Service	SFC (VNF ^{CPU})	Δ_d (ms)	λ_d (Mbps)	Proportion (%)
Video Streaming	NAT ² →FW ⁸ →TM ¹ →VOC ² →IDS ⁸	100	40.0	69.9
Web Service	NAT ² →FW ⁸ →TM ¹ →WOC ² →IDS ⁸	500	1.0	18.2
VoIP	NAT ² →FW ⁸ →TM ¹ →FW ⁸ →NAT ²	100	0.64	11.8
Online Gaming	NAT ² →FW ⁸ →VOC ² →WOC ² →IDS ⁸	60	0.5	0.1

TABLE IV
GEOVA PARAMETERS.

Parameter	Symbol	Value
Population size	μ	30
Generations	G_{max}	25
Mutation rate	p_m	0.08
Tournament size	k_t	4
Elitism size	ε	3

demands and the comparison is uninformative about the saturation regime of interest.

We report averages over the 30 scenarios of: acceptance rate (%); normalized power C_E (14); power per accepted demand (W); average end-to-end latency (ms); and execution time (s). All algorithms ran in Python 3.8.17 on an Intel® Core™ i7-7700 at 3.60 GHz, 16 GB RAM, Linux Mint.

VI. EXPERIMENTAL RESULTS

Fig. 1 and Table V summarize POPS, whose topological slack leaves the search room to reorganize. Under low load ($|D| = 60$), GEOVA admits 13.9 percentage points (pp) more demands than GREEDY while drawing 16.9% less power per accepted demand (Fig. 2). Both metrics improve together, and not by switching nodes off: because the access chain needs a vDU close to every ingress, all three algorithms keep a nearly identical active set (25.1 nodes for GEOVA against 25.3 for GREEDY at $|D| = 60$). GEOVA instead serves 57.6 demands over that footprint against 49.3, amortizing the same idle power over more admitted demands. The joint gain narrows as load grows — +9.2 pp and −11.8% at $|D| = 70$, +4.5 pp and −6.6% at $|D| = 80$.

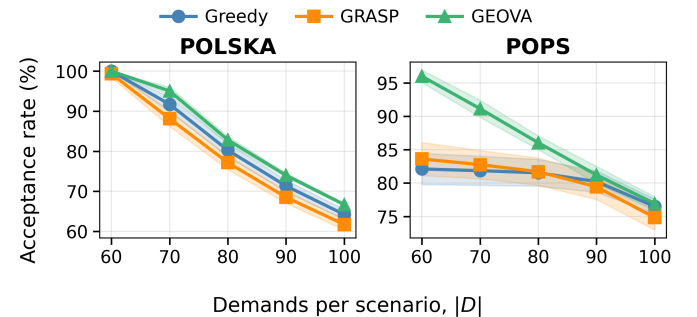


Fig. 1. Average acceptance rate ($\pm 95\%$ CI) on POLSKA (left) and POPS (right).

Under saturation the margin compresses. At $|D| = 90$ GEOVA still leads GREEDY by +1.0 pp (paired t -test over the 30 scenarios, $p = 0.03$); at $|D| = 100$ the two become statistically indistinguishable in acceptance (+0.4 pp, $p = 0.07$),

TABLE V

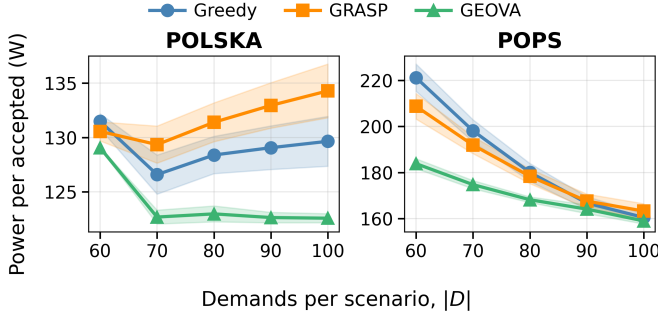
 PERFORMANCE ON POPS (27 NODES), AVERAGES OVER 30 SCENARIOS
PER $|D|$.

Algorithm	Acceptance (%)					Power per accepted (W)				
	60	70	80	90	100	60	70	80	90	100
GREEDY	82.1	81.9	81.5	80.2	76.5	221.2	198.1	180.1	166.8	160.5
GRASP	83.6	82.8	81.7	79.4	74.8	208.8	191.9	178.4	167.6	163.3
GEOVA	96.0	91.1	86.0	81.2	76.9	183.9	174.7	168.2	164.1	159.0

TABLE VI

 PERFORMANCE ON POLSKA (12 NODES), AVERAGES OVER 30
SCENARIOS PER $|D|$.

Algorithm	Acceptance (%)					Power per accepted (W)				
	60	70	80	90	100	60	70	80	90	100
GREEDY	100.0	91.7	80.2	71.3	64.2	131.5	126.6	128.4	129.1	129.7
GRASP	99.3	88.1	77.2	68.5	61.7	130.6	129.3	131.4	132.9	134.3
GEOVA	100.0	95.0	82.8	74.0	66.7	129.0	122.7	123.0	122.6	122.6


 Fig. 2. Power per accepted demand ($\pm$ 95% CI) on POLSKA (left) and POPS (right).

while GRASP falls significantly behind GEOVA (+2.1 pp, $p = 0.01$). Power per accepted demand remains the lowest for GEOVA at every $|D|$. The narrowing is expected rather than anomalous: as demand approaches capacity, few placements stay feasible and little is left for the evolutionary search to rearrange.

POLSKA saturates from $|D| = 70$ onward, and there the ranking is stable (Table VI): GEOVA leads GREEDY in acceptance at every load beyond $|D| = 60$, by +2.7 pp at $|D| = 90$ and +2.5 pp at $|D| = 100$ (both $p < 0.001$), and its power per accepted demand stays flat near 122.6 W while both baselines climb past 129 W. All 12 nodes stay active throughout, and from $|D| = 70$ the admitted count is essentially constant (66.5 for GEOVA, 64.2 for GREEDY), which is why the unit power flattens instead of falling monotonically. That the advantage persists on the smaller and tighter topology indicates it comes from how the search fills the active infrastructure, not from spare capacity.

No SLA violation occurred among the 24,000 demands generated, for any algorithm, since admission control refuses whatever cannot meet its budget. Per class, the worst delay GEOVA produced was 99.9 ms for Video Streaming and 99.4 ms for VoIP against their 100 ms budgets, and 142.4 ms for Web Service against 500 ms. Online Gaming, 0.1% of the mix, contributed 10 demands and none were admitted by any algorithm: its chain accrues 50.75 ms of fixed processing against a 60 ms budget, leaving 9.25 ms for propagation. Average latency stays at 52–54 ms on POLSKA and 61–67 ms on POPS, GEOVA running about 4 ms above GREEDY on POPS as a consequence of concentrating load. Execution time ranges from 0.03–0.09 s (GREEDY) to 5.5–9.3 s (GRASP) and 16–61 s (GEOVA), acceptable for the SMO planning cycle, which is triggered by traffic-matrix or topology changes rather than by per-request admission.

VII. CONCLUSION

This work presented GEOVA, a batch genetic algorithm for joint VNF placement and routing in O-RAN networks under SLA constraints. Across POLSKA and POPS, GEOVA improved acceptance and power per accepted demand at the same time — by up to +13.9 pp and −16.9% under low load — with the margin narrowing as the network saturates but never reversing. On the tighter POLSKA the advantage holds at every load (+2.5 pp at $|D| = 100$, $p < 0.001$), indicating that it stems from how the search fills the active infrastructure rather than from spare capacity. Future work will benchmark GEOVA against ant colony and particle swarm optimization, evaluate heavier traffic ($|D| > 100$), and extend the formulation to a multi-objective setting that jointly optimizes energy and latency.

REFERENCES

- [1] O-RAN Alliance, “O-RAN architecture description,” Technical Specification O-RAN.WG1.OAD, 2024.
- [2] M. Polese, L. Bonati, S. D’Oro, S. Basagni, and T. Melodia, “Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Communications Surveys & Tutorials*, 2023.
- [3] 3GPP, “Study on new radio access technology: Radio access architecture and interfaces,” TR 38.801, 2017.
- [4] ETSI, “Network Functions Virtualisation (NFV): Architectural framework,” ETSI GS NFV 002, 2014.
- [5] M. Klinkowski, “Optimized planning of DU/CU placement and flow routing in 5G packet xhaul networks,” *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 232–248, 2024.
- [6] K. D. R. Assis, R. C. Almeida Jr., A. F. Santos, A. Njanda, C. Natalino, and D. L. Cardoso, “A new path model for DU-CU placement in 5G Open Radio Access Network,” in *Proc. International Conference on Transparent Optical Networks (ICTON)*, 2025.
- [7] M. R. P. dos Santos, R. I. Tinini, and G. B. Figueiredo, “Energy-efficient flexible functional splitting in latency-constrained O-RAN,” *IEEE Transactions on Green Communications and Networking*, 2025.
- [8] R. Joda, T. Pamuklu, P. E. Iturria-Rivera, and M. Erol-Kantarci, “Deep reinforcement learning-based joint user association and CU-DU placement in O-RAN,” *IEEE Transactions on Network and Service Management*, 2022.
- [9] D. R. de Luna, F. M. Abinader Jr., P. R. B. da Silva, M. B. de Paiva, V. A. de Sousa Jr., and A. J. V. Neto, “Multi-objective optimization of Open RAN data center placement for enhanced 5G deployments,” in *Proc. IEEE NFV-SDN*, 2024.
- [10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [11] T. A. Feo and M. G. C. Resende, “Greedy randomized adaptive search procedures,” *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.
- [12] S. Orłowski, R. Wessäly, M. Pióro, and A. Tomaszewski, “SNDlib 1.0—Survivable network design library,” *Networks*, vol. 55, no. 3, pp. 276–286, 2010.
- [13] Rede Nacional de Ensino e Pesquisa (RNP), “Backbone RNP — map of points of presence,” 2024. [Online]. Available: <https://www.rnp.br/>.
- [14] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca, “Holu: Power-aware and delay-constrained VNF placement and chaining,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1524–1539, 2021.