CORA: CONSTRAINT-AWARE LEARNED IMPROVEMENT FOR VNF RECONFIGURATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Network function virtualization delivers services through chains of software functions. Overload-driven reconfiguration must satisfy coupled resource, bandwidth, queue-stability, and end-to-end delay constraints. Constructive heuristics and learned policies can leave residual inefficiency even after feasibility repair. We present Constraint-aware Optimization with Reconfiguration Actions (CORA), a reusable learned improvement heuristic for complete deployments. From a strictly feasible start admitted by bounded repair, a relational graph policy ranks move–swap actions under exact successor checks. Multi-start search returns the best feasible incumbent, ensuring non-degradation relative to the repaired start. Across 30 fat-tree-8 instances and three independently trained policies, CORA improves 96.6%, 96.4%, and 100% of paired feasible problem seeds from heuristic, evolutionary, and learned constructors, respectively. Mean objective changes are -0.01011, -0.02721, and -0.03319, with all 95% problem-seed bootstrap intervals below zero. CORA outperforms three classical searches under 25 complete objective evaluations, although equal-time results are mixed; all three replicas reach the conditional optimum on 20 enumerable instances.

1 INTRODUCTION

Network function virtualization (NFV) implements network services as ordered service function chains (SFCs) of software functions on commodity servers (ETSI Industry Specification Group, 2012; Mijumbi et al., 2016). Server failures, overload, and changing demand trigger migration of virtual network function instances (VNFIs). Reconfiguration must preserve CPU, memory, bandwidth, processing stability, end-to-end delay, and operational utilization limits. Because one VNFI may serve several SFCs, moving it simultaneously changes multiple routes and resource bottlenecks. This coupling makes exact optimization expensive and motivates heuristic and learned solvers (Moens & Turck, 2014; Bari et al., 2015; Luizelli et al., 2015; Li et al., 2021).

Constructive solvers determine a complete deployment through predefined rules or learned decisions. MSH-OR exemplifies the heuristic class through priority-based migration, whereas MAT-GNN combines graph encoding with autoregressive multi-agent construction (Li et al., 2021; Wen et al., 2022). Fast construction does not eliminate residual inefficiency: a feasible placement may admit a beneficial move or swap, while a learned proposal may require repair because prefix-wise checks do not guarantee all final-route constraints. Repair addresses violations, but need not optimize the recovered feasible deployment.

A second improvement stage can exploit information that partial construction does not expose: the complete routes, shared-instance interactions, and bottlenecks induced by all placement decisions. It can also preserve an existing constructor and fall back to its feasible output when a bounded search finds nothing better. The challenge is that a local edit may reroute several SFCs and violate a distant link or delay bound. Post-optimization must therefore combine useful search guidance with transition-level constraint validation.

We propose Constraint-aware Optimization with Reconfiguration Actions (CORA) as a provider-independent refinement mechanism for overload-driven migration. A bounded interface returns a strictly feasible start or reports failure. A relational graph policy then ranks move–swap candidates, an exact mask removes illegal successors, and multi-start search returns the best feasible incumbent. Learning determines search order without overriding feasibility or non-degradation. Across 30

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

frozen fat-tree-8 problem seeds and three independently trained policies, paired experiments show improvements for heuristic, evolutionary, and learned starts, with all mean-change confidence intervals below zero; matched-budget comparisons, ablations, and transfer tests further delimit this benefit.

Our contributions are threefold:

- We formulate strict-feasible refinement of complete VNF deployments through a bounded interface that separates start availability from quality.
- We introduce a graph-guided move–swap policy with exact feasibility masking, multi-start exploration, and best-incumbent return.
- We evaluate paired gains across three start classes, matched classical search, retrained component controls, topology transfer, and conditional small-instance optimality.

2 Related Work

VNF placement and reconfiguration. NFV couples placement, routing, and resource allocation (ETSI Industry Specification Group, 2012; Mijumbi et al., 2016; Herrera & Botero, 2016). Early optimization models enforce resource capacity and chain connectivity (Moens & Turck, 2014; Bari et al., 2015; Luizelli et al., 2015); Li et al. (2021) additionally models delay-aware migration, shared VNFIs, and overload thresholds. Learning-based approaches address placement (Solozabal et al., 2020), adaptive scheduling (Xu et al., 2020), and attention-guided reconfiguration (Setayesh & Wong, 2021; Tu et al., 2024). CORA targets residual improvement after complete deployment and common feasibility repair, rather than replacing these construction mechanisms.

Learning for constructive optimization. Pointer networks and attention models construct solutions sequentially (Vinyals et al., 2015; Bello et al., 2017; Kool et al., 2019); graph policies guide greedy construction and exact search (Khalil et al., 2017; Li et al., 2018; Gasse et al., 2019). Graph networks encode relational structure (Kipf & Welling, 2017; Gilmer et al., 2017; Battaglia et al., 2018), while multi-agent Transformers factorize coordinated decisions autoregressively (Wen et al., 2022). Our learned start provider follows this principle. Following broader evaluations of learning for optimization (Bengio et al., 2021; Mazyavkina et al., 2021), we assess refinement under controlled budgets and distribution changes.

Learned improvement heuristics. Tabu search, variable-neighborhood search, and iterated local search improve complete solutions (Glover, 1989; Hansen & Mladenović, 2001; Lourenço et al., 2003). Neural approaches learn rewrites, neighborhood choices, and iterative exchanges (Chen & Tian, 2019; Hottung & Tierney, 2020; Ma et al., 2021; Wu et al., 2022; Ma et al., 2023). The closest precedent is graph-guided improvement for job-shop scheduling (Zhang et al., 2024). CORA adopts complete-state search for VNF migration, where a single edit affects shared services and remote links. Exact successor validation handles this coupling; paired classical-search comparisons and retrained ablations measure the value added by learned ranking.

3 Preliminaries

3.1 Problem Definition

We study the delay-aware VNF migration problem of Li et al. (2021). The physical data-center network is an undirected graph $G^P = (\mathcal{U}, \mathcal{E})$, where $\mathcal{P} \subset \mathcal{U}$ contains physical machines (PMs) and the remaining nodes are switches. Each PM $p \in \mathcal{P}$ has CPU and memory capacities C_p^{cpu} and C_p^{mem}, and each physical link $e \in \mathcal{E}$ has bandwidth capacity C_e^{bw} and propagation delay d_e. The set of deployed VNF instances is $\mathcal{V}$. VNFI $i \in \mathcal{V}$ has a type, a host $x_i \in \mathcal{P}$, and fixed CPU and memory reservations counted once even when several SFCs share the instance.

The request set is $\mathcal{R}$. Request $r \in \mathcal{R}$ contains ingress s_r, egress t_r, an ordered type sequence $(f_{r,1}, \ldots, f_{r,L_r})$, bandwidth demand b_r, arrival rate λ_r, and maximum end-to-end delay D_r. A fixed assignment maps each required type to a compatible VNFI. Once placement x is known, the route of r concatenates shortest physical paths between s_r, the hosts of its assigned VNFIs in service order, and t_r. The integer $\rho_r^e(x) \geq 0$ counts traversals of link e, including repeated use by different

chain segments. Let $\mathcal{V}_q \subseteq \mathcal{V}$ denote the triggered VNFIs; all other placements remain fixed during refinement.

3.2 Feasible Deployment

The same final-state constraints validate repaired starts and search successors. VNFI reservations are aggregated on hosts, and routed traffic on links. For every SFC–VNFI pair (r, i), the arrival rate λ_r must remain below its allocated service rate $\mu_{r,i}$; each SFC's propagation and processing delay $D_r(x)$ must not exceed D_r.

Let $U_{\max}(x)$ denote the largest CPU, memory, or link utilization, measured relative to the corresponding physical capacity. The operational threshold $\beta \in (0, 1]$ is stricter than, or equal to, the physical limit. We use

$$\mathcal{F}_\beta = \{x : U_{\max}(x) \le \beta,\ \lambda_r < \mu_{r,i}\ \forall (r,i),\ D_r(x) \le D_r\ \forall r\}. \tag{1}$$

In the modeled overload setting, $\mathcal{F}_\beta$ certifies feasibility and trigger resolution: physical capacity alone is insufficient when utilization exceeds β. Appendix A gives the full constraint expressions.

3.3 Optimization Objective

Every candidate is compared with the same pre-migration deployment x^0. The term $\Delta_D(x; x^0)$ is the change in total delay of SFCs containing a triggered VNFI, normalized by their reference delay. The term $\Delta_L(x; x^0)$ is the normalized change in the mean variance of CPU, memory, and link utilization. The joint objective is

$$J(x; x^0) = \alpha_D \Delta_D(x; x^0) + \alpha_L \Delta_L(x; x^0), \qquad \alpha_D + \alpha_L = 1. \tag{2}$$

Lower values are better. A negative objective indicates a weighted improvement relative to x^0, not a negative physical delay or load. The fixed denominators and numerical safeguards are specified in Appendix A.

A provider supplies a complete deployment x_q and the triggered set $\mathcal{V}_q$. The bounded repair interface returns $\bar{x}_q \in \mathcal{F}_\beta$ or reports unavailability. Given an available start, the refinement target is

$$\min_{x \in \mathcal{N}_{H,R}(\bar{x}_q) \cap \mathcal{F}_\beta} J(x; x^0) \quad \text{s.t.} \quad J(x; x^0) \le J(\bar{x}_q; x^0). \tag{3}$$

Here $\mathcal{N}_{H,R}$ denotes placements visited by at most R trajectories of H steps each, including the start. CORA approximates refinement with a bounded search; its best-incumbent return rule enforces non-degradation without claiming exhaustive minimization of the reachable region.

4 Methodology

4.1 The Search Procedure as an MDP

We formulate finite-horizon improvement using a complete deployment, since each edit depends on routes and loads induced by earlier decisions. The controller state s_t contains $x_t \in \mathcal{F}_\beta$, the fixed reference x^0, triggered set $\mathcal{V}_q$, best objective so far, visited placements, and remaining steps. These variables make reward and cycle exclusion well defined. The policy observes the deployment graph and candidate features, not all controller memory. Complete placements enable exact successor validation.

The raw action neighborhood contains three operation types:

$$\widetilde{\mathcal{A}}(x_t) = \{\mathrm{move}(i,p)\} \cup \{\mathrm{swap}(i,j)\} \cup \{\mathrm{stay}\}, \tag{4}$$

where $i, j \in \mathcal{V}_q$ and $p \in \mathcal{P}$. A move relocates one triggered VNFI to another PM. A swap exchanges the hosts of two triggered VNFIs and can escape a one-move local optimum when neither destination has enough spare capacity for an independent move. Stay is an implicit stop/fallback, not a learned logit. The implementation ranks move–swap actions and stops when no unseen legal successor remains. To bound online cost, move destinations are shortlisted by residual capacity and topological distance; the shortlist never certifies feasibility.

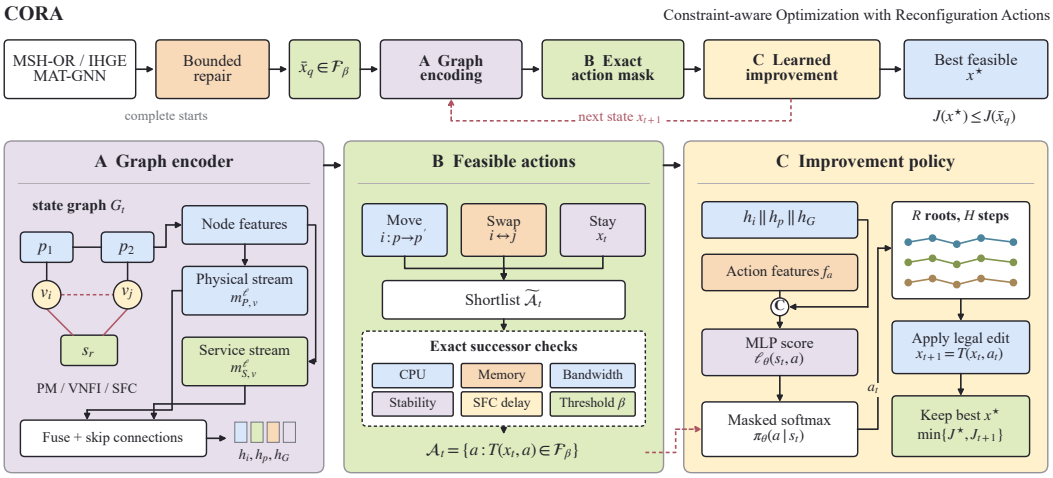

Figure 1: CORA's provider-independent improvement pipeline. A complete MSH-OR, IHGE, or MAT-GNN deployment first enters a common bounded interface that admits a strictly feasible start or reports failure. Panel A embeds the heterogeneous deployment graph; Panel B constructs move–swap candidates, with stay as an implicit fallback, and removes every illegal successor by exact constraint checks; Panel C ranks the remaining actions, explores multiple trajectories, and returns the best feasible incumbent. Learning controls search order but cannot bypass the feasibility filter or the non-degradation rule.

Transition $T(x_t, a)$ applies the edit and recomputes affected routes. The mask

$$m_t(a) = \mathbb{I}[T(x_t, a) \in \mathcal{F}_\beta] \tag{5}$$

defines the legal action set $\mathcal{A}_t = \{a \in \widetilde{\mathcal{A}}(x_t) : m_t(a) = 1\}$. Cycle exclusion further restricts this set before ranking; it never relaxes feasibility. Consequently $x_t \in \mathcal{F}_\beta$ and $a_t \in \mathcal{A}_t$ imply $x_{t+1} = T(x_t, a_t) \in \mathcal{F}_\beta$. Feasibility is therefore an invariant of search rather than a reward-shaping preference.

Following learned improvement heuristics, the reward combines stepwise progress with a bonus for improving the incumbent. Let $J_t = J(x_t; x^0)$ and $J_t^\star = \min_{\tau \le t} J_\tau$. We use

$$r_t = (J_t - J_{t+1}) + \eta \max(J_t^\star - J_{t+1}, 0), \tag{6}$$

where $\eta = 0.5$. Legal worsening steps are allowed to explore beyond a local minimum; only the returned incumbent is protected. The returned state is

$$x_q' = \arg \min_{x \in \{\bar{x}_q\} \cup \mathcal{X}_{\text{visited}}} J(x; x^0). \tag{7}$$

Here $\mathcal{X}_{\text{visited}}$ contains at most HR successors across all restarts. Because $\bar{x}_q$ is included, Eq. (7) gives the non-degradation guarantee in Eq. (3).

4.2 Policy Module

Graph embedding. We use a graph representation because the effect of a migration is relational: a VNFI consumes resources on one PM, shares traffic with several SFCs, and changes paths over a physical topology. The heterogeneous state graph $G_t = (\mathcal{N}, \mathcal{E}_P, \mathcal{E}_S)$ contains PM, VNFI, and SFC nodes. Physical-context edges $\mathcal{E}_P$ connect each VNFI to its host and each PM to its three nearest PMs by path delay; these are not literal switch-level links. Service edges $\mathcal{E}_S$ connect SFCs to assigned VNFIs and consecutive VNFIs within each chain. Shared VNFIs connect multiple SFC nodes. Bidirectional edges and normalized type, resource, traffic, and delay features provide the policy context.

Separate message channels preserve physical and service relations. With $h_v^0 = \sigma(W_0 z_v + b_0)$ and ReLU σ, layer ℓ computes

$$m_{P,v}^{\ell} = \sigma\big(W_P^{\ell}\,\mathrm{Mean}_{u:(u,v)\in\mathcal{E}_P}\, h_u^{\ell-1} + b_P^{\ell}\big)\,, \tag{8}$$

$$m_{S,v}^{\ell} = \sigma\big(W_S^{\ell}\,\mathrm{Mean}_{u:(u,v)\in\mathcal{E}_S}\, h_u^{\ell-1} + b_S^{\ell}\big)\,, \tag{9}$$

$$h_v^{\ell} = \sigma\big(W_C^{\ell}[m_{P,v}^{\ell} \| m_{S,v}^{\ell}] + b_C^{\ell}\big)\,. \tag{10}$$

The final node embedding sums skip-connected layer outputs, $h_v = \sum_{\ell=0}^{L} h_v^{\ell}$, and the graph context is $h_G = |\mathcal{N}|^{-1}\sum_v h_v$. The streams distinguish shared infrastructure from shared service. Section 5.4 tests their empirical contribution rather than assuming both are necessary.

Action selection. Each legal move or swap receives features f_a: projected host utilization, traffic, distance, delay change, incident-link load, source utilization, and a swap indicator. Swaps account for both directions. Four further features record the action-induced changes in CPU-, memory-, and link-utilization variance and in normalized affected-SFC delay. They are computed from the current state and candidate transition; they contain objective-relevant information and incur computation, but do not query the fixed-reference teacher label. The logit is

$$\ell_\theta(s_t, a) = \mathrm{MLP}_\theta([h_i \| h_{\mathrm{target}(a)} \| h_G \| f_a]) + b(a). \tag{11}$$

The fixed prior $b(a)$ favors low utilization, short moves, and reduced delay (Appendix C). The target is a PM for moves or a VNFI for swaps. Masking precedes normalization:

$$\pi_\theta(a \mid s_t) = \frac{m_t(a)\exp(\ell_\theta(s_t,a)/\tau)}{\sum_{a'\in\widetilde{\mathcal{A}}_t^{\mathrm{unseen}}} m_t(a')\exp(\ell_\theta(s_t,a')/\tau)}. \tag{12}$$

Here $\widetilde{\mathcal{A}}_t^{\mathrm{unseen}}$ contains shortlisted move–swap actions after cycle exclusion; $\tau = 1$ in training. Restart r takes root rank r (clamped to the last available rank), then acts greedily. Each trajectory stops when no unseen legal successor remains. Learning orders exploration; the simulator determines legality.

4.3 Policy Training

The policy is trained on feasible development placements using buffered REINFORCE-style updates. A buffer $\mathcal{B} = [u, v)$ contains at most $n = 4$ transitions. For $t \in \mathcal{B}$, its truncated return is

$$G_t^{\mathcal{B}} = \sum_{k=0}^{v-t-1} \gamma^k r_{t+k}, \qquad \gamma = 0.99. \tag{13}$$

There is no learned critic or bootstrapped tail. Before each update, a scalar baseline is updated as $b \leftarrow 0.95b + 0.05\,\mathrm{Mean}_{t\in\mathcal{B}}\, G_t^{\mathcal{B}}$, starting from zero. With $A_t = G_t^{\mathcal{B}} - b$, the loss is

$$\mathcal{L}_\pi(\theta) = -\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\left[\log\pi_\theta(a_t \mid s_t)\,\mathrm{stopgrad}(A_t) + \lambda_{\mathrm{ent}}\mathcal{H}(\pi_\theta(\cdot \mid s_t))\right]. \tag{14}$$

Here $\lambda_{\mathrm{ent}} = 10^{-3}$. This practical surrogate follows policy-gradient training (Williams, 1992; Sutton et al., 2000); truncation and the current-buffer baseline mean it is not an unbiased full-horizon gradient estimator. We evaluate its utility empirically.

To sharpen distinctions among legal actions, we add a development-only ranking teacher during early training. At a labeled state, the teacher evaluates every shortlisted legal one-step successor and identifies the best improving action a^+. For hard negatives $a^- \in \mathcal{H}_t$, we add

$$\mathcal{L}_{\mathrm{rank}} = \frac{1}{|\mathcal{H}_t|}\sum_{a^-\in\mathcal{H}_t}\mathrm{softplus}\big(\ell_\theta(s_t,a^-) - \ell_\theta(s_t,a^+) + m(a^-)\big)\,, \tag{15}$$

where the margin $m(a^-)$ is the normalized one-step objective gap. The ranking weight decays over 240 informative labeled states. Each labeled state also receives three replay updates; later episodes use only the policy loss. Teacher labels are never queried at inference. Appendix B specifies the update schedule, and Appendix H audits ranking.

The policy scores legal candidates after graph encoding and fully evaluates only the selected successor. Candidate features still compute local delay and variance changes. Thus, fewer complete objective calls do not imply an equal reduction in arithmetic work; Section 5.3 reports both budgets.

4.4 Incremental Feasibility Evaluation

Exact masking exploits local changes. For $i : p \to q$, CPU and memory totals change only on p, q; affected requests are $\mathcal{R}(a) = \{r : i \in r\}$. A swap uses the union for both VNFIs. We remove old routed traffic, add new traffic, and check each affected SFC's full delay. Unchanged constraints retain their validated values.

Let $\Delta u_p^c(a)$ and $\Delta u_p^m(a)$ denote compute deltas and $\Delta u_e^b(a)$ the routed bandwidth delta. A candidate is retained iff

$$u_p^c + \Delta u_p^c(a) \le \beta C_p^{\mathrm{cpu}}, \qquad u_p^m + \Delta u_p^m(a) \le \beta C_p^{\mathrm{mem}}, \tag{16}$$

$$u_e^b + \Delta u_e^b(a) \le \beta C_e^{\mathrm{bw}}, \qquad D_r(T(x_t, a)) \le D_r, \tag{17}$$

for every changed PM, link, and affected request. Processing stability in Eq. (20) is inherited because allocated service rates are placement-invariant in this simulator. These are the modeled constraints evaluated incrementally, not learned approximations. If the current state is not strictly feasible, the checker falls back to full validation. Admitted CORA states remain in $\mathcal{F}_\beta$, so their successors can use the incremental checker.

Constraint checking touches changed resources and affected routes rather than every SFC. Horizon H, restarts R, and candidate limit K control online cost; Appendix D gives the full complexity analysis.

5 Experiments

5.1 Experimental Setup

We evaluate paired improvement, budget efficiency, component contributions, and transfer. The primary suite comprises 30 frozen fat-tree-8 problem seeds: ten each for 90 SFCs/20 VNFIs, 60 SFCs/30 VNFIs, and 30 SFCs/40 VNFIs. MSH-OR, IHGE, and MAT-GNN instantiate heuristic, evolutionary, and learned start generators. Their outputs use the same bounded repair interface (candidate limit 8, beam width 3, depth 3, eight expansions). CORA uses horizon $H = 8$, three ranked-root restarts, candidate limit $K = 24$, $\alpha_D = \alpha_L = 0.5$, and $\beta = 0.60$. Three independently trained policy replicas are evaluated without test-set selection. The generator models overload, not physical server removal, with a shared ingress/egress and fixed SFC–VNFI assignments (Appendix C). All online timings use an Intel Core i5-1135G7 CPU; training details are in Appendix C.

For each available strict-feasible start, we compute $\Delta J_{q,s} = J(x'_{q,s}; x_s^0) - J(\bar{x}_{q,s}; x_s^0)$. We average available replicas within each problem seed before reporting means and 95% confidence intervals from 10,000 seed-bootstrap resamples (Efron, 1979). Improvement means $\Delta J < -10^{-10}$. Availability instead counts all scheduled provider–policy rows, including failures. Each provider's objective analysis uses its own paired feasible subset; it is not a ranking of providers on different instances. Matched-search, ranking, and component studies reuse the first five seeds of each regime; transfer and exact audits use separate seeds. Intervals quantify problem-instance variation conditional on the trained replicas, not independent training uncertainty.

5.2 Improvement Across Start Distributions

We test reusable refinement by applying CORA to each provider's identical repaired start. Table 1 separates quality from availability across three regimes; Figure 2 shows paired uncertainty.

The interface admits 96.7%, 93.3%, and 70.0% of the 90 scheduled rows for heuristic, evolutionary, and learned starts, respectively. These rows represent 29, 28, and 27 independent problem seeds. Conditional on admission, CORA improves 96.6%, 96.4%, and 100% of the represented seeds, with mean objective changes of -0.01011, -0.02721, and -0.03319. All nine regime–provider intervals and all three aggregate intervals lie below zero. No returned solution is worse than its repaired start, consistent with the incumbent-safe rule.

In the 30/40 regime, mean changes of -0.01137, -0.04325, and -0.05497 indicate more local headroom in evolutionary and learned starts. This supports reusable refinement, not a cross-provider ranking on unequal feasible subsets. Learned-start availability remains an end-to-end limitation. Table 1 reports paired start/total CPU time; Appendix Fig. 6 provides the operational breakdown.

Table 1: Paired refinement by SFC/VNFI regime (ten seeds, three policies each). n: paired feasible problem seeds; Improve: seeds with $\Delta J < -10^{-10}$. Means and bootstrap intervals use provider-specific paired subsets, not a common subset for ranking providers. MSH/IHGE/MAT denote heuristic/evolutionary/learned starts.

Regime	Start	n	J_{start}	J_{CORA}	ΔJ	Improve	95% CI for ΔJ
90/20	MSH	9	-0.0919	-0.0997	-0.0078	88.9%	[-0.0133, -0.0033]
	IHGE	9	-0.0875	-0.0995	-0.0120	88.9%	[-0.0187, -0.0062]
	MAT	9	-0.0643	-0.0808	-0.0165	100.0%	[-0.0224, -0.0114]
60/30	MSH	10	-0.1849	-0.1958	-0.0109	100.0%	[-0.0191, -0.0044]
	IHGE	9	-0.1519	-0.1765	-0.0246	100.0%	[-0.0370, -0.0136]
	MAT	9	-0.1356	-0.1637	-0.0281	100.0%	[-0.0376, -0.0186]
30/40	MSH	10	-0.2029	-0.2143	-0.0114	100.0%	[-0.0170, -0.0061]
	IHGE	10	-0.1340	-0.1773	-0.0432	100.0%	[-0.0511, -0.0345]
	MAT	9	-0.1298	-0.1848	-0.0550	100.0%	[-0.0619, -0.0481]
Overall	MSH	29	-0.1623	-0.1724	-0.0101	96.6%	[-0.0140, -0.0067]
	IHGE	28	-0.1248	-0.1520	-0.0272	96.4%	[-0.0344, -0.0205]
	MAT	27	-0.1099	-0.1431	-0.0332	100.0%	[-0.0407, -0.0258]

Start	Available rows	Paired seeds	Start time (s)	Total time (s)	Worse
MSH	96.7%	29/30	3.44	7.78	0/ 29
IHGE	93.3%	28/30	12.82	16.79	0/ 28
MAT	70.0%	27/30	1.41	4.62	0/ 27

Start time includes construction and bounded repair; total time additionally includes CORA. Both times average the same strict-feasible replica pairs within problem seed. Regime denotes SFCs/VNFIs.

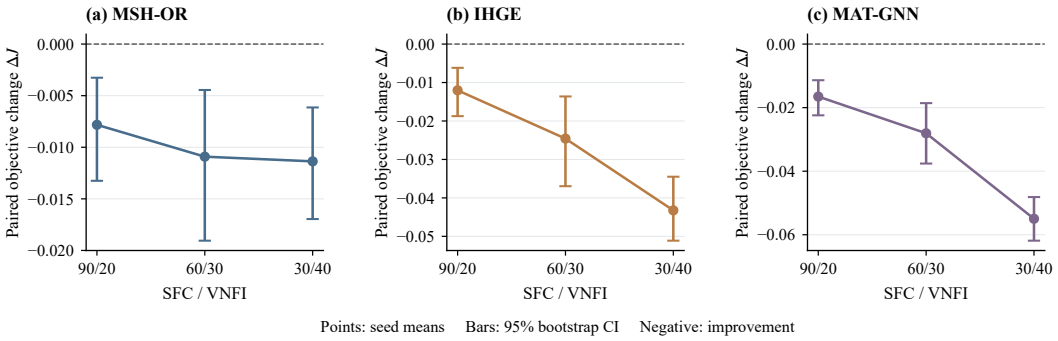

Figure 2: Paired refinement by start class and migration regime. Each point is the mean within-provider objective change; bars are 95% problem-seed bootstrap intervals. All intervals exclude zero. Starts are identical before and after CORA; cross-provider heights do not compare final deployment quality.

5.3 Comparison Under Matched Search Budgets

On 41 feasible starts from 15 scheduled problem seeds, using policy 97001, we compare learned ordering with additional classical search. First improvement, best improvement, and tabu receive identical starts, neighborhoods, objectives, and constraint checks. We match either 25 complete objective calls, including the start, or the CPU time consumed by CORA on that start. Both controls are needed: neural scoring, exact constraint checks, and incremental action features cost time but are not complete objective calls. Appendix L specifies the stopping and tabu rules.

Under 25 evaluations, CORA achieves mean ΔJ of -0.00960, -0.02583, and -0.03627 for heuristic, evolutionary, and learned starts. The strongest classical values are -0.00438, -0.01310, and -0.01707. Every provider–baseline paired advantage interval excludes zero (Figure 3a,b). Thus, the learned ordering extracts more improvement from this restricted complete-evaluation budget.

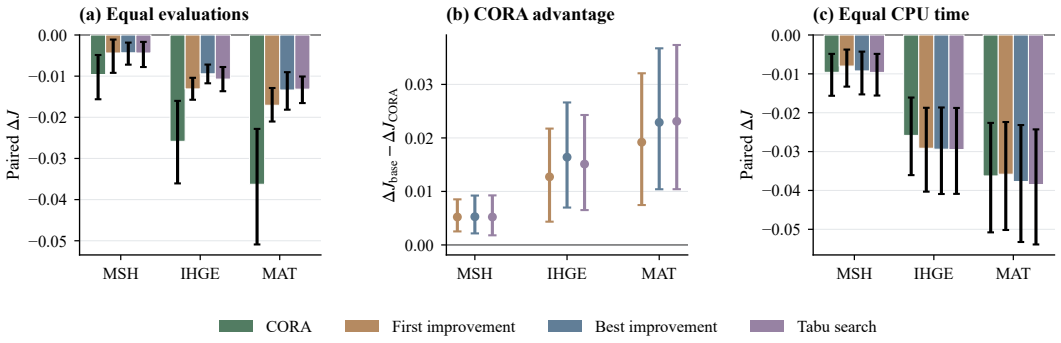

Figure 3: Classical-search comparison from identical strict-feasible starts. (a) Objective change under 25 complete evaluations. (b) Paired advantage of CORA over each baseline; positive favors CORA. (c) Objective change under matched wall-clock time. Error bars are 95% problem-seed bootstrap intervals.

The wall-clock result is mixed (Figure 3c). On heuristic starts, CORA improves over first improvement, but paired intervals against best improvement and tabu include zero. On evolutionary starts it trails the classical methods by approximately 0.0033–0.0036; on learned starts neither descent comparison excludes zero, but CORA trails tabu by 0.00221. The conclusion is therefore evaluation efficiency, not universal speed superiority. A practical deployment must weigh the cost of exact evaluation against graph encoding and candidate scoring.

5.4 Component Ablation

We isolate representation, supervision, neighborhoods, restarts, and incumbent protection on 41 feasible provider starts from 14 usable problem seeds among 15 scheduled seeds. Representation and ranking-loss variants are retrained for 160 episodes with three policy seeds; search-only controls reuse the full policy. Averaging replicas and providers within seed controls start difficulty without treating correlated rows as independent observations.

Removing ranking supervision changes mean ΔJ from -0.02371 to -0.00508; the paired loss is 0.01863 with 95% interval $[0.01061, 0.02749]$. Restricting the neighborhood to moves loses 0.01147 $[0.00751, 0.01563]$, showing that learned ordering and swaps contribute complementary gains (Figure 4a,b). Three ranked roots improve over one greedy root by 0.00112 $[0.00060, 0.00168]$. For physical-only and service-only encoders, both paired difference intervals include zero. These tests do not establish an advantage for the dual-stream encoder, nor do they prove equivalence.

Safety controls test a different property from average quality. Across starts, the mean rejected-action fraction is 78.1%, and the unmasked top-ranked action is illegal in 43.1% of start–policy decisions. Removing best-incumbent return causes one of 14 problem seeds to worsen, whereas full CORA has no worsening. These results support separating learned preferences from explicit feasibility and return guarantees. The breadth sweep also reveals a cost trade-off: $K = 8, 24, 48$ achieves mean changes of -0.02040, -0.02371, -0.02675 in $2.87, 4.91, 8.09$ seconds, respectively (Figure 4c). Appendix I contains all twelve variants and safety plots; Appendix H directly audits action-ranking quality.

5.5 Topology Transfer and Scale Generalization

To test reuse beyond the training topology and size, we evaluate without retraining on fat-tree-4, VL2, BCube, ordinary fat-tree-8, and a 120-SFC fat-tree-8 family. Each has five problem seeds and heuristic starts passed through the same interface. This tests topology transfer, not simultaneous transfer of topology and provider distribution.

Mean paired changes are -0.05706, -0.03716, -0.15002, -0.00918, and -0.00459, respectively; all five seed-bootstrap intervals lie below zero (Figure 5a). Availability is 100% except for ordinary fat-tree-8, where four of five seeds are admitted. Every available seed improves except one tie in the

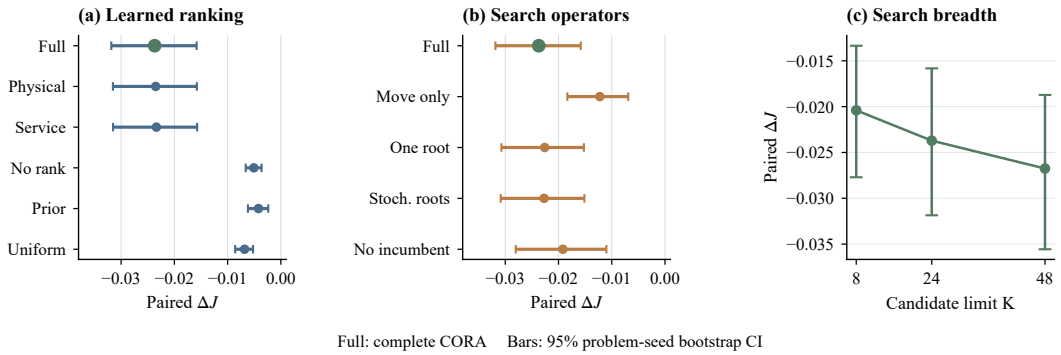

Figure 4: CORA component ablation. (a) Relation and ranking controls. (b) Neighborhood, restart, and incumbent mechanisms. (c) shortlist-size sensitivity. Points are problem-seed macro means; bars are 95% bootstrap intervals. Lower is better.

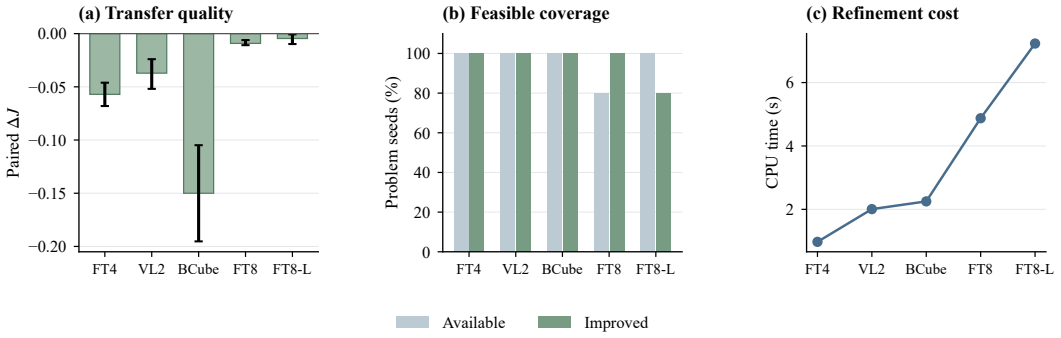

Figure 5: Zero-shot topology and scale transfer of three fat-tree-8-trained policies. (a) Paired objective change with 95% seed-bootstrap intervals. (b) strict-start availability and conditional improvement rate. (c) CPU refinement time. The larger family contains 120 SFCs.

larger family, giving 80% improvement there. Mean refinement time increases from 0.97 seconds on fat-tree-4 to 7.23 seconds on the larger family (Figure 5b,c).

Transfer holds on these synthetic families, conditional on successful repair; five seeds per family do not establish production robustness. Gains also depend on initial quality and normalization. Complementary exhaustive enumeration in Appendix J finds all three replicas conditionally optimal on 20 small instances, without establishing global optimality at larger scales.

6 Conclusion

Constraint-aware Optimization with Reconfiguration Actions separates deployment construction from learned refinement. Exact masking preserves modeled feasibility, while incumbent retention prevents degradation from the repaired start. Paired evaluations support improvement across three start classes, effective ranking under limited complete evaluations, and transfer within the tested families. These results establish reusable refinement, not universal runtime superiority or production safety; operator-trace validation and more efficient implementations remain important next steps.

AI USE STATEMENT

Generative AI tools were used solely to assist with language polishing, improving readability, and revising selected portions of the manuscript. These tools were not used to generate data, develop the proposed method or its conceptual framework, formulate or prove mathematical results, propose hypotheses, design experiments, implement methods, analyze data, or interpret experimental findings. All AI-assisted revisions were carefully reviewed and verified by the authors. The authors assume full responsibility for the final content of this work, including all text, claims, and conclusions.

ETHICS STATEMENT

This study evaluates VNF reconfiguration on synthetic network instances, without using human-subject data or production traffic. CORA incorporates explicit feasibility checks and best-incumbent retention to preserve the modeled constraints and prevent objective degradation relative to a feasible starting deployment. These safeguards support constraint-aware network optimization. Practical deployment should complement them with operator-led validation of migration dynamics, service continuity, and controller reliability.

REPRODUCIBILITY STATEMENT

The accompanying package contains frozen result records and plotting scripts; these reproduce reported summaries but do not constitute a full training release. Appendix C reports hardware, model settings, and problem seeds; Algorithms 1 and 2 specify training and inference. Paired aggregation, failed-start accounting, and claim boundaries are stated explicitly to distinguish reproducible measurements from broader deployment assumptions.

REFERENCES

Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. On orchestrating virtual network functions. In *11th International Conference on Network and Service Management*, pp. 50–56, 2015. doi: 10.1109/CNSM.2015.7367338.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2017.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. doi: 10.1016/j.ejor.2020.07.063.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979.

ETSI Industry Specification Group. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. Technical report, European Telecommunications Standards Institute, 2012.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, volume 70, pp. 1263–1272, 2017.

Fred Glover. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

Jhonatan G. Herrera and Juan F. Botero. Resource allocation in NFV: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016. doi: 10.1109/TNSM.2016.2598420.

André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *European Conference on Artificial Intelligence*, pp. 443–450, 2020.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Biyi Li, Bo Cheng, Xuan Liu, Meng Wang, Yi Yue, and Junliang Chen. Joint resource optimization and delay-aware virtual network function migration in data center networks. *IEEE Transactions on Network and Service Management*, 18(3):2960–2974, 2021. doi: 10.1109/TNSM.2021.3067883.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated local search. In *Handbook of Metaheuristics*, pp. 320–353. Springer, 2003.

Marcelo Caggiani Luizelli, Leonardo Richter Bays, Luciana Salete Buriol, Marinho Pilla Barcellos, and Luciano Paschoal Gaspary. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *IFIP/IEEE International Symposium on Integrated Network Management*, pp. 98–106, 2015. doi: 10.1109/INM.2015.7140281.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021. doi: 10.1016/j.cor.2021.105400.

Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016. doi: 10.1109/COMST.2015.2477041.

Hendrik Moens and Filip De Turck. VNF-P: A model for efficient placement of virtualized network functions. In *10th International Conference on Network and Service Management*, pp. 418–423, 2014.

Mehdi Setayesh and Vincent W. S. Wong. Service function chain reconfiguration in 5g core networks using deep learning. In *IEEE Global Communications Conference*, 2021. doi: 10.1109/GLOBECOM46510.2021.9685869.

Ruben Solozabal, Josu Ceberio, Aitor Sanchoyerto, Luis Zabala, Bego Blanco, and Fidel Liberal. Virtual network function placement optimization with deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 38(2):292–303, 2020. doi: 10.1109/JSAC.2019.2959183.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

Huaqing Tu, Ziqiang Hua, Qi Xu, Tao Zou, Huifeng Zhang, Hongli Xu, and Zuqing Zhu. OptRec: An efficient DRL-based SFC reconfiguration optimization framework in programmable networks. In *IEEE International Conference on Communications*, 2024. doi: 10.1109/ICC51166.2024.10622172.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16509–16521, 2022.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33 (9):5057–5069, 2022. doi: 10.1109/TNNLS.2021.3068828.

Xiaolong Xu, Qiong Wu, Lingling Qi, Wanchun Dou, Shaohua Wan, and Zibin Zhang. Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning. *IEEE Access*, 8:164922–164935, 2020. doi: 10.1109/ACCESS.2020.3022038.

Cong Zhang, Zhiguang Cao, Wen Song, Yaoxin Wu, and Jie Zhang. Deep reinforcement learning guided improvement heuristic for job shop scheduling. In *International Conference on Learning Representations*, 2024.

Appendix of "CORA: Constraint-Aware Learned Improvement for VNF Reconfiguration"

A Detailed Constraint Model

This appendix expands the definitions used in Section 3. The constraints and normalization are unchanged; moving their separate expressions here keeps the main formulation focused on the optimization task. The definitions below follow the evaluated simulator, including repeated link traversals, fixed VM reservations, and per-request service allocations.

A.1 Capacity, stability, and delay constraints

The feasible set must be stated explicitly because it defines both the environment and the action mask. For every PM, placement must respect CPU and memory:

$$\sum_{i:x_i=p} c_i^{\mathrm{cpu}}(x) \le C_p^{\mathrm{cpu}}, \qquad \sum_{i:x_i=p} c_i^{\mathrm{mem}}(x) \le C_p^{\mathrm{mem}}, \qquad \forall p \in \mathcal{P}. \tag{18}$$

Here $c_i^{\mathrm{cpu}}(x)$ and $c_i^{\mathrm{mem}}(x)$ are the fixed type-dependent VM reservations; sharing does not multiply these reservations. Every physical link must satisfy

$$\sum_{r\in\mathcal{R}} b_r \rho_r^e(x) \le C_e^{\mathrm{bw}}, \qquad \forall e \in \mathcal{E}. \tag{19}$$

Let Q_i be the processing capacity of VNFI i, N_i its number of assigned SFCs, and $\delta_{r,i}$ the processing density for request r. The simulator allocates $\mu_{r,i} = Q_i/(N_i\delta_{r,i})$ packets per second. Its per-request queue approximation requires

$$\lambda_r < \mu_{r,i}, \qquad \forall r \in \mathcal{R},\ i \in r. \tag{20}$$

With propagation delays in milliseconds, end-to-end delay is

$$D_r(x) = \sum_{e\in\mathcal{E}} \rho_r^e(x) d_e + \sum_{i\in r} \frac{1000}{\mu_{r,i} - \lambda_r} \le D_r, \qquad \forall r \in \mathcal{R}. \tag{21}$$

Migration is triggered by an operational threshold $\beta \in (0, 1]$ that is no greater than the physical capacity. We require every CPU, memory, and link utilization after reconfiguration to remain below β:

$$U_{\max}(x) = \max\left\{ \max_p \frac{u_p^{\mathrm{cpu}}(x)}{C_p^{\mathrm{cpu}}}, \max_p \frac{u_p^{\mathrm{mem}}(x)}{C_p^{\mathrm{mem}}}, \max_e \frac{u_e^{\mathrm{bw}}(x)}{C_e^{\mathrm{bw}}} \right\} \le \beta. \tag{22}$$

We write $\mathcal{F}_\beta$ for the placements satisfying Eqs. (18)–(22). This distinction matters: a placement may be physically feasible at utilization below one but still fail to resolve the overload trigger at threshold β.

A.2 Normalization constants

We optimize the same delay–load trade-off throughout training and evaluation. Let x^0 be the pre-migration reference deployment and let $\mathcal{R}_q$ be the SFCs that contain a triggered VNFI. The normalized delay change is

$$\Delta_D(x; x^0) = \frac{\sum_{r\in\mathcal{R}_q} \left[D_r(x) - D_r(x^0)\right]}{\max\left(\left|\sum_{r\in\mathcal{R}_q} D_r(x^0)\right|, 1\right)}. \tag{23}$$

For CPU, memory, and link utilization vectors $u^c(x)$, $u^m(x)$, and $u^b(x)$, define

$$L(x) = \frac{1}{3}\left[\mathrm{Var}(u^c(x)) + \mathrm{Var}(u^m(x)) + \mathrm{Var}(u^b(x))\right], \tag{24}$$

and its normalized change $\Delta_L(x; x^0) = (L(x) - L(x^0))/\max(|L(x^0)|, 10^{-8})$.

These terms enter Eq. (2). The delay denominator is fixed by the original affected-SFC delays, and the load denominator is fixed by the original utilization variance. Neither denominator is recomputed from a candidate solution. Thus all successors of one instance are evaluated against the same reference, including after bounded repair.

B BUFFERED POLICY TRAINING

Algorithm 1 matches the buffered update in Eq. (13). Training and evaluation problem seeds are disjoint. Invalid or already visited successors never enter the categorical distribution. The final short buffer is updated rather than discarded.

Algorithm 1 Buffered training for the masked improvement policy

Require: development distribution $\mathcal{D}$, horizon H, buffer limit n
initialize policy θ, baseline $b = 0$, informative-label count $j = 0$
for each episode **do**
  sample $(x^0, \mathcal{V}_q)$; obtain a feasible start or skip the attempt
  $x_0 \leftarrow \bar{x}_q$; $J^\star \leftarrow J(x_0; x^0)$
  reset buffer, visited signatures, and episode replay records
  for $t = 0, \ldots, H-1$ **do**
    build G_t and shortlisted $\widetilde{\mathcal{A}}(x_t)$
    retain feasible, unvisited successors; stop if none remain
    if teacher budget permits, compute weighted ranking loss and save labels
    sample $a_t \sim \pi_\theta(\cdot \mid s_t)$ over legal actions
    $x_{t+1} \leftarrow T(x_t, a_t)$; compute r_t by Eq. (6)
    $J^\star \leftarrow \min(J^\star, J(x_{t+1}; x^0))$
    record successor signature and append transition to the buffer
    if buffer has n transitions, update b, θ and clear the buffer
  end for
  update from any remaining transitions using Eqs. (13)–(15)
  perform three ranking replay updates per saved informative state
end for

C HARDWARE AND CONFIGURATIONS

All reported online times were obtained on a laptop-class Intel Core i5-1135G7 CPU (four cores, eight logical processors) with 7.8 GB RAM. Neural inference used CPU. The primary policy has two message-passing layers, 48-dimensional hidden states, a 48–24–1 action head, candidate limit 24, horizon 8, and three deterministic restarts whose root actions are the first, second, and third policy-ranked legal actions. All later decisions are greedy. Provider training is separate from CORA training. Each CORA policy is trained for 160 completed episodes with horizon eight. A hard-negative ranking loss supplements policy gradients during the first 240 labeled states; subsequent updates are on-policy. The learned constructor uses 24 imitation episodes followed by 72 training episodes. Evaluation problem seeds are excluded from CORA training; trained policy seeds identify model replicas, not new independent problem instances.

The confirmatory problem-seed ranges are 269000–269009, 270000–270009, and 271000–271009 for the three main regimes. CORA policy seeds are 97001–97003; MAT-GNN provider replicas use seeds 95000–95002. The matched-search, component, and ranking audits reuse seeds 269000–269004, 270000–270004, and 271000–271004 from the primary suite. They are diagnostic controls, not independent replications of the main result. Exact-gap and transfer instances use separate seed ranges. Raw JSON files contain every provider, policy, and problem-seed row; tables aggregate policy replicas within problem seed before resampling.

Training configuration. Adam uses learning rate 3×10^{-4} and gradient clipping at norm one. Training uses $\gamma = 0.99$, $n = 4$, entropy weight 10^{-3}, and reward bonus $\eta = 0.5$. The initial ranking weight is 0.5 and decreases linearly with the number of informative labels used, up to 240. At most two states per episode are queried. The teacher selects up to seven highest-scoring nonoptimal actions; tied best successors use the highest current logit. Margins divide the objective gap by the larger of the candidate objective range and 10^{-3}, then clip to $[0, 1]$. Three replay updates reuse each informative state's labels at the end of that episode, without additional teacher evaluations. The training mixture contains fat-tree-8 problems with 30 or 60 SFCs and two or three VNFIs per

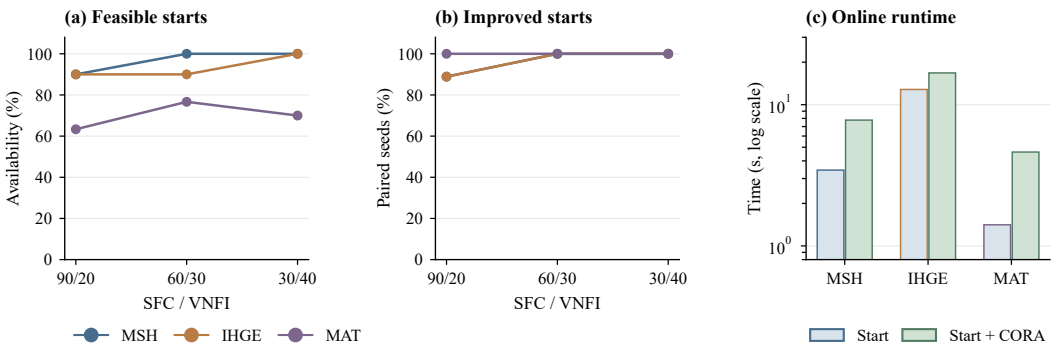

Figure 6: Operational profile. (a) Strict-start availability after bounded repair, (b) fraction of paired problem seeds improved by CORA, and (c) provider-plus-bridge versus end-to-end runtime on a logarithmic scale. Availability and objective quality are intentionally reported separately.

type. Starts are supplied by the heuristic constructor; failed starts are skipped and counted. Thus provider-independent inference does not imply training on all provider distributions.

Action prior and representation. The twelve-dimensional action vector starts with projected peak CPU and memory utilization on changed hosts, normalized traffic, normalized move distance, mean delay change normalized by SFC bounds, peak incident-link utilization at destination hosts, source utilization, and a swap indicator. Its last four entries are the variance changes and normalized delay change described in Section 4.2. With one-based feature indices, the fixed prior is $b(a) = -0.75f_{a,1} - 0.75f_{a,2} - 0.30f_{a,4} - 0.20f_{a,5}$. The graph has twelve-dimensional typed node features; duplicate directed relations produced by chain occurrences are retained in mean aggregation. Physical proximity uses a three-nearest-PM graph, while exact checks always use the original switch-level topology. The candidate limit is per triggered VNFI, not the total action count; swaps are additionally enumerated. Numerical feasibility checks use tolerances near 10^{-9}; "exact" means model-based rather than learned checking, not exact arithmetic.

Synthetic-instance construction. The generator uses ten function types and chain lengths of three to five. PM CPU and memory capacities are sampled uniformly from $[10, 30]$ and $[32, 64]$; type reservations from $[1, 2]$ and $[2, 4]$, respectively. Link capacities are $[500, 1000]$ Mbps and propagation delays $[2, 5]$ ms. Packet rates are $[1000, 4000]$ per second. All requests in one instance share one ingress/egress access switch. Requests are assigned to replicas by largest-rate-first balancing; initial placement packs nearby hosts subject to physical capacity. This is a controlled mapper, not a reproduction of an optimal mixed-integer initial placement.

Processing densities are calibrated to $\mu_{r,i} = 1.30\lambda_r$, so queue stability is satisfied and placement-invariant in these experiments. Deadlines initially drawn from $[50, 100]$ ms are increased when necessary to at least $1.20D_r(x^0)$. Bandwidth is initially sampled from $[0.5, B_{\max}]$ Mbps, where $B_{\max} = \max(0.5, \min(10, 0.9 \min_e C_e^{\mathrm{bw}}/(2|\mathcal{R}|)))$. If peak initial link utilization exceeds 0.9β, all bandwidth demands are scaled down together to that target. These choices create physically feasible but potentially CPU/memory-threshold-violating starts. They limit claims about real traffic, queue instability, link-failure recovery, and naturally occurring delay distributions.

D Computational Complexity Analysis

Let $N = |\mathcal{N}|$, $E = |\mathcal{E}_P| + |\mathcal{E}_S|$, hidden width d, total candidate count M, horizon H, and restart count R. Sparse mean message passing costs $O(Ed + Nd^2)$ per layer. The batched action head costs $O(Md^2)$. Incremental feasibility checking depends on changed hosts, affected requests, and their old/new route lengths; write this cost as C_Δ. Online search is therefore

$$O\big(HR(L(Ed + Nd^2) + Md^2 + MC_\Delta + C_G)\big)\,.$$

Here C_G accounts for state-feature assembly and the selected successor's evaluation; C_Δ includes candidate feasibility and local-feature costs. With $q = |\mathcal{V}_q|$ and per-VNFI limit K, $M \le qK + q(q-$

$1)/2$ before filtering. Graph assembly, sorting, and state-copy overhead remain relevant in the Python implementation; paths and resource totals can be cached. Exhaustive best-improvement also pays a complete objective evaluation for each legal successor, which is why we report both wall-clock time and objective-evaluation count.

E Incremental Feasibility Evaluation

The delta checker is exact under a feasible current placement. Any unchanged constraint remains satisfied. For changed PMs, resource deltas account for both removal and addition; this is necessary for swaps between unequal VNFIs. For each affected SFC, the checker reconstructs its entire ordered host list, recomputes end-to-end delay, removes bandwidth from every edge of the old concatenated path, and adds it to every edge of the new path. The final load of every changed edge is checked against both physical capacity and βC_e^{bw}. Queue stability is placement invariant in the current simulator and inherited from the validated start; a future placement-dependent allocation model would require new delta checks.

F Provider Details

MSH-OR prioritizes triggered VNFIs by service sharing and traffic, filters resource-ineligible destinations, normalizes delay and load terms, and commits the lowest weighted score. IHGE represents a complete plan as an individual, uses best-fit placement for initialization and repair, and applies roulette selection, adaptive crossover, and mutation. The main experiment limits IHGE to population 6 and four generations; this is a bounded provider configuration, not an equal-time comparison or a claim of evolutionary convergence.

MAT-GNN assigns one agent to each triggered VNFI. The heterogeneous encoder contextualizes agents, and a causal Transformer decoder predicts destinations autoregressively so later agents observe earlier decisions. Dynamic prefix-wise masks remove immediately illegal PM choices. The complete output still passes through the common bridge because several routes and utilization constraints become fully determined only after all agents have acted. This provider is included to expose CORA to a learned start distribution. It is an adaptation of MAT, not a claim to have introduced the original architecture.

The common interface checks the complete proposal first. If repair is needed, a policy-free bounded search may traverse threshold- or delay-violating intermediate states while retaining physical resource constraints. Only a fully validated output enters CORA. Failure means no admissible start was found within that repair budget, not that the instance is infeasible. Safety of CORA's feasible search does not apply to these internal repair states or to the physical transients of live migration.

G CORA PSEUDOCODE

Algorithm 2 CORA inference from a complete provider output

Require: reference x^0, provider output x_q, triggered set $\mathcal{V}_q$, policy π_θ, horizon H, restarts R

$(\bar{x}_q, \text{available}) \leftarrow \textsc{BoundedRepair}(x_q)$
if not available **then**
  return unavailable
end if
$x^\star \leftarrow \bar{x}_q$; $J^\star \leftarrow J(\bar{x}_q; x^0)$
for $r = 1, \dots, R$ **do**
  $x_0 \leftarrow \bar{x}_q$; $\mathcal{H} \leftarrow \{\text{sig}(x_0)\}$
  for $t = 0, \dots, H-1$ **do**
    build graph embeddings for x_t
    enumerate shortlisted move–swap actions
    $\mathcal{A}_t \leftarrow \{a : T(x_t, a) \in \mathcal{F}_\beta,\ \text{sig}(T(x_t, a)) \notin \mathcal{H}\}$
    if $\mathcal{A}_t = \varnothing$ **then**
      break
    end if
    if $t = 0$ **then**
      $a_t \leftarrow$ action of rank $\min(r, |\mathcal{A}_t|)$ in the legal root ordering
    else
      $a_t \leftarrow \arg\max_{a \in \mathcal{A}_t} \pi_\theta(a \mid s_t)$
    end if
    $x_{t+1} \leftarrow T(x_t, a_t)$; $\mathcal{H} \leftarrow \mathcal{H} \cup \{\text{sig}(x_{t+1})\}$
    if $J(x_{t+1}; x^0) < J^\star$ **then**
      $x^\star \leftarrow x_{t+1}$; $J^\star \leftarrow J(x_{t+1}; x^0)$
    end if
  end for
end for
return $x^\star$

H POLICY TRAINING AND RANKING AUDIT

The training logs retain attempted and completed episodes, reward, policy loss, ranking loss, and wall-clock duration for every model seed. Episodes for which the bounded interface cannot supply a strict-feasible start do not contribute a policy gradient. This accounting avoids treating repair failure as zero reward and mixing interface availability with action-ranking quality. Three policy replicas use training seeds 97001–97003; none is selected using test outcomes.

Figure 7 evaluates the intended mechanism directly on 41 frozen states. The 160-episode policy places an oracle-best action in its top five for 92.7% of states and cuts selected-action regret by 82.2% relative to the 24-episode reference policy. These are distinct training configurations, not two checkpoints of one controlled learning curve. The comparison audits the selected policies; it cannot isolate training duration alone. The oracle supplies labels only for this audit and the early development-state ranking loss; it is absent from deployment.

I COMPONENT ABLATION

We isolate the representation, ranking objective, neighborhood, restart rule, incumbent protection, and candidate limit under the same strict-feasible-start interface. Relation-stream and ranking-loss variants are retrained from scratch for 160 episodes with three independent policy seeds. Inference-only controls reuse the same starts and search horizon. Policy replicas are first averaged within a provider–problem row; the reported mean and bootstrap interval then use the problem seed as the sampling unit. The suite contains 41 provider-start rows from 14 usable problem seeds among 15 scheduled seeds.

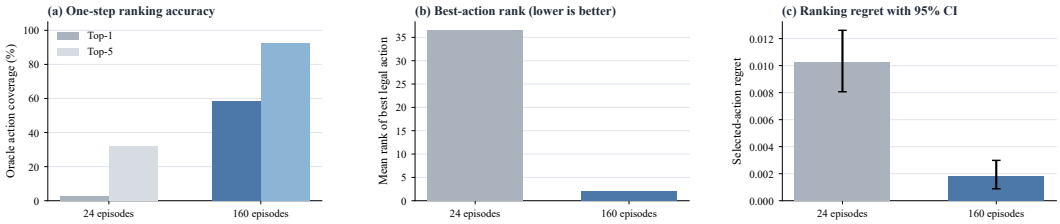

Figure 7: Mechanism audit on 41 strict-feasible held-out states. (a) Top-k inclusion of the oracle-best legal action. (b) Mean predicted rank. (c) regret of the policy's top-ranked action. Lower rank and regret are better. Training and audit seeds are disjoint, but the audit reuses primary test instances. The two policies differ in training configuration.

Table 2: Component ablation on identical bounded-repaired starts. Lower ΔJ is better. Intervals bootstrap independent problem seeds after averaging providers and policy replicas. Runtime is CPU refinement time per provider-start row.

Variant	Mean ΔJ [95% CI]	Improve (%)	Worse (%)	Time (s)
Full CORA	-0.02371 $[-0.03185, -0.01582]$	100.0	0.0	4.91
Topology only	-0.02351 $[-0.03154, -0.01578]$	100.0	0.0	4.87
Service only	-0.02338 $[-0.03152, -0.01573]$	100.0	0.0	4.91
No ranking loss	-0.00508 $[-0.00660, -0.00362]$	100.0	0.0	4.67
Local prior only	-0.00421 $[-0.00618, -0.00237]$	92.9	0.0	4.52
Uniform scores	-0.00682 $[-0.00858, -0.00523]$	100.0	0.0	2.02
Move only	-0.01224 $[-0.01834, -0.00690]$	92.9	0.0	3.06
Single root	-0.02260 $[-0.03072, -0.01523]$	100.0	0.0	1.67
Stochastic roots	-0.02271 $[-0.03086, -0.01514]$	100.0	0.0	4.73
No incumbent protection	-0.01918 $[-0.02803, -0.01101]$	92.9	7.1	4.90
$K = 8$	-0.02040 $[-0.02771, -0.01335]$	100.0	0.0	2.87
$K = 48$	-0.02675 $[-0.03556, -0.01872]$	100.0	0.0	8.09

Table 2 reports all twelve variants, including local-prior and uniform-score controls omitted from the main narrative. Each retains the same legal-action interface, so differences measure ranking or search behavior rather than relaxed feasibility. Figure 8 separates mask rejection, returned-solution degradation, and the candidate-limit cost trade-off. These controls support the specific interpretations in Section 5.4; they do not establish that every architectural component is independently necessary. The stochastic-restart control keeps the first trajectory greedy and samples all actions in later trajectories at temperature 0.8; it changes more than root selection alone. Uniform scores use the same stochastic-restart rule.

J CONDITIONAL OPTIMALITY AUDIT

For the small-instance audit, all placements of at most three triggered VNFIs are enumerated while all other placements remain fixed. Feasibility and the objective are computed by the same simulator used by every method. This supplies a known conditional optimum rather than a claimed optimum of the full problem. All three 160-episode replicas match the conditional optimum on every one of 20 held-out instances. Instances are retained only when one to three VNFIs are triggered and the bounded interface provides a feasible start. Thus this is a filtered tractable audit, not the success rate over all generated small cases. The result validates the search mechanism in a tractable regime but is not a proof of global optimality at fat-tree-8 scale.

K HELD-OUT TOPOLOGIES AND SCALES

The same fat-tree-8-trained graph policy is evaluated without retraining on fat-tree-4, VL2, BCube, ordinary fat-tree-8, and a larger fat-tree-8 family with 120 SFCs and approximately 40 VNFIs. Message passing accepts different graph sizes without padding because aggregation is permutation

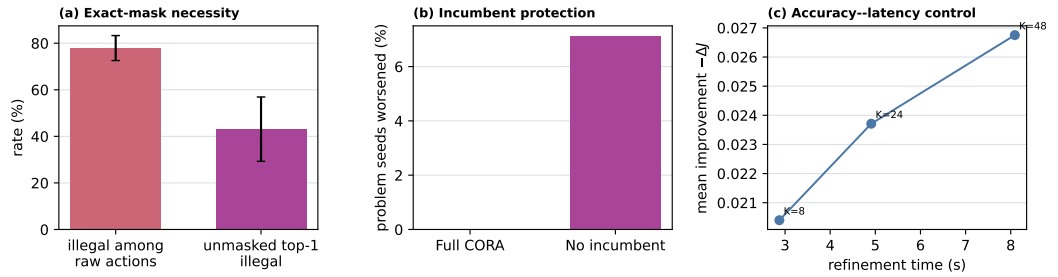

Figure 8: Safety and compute controls. (a) Fraction of raw actions rejected by exact masking (averaged within each start) and unmasked top-1 illegality (averaged over policy replicas and starts). (b) problem-seed worsening rate with and without best-incumbent return. (c) objective improvement versus CPU time as K varies.

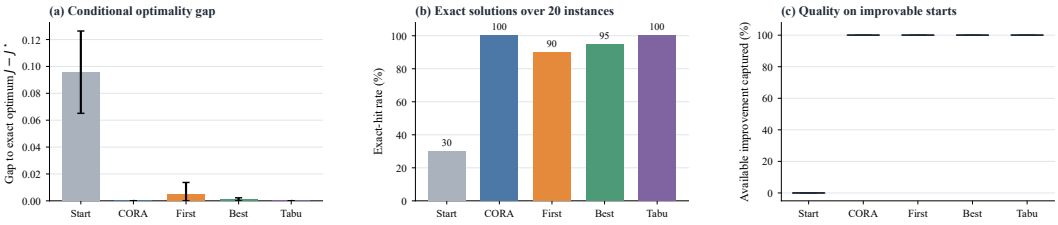

Figure 9: Exact conditional-optimality audit on 20 fat-tree-4 instances. (a) Absolute objective gap. (b) exact-hit rate. (c) fraction of available improvement captured on the 14 starts with a positive initial gap; the six already optimal starts are excluded from panel (c) only. Enumeration moves only triggered VNFIs; the global VNF placement problem remains combinatorial.

equivariant. The principal failure mode is bridge unavailability, not an illegal CORA transition. Figure 5 separates this availability from paired quality and runtime. Five seeds per family are sufficient to expose gross failures but not to establish universal topology generalization.

L MATCHED CLASSICAL SEARCH DETAILS

First improvement accepts the first improving legal action in a seeded random order. Best improvement chooses the lowest-objective legal successor. Tabu search selects the best admissible successor, forbids recently visited placement signatures for five iterations, and permits aspiration when a tabu successor improves the global incumbent. All methods share move generation, exact feasibility checks, objective code, and the initial placement.

The evaluation-matched comparison counts every complete objective call, including the initial state, and stops at 25. It shows statistically positive paired CORA advantages for every provider–baseline pair. In the time-matched comparison, classical methods may perform hundreds or thousands of objective calls during the measured CORA CPU time. The mixed outcome reported in Section 5.3 is consequently the correct claim boundary: the learned ordering is evaluation-efficient, while the unbatched Python prototype is not uniformly wall-clock superior. A scan interrupted by a budget returns the best candidate evaluated so far; best improvement need not finish its full neighborhood. Time checks occur between operations, so one non-preemptible candidate-generation or evaluation operation can overrun the deadline. Complete-call accounting excludes CORA's local objective-component features, which is why the call-count result is not an equal-information-oracle result.

M STATE TRANSITION EXAMPLE

Consider moving VNFI i from p_1 to p_3. The candidate is first rejected if CPU or memory on p_3 exceeds β. For every SFC sharing i, the old ingress–VNFI–egress path is removed from the link-load cache and the new path through p_3 is added. If one changed edge exceeds the bandwidth threshold or

one complete SFC delay exceeds its bound, the mask is zero and the policy never sees the action as selectable. If all checks pass, the graph is re-encoded after the transition because host, route, and utilization features have changed.

N LIMITATIONS

The study uses synthetic data-center instances rather than operator traces, and its largest reported family contains 120 SFCs on fat-tree-8. The bounded bridge can fail, most transfer families contain only five problem seeds, and the unbatched learned ranker does not dominate classical local search under every wall-clock comparison. The non-degradation guarantee is relative to the repaired start, not to the raw infeasible proposal. Exact optimality is shown only in the conditional small-instance space and does not imply a global optimum. Finally, single-machine CPU timings should not be extrapolated to a distributed controller without implementation profiling. The present controls do not isolate a feature-only MLP from the graph encoder, or policy-gradient training from ranking-only training; the separate value of those ingredients remains unresolved. Reusing primary seeds for auxiliary diagnostics does not provide independent replication, and the matched-search study evaluates only one CORA replica. Reported 95% intervals are marginal, not simultaneous multiple-comparison guarantees.