Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3769697.3771243

RESEARCH-ARTICLE

NeRM-Net: Reflective Evolution of Routing Strategies for Dynamic Communication Networks

SHUHAN GUO, Tsinghua University, Beijing, China

NAN YIN

SHUZHI LIU, Nanyang Technological University, Singapore City, Singapore

ZHONGHENG LIU, Tsinghua University, Beijing, China

WEI HUANGFU, University of Science and Technology Beijing, Beijing, China

QUANMING YAO, Tsinghua University, Beijing, China

Open Access Support provided by:

University of Science and Technology Beijing

Nanyang Technological University

Tsinghua University

PDF Download
3769697.3771243.pdf
17 March 2026
Total Citations: 0
Total Downloads: 102

Published: 08 December 2025

Citation in BibTeX format

CoNEXT '25:The 21st International Conference on emerging Networking EXperiments and Technologies
December 1 - 4, 2025
Hong Kong, Hong Kong

Conference Sponsors:
SIGCOMM

NeRM-Net: Reflective Evolution of Routing Strategies for Dynamic Communication Networks

Shuhan Guo
Department of Electronic Engineering
Tsinghua University
Beijing, China
guoshuhan@tsinghua.edu.cn

Nan Yin
Hong Kong Space Robotics and Energy Centre Limited
Hong Kong, China
srenany@ust.hk

Shuzhi Liu
College of Computing and Data Science
Nanyang Technological University
Singapore, Singapore
SHUZHI001@e.ntu.edu.sg

Zhongheng Liu
Department of Electronic Engineering
Tsinghua University
Beijing, China
zhonghen23@mails.tsinghua.edu.cn

Wei Huangfu
School of Computer & Communication Engineering
University of Science and Technology Beijing
Beijing, China
huangfuwei@ustb.edu.cn

Quanming Yao*
Department of Electronic Engineering, State Key laboratory of Space Network and Communications
Tsinghua University
Beijing, China
qyaoaa@tsinghua.edu.cn

Abstract

Modern networks demand routing strategies that jointly optimize heterogeneous constraints like bandwidth, latency, and compute resources. While LLM-driven algorithm generation excels in static domains, it struggles in dynamic network environments where these constraints are interdependent and topology-aware. We introduce NeRM-Net, a reflective co-evolution framework that jointly optimizes task specifications and routing algorithms through network-aware feedback. At its core, an adaptive environmental feedback module captures dynamic network states and performance outcomes. This module steers two complementary processes: (i) structure-conditioned prompt evolution, which aligns problem descriptions with graph-level constraints for more effective guidance, and (ii) feasibility-guided algorithm refinement, which evolves routing logic by analyzing constraint violations to improve solution quality. Experiments on Abilene and GEANT topologies show NeRM-Net consistently outperforms baselines in flow acceptance and resource efficiency, demonstrating its effectiveness and scalability.

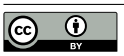
CCS Concepts

• **Networks** → **Network management**; • **Computing methodologies** → **Planning and scheduling**; • **Software and its engineering** → *Automatic programming*.

Keywords

Network strategy optimization; Resource-constrained routing; LLM-driven algorithm evolution

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
PN '25, Hong Kong, Hong Kong
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2243-1/2025/12
https://doi.org/10.1145/3769697.3771243

ACM Reference Format:
Shuhan Guo, Nan Yin, Shuzhi Liu, Zhongheng Liu, Wei Huangfu, and Quanming Yao. 2025. NeRM-Net: Reflective Evolution of Routing Strategies for Dynamic Communication Networks. In *Proceedings of the ACM Conext-2025 Workshop on the Polymorphic Network (PN '25), December 1–4, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3769697.3771243

1 Introduction

Modern communication networks increasingly operate in dynamic environments where traffic demands, resource availability, and even network topology evolve over time. This dynamism presents the central challenge for routing optimization: strategies must not only satisfy multi-dimensional constraints but also adapt in real-time to environmental fluctuations. The polymorphic smart network architecture further envisions full-dimensionally definable, adaptive infrastructures for future heterogeneous environments [8]. Traditional approaches [9, 13] often rely on hand-crafted heuristics like shortest-path or ant colony optimization (ACO), which prioritize simplicity and efficiency. While simple heuristics offer speed and interpretability, they fall short in scenarios requiring nuanced trade-offs among these complex constraints. Recognizing this, recent research efforts have shifted toward designing customized algorithms for specific network environments, such as hierarchical topologies [2], multi-tier data centers [6], or edge-cloud coordination setups [15]. While tailored solutions often leverage hand-crafted policies, topology-specific rules, or offline-optimized control paths to enhance flow deployment, their rigid design hinders generalization across diverse topologies, dynamic workloads, and evolving service demands. This lack of adaptability results in performance degradation under real-world conditions with fluctuating resource availability and latency-sensitive constraints. Related efforts have explored verification challenges in programmable networks [16].

Recent advances in code-evolving frameworks [7, 10, 14, 17] powered by large language models (LLMs) offer a new paradigm: generating problem-specific algorithms through iterative code refinement and task adaptation. Among these, the NeRM framework [7]

has shown strong performance in classical combinatorial tasks by jointly evolving natural language prompts and executable code. However, when extended to network flow deployment, these methods reveal critical limitations that hinder their effectiveness. Networking tasks present unique difficulties not found in abstract optimization problems. They involve stateful constraints that are tightly coupled with graph topology, such as queue-induced latency and shared link contention, which are difficult to encode in high-level prompt description. Additionally, LLM-generated algorithms may violate feasibility conditions or overlook critical trade-offs between delay and resource distribution without explicit structural awareness. The general co-evolution mechanism in LLM-based code generation also struggles to model dynamic network state effectively, and its reliance on full execution for evaluation leads to substantial computational overhead during deployment.

To address these limitations, we propose a network-specialized evolution framework that systematically co-optimizes prompt design and algorithm generation. The framework integrates three core components: topology-aware prompt mutation, which ensures that high-level task descriptions remain aligned with graph-structured constraints; feasibility-guided algorithm refinement, which incrementally improves routing heuristics by identifying and correcting constraint violations; and adaptive environmental assessment and feedback, which inject dynamic feedback into the co-optimization of prompts and algorithms, enabling better adaptation to input-scale-specific characteristics. Given a set of flow requests, each defined by a source-destination pair and a delay threshold, the goal is to find feasible routing paths under compute, storage, and bandwidth constraints at both nodes and links, so as to maximize the flow acceptance rate while minimizing average resource consumption. Our framework tightly integrates network structure and runtime feedback into the evolutionary process, enabling adaptive and efficient routing strategy generation tailored for dynamic and heterogeneous network environments.

Our contributions can be summarized as follows:

- We formulate a dynamic network routing problem model that imposes strict delay and resource constraints on both nodes and links, and there are environmental topology changes on the network.
- We propose a topology-aware reflective evolution framework that applies the LLM-based code generation to resource-constrained flow routing with latency guarantees.
- Experiments on real-world topologies show improved flow acceptance and resource efficiency over different methods and environment shifts.

2 Preliminary

2.1 Network Flow Management Problem

Formally, the network is modeled as a directed graph $G = (V, E)$, where V denotes the set of nodes and E the set of directed links. Similar to the setting common in virtual network function [3]. Each node $v \in V$ is associated with available computational and storage capacities, denoted as c_v and s_v, respectively. Each link $e \in E$ is characterized by its bandwidth b_e and latency d_e. A communication request is defined as a tuple $r_i = (s_i, t_i, d_i^{\max}, \mathbf{r}_i)$, where s_i and t_i denote the source and destination nodes, $d_i^{\max}$ specifies the maximum allowable end-to-end latency, and $\mathbf{r}_i = [c_i, s_i, b_i]$ represents the required computational, storage, and forwarding resources, respectively. The goal is to determine a set of routing paths $\{P_i\}$, one for each request r_i, such that:

- **Delay constraint:** $\sum_{e \in P_i} d_e \leq d_i^{\max}$,
- **Resource constraint:** The total resource consumption across all paths does not exceed the capacities of nodes and links,
- **Throughput maximization:** Maximize the number of successfully routed requests: $[\max_{\{P_i\}} \sum_i \mathbb{I}[P_i \text{ is feasible}]]$

This problem extends classical delay-constrained routing by incorporating multi-dimensional resource constraints at both node and link levels. It can be viewed as a generalized knapsack problem over graphs and is inherently NP-hard, similar to the challenges faced in virtual network embedding with learned heuristics [12]. The joint optimization of throughput and balanced resource utilization introduces significant challenges, particularly in large-scale dynamic environments [11].

2.2 LLM for Strategy Evolution

LLMs have demonstrated strong capabilities in code generation and algorithm design by translating natural language prompts into executable programs. The standard pipeline typically involves three stages: (1) encoding the task description as a prompt, (2) invoking an LLM to generate one or more candidate programs, and (3) executing these candidates to evaluate their performance.

While effective, this paradigm largely treats the prompt and code generation as loosely connected stage. Most existing methods fix the prompt throughout the process, focusing solely on generating and selecting optimal code from a large candidate pool. This one-shot prompting strategy overlooks the impact of task specification quality on the alignment and effectiveness of the generated algorithms. To address these limitations, NeRM proposes a biologically inspired co-evolution framework that jointly evolves prompts and codes. NeRM introduces two interleaved modules: Metamorphosis on Prompts (MoP), which refines task specifications through reflection and prompt mutation, and Metamorphosis on Algorithms (MoA), which evolves heuristic programs conditioned on the refined prompts. These two modules form a nested loop, enabling continual improvement of both the input specification and the resulting algorithm. NeRM further incorporates a predictor-assisted evaluation mechanism that uses LLM embeddings and a pairwise ranking model to estimate code quality, allowing early elimination of weak candidates and reducing the cost of full execution. By integrating prompt evolution, algorithm refinement, and efficient evaluation, NeRM establishes a more adaptive, scalable, and performance-aware paradigm for LLM-driven algorithm design.

3 Methodology

We consider the problem of resource-constrained routing in enterprise-scale networks, where each node has limited computational and storage capacities, and each link is characterized by constrained bandwidth and latency. Given a collection of communication requests, the objective is to determine feasible routing paths that jointly satisfy delay and resource constraints while maximizing the number of successful transmissions. This problem, formalized as a

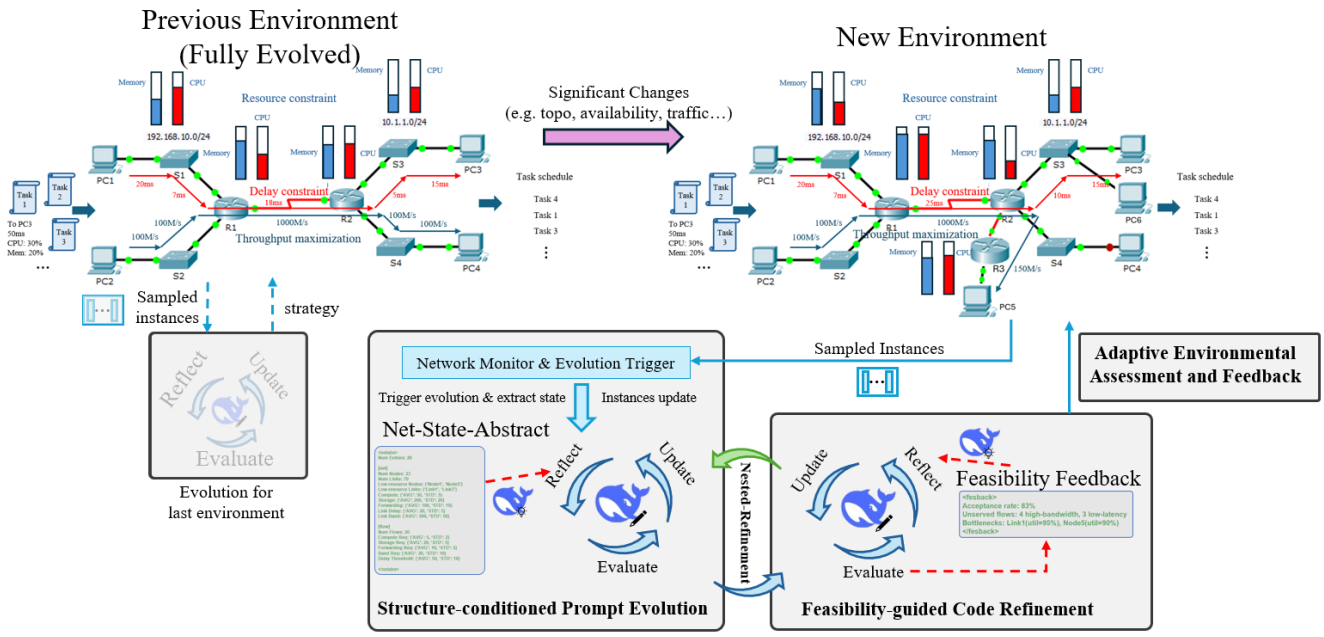


Figure 1: Overview of NeRM-Net, highlighting the iterative optimization loop between Structure-conditioned Prompt Evolution (SPE) and Feasibility-guided Code Evolution (FCE). The framework integrates adaptive environmental feedback (tracking topology dynamics and resource states) to drive SPE-FCE interactions, enhancing the generation of routing strategies tailored to dynamic, resource-constrained networks.

graph-based generalization of the knapsack problem, is NP-hard and requires the coordination of heterogeneous resources under strict performance constraints.

To tackle this challenge, we adopt a dual-stage learning framework that iteratively refines both the formulation of the problem and the generation of heuristic algorithms. The approach consists of two interacting components: one that evolves the task description in natural language, capturing high-level problem characteristics, and another that synthesizes and optimizes routing heuristics based on the evolving specification. These components operate in a closed loop, allowing the system to gradually align task understanding with effective algorithmic design.

3.1 Structure-conditioned Prompt Evolution

In the context of our routing problem, a prompt serves as a natural language description of the optimization objective and the operational constraints. A typical prompt may express that the system must prioritize requests with tight latency bounds, balance computational loads across underutilized nodes, or avoid congested links. The expressiveness of the prompt plays a crucial role in shaping the behavior of the language model during heuristic generation. However, hand-crafted prompts often suffer from incompleteness or misalignment with the underlying graph-level constraints.

To address this, we incorporate a reflection-driven prompt evolution mechanism. The process begins with a diverse set of initial prompts, which may describe the problem from different perspectives, such as resource-aware path selection, delay-sensitive forwarding, or queue-state monitoring. Each prompt is used to guide the generation of candidate routing algorithms, whose performance is then evaluated over a batch of communication requests. Based on the feedback, a language model analyzes historical performance patterns and refines the prompt by adjusting its emphasis on key constraints or introducing new semantic elements. For instance, a prompt that initially emphasizes delay minimization may evolve to incorporate resource balancing when low-throughput performance is observed. Through this iterative refinement, the prompts become increasingly aligned with the latent structure of the problem and serve as effective guidance for subsequent code generation.

3.2 Feasibility-guided Code Refinement

Conditioned on the current prompt, the framework generates heuristic algorithms expressed as executable code for selecting routing paths. Each candidate algorithm is a mapping from request specifications and network states to path selection decisions, incorporating logic for constraint checking and resource tracking. The initial code candidates are generated via a language model and encode various routing strategies, such as greedy link selection based on residual bandwidth, shortest-delay path traversal with feasibility pruning, or node-level admission control based on resource profiles.

To continuously improve the quality of the heuristics, we adopt an evolutionary process in which candidate algorithms are refined through multiple rounds of generation, evaluation, and variation. A reflection module analyzes previously generated code to identify reusable components and extract implicit design principles, such as prioritizing short paths under tight latency constraints or distributing flows across disjoint paths to reduce contention. Based on these insights, new candidate algorithms are synthesized by modifying conditionals, altering scoring functions, or recombining segments from high-performing code. This evolution in code space enables

Table 1: Details of Structure-conditioned Prompt Evolution (SPE) and Feasibility-guided Code Evolution (FCE)

Module	Structure-conditioned Prompt Evolution (SPE)	Feasibility-guided Code Evolution (FCE)
Initial population	Combines manually crafted topology-aware prompts (extract network features like node types and link tiers) and LLM-generated variants to ensure initial alignment with network structure.	Initializes with LLM-generated code candidates based on SPE prompts, supplemented by feasible code snippets (e.g., basic constraint-checking logic) to ensure baseline feasibility.
Reflection	Analyzes performance gaps (e.g., underutilized nodes, latency overruns) to refine prompts, emphasizing overlooked structural constraints (e.g., "prioritize links in GEANT's ultra-high tier").	Identifies feasibility violations (e.g., resource exhaustion) from execution logs, extracting rules like "prune paths with node storage < request demand" for code refinement.
Evolution	Evolves prompts via structure-targeted mutation (e.g., adjusting emphasis on node/link attributes) and crossover of high-performing topology descriptions.	Evolves code by fusing feasible segments (e.g., residual bandwidth check + latency pruning) and mutating constraint logic to adapt to dynamic network states.
Update	Retains top-performing prompts/algorithms via validation on diverse topologies.	

the system to explore a rich set of strategies that adapt to the combinatorial nature of the routing problem and progressively approach optimal or near-optimal solutions under practical constraints.

3.3 Adaptive Environmental Assessment and Feedback

The adaptive environmental assessment and feedback module acts as a dynamic bridge between network input features and the two core modules. It captures multi-dimensional network inputs: static topology attributes (e.g., node heterogeneity like compute/storage-intensive nodes, link hierarchy such as ultra-high bandwidth tiers) and real-time dynamic states (node resource utilization, link bandwidth consumption, latency fluctuations, and traffic request characteristics like delay thresholds).

The prompt and code evolution modules interact in a tight loop. The performance of algorithms provides signal for improving the prompt, while improved prompts guide the generation of more effective algorithms. This co-adaptive design enables the system to optimize problem abstraction and solution synthesis jointly. As a result, the framework can discover routing policies that are tailored to the structure and dynamics of large-scale enterprise networks.

4 Experiments

4.1 Experimental Settings

Dataset. We evaluate the proposed NeRM-Net on two standard network topologies widely used in network research: the Abilene network and the GEANT network [1]. For the Abilene network, we form a relatively simpler scenario. The compute and storage resources of each node are set to 10 units. The edges are bidirectional, where each link has a bandwidth capacity of 10 units and varying delay characteristics ranging from 2 to 4 time units. For the GEANT network, we form a more complex scenario. Nodes are assigned heterogeneous compute (20–120 units) and storage (20–120 units) resources based on their roles: compute-intensive nodes prioritize higher compute capacity, storage-intensive nodes focus on larger storage, and balanced nodes maintain moderate levels of both. Network edges are bidirectional, with bandwidth (10–120 units) and delay (1–20 units) categorized into four tiers (ultra-high, high, medium, low) to create differentiated link characteristics.

We generate datasets containing multiple network request instances, where each instance represents a collection of network service requests. The source node s_i and the destination node t_i are selected from the network randomly. The compute resource c_i, storage resource s_i, and bandwidth resource b_i requirements and delay threshold $d_i^{\max}$ are sampled uniformly from $[1, 3]$, $[1, 3]$, $[2, 5]$ and $[10, 20]$ respectively. Additionally, we generate datasets of varying complexity, e.g., 20, 50, and 100 requests per instance for small-, medium-, and large-scale problems, respectively. Each scale includes 10 training, 20 validation, and 20 test instances.

Metrics. The evaluation metric is the optimization objective, defined as the number of successfully routed requests and denoted as Obj. We also compute the performance gap between each method and the optimal solution. For 20-flow scenarios, the *optimal* method adopts the arrangement derived from depth-first search as the optimal solution. For 50-flow and 100-flow scenarios, due to the excessive time consumption of depth-first search, we take the average-weight method without priority as the benchmark. Positive values indicate the performance gap from the optimal value, whereas negative values reflect improvement over the benchmark.

Baselines. The baseline algorithms include Optimal [4], Average (simply without priority), Greedy [9], and Ant Colony Optimization (ACO) [5]. The Optimal algorithm exhaustively searches all possible priority orderings of requests to find the best deployment. The Average algorithm assigns equal priority to all requests, effectively resulting in random deployment. The Greedy algorithm prioritizes requests based on a heuristic function; in our baseline, this function is simplified to the request ID, leading to a strict arrival-order deployment. ACO selects requests in each iteration using a combination of pheromone levels and a heuristic function, iteratively constructing a deployment sequence. In the baseline setting, the heuristic function is uniform across all requests.

4.2 Performance Comparison

Effective of NeRM-Net. We first evaluate the overall effectiveness of NeRM-Net against traditional routing algorithms. As shown in **Table 2**, NeRM-Net significantly outperforms all baselines across problems of varying scale. For the 20-flow instances where an optimal solution is computable, NeRM-Net achieves a near-optimal objective value of 11.11, reducing the performance gap to just 2.81%. In contrast, traditional methods like Greedy and ACO lag considerably, with gaps exceeding 16%. The superiority of NeRM-Net becomes even more pronounced in larger-scale scenarios. For the 50-flow and 100-flow problems, NeRM-Net improves upon the Average baseline by 31.01% and 57.49%, respectively, demonstrating its

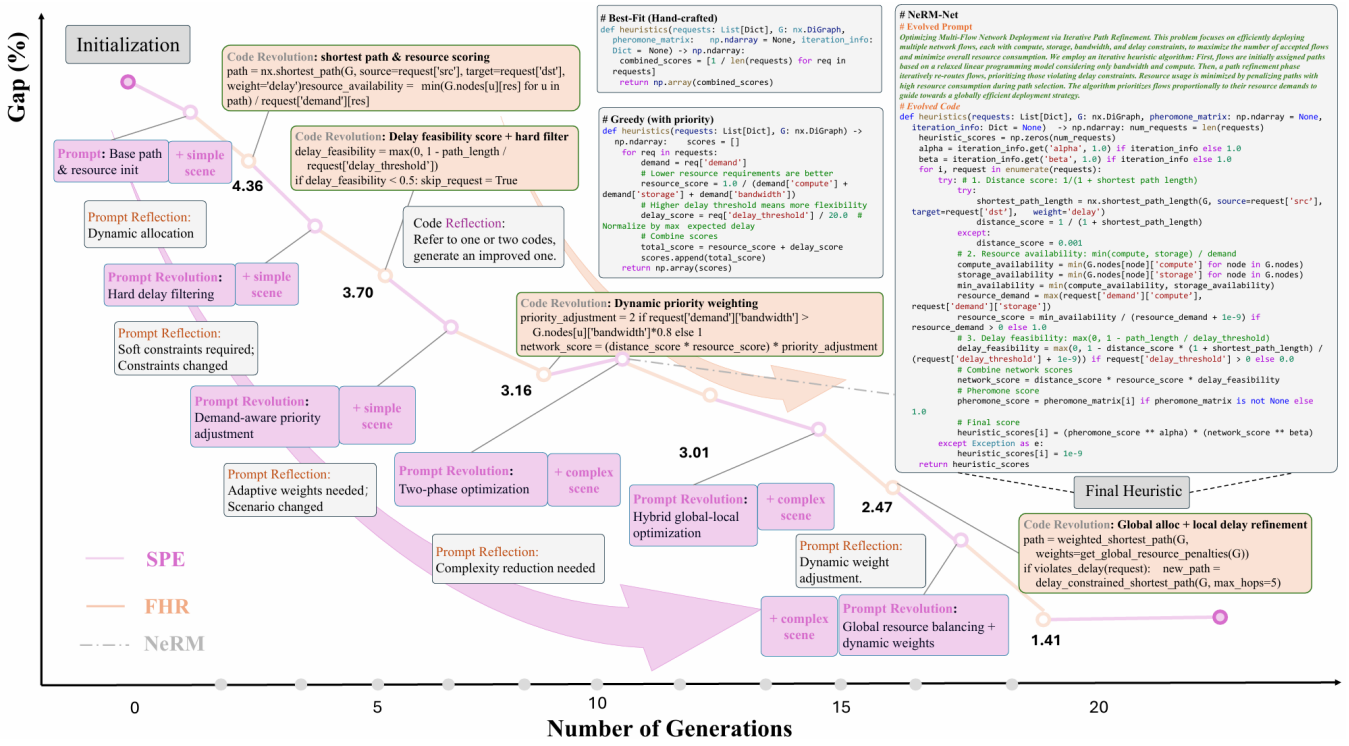


Figure 2: Evolution process of NeRM-Net. We outline the key prompts and best codes generated from NeRM-Net. Moreover, we present the best algorithm in the final iteration and compare it with typical human-designed strategies.

exceptional scalability and ability to discover high-quality heuristics for complex, resource-constrained routing tasks. This highlights the framework's capacity to evolve sophisticated policies that effectively navigate the trade-offs between multiple network constraints, a task where conventional heuristics fall short.

Effect of Different LLMs. The performance of NeRM-Net is inherently linked to the capabilities of its underlying Large Language Model (LLM). We conducted experiments with several state-of-the-art LLMs, with results summarized in **Table 3** (the second one). The findings indicate that more powerful models generally yield better results. For instance, **Gemini-1.5 Pro** achieves a remarkable near-optimal performance on the 20-flow problem, closing the gap to just 0.19%. On the highly complex 100-flow instances, **Llama-3 70B** delivers the best performance. Notably, even with a widely accessible model like **GPT-3.5-Turbo**, our framework substantially outperforms all traditional baselines, confirming the robustness and versatility of the NeRM-Net architecture. This shows that NeRM-Net can effectively leverage the reasoning power of various LLMs to tackle complex optimization challenges.

Adaptive Ability of Typical Methods.

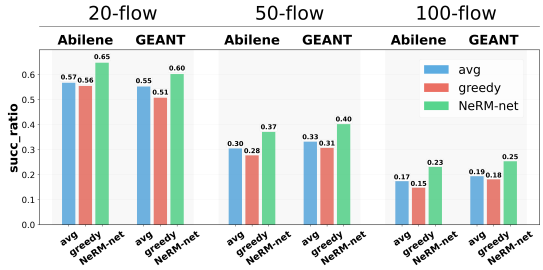
We conduct a detailed analysis across various network service quality and resource utilization metrics. As illustrated in **Figure 3** and **Figure 4**, these experiments cover different problem scales (20, 50, and 100 flows) and network topologies (Abilene and GEANT). Each figure is divided into three gray-scaled sections representing results under 20-flow, 50-flow, and 100-flow scenarios respectively, and each gray-scaled section contains two subsections for Abilene and GEANT topologies, with each subsection presenting the performance bar charts of the avg, greedy, and NeRM-Net approaches. For evolutionary algorithms, we run them on Abilene topology and fine-tune them on GEANT topology. For fixed strategy methods, we run the same algorithm on different topologies.

Table 2: Comparisons of different methods. Bold results indicate the best performance.

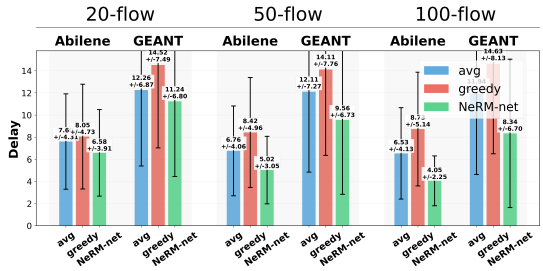
Method	20-flow		50-flow		100-flow	
	obj	gap(%)	obj	gap(%)	obj	gap(%)
optimal	11.43	0.00	-	-	-	-
Average	9.55	16.45	12.65	0.00	12.88	0.00
Greedy	9.57	16.26	12.00	5.10	12.82	0.49
ACO	9.55	16.45	11.95	5.48	13.18	-2.34
NeRM-Net	**11.11**	**2.81**	**16.57**	**-31.01**	**20.29**	**-57.49**

Table 3: Performance comparison using different LLMs as the backbone of NeRM-Net. Bold values indicate the best performance, while underlined values represent the second-best performance.

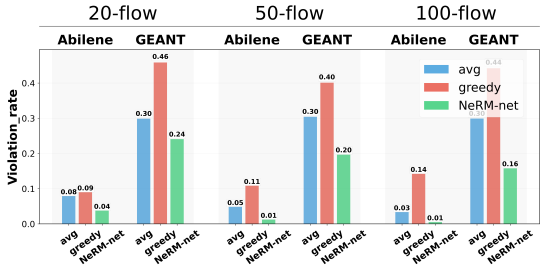
Method	20-flow		50-flow		100-flow	
	obj	gap(%)	obj	gap(%)	obj	gap(%)
optimal	11.43	0.00	-	-	-	-
Average	9.55	16.45	12.65	0.00	12.88	0.00
GPT-3.5-Turbo	11.15	2.44	16.47	-30.24	20.19	-56.74
GPT-4-Turbo	11.22	1.85	13.69	-8.26	20.06	-55.72
Llama-3 70B	10.40	9.03	14.51	-14.74	**22.05**	**-71.16**
Gemini-1.5 pro	**11.41**	**0.19**	**16.88**	**-33.48**	21.22	-64.72
DeepSeek-V3	10.30	9.90	15.29	-20.91	21.20	-64.56
GLM-4-Flash	11.11	2.81	16.57	-31.01	20.29	-57.49



(a) success ratio

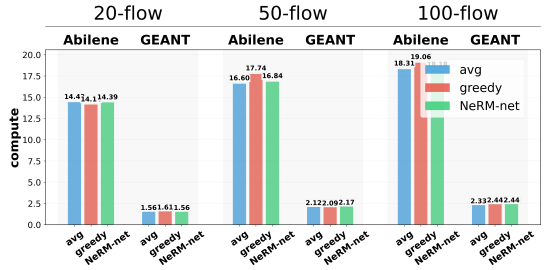


(b) delay variance

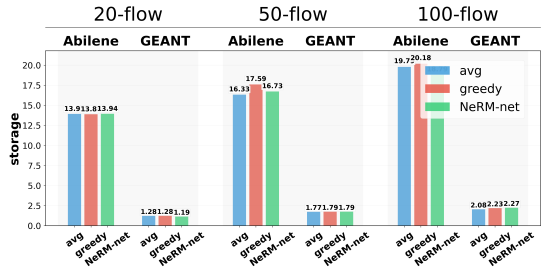


(c) violation ratio

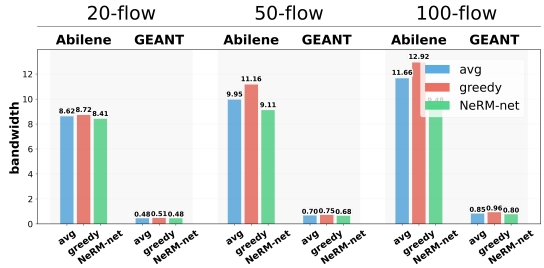
Figure 3: Network indexes: better and stabler across different problem sizes and topologies.



(a) average computing resource utilization



(b) average storage resource utilization



(c) average bandwidth resource utilization

Figure 4: Network utilization: reduce average utilization while keeping high performance.

The results highlight NeRM-Net's superior performance in maintaining high network service quality. As shown in **Figure 3a**, NeRM-Net consistently achieves the highest success ratio across all tested scenarios. Furthermore, it maintains the lowest violation rate (**Figure 3c**), demonstrating its effectiveness in adhering to delay constraints where other methods, such as the *greedy* approach, fall short. **Figure 3b** further reveals that NeRM-Net offers a more stable and predictable service by achieving lower average delay with significantly smaller variance, a critical factor for ensuring Quality of Service (QoS). Crucially, NeRM-Net achieves this enhanced performance with remarkable resource efficiency. **Figure 4** shows that the average utilization of compute, storage, and bandwidth resources by NeRM-Net is comparable to, and in many cases lower than, the baseline methods. For example, in the 50-flow Abilene scenario, NeRM-Net delivers a much higher success ratio than the *avg* baseline (0.371 vs. 0.305) while consuming less bandwidth (9.108 vs. 9.950). This demonstrates that NeRM-Net evolves sophisticated heuristics capable of identifying more efficient routing paths that intelligently balance load across the network. In summary, these findings underscore NeRM-Net's strong adaptive ability to deliver high-quality, efficient solutions across diverse network conditions.

5 Conclusion

This work proposes a network-specialized reflective evolution framework for resource-constrained routing in dynamic networks. NeRM-Net integrates topology-aware prompt evolution, feasibility-guided code refinement, and lightweight surrogate evaluation to address limitations of traditional methods. Experiments on Abilene and GEANT topologies show NeRM-Net outperforms baselines across different number of flow sizes, with up to 57.49% higher flow acceptance and efficient resource use. NeRM-Net advances adaptive network optimization. Future work will scale to hybrid architectures and enhance real-time tracking.

Acknowledgments

This work is supported by National Key Research and Development Program of China (under Grant No.2023YFB2903904).

References

[1] 2020. GEANT/Abilene Network Topology Data and Traffic Traces.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*.

[3] Xiaoliang Chen, Zuqing Zhu, Roberto Proietti, and S. J. Ben Yoo. 2019. On Incentive-Driven VNF Service Chaining in Inter-Datacenter Elastic Optical Networks: A Hierarchical Game-Theoretic Mechanism. *IEEE Transactions on Network and Service Management* (2019).

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press.

[5] M. Dorigo and G. Di Caro. 1999. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*.

[6] M.A. Gunavathie and S. Umamaheswari. 2024. Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. *Expert Systems with Applications* (2024).

[7] Shuhan Guo, Nan Yin, James Kwok, and Quanming Yao. 2025. Nested-Refinement Metamorphosis: Reflective Evolution for Efficient Optimization of Networking Problems. In *Findings of the Association for Computational Linguistics: ACL 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.).

[8] Yuxiang Hu, Dan Li, Penghao Sun, Peng Yi, and Jiangxing Wu. 2020. Polymorphic Smart Network: An Open, Flexible and Universal Architecture for Future Heterogeneous Networks. *IEEE Transactions on Network Science and Engineering* (2020).

[9] T. Korkmaz and M. Krunz. 2001. Multi-constrained optimal path selection. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*.

[10] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Forty-first International Conference on Machine Learning*.

[11] Yaxi Liu, Wei Huangfu, Haijun Zhang, and Keping Long. 2019. An Efficient Stochastic Gradient Descent Algorithm to Maximize the Coverage of Cellular Networks. *IEEE Transactions on Wireless Communications* (2019).

[12] Xinglong Pei, Shuhan Guo, Yuxiang Hu, Ziyong Li, Quanming Yao, Dan Li, Jinchuan Pei, and Yongji Dong. 2025. Graph Pointer Network Assisted Deep Reinforcement Learning for Virtualized Network Embedding. *IEEE Transactions on Green Communications and Networking* (2025).

[13] D.S. Reeves and H.F. Salama. 2000. A distributed algorithm for delay-constrained unicast routing. *IEEE/ACM Transactions on Networking* (2000).

[14] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, and et al. 2024. Mathematical discoveries from program search with large language models. *Nature* (2024).

[15] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* (2017).

[16] Ying Yao, Le Tian, and Yuxiang Hu. 2025. NetChecker: enabling real-time and error-locatable runtime verification for programmable networks. *Journal of King Saud University Computer and Information Sciences* (2025).

[17] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. 2024. ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution. In *Advances in Neural Information Processing Systems*.