Virne: A Comprehensive Benchmark for Deep RL-based Network Resource Allocation in NFV

Tianfu Wang[1,5,*], **Liwei Deng**[2,*], **Xi Chen**[1], **Junyang Wang**[1], **Huiguo He**[3], **Leilei Ding**[1], **Wei Wu**[1], **Qilin Fan**[4], **Hui Xiong**[5,†]
[1]University of Science and Technology of China
[2]University of Electronic Science and Technology of China, [3]Sun Yat-sen University
[4]Chongqing University, [5]Hong Kong University of Science and Technology (Guangzhou)
[*]tianfuwang@mail.ustc.edu.cn, [†]xionghui@ust.hk

Abstract

Resource allocation (RA) is critical to efficient service deployment in Network Function Virtualization (NFV), a transformative networking paradigm. Recently, deep Reinforcement Learning (RL)-based methods have been showing promising potential to address this complexity. However, the lack of a systematic benchmarking framework and thorough analysis hinders the exploration of emerging networks and the development of more robust algorithms while causing inconsistent evaluation. In this paper, we introduce Virne, a comprehensive benchmarking framework for the NFV-RA problem, with a focus on supporting deep RL-based methods. Virne provides customizable simulations for diverse network scenarios, including cloud, edge, and 5G environments. It also features a modular and extensible implementation pipeline that supports over 30 methods of various types, and includes practical evaluation perspectives beyond effectiveness, such as solvability, generalization, and scalability. Furthermore, we conduct in-depth analysis through extensive experiments to provide valuable insights into performance trade-offs for efficient implementation and offer actionable guidance for future research directions. Overall, with its diverse simulations, rich implementations, and extensive evaluation capabilities, Virne could serve as a comprehensive benchmark for advancing NFV-RA methods and deep RL applications. The code is publicly available at https://github.com/GeminiLight/virne.

1 Introduction

Network Function Virtualization (NFV) has emerged as an essential enabler for modern networks, such as cloud data centers, edge computing and 5G, due to its remarkable flexibility and scalability [1]. By transforming traditional hardware-bound network service functions into flexible software modules, NFV enables the deployment of Virtual Network Functions (VNFs) on general-purpose servers [2]. A central challenge in NFV is Resource Allocation (NFV-RA) that is essential for effective resource management and ensuring service quality [3]. As illustrated in Figure 1, NFV-RA involves mapping service requests (modeled as virtual networks of interconnected VNFs) onto shared physical infrastructure while satisfying various constraints. It is an NP-hard combinatorial optimization problem [4], highlighting the need for efficient solution strategies.

Traditional approaches to NFV-RA, such as exact solvers [5, 6] that seek optimal solutions or heuristics [7–9] dependent on manual design, suffer from neither excessive time consumption nor suboptimal performance. Recently, Reinforcement Learning (RL) has shown promise in solving NFV-RA, which enables the autonomous learning of efficient heuristics by interacting with simulated network environments [10–20]. This training paradigm operates without requiring high-quality

Preprint.

Table 1: Comparison of NFV-RA benchmarks.

	Supported Simulation	Implemented Algorithms	RL & Gym Support	Evaluation Perspectives	Last Update
VNE-Sim [21]	Cloud	3 heuristics	✗	Effectiveness	2014
ALEVIN [22]	Cloud	5 heuristics	✗	Effectiveness	2016
ALib [23]	Cloud	1 exact	✗	Effectiveness	2019
SFCSim [24]	Cloud	3 heuristics	✗	Effectiveness	2022
Iflye [25]	Cloud	3 heuristics	✗	Effectiveness	2024
Virne (Ours)	Cloud, Edge, 5G Slicing, etc.	30+ algorithms (10+ non-RL)	✓	Effectiveness & 3 Practicality	2025

labeled datasets, which are difficult to obtain due to the computational complexity and privacy concerns, making it particularly well-suited for NFV-RA.

However, the advancement of RL for NFV-RA is significantly hampered by the lack of standardized, comprehensive benchmarking. On the one hand, existing benchmarks, as summarized in Table 1, are limited to specific scenarios (e.g., cloud) and a narrow range of non-RL methods. On the other hand, the increasing complexity of modern networks leads to fragmented problem definitions and inconsistent simulations, making fair comparisons and robust evaluations difficult. We summarize the current state of NFV-RA research in Table 4 to highlight these issues. Developing a framework to address these issues poses significant technical challenges, including unifying diverse models, standardizing numerous algorithms, and designing comprehensive evaluations. A unified, accessible framework for reproducible research and standardized evaluation is urgently needed.

In this paper, we introduce **Virne**, a comprehensive benchmarking framework for NFV-RA that offers diverse simulations, unified implementations and thorough evaluations. Firstly, Virne is designed to serve as a unified and readily accessible framework for researchers from both the machine learning and networking communities. It provides a highly customizable simulation environment capable of accurately modeling a wide array of NFV scenarios, from cloud data centers to edge and 5G networks, allowing for the exploration of various resource types, constraints, and service requirements. Secondly, Virne features a modular architecture that simplifies the implementation of various NFV-RA algorithms, which facilitates both efficient utilization and new algorithm development. It includes over 30 NFV-RA algorithms, covering both exact, heuristic and advanced learning-based methods. Beyond standard performance metrics, Virne enables in-depth analysis through practical evaluation perspectives such as solution feasibility, generalization across varying network conditions, and scalability with increasing problem size. Through extensive empirical studies conducted within Virne, we offer valuable insights into the effectiveness and characteristics of various algorithms, providing data-driven guidance for future research directions and practical deployments.

Our main contributions, aimed at accelerating data-centric ML research in network optimization, are:

- **Comprehensive Simulation and Gym-style Environments**. We propose Virne, the most comprehensive benchmark for NFV-RA to date, along with gym-style environments. It offers highly customizable simulation and thorough evaluation across diverse scenarios.
- **Unified Implementations of Deep RL-based Methods**. To facilitate convenient utilization and future research, we design a modular and easily expandable pipeline that streamlines the implementation process and integrates a wide range of NFV-RA algorithms.
- **Insightful Findings on ML Applications**. Through extensive experiments from multiple perspectives, our empirical results reveal the impact of different modules and nuanced performance comparisons, providing actionable insights for future applied ML research.

2 NFV-RA Problem Definitions

Due to distinct considerations of network scenarios, existing studies formulate NFV-RA varying in system models, constraints, and objectives. To enhance clarity and consistency, we provide a unified definition that combines a basic cost optimization model with extensions for emerging scenarios.

2.1 Basic Definition for Cost Optimization

System Model In NFV, user network services and the physical infrastructure are virtualized as virtual networks (VNs) and physical network (PN), respectively. As illustrated in Figure 1, in online network systems, service requests represented as VNs are continuously arriving at PN, seeking the physical resource with specific Quality-of-Service requirements. Each arrived VN and the corresponding snapshot of PN consist of an instance $I = (\mathcal{G}_v, \mathcal{G}_p)$, where the PN $\mathcal{G}_p$ and VN $\mathcal{G}_v$ are modeled as undirected graphs, $\mathcal{G}_p = (\mathcal{N}_p, \mathcal{L}_p)$ and $\mathcal{G}_v = (\mathcal{N}_v, \mathcal{L}_v, \omega, \varpi)$, respectively. Here, $\mathcal{N}_p$ and $\mathcal{L}_p$ denote the sets of physical nodes and links, indicating servers and their interconnections; $\mathcal{N}_v$ and $\mathcal{L}_v$ denote the sets of virtual nodes and links, representing services and their relationships; ω and ϖ denote the arrival time and lifetime of VN. We denote $C(n_p)$ as available computing resources for physical node $n_p \in \mathcal{N}_p$, and $B(l_p)$ as available bandwidth resources of physical link $l_p \in \mathcal{L}_p$. Besides, $C(n_v)$ and $B(l_v)$ denote the demands for computing resource by a virtual node $n_v \in \mathcal{N}_v$ and bandwidth resource by a virtual link $l_v \in \mathcal{L}_v$. Over a specific period, we collect all instances into a set $\mathcal{I}$.

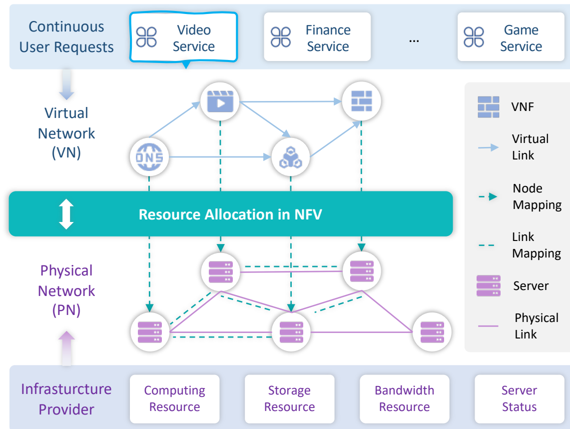


Figure 1: A brief illustration of the NFV-RA problem.

Embedding Constraints Mapping a VN $\mathcal{G}_v$ onto the sub-PN $\mathcal{G}_{p'}$ is formulated as a mapping function $f_{\mathcal{G}} : \mathcal{G}_v \rightarrow \mathcal{G}_{p'}$. It consists of two sub-mapping processes, i.e., node mapping $f_{\mathcal{N}}$ and link mapping $f_{\mathcal{L}}$. Node mapping $f_{\mathcal{N}}$, aims to find a suitable physical node $n_p = f_{\mathcal{N}}(n_v)$ to place each virtual node n_v while adhering to the one-to-one mapping and resource availability constraints. Concretely, virtual nodes in the same VN must be placed in different physical nodes and each physical node can host at most one virtual node. And the physical node n_p must have enough available resources to place the corresponding virtual node n_v, i.e., $C(n_v) \leq C(n_p), \forall n_v \in \mathcal{N}_v$. Link mapping aims to find a connected physical path $\rho_p = f_{\mathcal{L}}(l_v)$ to route each virtual link l_v while satisfying the path connectivity and resource capacity constraints. Concretely, the physical path ρ_p should connect physical nodes hosting the two endpoints of virtual link l_v. And each physical link $l_p \in \rho_p$ of physical path ρ_p should have enough bandwidth to route the corresponding virtual link l_v, i.e., $B(l_v) \leq B(l_p), \forall l_v \in \mathcal{L}_v, l_p \in \rho_p$.

Optimization Objective Considering the randomness of online service requests, NFV-RA mainly aims to maximize the resource utilization of each instance I, which facilitates long-term resource profit and request acceptance [15, 26, 27]. To assess the quality of solution $S = f_{\mathcal{G}}(I)$ to the instance I, the revenue-to-cost ratio (R2C) is a widely used metric, defined as follows:

$$\max \text{R2C}\,(S) = (\varkappa \cdot \text{REV}\,(S))\,/\text{COST}\,(S)\,. \tag{1}$$

Here, $\varkappa$ is a binary variable indicating the feasibility of the solution S: $\varkappa = 0$ if S violates some constraints, and $\varkappa = 1$ otherwise. REV(S) and COST(S) denotes the generated revenue and incurred cost by embedding VN $\mathcal{G}_v$. If $\varkappa = 1$, REV(S) denotes the revenue from the VN, calculated as $\sum_{n_v \in \mathcal{N}_v} C(n_v) + \sum_{l_v \in \mathcal{L}_v} B(l_v)$ and COST(S) denotes the resource consumption of PN, calculated as $\sum_{n_v \in \mathcal{N}_v} C(n_v) + \sum_{l_v \in \mathcal{L}_v} (|f_{\mathcal{L}}(l_v)| \times B(l_v))$. Here, $|f_{\mathcal{L}}(l_v)|$ quantifies the length of the physical path ρ_p routing the virtual link l_v. See Appendix A.1 for the detailed formulation.

2.2 Extensions in Emerging Networks

The application of NFV is pivotal in emerging network scenarios, such as heterogeneous resourcing networks, latency-aware edge networks, and energy-efficient green networks. These scenarios require additional consideration of their unique challenges. To better align NFV-RA with practical requirements, we discuss several extended definitions of NFV-RA in popular scenarios in Appendix A.2.

3 Virne: A Comprehensive NFV-RA Benchmark

To provide high-quality simulation and implementation, we introduce three design principles for Virne, following established software engineering practices [28]: (a) *Versatile customization* allows

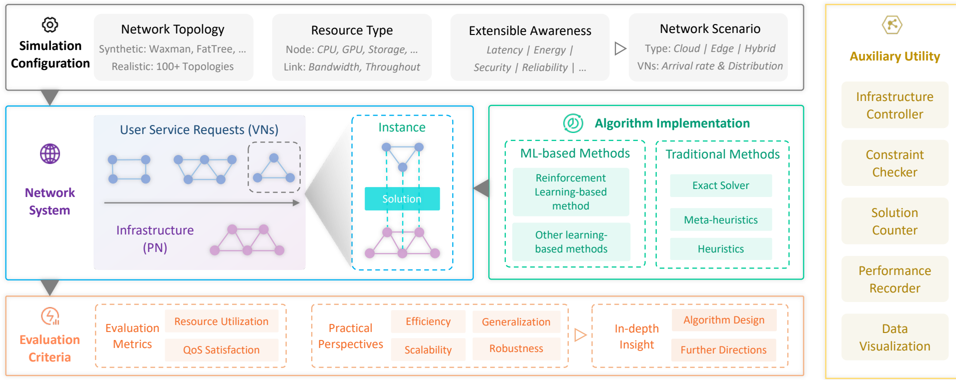


Figure 2: The architecture of Virne benchmark. Virne offers a streamlined workflow for supporting comprehensive experimentation of NFV-RA algorithms. The process begins with customizing simulation configurations that define network scenarios and conditions. Then, the network system is instantiated, triggering a series of service request events to process. At each event, the selected NFV-RA algorithm interacts with the system to resolve the instance. For each simulation, both the processing and final results are automatically recorded for subsequent analysis.

the platform to meet diverse simulation needs on varying network scenarios and conditions, ensuring adaptability. (b) *Scalable modularity* built platforms with a modular architecture to support flexible configurations and easy extensibility. (c) *Intuitive usability* prioritizes a user-friendly interface, enabling users to focus on experiment outcomes rather than implementation complexities. Guided by these principles, as illustrated in Figure 2, we implement Virne with five key modules as follows.

3.1 Simulation Configuration

Modern network systems are diverse and intricate, associated with various realistic factors such as resource availability and service requirements. To support high customization for simulating different network scenarios and conditions, Virne abstracts both the PN and VN as graphs, with customizable attributes for nodes, links, and the overall network. Specifically, Virne provides the following key customizable elements: (a) *Network Topologies*. Users can select from various topological synthesis methods or real-world physical infrastructure topologies, such as those from SDNLib[1]. (b) *Resource Availability*. Virne enables users to define multiple resource types (e.g., CPU, GPU, bandwidth) and their availability across different levels of the network (i.e., nodes, links and graph), allowing the reflection of the specific resource characteristics of different network environments (c) *Service Requirements*. Users can specify additional service requirements, such as latency, energy efficiency, and reliability, to ensure that the simulation reflects the needs of different network applications. By adjusting these parameters, Virne provides the flexibility to simulate a wide array of network scenarios, including cloud-based infrastructures, edge computing and 5G networks. This customization enables comprehensive testing, accommodating fluctuating resource demands and evolving network conditions. We provide configuration examples in Figure 4 and Appendix C.1.

3.2 Network System

With the above-mentioned configurations, Virne automatically creates a network system that functions as an event-driven simulator, which mainly consists of a PN as infrastructure and a series of sequential arrived VN requests. Virne treats each VN request arrival as a discrete event that triggers the corresponding resource scheduling and allocation procedures. An event represents the interaction between the VN request $\mathcal{G}_v$ and a snapshot of the PN $\mathcal{G}_p$ at the time of the request's arrival, denoted as an instance I. The specific NFV-RA algorithm then solves the corresponding NFV-RA problem for that instance to obtain a solution, $S = f_{\mathcal{G}}(I)$. Next, the system evaluates the feasibility of the solution based on the resource availability constraints and service requirements satisfaction. If feasible, the VN request is accepted; otherwise, it is rejected. This event-driven mechanism is designed to simulate realistic, complex network environments while enabling efficient handling of diverse network scenarios and fluctuating conditions.

[1]https://sndlib.put.poznan.pl/

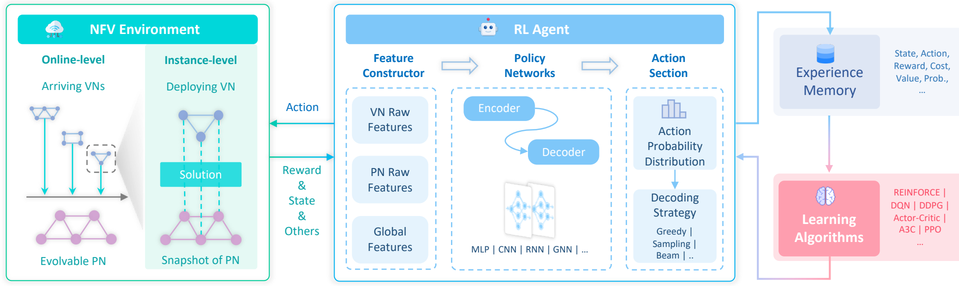


Figure 3: A unified pipeline of gym-style environment and RL-based NFV-RA methods in Virne.

3.3 Algorithm Implementation

To facilitate direct utilization and further extensions, Virne integrates diverse NFV-RA algorithms, covering both learning-based and traditional methods (see Appendix C.3). These algorithms are systematically organized to streamline the implementation process. Here, we highlight our unified pipeline of gym-style environments and RL-based NFV-RA methods, as illustrated in Figure 3.

3.3.1 NFV-RA as a Markov Decision Process

To address the randomness of online networks, most existing works model NFV-RA solution construction as a Markov Decision Process (MDP), which sequentially selects a physical node to place a virtual node at each decision step t, until all virtual nodes are placed or any constraints are violated.

Formally, we define this process as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \lambda)$. Concretely, $\mathcal{S}$ is the state space, where $s_t \in \mathcal{S}$ represents the embedding status of VN and PN. $\mathcal{A}$ is the action space, where a_t referring to a set of physical nodes. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function, indicating the conditional transition probabilities between states. During one transition, the system will attempt to place the selected action, i.e., a physical node $a_t = n_p$ to route the to-be-decided virtual node n_v. If node placement succeeds, link routing will be conducted to route the virtual links connecting n_v and its already-placed neighbor nodes. The shortest-path algorithms are used to identify the shortest and constraint-satisfied physical paths for these virtual links. Only when both node placement and link routing are successful, the physical network's available resources are updated with the VNR's requirements. Otherwise, the VN is rejected. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a designed reward function to guided optimization. $\lambda \in [0, 1)$ is the discount factor.

At each decision timestep y, observing the state s_t of the environment, the agent takes an action $a_t \sim \pi(\cdot|s_t)$ according to a policy π_θ parameterized by θ. Then, the environment will transit to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and feedback a reward $R(s_t, a_t)$ and. During interactions, a trajectory memory collects these step information as experiences. Using these experiences, the objective of the agent is to learn an optimal policy for resource allocation that maximizes the expected sum of discounted reward, i.e., the solution quality:

$$\pi^*_\theta = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta} \left[\sum_{i=0}^{T} \gamma^i R(s_t, a_t) \right] \tag{2}$$

Note that the above MDP introduced is the most adopted version. As a holistic framework, Virne also supports the emerging MDP version in recent studies. Due to the presentation clarity and page limit, we offer other MDP versions of NFV-RA in Appendix C.2.

3.3.2 Unified Pipeline for Efficient Implementations

Considering this widely-used MDP model, we generally unify existing NFV-RA RL-based algorithms into three key components: MDP modeling, policy architecture, and training methods. These methods model the NFV-RA solution construction process as above-introduced MDPs with a specific reward design and feature engineering. Subsequently, they use various neural networks to build policy architectures, such as Convolutional Neural Networks (CNN) [29] and Graph Convolutional Network (GCN) [30]. These policies are trained with a selected RL method, such as Asynchronous Advantage Actor-critic (A3C) [31] and Proximal Policy Optimization (PPO) [32].

Following this insight, Virne implements NFV-RA algorithms through a modular and extensible pipeline comprising several core components as illustrated in Figure 3. Specifically, an *instance-level environment* enable interactions with the agent for sequentially constructing solutions, where a customized *reward function* provides feedback at each step. The RL agent consists of a *feature constructor*, which extracts relevant features as inputs for the neural networks, and a *neural policy*, implemented with various architectures. To support RL training, an *experience memory* module continuously collects interaction data, while a designated *training method* optimizes the policy. This modular design is central to Virne's ability to integrate diverse approaches seamlessly, due to the high customization of each module. By standardizing the implementation process, Virne ensures consistency, reusability, and reduced complexity of implementation, which accelerates the development of new algorithms by providing a unified pipeline.

3.3.3 Implemented RL-based NFV-RA Algorithms

As summarized in Table 4 in Appendix B.2, various studies on NFV-RA often share similar core methodologies in their core RL framework design but differ primarily in terms of the specific network scenarios considered or specialized implementation techniques employed. Virne offers a comprehensive suite of state-of-the-art reinforcement learning methods for NFV resource allocation. The framework flexibly combines various RL training approaches (including MCTS, PG, A3C, and PPO) with different neural network architectures (such as MLP, CNN, S2S, GCN, GAT, BiGCN, BiGAT, and HeteroGAT). It also incorporates various implementation enhancements like custom reward functions, feature engineering combinations, and action masking mechanisms. See Appendix C.3.1 for descriptions of these RL methods, neutral policies, and implementation techniques.

To provide clarity in both implementation and evaluation, we categorize these works under common names that reflect RL algorithms and their neural policy architectures. To further streamline the training process and avoid unnecessary inconsistencies, we adopt PPO as the default RL algorithm in our experiments, due to its recognized training efficiency and strong empirical performance [33]. For example, the PPO-GCN method uses graph convolutional networks as feature encoders while employing Proximal Policy Optimization for efficient training. In the subsequent experiments, we will first investigate the impact of distinct implementation techniques, such as reward function and feature engineering, and identify the most efficient configurations for general implementation. Then, all of them will be evaluated for their effectiveness in the cloud and other popular scenarios.

3.4 Auxiliary Utility

To enhance usability and streamline analysis, Virne includes several key auxiliary utilities, as shown in Figure 2. These are designed to simplify the process of managing simulations and interpreting results, particularly including: (a) A *system controller* manages the simulation of physical and virtual networks, providing method choices for customization; (b) A *solution monitor* tracks solution feasibility and performance during execution, helping users assess whether solutions meet their defined criteria; (c) *Visualization tools* offer interactive and visual representations of simulation results, providing users with an intuitive way to analyze network system behaviors. These utilities support more advanced customizations for algorithm design and result analysis.

3.5 Evaluation Criteria

To offer a systematic evaluation, Virne provides a suite of metrics and multiple practical perspectives.

Performance Metrics. Virne includes the critical performance metrics of NFV-RA algorithms [34, 35], including request acceptance rate (RAC), long-term revenue-to-cost (LRC), long-term average revenue (LAR) and average solving time (AST). See Appendix C.4 for their definitions.

Practicality Perspectives. To comprehensively assess the practicality of NFV-RA algorithms, we examine multiple perspectives that extend beyond mere effectiveness. In particular, we consider three aspects of NFV-RA algorithms: (a) *Solvability* denotes the ability to find feasible solutions; (b) *Generalization* indicates reliable performance across various network conditions; (c) *Scalability* measures how effectively it accommodates increases in network size and complexity. Gaining these insights with our pre-provided evaluation interfaces, Virne helps users understand the practical viability of an algorithm and guides further development and application.

4 Empirical Analysis

4.1 Experimental Setup

We consider the most general scenarios as main network systems, i.e., cloud computing, and describe default settings for simulation and implementation. Subsequently, some parameters may be changed to simulate diverse network scenarios and conditions.

Simulation Settings. We adopt the widely-used topologies as physical networks: synthetic WX100 [36], and real-world GEANT and BRAIN [37], which cover various network scales and densities. See Appendix D.1.2 for their detailed descriptions. Computing resources of physical nodes and bandwidth resources of physical links are uniformly distributed within the same range of units, i.e., $\mathcal{X}_{C(n_p)} \sim \mathcal{U}(50, 100)$ and $\mathcal{X}_{B(l_p)} \sim \mathcal{U}(50, 100)$. In default settings, for each simulation run, we create 1000 VN with varying sizes following a uniform distribution, $\mathcal{X}_{|\mathcal{G}_v|} \sim \mathcal{U}(2, 10)$. The computing resource demands of the nodes and the bandwidth requirements of the links within each VN are uniformly distributed, i.e., $\mathcal{X}_{C(n_v)} \sim \mathcal{U}(0, 20)$ and $\mathcal{X}_{B(l_v)} \sim \mathcal{U}(0, 50)$, respectively. The virtual nodes in each VN are randomly interconnected with a probability of 50%. The lifetime of each VN follows an exponential distribution with an average of 500 time units. The arrival of these VNs follows a Poisson process with an average rate η, where η denotes the average arrived VN count per unit of time. Due to the varying physical resource supply in these topologies caused by distinct scale and density, we use different η, i.e., (0.16, 0.016, 0.004) for (WX100, GEANT, BRAIN), to accommodate the reasonable request demand. In the subsequent experiments, we manipulate the distribution settings of VNs and change the PN topologies to simulate various network systems.

Algorithm Implementation. In Appendix D.1.1, we provide the details of parameter setting, experimental methods on training and testing, and descriptions of computing resources.

4.2 Performance Comparison

4.2.1 Exploration on Implementation Techniques

Variations in core implementation techniques, such as reward function designs, feature engineering, action masking strategies, lead to substantial differences in the performance of RL-based NFV-RA algorithms. We study their impacts by systematically evaluating three representative policy architectures (PPO-MLP, PPO-ATT, PPO-DualGAT) in WX100, summarized in Table 2.

Impact on Reward Function Design. The design and magnitude of intermediate rewards strongly affect training and final performance. Across all solvers, a moderate fixed intermediate reward (`fixed, 0.1`) consistently yields the best or near-best results. In contrast, very small or no intermediate rewards result in clear performance drops, highlighting the importance of sufficient reward signals for effective exploration. Interestingly, the adaptive intermediate reward, which intuitively normalizes the total reward for different VN sizes, does not achieve optimal performance in practice. This suggests that, ❶ *while normalization is conceptually appealing, a fixed and moderate reward signal is more effective for guiding policy learning in the NFV-RA context.*

Impact on Feature Engineering Combination. We compare the use of Status (S), Topological (T), and their combination in the feature constructor. The results show that combining both Status and Topological features (✓, ✓) generally leads to the best or second-best outcomes for acceptance rate and revenue, regardless of the solver architecture. For example, PPO-ATT+ achieves the best acceptance rate (0.712) and LRC (0.661) with both features, and a marked drop when using only one. This demonstrates that ❷ *topological metrics serve as a valuable augmentation, capturing global network context and node importance, even for graph neural networks such as GATs.*

Impact on Action Mask Strategy Applying action masking (✓), which prevents the agent from selecting infeasible actions (e.g., resource availability satisfaction), consistently improves performance across all methods. For instance, in PPO-MLP+ and PPO-DualGAT+, enabling action masking increases acceptance rate by up to 0.053 compared to otherwise identical configurations without masking. This demonstrates that ❸ *explicit constraint enforcement is vital for robust RL-based NFV-RA solutions, since the constraints of NFV-RA are intricate and hard.*

This study reveals that the most effective implementation techniques of the RL-based NFV-RA method are achieved by combining moderate intermediate rewards, comprehensive feature engineering, and action masking. In subsequent experiments, we consider them as the default implementation.

Table 2: Impact of key implementation techniques. Reward function is specified by its type (fixed or adaptive) and the intermediate reward value (0, 0.01, 0.1, 0.2; where 0 indicates no intermediate reward). Features indicate whether node Status (S) and/or Topological (T) metrics were used (✓ for used, ✗ for not used). Action Mask indicates whether action masking was applied (✓) or not (✗).

Reward Function		Features		Action Mask	PPO-MLP			PPO-ATT			PPO-DualGAT		
Type	Value	S	T		RAC	LRC	LAR	RAC	LRC	LAR	RAC	LRC	LAR
Fixed	0.1	✓	✓	✗	0.666	<u>0.670</u>	11745.0	0.667	0.644	12174.0	0.648	0.750	12430.1
Fixed	0.1	✓	✗	✓	0.678	**0.675**	11159.0	0.660	<u>0.667</u>	11134.3	0.766	**0.754**	<u>15054.7</u>
Fixed	0.1	✗	✓	✓	0.703	0.654	12693.8	0.579	0.575	9323.5	0.733	0.685	13333.1
Adaptive	-	✓	✓	✓	0.705	0.619	12139.6	0.702	0.643	12368.6	<u>0.772</u>	0.744	14216.6
Fixed	0.2	✓	✓	✓	0.616	0.643	11869.4	0.706	**0.675**	12420.7	0.769	0.731	15045.5
Fixed	0.01	✓	✓	✓	<u>0.709</u>	0.645	<u>12719.1</u>	0.517	0.601	8339.7	0.766	0.748	14516.6
Fixed	0	✓	✓	✓	0.560	0.596	8259.5	**0.716**	0.658	<u>12629.0</u>	0.741	<u>0.753</u>	14047.6
Fixed	0.1	✓	✓	✓	**0.719**	0.647	**12944.4**	<u>0.712</u>	0.661	**12657.7**	**0.781**	0.738	**15138.6**

4.2.2 Effectiveness in Online Environments

To evaluate the effectiveness of these NFV-RA algorithms, we conduct online simulations across three distinct network topologies: WX100, GEANT, and BRAIN, each with different traffic rates (η) as specified. Table 3 reports experimental results of the implemented RL-based and traditional algorithms on key metrics. Particularly, RL agents equipped with dual graph neural network architectures, such as PPO-DualGAT and PPO-DualGCN, consistently dominate across diverse topologies (WX100, GEANT, BRAIN). PPO-DualGAT, for instance, achieves a remarkable 78.10% RAC and the highest LAR (14938.60) in WX100, while PPO-DualGCN excels in the GEANT network with a 60.40% RAC and 0.75 LRC. This suggests that ❹ *the ability of these dual-GNN models to concurrently process and relate features from both the virtual and physical network graphs provides a significant edge in finding high-quality embeddings.* Traditional heuristics, though extremely fast (e.g., GRC-Rank with AST $\sim$0.01-0.04s), generally lag in solution quality (e.g., GRC-Rank LRC 0.56 in WX100 vs. PPO-DualGAT's 0.74). Meta-heuristics or MCTS can achieve competitive RACs (74.30% for MCTS in WX100), yet their substantial computational overhead (AST >3s for MCTS) renders them less viable for agile, real-time NFV deployments compared to the sub-second inference of RL agents.

4.3 Evaluation from Practicality Perspectives

Generalization on Network Conditions Network conditions are inherently complex and subject to continuous changes, such as fluctuations in request frequencies and varying resource demands. As such, evaluating the generalization of these trained NFV-RA policies is critical to ensure they can adapt effectively to different, evolving network environments. To address this, we conduct the experiments in various network conditions to study the generalization to conditions of pretrained models, including *(a) Evaluation on Varying Traffic Rates* and *(b) Evaluation on Fluctuating Demand Distribution*. See Appendix D.2 for detailed experimental setup and result analysis.

Solvability via Offline Evaluation Traditional online testing of NFV-RA algorithms, while reflecting real-world scenarios, complicates direct comparisons of algorithmic solvability. The continuous evolution of the PN state, coupled with dynamic VN arrivals and lifetimes, makes it difficult to isolate performance. Specifically, it is challenging to determine if solution failures arise from an algorithm's inherent limitations or from transient unsolvable network states, thereby hindering fair comparisons on solvability. To address this, Virne provides a controlled offline evaluation environment. This features a series of static instances, each comprising a PN and a VN, evaluating algorithms' abilities to find high-quality, feasible solutions under reproducible conditions. Through such evaluation, we assess the fundamental solvability of each method, not only across an entire dataset but also for distinct VN scales. For detailed experimental setup and results, please see Appendix D.3.

Scalability on Network Sizes Network systems are growing in size as the physical infrastructure extends and service requests increase. Thus, assessing the scalability of NFV-RA policies is essential to ensure they can maintain their effectiveness as the network size and complexity increase. To assess this, Virne supports scalability evaluations through two primary perspectives: *(a) Performance*

Table 3: Performance comparison of implemented RL-based and traditional NFV-RA algorithms.

Solver	**WX100** ($\eta = 0.14$)				**GEANT** ($\eta = 0.016$)				**BRAIN** ($\eta = 0.004$)			
	RAC↑	**LRC**↑	**LAR**↑	**AST**↓	**RAC**↑	**LRC**↑	**LAR**↑	**AST**↓	**RAC**↑	**LRC**↑	**LAR**↑	**AST**↓
PPO-MLP	71.90	0.65	12944.40	0.13	55.80	0.67	645.04	<u>0.03</u>	51.30	0.69	155.10	0.14
PPO-CNN	71.70	0.65	12964.87	0.13	54.80	0.65	643.83	0.09	51.10	0.69	151.51	0.13
PPO-ATT	71.20	0.66	12657.69	0.14	54.50	0.65	707.01	0.10	51.00	0.68	<u>156.40</u>	0.15
PPO-GCN	66.80	0.64	11462.65	0.14	58.70	<u>0.72</u>	<u>763.68</u>	0.12	49.50	0.71	125.63	0.09
PPO-GAT	71.90	0.70	13178.13	0.15	58.40	0.70	724.31	0.07	44.60	0.51	95.32	0.09
PPO-GCN&S2S	65.80	0.63	11501.94	0.13	58.50	<u>0.72</u>	718.76	0.06	44.40	0.59	99.83	0.18
PPO-GAT&S2S	67.90	0.67	12445.03	0.16	57.30	0.69	754.61	0.15	51.80	0.68	136.67	0.19
PPO-DualGCN	70.20	<u>0.71</u>	<u>13467.57</u>	0.17	**60.40**	**0.75**	**791.75**	0.22	<u>54.80</u>	**0.78**	176.15	0.23
PPO-DualGAT	**78.10**	**0.74**	**14938.60**	0.18	<u>59.10</u>	<u>0.72</u>	739.27	0.10	**58.90**	<u>0.75</u>	**180.78**	0.13
PPO-HeteroGAT	72.50	0.66	12691.03	0.27	53.30	0.66	621.47	0.30	49.30	0.66	133.52	0.38
MCTS	<u>74.30</u>	0.44	12642.27	3.38	48.20	0.45	494.64	2.96	40.80	0.47	83.91	3.59
SA-Meta	65.50	0.63	10467.60	1.58	38.60	0.62	396.49	0.49	36.10	0.58	75.50	1.13
GA-Meta	71.70	0.59	11977.41	3.22	49.90	0.58	517.63	2.34	42.50	0.57	85.45	3.75
PSO-Meta	69.10	0.52	10706.48	4.29	45.30	0.46	457.93	3.68	41.50	0.46	80.67	4.20
TS-Meta	65.70	0.66	11141.91	1.35	40.90	0.69	402.04	0.62	37.10	0.64	68.41	1.11
NRM-Rank	60.70	0.52	9826.94	0.07	37.90	0.51	394.29	<u>0.03</u>	48.30	0.64	142.99	<u>0.04</u>
RW-Rank	60.10	0.56	9396.32	**0.04**	38.70	0.52	418.14	**0.01**	50.20	0.65	147.64	0.05
GRC-Rank	58.90	0.56	9269.03	**0.04**	36.70	0.47	353.21	**0.01**	48.40	0.64	144.55	**0.03**
PL-Rank	68.10	0.67	11570.27	0.32	55.30	0.66	661.93	0.04	48.70	0.70	136.79	0.36
NEA-Rank	64.00	0.66	10837.51	0.83	47.20	0.62	543.18	0.11	53.90	0.70	148.70	1.46
RW-BFS	40.00	0.57	7771.38	<u>0.05</u>	56.70	0.64	736.28	**0.01**	48.00	0.64	140.38	0.16

on Large-scaled Networks and *(b) analysis of Solving Time Scale* as problem size grows. Detailed experimental setups and results for these evaluations are placed in Appendix D.4.

4.4 Validation on Emerging Networks

NFV is widely applied across diverse modern networks, where specific scenarios often present unique characteristics. To further validate the adaptability of NFV-RA algorithms in emerging networks, we conduct evaluations in two key environments: *(a) heterogeneous resourcing networks*, which involve varied resource types, and *(b) latency-aware edge networks*, where delay constraints should be satisfied. See Appendix D.5 for experimental analysis.

4.5 Discussion on Future Direction

While RL-based methods for NFV-RA show promise, there are still practical challenges through our empirical observation. To advance deep RL for NFV-RA, we identify several future directions that also pose significant on existing ML methods. They are included but not limited to (a) *develop sophisticated representation learning for cross-graph statuses and attributed constraints*, (b) *achieving generalizable policies adaptable to varying network scales and dynamic conditions*, (c) *creating robust learning frameworks to enable learning in conflicting operational constraints*, and (d) *engineering highly scalable algorithms for extremely large-scale physical infrastructures*. A more detailed exploration of these prospective is provided in the supplementary material.

5 Conclusion

In this paper, we introduce **Virne**, a comprehensive and unified benchmarking framework specifically designed for evaluating deep RL-based algorithms for NFV-RA problem. Virne supports diverse network scenarios, including cloud, edge, and 5G environments, facilitated by customizable simulations. In addition, we provide a modular and extensible pipeline that integrates over 30 diverse algorithms, particularly deep RL-based methods, enabling extensible implementation. Beyond traditional effectiveness metrics, Virne offers practical evaluation perspectives such as solvability, generalization, and scalability. Furthermore, we conduct extensive experiments to evaluate the deep RL-based NFV-RA method from comprehensive perspectives. We provide crucial insights on the impact of implementation techniques and algorithm performance trade-offs. Our extensive empirical analysis also reveals valuable findings on potential challenges of RL-based NFV-RA methods, guiding future research on the interaction of data-driven networking optimization and applied machine learning.

References

[1] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[2] Weihua Zhuang, Qiang Ye, Feng Lyu, Nan Cheng, and Ju Ren. SDN/NFV-empowered future IoV with enhanced communication, computing, and caching. *Proceedings of the IEEE*, 108(2):274–291, 2020.

[3] Song Yang, Fan Li, Stojan Trajanovski, Ramin Yahyapour, and Xiaoming Fu. Recent advances of resource allocation in network function virtualization. *IEEE Transactions on Parallel and Distributed Systems*, 32(2):295–314, 2021.

[4] Matthias Rost and Stefan Schmid. On the hardness and inapproximability of virtual network embeddings. *IEEE/ACM Transactions on Networking*, 28(2):791–803, 2020.

[5] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE International Conference on Computer Communications*, pages 783–791, 2009.

[6] Nashid Shahriar, Shihabur Rahman Chowdhury, Reaz Ahmed, Aimal Khan, Siavash Fathi, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Virtual network survivability through joint spare capacity allocation and embedding. *IEEE Journal on Selected Areas in Communications*, 36(3):502–518, 2018.

[7] Long Gong, Yonggang Wen, Zuqing Zhu, and Tony Lee. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *IEEE International Conference on Computer Communications*, pages 1–9, 2014.

[8] Peiying Zhang, Haipeng Yao, and Yunjie Liu. Virtual network embedding based on computing, network, and storage resource constraints. *IEEE Internet of Things Journal*, 5(5):3298–3304, 2018.

[9] Weibei Fan, Fu Xiao, Mengjie Lv, Lei Han, Junchang Wang, and Xin He. Node essentiality assessment and distributed collaborative virtual network embedding in datacenters. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1265–1280, 2023.

[10] Soroush Haeri and Ljiljana Trajković. Virtual network embedding via Monte Carlo tree search. *IEEE Transactions on Cybernetics*, 48(2):510–521, 2017.

[11] Yikai Xiao, Qixia Zhang, Fangming Liu, Jia Wang, Miao Zhao, Zhongxing Zhang, and Jiaxing Zhang. Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning. In *IEEE/ACM International Symposium on Quality of Service*, 2019.

[12] Haipeng Yao, Sihan Ma, Jingjing Wang, Peiying Zhang, Chunxiao Jiang, and Song Guo. A continuous-decision virtual network embedding scheme relying on reinforcement learning. *IEEE Transactions on Network and Service Management*, 17(2):864–875, 2020.

[13] Tianfu Wang, Qilin Fan, Xiuhua Li, Xu Zhang, Qingyu Xiong, Shu Fu, and Min Gao. DRL-SFCP: Adaptive service function chains placement with deep reinforcement learning. In *IEEE International Conference on Communications*, pages 1–6, 2021.

[14] Zhongxia Yan, Jingguo Ge, Yulei Wu, Liangxiong Li, and Tong Li. Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE Journal on Selected Areas in Communications*, 38(6):1040–1057, 2020.

[15] Peiying Zhang, Yu Su, Jingjing Wang, Chunxiao Jiang, Ching-Hsien Hsu, and Shigen Shen. Reinforcement learning assisted bandwidth aware virtual network resource allocation. *IEEE Transactions on Network and Service Management*, 19(4):4111–4123, 2022.

[16] Mahdi Dolati, Seyedeh Bahereh Hassanpour, Majid Ghaderi, and Ahmad Khonsari. DeepViNE: Virtual network embedding with deep reinforcement learning. In *IEEE International Conference on Computer Communications Workshops*, pages 879–885, 2019.

[17] Tianfu Wang, Li Shen, Qilin Fan, Tong Xu, Tongliang Liu, and Hui Xiong. Joint admission control and resource allocation of virtual network embedding via hierarchical deep reinforcement learning. *IEEE Transactions on Services Computing*, pages 1–14, 2023.

[18] Haoyu Geng, Runzhong Wang, Fei Wu, and Junchi Yan. Gal-vne: Solving the vne problem with global reinforcement learning and local one-shot neural prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 531–543, 2023.

[19] Tianfu Wang, Qilin Fan, Chao Wang, Long Yang, Leilei Ding, Nicholas Jing Yuan, and Hui Xiong. Flagvne: A flexible and generalizable reinforcement learning framework for network resource allocation. In *International Joint Conference on Artificial Intelligence*, 2024.

[20] Sheng Wu, Ning Chen, Ailing Xiao, Peiying Zhang, Chunxiao Jiang, and Wei Zhang. Ai-empowered virtual network embedding:a comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2024.

[21] Soroush Haeri and Ljiljana Trajković. Vne-sim: A virtual network embedding simulator. In *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques*, SIMUTOOLS'16, page 112–117, 2016.

[22] Michael Till Beck, Claudia Linnhoff-Popien, Andreas Fischer, Fabian Kokot, and Hermann de Meer. A simulation framework for virtual network embedding algorithms. In *2014 16th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 1–6, 2014.

[23] Matthias Rost, Elias Döhne, and Stefan Schmid. Parametrized complexity of virtual network embeddings: dynamic & linear programming approximations. *SIGCOMM Comput. Commun. Rev.*, 49(1):3–10, February 2019.

[24] Lingyi Xu, Hefei Hu, and Yuanan Liu. Sfcsim: a network function virtualization resource allocation simulation platform. *Cluster Computing*, 26(1):423–436, July 2022.

[25] Stefan Tomaszek. Modellbasierte einbettung von virtuellen netzwerken in rechenzentren. 2021.

[26] Nan He, Song Yang, Fan Li, Stojan Trajanovski, Liehuang Zhu, Yu Wang, and Xiaoming Fu. Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1186–1201, 2023.

[27] Xiaoquan Zhang, Lin Cui, Fung Po Tso, Zhetao Li, and Weijia Jia. Dapper: Deploying service function chains in the programmable data plane via deep reinforcement learning. *IEEE Transactions on Services Computing*, pages 1–14, 2023.

[28] Hans Van Vliet, Hans Van Vliet, and JC Van Vliet. *Software engineering: principles and practice*, volume 13. John Wiley & Sons Hoboken, NJ, 2008.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[31] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2022.

[34] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906, 2013.

[35] Sheng Wu, Ning Chen, Ailing Xiao, Peiying Zhang, Chunxiao Jiang, and Wei Zhang. Ai-empowered virtual network embedding: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2024.

[36] B.M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[37] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0—survivable network design library. *Netw.*, 55(3):276–286, may 2010.

[38] Peiying Zhang, Yi Zhang, Neeraj Kumar, Mohsen Guizani, Ahmed Barnawi, and Wei Zhang. Energy-aware positioning service provisioning for cloud-edge-vehicle collaborative network based on drl and service function chain. *IEEE Transactions on Mobile Computing*, 23(4):3424–3435, 2024.

[39] Yawen Tan, Jiajia Liu, and Jiadai Wang. 5g end-to-end slice embedding based on heterogeneous graph neural network and reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[40] Yongshuai Liu, Jiaxin Ding, and Xin Liu. A constrained reinforcement learning based approach for network slicing. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6, 2020.

[41] Aleteng Tian, Bohao Feng, Yunxue Huang, Huachun Zhou, Shui Yu, and Hongke Zhang. Drl-based two-stage sfc deployment approach under latency constraints. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2024.

[42] Xiaoyuan Fu, F. Richard Yu, Jingyu Wang, Qi Qi, and Jianxin Liao. Dynamic service function chain embedding for nfv-enabled iot: A deep reinforcement learning approach. *IEEE Transactions on Wireless Communications*, 19(1):507–519, 2020.

[43] Ilora Maity, Thang X. Vu, and Symeon Chatzinotas. Revine: Reinforcement learning-based virtual network embedding in satellite-terrestrial networks. *IEEE Transactions on Communications*, 72(10):6316–6327, 2024.

[44] Yi Zhang, Peiying Zhang, Chunxiao Jiang, Shangguang Wang, Hongxia Zhang, and Chunming Rong. Qos aware virtual network embedding in space-air-ground-ocean integrated network. *IEEE Transactions on Services Computing*, 17(4):1712–1723, 2024.

[45] Sihan Ma, Haipeng Yao, Tianle Mai, Jingkai Yang, Wenji He, Kaipeng Xue, and Mohsen Guizani. Graph convolutional network aided virtual network embedding for internet of thing. *IEEE Transactions on Network Science and Engineering*, 10(1):265–274, 2023.

[46] Peiying Zhang, Chao Wang, Chunxiao Jiang, and Abderrahim Benslimane. Security-aware virtual network embedding algorithm based on reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 8:1095–1105, 2021.

[47] 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.

[48] Rania Sahraoui, Omar Houidi, and Fetia Bannour. Energy-aware vnf-fg placement with transformer-based deep reinforcement learning. In *2024 IEEE/IFIP Network Operations and Management Symposium (NOMS 2024)*, 2024.

[49] Xiancui Xiao. Dvne-drl: dynamic virtual network embedding algorithm based on deep reinforcement learning. *Scientific Reports*, 13(1):19789, 2023.

[50] Tianfu Wang, Long Yang, Chao Wang, Chuan Qin, Liwei Deng, Li Shen, and Hui Xiong. Towards constraint-aware learning for resource allocation in nfv-enabled networks. *arXiv preprint arXiv:2410.22999*, 2024.

[51] Weibei Fan, Fu Xiao, Xiaobai Chen, Lei Cui, and Shui Yu. Efficient virtual network embedding of cloud-based data center networks into optical networks. *IEEE Transactions on Parallel and Distributed Systems*, 32(11):2793–2808, 2021.

[52] Chinmaya Kumar Dehury and Prasan Kumar Sahoo. DYVINE: Fitness-based dynamic virtual network embedding in cloud computing. *IEEE Journal on Selected Areas in Communications*, 37(5):1029–1045, 2019.

[53] Aris Leivadeas, Chrysa Papagianni, and Symeon Papavassiliou. Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning. *IEEE Transactions on Parallel and Distributed Systems*, 24:1077–1086, 2013.

[54] Ilhem Fajjari, Nadjib Aitsaadi, Guy Pujolle, and Hubert Zimmermann. VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic. In *IEEE International Conference on Communications*, pages 1–6, 2011.

[55] Peiying Zhang, Haipeng Yao, Maozhen Li, and Yunjie Liu. Virtual network embedding based on modified genetic algorithm. *Peer-to-Peer Networking and Applications*, 12(2):481–492, 2019.

[56] Sen Su, Zhongbao Zhang, Alex X. Liu, Xiang Cheng, Yiwen Wang, and Xinchao Zhao. Energy-aware virtual network embedding. *IEEE/ACM Transactions on Networking*, 22(5):1607–1620, 2014.

[57] Chunxiao Jiang and Peiying Zhang. *VNE-HPSO Virtual Network Embedding Algorithm Based on Hybrid Particle Swarm Optimization*, pages 129–152. Springer Singapore, Singapore, 2021.

[58] Andreas Blenk, Patrick Kalmbach, Patrick van der Smagt, and Wolfgang Kellerer. Boost online virtual network embedding: Using neural networks for admission control. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 10–18, 2016.

[59] Hiren Kumar Thakkar, Chinmaya Kumar Dehury, and Prasan Kumar Sahoo. Muvine: Multi-stage virtual network embedding in cloud data centers using reinforcement learning-based predictions. *IEEE Journal on Selected Areas in Communications*, 38(6):1058–1074, 2020.

[60] 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.

[61] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

[62] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *International Conference on Machine Learning*, pages 42769–42789. PMLR, 2023.

[63] Hang Zhao, Qijin She, Chenyang Zhu, Yin Yang, and Kai Xu. Online 3d bin packing with constrained deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 741–749, 2021.

[64] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[65] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[66] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[67] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219, 2012.

[68] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, 2011.

[69] Wenbo Wang, Xiaolin Chang, Jiqiang Liu, and Bin Wang. Simulated annealing based resource allocation for cloud data centers. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, page 81–82, 2013.

[70] Wenzhe Wang, Peilin Hong, Defang Lee, Jianing Pei, and Lei Bo. Virtual network forwarding graph embedding based on tabu search. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, 2017.

[71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. volume 32, pages 8026–8037, 2019.

[72] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[73] Qihang Guo, Xibei Yang, Weiping Ding, and Yuhua Qian. Cross-graph interaction networks. *IEEE Trans. on Knowl. and Data Eng.*, 37(5):2341–2355, February 2025.

[74] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

[75] Runzhong Wang, Yang Li, Junchi Yan, and Xiaokang Yang. Learning to solve combinatorial optimization under positive linear constraints via non-autoregressive neural networks. *arXiv preprint arXiv:2409.04495*, 2024.

[76] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Appendix

A NFV-RA Problem Definitions

A.1 Basic Formulation of NFV-RA

Here, we provide the mathematical formulation of the basic cost minimization model, including constraints and the objective.

A.1.1 Constraint Conditions

During the embedding process of a VN $\mathcal{G}_v$ onto the PN $\mathcal{G}_p$, we need to decide two types of boolean variables: (1) $x_i^m = 1$ if virtual node n_v^m is placed in physical node n_p^i, and 0 otherwise; (2) $y_{i,j}^{m,w} = 1$ if virtual link $l_{m,w}^v = (n_v^m, n_v^w)$ traverses physical link $l_{i,j}^p = (n_p^i, n_p^j)$, and 0 otherwise. Here, m and w are identifiers for virtual nodes, while i and j are identifiers for physical nodes. A VN request is successfully embedded if a feasible mapping solution is found, satisfying the following constraints:

$$\sum_{n_p^i \in n_p} x_i^m = 1, \forall n_v^m \in n_v, \tag{3}$$

$$\sum_{n_v^m \in \mathcal{N}_v} x_i^m \leq 1, \forall n_p^i \in \mathcal{N}_p, \tag{4}$$

$$x_i^m C(n_v^m) \leq C(n_p^i), \forall n_v^m \in \mathcal{N}_v, n_p^i \in \mathcal{N}_p, \tag{5}$$

$$\sum_{n_p^i \in \Omega(n_p^k)} y_{i,k}^{m,w} - \sum_{n_p^j \in \Omega(n_p^k)} y_{k,j}^{m,w} = x_k^m - x_k^w, \forall l_{m,w}^v \in \mathcal{L}_v, n_v^k \in \mathcal{N}_p, \tag{6}$$

$$y_{i,j}^{m,w} + y_{j,w}^{m,w} \leq 1, \forall l_{m,w}^v \in \mathcal{L}_v, l_{i,j}^p \in \mathcal{L}_p, \tag{7}$$

$$\sum_{l_{m,w}^v \in \mathcal{L}_v} (y_{i,j}^{m,w} + y_{j,i}^{m,w}) B(l_{m,w}^v) \leq B((l_{i,j}^p)), \forall (l_{i,j}^p) \in \mathcal{L}_p. \tag{8}$$

Here, $\Omega(n_p^k)$ denotes the neighbors of n_p^k. Constraint (3) ensures that every virtual node is mapped to one and only one physical node. Conversely, constraint (4)limits each physical node to hosting at most one virtual node, thus enforcing a unique mapping relationship. Constraint (5) verifies that virtual nodes are allocated to physical nodes with adequate resources. Following the principle of flow conservation, constraint (6) guarantees that each virtual link (n_v^m, n_v^w) is routed along a physical path from n_p^i (the physical node where n_v^m is placed) to n_p^j (the physical node where n_v^w is placed). Constraint (7) eliminates the possibility of routing loops, thereby ensuring that virtual links are routed acyclically. Lastly, constraint (8) ensures that the bandwidth usage on each physical link remains within its available capacity. Overall, constraints (3,4,5) enforce the one-to-one placement and computing resource availability required in the node mapping process. And constraints (6,7,8) ensure the path connectivity and bandwidth resource availability required in the link mapping process.

A.1.2 Optimization Objective

Revenue-to-Consumption (R2C) ratio is a widely used metric to measure the quality of solution $S = f_{\mathcal{G}}(I)$. The objective function is maximize the resource utilization as follows:

$$\max \text{R2C}(S) = \varkappa \cdot (\text{REV}(S) / \text{COST}(S)), \tag{9}$$

where $\varkappa$ is a binary variable indicating the solution's feasibility; $\varkappa = 1$ for a feasible solution and $\varkappa = 0$ otherwise. When the solution is feasible, REV(S) represents the revenue from the VN, calculated as $\sum_{n_v \in \mathcal{N}_v} C(n_v) + \sum_{l_v \in \mathcal{L}_v} B(l_v)$. If $\varkappa = 1$, COST(S) denotes the resource consumption of PN, calculated as $\sum_{n_v \in \mathcal{N}_v} C(n_v) + \sum_{l_v \in \mathcal{L}_v} (|f_{\mathcal{L}}(l_v)| \times B(l_v))$. Here, $|f_{\mathcal{L}}(l_v)|$ quantifies the length of the physical path ρ_p routing the virtual link l_v.

A.2 Extensions in Emerging Networks

The application of NFV is pivotal in emerging network scenarios, which require additional considerations of their unique challenges. Particularly, we discuss several extended definitions of NFV-RA in popular scenarios to better align with their practical requirements.

Resource Heterogeneity In modern data center networks, computing resources are often heterogeneous, meaning physical nodes may have varying capacities in terms of CPU, GPU, memory, etc. To account for this, we extend the basic model by incorporating heterogeneous resources, where both virtual and physic nodes are associated with a set of computing resources $\mathcal{C}$. Thus, a physical node n_p for placing virtual node n_v must have enough resources across all types, i.e., $C(n_v) \leq C(n_p), \forall n_v \in \mathcal{N}_v, C \in \mathcal{C}, n_p = f_{\mathcal{N}}(n_v)$.

Latency Requirement In time-sensitive networks (e.g., edge computing and 5G), satisfying latency requirements is crucial. We consider the latency requirement of virtual link l_v as $D(l_v)$ and the incurred latency of physical link p_v as $D(l_p)$. The latency of physical path ρ_p that routes l_v should not exceed such specified threshold, i.e., $D(\rho_p) \leq D(l_v)$, where $D(\rho_p) = \sum_{l_p \in \rho_p} D(l_p)$.

Energy Efficiency Energy consumption is a significant concern in green data centers due to high sustainability and economic efficiency. Energy-efficient NFV-RA also considers the energy consumption minimization of physical infrastructure. We denote the energy consumed by the physical node n_p as $E(n_p)$, associated with its status (idle or active) and workload. The objective function is to optimize both resource utilization and energy consumption, formulated as: $\max -w_a \cdot \sum_{n_p \in \mathcal{N}_p} E(n_p) + w_b \cdot \text{R2C}(S)$, where w_a and w_b are weights of different objectives.

Note that these definitions extend the basic concepts of modeling, constraints, and objectives. Variations of NFV-RA for other scenarios can be easily derived using the approaches discussed above.

B Related Work

As network scenarios become increasingly diverse and complex (e.g., cloud, edge, 5G), many studies explore NFV-RA in emerging networks while addressing additional factors (e.g., latency, energy, heterogeneity). Despite these variations, many of these studies share a common core methodology. In this section, we focus on the methodological aspects and review the development of key approaches in NFV-RA. Then, we describe existing benchmarks to emphasize the gap between Virne. Finally, we compare NFV-RA with other COPs, highlighting the need for specific designs.

B.1 Traditional NFV-RA Algorithms

Earlier studies for the NFV-RA problem employed *exact solvers*, such as integer linear programming (ILP)[6] and mixed integer programming (MIP)[5], to find optimal solutions. However, their high computational complexity makes them impractical for real-world scenarios, especially in larger networks where dynamic service requests require rapid solutions. To address these limitations, *heuristic-based methods* emerged as alternatives, offering suboptimal yet computationally efficient solutions. Among these, node-ranking strategies stand out as a prominent approach [7, 8, 51, 9]. These strategies construct solutions by prioritizing the virtual and physical nodes based on specific metrics to guide the node mapping process, followed by link mapping. For instance, Node Resource Management (NRM) [8] ranks nodes by weighting both node and link resources, while the Node Essentiality Assessment (NEA) [9] incorporates topology connectivity into decision-making. Although reducing computational overhead, they rely heavily on manual design with domain-specific knowledge and lack adaptation to diverse scenarios, which can lead to inferior results in complex NFV-RA requirements. Furthermore, *meta-heuristics* have been adopted to solve NFV-RA by modeling the solution search as an evolutionary process [52–54]. These algorithms, such as Genetic Algorithms (GA) [55, 52] and Particle Swarm Optimization (PSO) [56, 57], explore the solution space by iteratively generating and evolving a population of candidate solutions. However, they are computationally expensive and hyperparameter-sensitive, limiting their practicality.

B.2 RL-based NFV-RA Algorithms

In recent years, machine learning (ML)-based methods have gained prominence in solving NFV-RA problems due to their superior performance and adaptability to dynamic network conditions [35]. Several works [58, 59] have explored predictive models trained on high-quality datasets to forecast the future service requests. However, acquiring labeled, high-quality datasets for large-scale and unseen network scenarios remains impractical, limiting their applicability. More dominantly, RL

Table 4: Summary of existing studies on RL-based NFV-RA algorithms. (a) Existing studies investigate RL-based NFV-RA algorithms in various network scenarios, such as cloud computing and latency-aware edge computing. (b) These studies employ different RL methods, such as DQN and PPO, to optimize customized neural policies, including CNN- or GNN-based architectures. (c) They also vary in implementation techniques, including whether they incorporate intermediate rewards in the reward function, use action masking mechanisms to prevent selecting nodes with insufficient resources, or leverage topological features as augmented inputs. (d) Additionally, we summarize the benchmarks used in these studies, noting whether they rely on existing benchmarks or use custom ones that are not publicly available.

	Network Scenario		**Core RL Framework**		**Implementation Techniques**			**Code Base**
	Network System	*Additional Considerations*	*Training Method*	*Neural Policy*	*Intermediate Rewards*	*Action Masking*	*Topological Features*	*Used Benchmark*
[26]	Edge	/	A2C	LSTM, Attention	✓	✗	✗	Not available
[38]	Edge	Energy, Latency	PG	CNN	✗	✓	✓	Not available
[39]	5G Slicing	Security	DQN	Heterogeneous GCN	✓	✗	✗	Not available
[40]	5G Slicing	Latency	PPO	MLP	✗	✓	✗	Not available
[41]	Internet of Things	Latency	PPO	GNN	✓	✓	✓	Not available
[42]	Internet of Things	Latency	DQN	CNN	✓	✗	✗	Not available
[43]	Satellite	Latency	DQN	MLP	✗	✗	✗	Not available
[27]	Data Plane	Latency	A3C	GCN	✓	✓	✓	Not available
[44]	Space-Air-Ground	Latency	PG	CNN	✗	✗	✗	Not available
[10]	Cloud	/	MCTS	/	✗	✗	✗	VNE-Sim
[12]	Cloud	/	PG	RNN	✗	✗	✗	Not available
[45]	Cloud	/	PG	GCN	✗	✗	✗	Not available
[46]	Cloud	Security	PG	CNN	✓	✗	✓	Not available
[47]	Cloud	Energy	PG	LSTM; Attention	✓	✗	✗	Self-implemented
[15]	Cloud	/	A3C	GCN, GRU	✓	✓	✗	VNE-Sim
[48]	Cloud	Energy	A2C	Attention	✓	✓	✗	**Virne**
[49]	Cloud	/	PG	MLP	✓	✓	✓	Not available
[16]	Cloud	/	PG	CNN	✓	✗	✗	Not available
[14]	Cloud	/	A3C	GCN	✓	✓	✓	VNE-Sim
[11]	Cloud	Latency	PG	MLP	✓	✗	✗	Self-implemented
[13]	Cloud	/	A3C	GCN, GRU	✗	✓	✓	**Virne**
[18]	Cloud	/	PPO	GNN	✓	✓	✓	**Virne**
[19]	Cloud	Heterogeneity	PPO	Cross-GCN	✓	✓	✓	**Virne**
[17]	Cloud	/	PPO	Edge-aware GAT	✓	✓	✓	**Virne**
[50]	Cloud, Edge	Latency	PPO	Heterogeneous GAT	✗	✓	✓	**Virne**

has demonstrated its promise for NFV-RA tasks [10–19], mainly due to its label-free nature and adaptation to handle dynamics of network. RL-based methods model NFV-RA as a Markov Decision Process (MDP), allowing an agent to learn optimal policies through iterative interactions with the environment. In general, we unified existing RL-based methods for NFV-RA under a framework with three core components: MDP modeling, policy architecture, and training methods. These methods conceptualize the node mapping allocation process as a sequential decision-making task, where physical nodes are incrementally selected to host VNFs. To execute these decisions, various policy network architectures are designed to represent the network state and generate corresponding actions. Then, they leverage a specific RL method to optimize the policy network with collected experience during interactions. For example, [11] utilized a multilayer perceptron (MLP) and trained it using the policy gradient (PG) algorithm, while the work [14] combined MLPs with graph convolutional networks (GCNs) and employed the asynchronous advantage actor-critic (A3C) algorithm. While RL-based methods offer notable advantages in terms of performance and adaptability, there is a lack of in-depth analysis of the impact of specific model design choices and consistent performance comparison. Moreover, systematically identifying the remaining significant challenges in RL-based NFV-RA is critical to open opportunities for future exploration and innovation.

B.3 Existing NFV-RA Benchmarks

For the NFV-RA problem, several benchmarks have been developed for simulation and evaluation. As summarized in Table 1, the most notable existing benchmarks include: VNE-Sim[2] [21], ALEVIN [3]

[2] https://tehreemf.wixsite.com/vne-sim
[3] https://sourceforge.net/p/alevin/wiki/home/

[22], ALib[4] [22], SFCSim[5] [24], and Iflye[6] [25]. For instance, VNESim[21] supports three heuristic algorithms and focuses exclusively on cloud-based simulations, evaluating solutions based on their effectiveness. Similarly, SFCSim [24], incorporates three additional heuristic methods, continuing the trend of focusing on cloud-based scenarios However, most of these benchmarks are limited to cloud-based simulations and lack the flexibility to accommodate more modern network scenarios, such as edge computing or 5G environments. Moreover, these benchmarks generally implement only a few exact and heuristic algorithms and focus mainly on effectiveness evaluation. These limitations highlight the need for more comprehensive benchmarks that can assess algorithms across a broader range of network environments and evaluation perspectives.

B.4 RL for Combinatorial Optimization

RL has gained significant attention for solving Combinatorial Optimization Problems (COPs), where the goal is to learn high-quality solutions [60]. These problems span various domains, including the Traveling Salesman Problem (TSP) [61], Vehicle Routing Problem (VRP) [62], and bin packing [63]. RL-based approaches to solving these problems generally fall into two categories: construction methods that build a solution incrementally from scratch and improvement methods that start with an initial solution and iteratively refine it. Compared with these classic COPs, the NFV-RA problem presents unique challenges, highlighting the need for specialization of RL approach design. First, NFV-RA requires real-time decision-making to meet strict network service requirements. NFV-RA algorithms must execute decisions almost immediately, making construction methods more suitable to incrementally build solutions within time constraints. Second, NFV-RA operates in highly dynamic environments where both service demands and network resources fluctuate in real time. This variability requires RL methods that can adapt to changes in network topologies, resource availability, and demand patterns. Lastly, the state space of NFV-RA is highly complex, involving intricate interactions such as cross-graph mapping and bandwidth-constrained routing. These factors require RL methods to navigate a multidimensional decision space, accounting for diverse resource types and connectivity requirements.

C Benchmark Details

C.1 Simulation Configuration

Virne offers a highly customizable simulation framework designed to accommodate the diverse and complex nature of modern network environments. As illustrated in Figure 4, users can define the network scenarios and conditions within the simulator through configuration files. These configurations cover two key components: PN settings and VN requests, providing users the flexibility to simulate a variety of network scenarios.

Network Topologies Virne provides a range of customizable network topologies to simulate different network conditions. Users can select between: *(a) synthetic topologies* that are generated using algorithms such as Waxman and FatTree, which provide structured and predictable network designs for controlled testing; and *(b) realistic topologies* that are sourced from SDNLib and the Topology Zoo, offering more realistic and complex network structures that mimic real-world environments. Users can specify both the PN and VN topologies through their configuration files, allowing the reflection of various network scales and topological characteristics.

Resource Availability Virne supports the customization of resource availability across multiple levels of the network, including nodes, links, and graph. This flexibility enables simulations that represent different resource conditions, such as: *(a) resource types*: Users can define various types of resources, including computing resources (e.g., CPU, GPU, memory) and network resources (e.g., bandwidth). This allows for the modeling of diverse resource requirements in both PN and VN settings; *(b) availability distribution*: The resources can be distributed across the network using different statistical models, such as uniform or exponential distributions, which reflect real-world

[4] https://github.com/vnep-approx/alib
[5] https://github.com/SFCSim/SFCSim
[6] https://github.com/Echtzeitsysteme/iflye

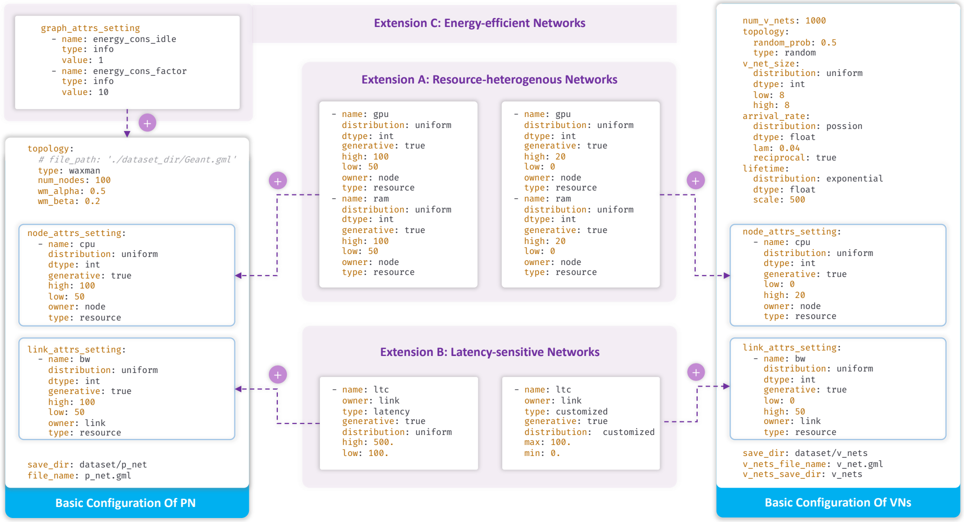


Figure 4: An example of basic configurations on both PN and VNs, along with their extensions. By adding specific settings on the levels of node, link, or graph, Virne can be easily extended to support emerging networks with additional awareness.

variations in resource supply and demand. This level of customization provides users with the ability to model networks with differing levels of resource availability.

Service Requirements Virne allows users to define specific service requirements, which add complexity and realism to the simulations. These requirements can be tailored to address a wide range of network use cases, reflecting varying demands and constraints. As illustrated in Figure 4, we introduce three key types of service awareness that can be customized:

- *Heterogeneous resources*. To configure this, users can add two new types of node resources (e.g., CPU, GPU) into the *node_attrs_setting* section of the PN and VN configuration files.
- *Latency constraints*. By introducing the latency into the *link_attrs_setting* section of the PN and VN configuration files, users can specify maximum latency thresholds for virtual links and the physical paths used for routing.
- *Energy efficiency*. By including energy consumption parameters into the *graph_attrs_setting* section of the PN configuration files, users can simulate green data centers and optimize resource allocation with sustainability in mind.

Through these service requirements, Virne enables the simulation of networks with diverse operational demands, providing users with the flexibility to model complex and dynamic network systems.

C.2 MDP Variation for NFV-RA

Apart from the MDP formulations introduced in Section 3.3.1, several extended MDPs for NFV-RA are also proposed to handle the specific aspects of NFV-RA problems.

C.3 Implemented NFV-RA Algorithms

C.3.1 RL-based Algorithm Implementations

Our RL-based implementations are structured around core components: RL training algorithms that define the learning paradigm, neural policy architectures that parameterize the agent's decision-making function, and additional techniques that can enhance learning or address specific challenges in NFV-RA. This modularity allows researchers to easily experiment with different combinations and contribute new components.

RL training Methods Virne supports several foundational and advanced RL training algorithms, which are used to guide the learning process of the neural policies.

- *Monte Carlo Tree Search (MCTS) [10]* is a planning algorithm that explores the decision space by building a search tree, balancing exploration and exploitation to find action sequences.
- *Policy Gradient (PG or REINFORCE) [64]* is a policy-based RL method that learns a parameterized policy to maximize expected returns, i.e., solution quality of NFV-RA.
- *Asynchronous Advantage Actor-Critic (A3C) [31]* is an actor-critic-based RL algorithm that uses multiple parallel actors, each interacting with its own copy of the environment. It learns both a policy (actor) and a value function (critic) to estimate the advantage of taking certain actions, leading to more efficient learning than pure PG methods.
- *Proximal Policy Optimization (PPO) [32]* is a popular RL method known for its stability, ease of implementation, and strong empirical performance across a wide range of tasks. It achieves policy optimization stability by using a clipped surrogate objective.
- *Deep Q-Network (DQN) and variants [65]* are value-based RL methods that aim to learn the state-action value space. However, NFV-RA often involves large, structured action spaces not directly amenable to these methods.

Neural Policy Architectures The neural policy architecture defines how the RL agent perceives the environment (state representation) and decides on actions. Virne implements a range of architectures to capture attribute and structural information in both the PN and VN.

- *Multi-Layer Perceptron (MLP)-based Policy [40, 43, 49, 11].* It concatenates the features of the current to-be-placed virtual node with every physical node's features. Then, it fed them into multiple fully connected layers and outputs probabilities for selecting each physical node for placement or parameters for action selection.
- *Convolutional Neural Network (CNN)-based Policy* [38, 42, 44, 46, 16]. They use CNN to process graph-structured data by treating node features and adjacency information as grid-like inputs. Similar to the MLP, virtual link demands are integrated as node features.
- *Attention-based Policy*[26, 47]. It first embeds the features of the current to-be-placed virtual node to form a query. Then, the features of each physical node are similarly embedded to form keys and values. The attention mechanism then computes a weighted sum of physical node values based on their compatibility with the virtual node query.
- *Graph Convolutional Network (GCN)-based policy [27, 45, 14]* . It uses GCN to learn node representations by aggregating information from their local neighborhoods. In Virne, GCNs are used to process the PN topology, often with virtual node demands incorporated as additional node features on the PN nodes.
- *Graph Convolutional Network (GCN)-based policy*. It uses GCN to learn node representations by aggregating information from their local neighborhoods. GCNs is mainly used to process the PN topology, often with virtual node demands incorporated as additional node features on the PN nodes.
- *GCN & Sequence-to-Sequence (GCN&S2S) Policy [13].* This hybrid architecture combines GCNs for graph representation learning with a Sequence-to-Sequence (S2S) model [66], an RNN-based encoder-decoder with attention, to generate an ordered sequence of actions.
- *GAT&S2S Policy [17].* Similar to GCN&S2S, but replaces the GCN encoder(s) with GAT(s).
- *Graph Attention Network (GAT)-based policy* [17]. GATs extend GCNs by incorporating attention mechanisms into the neighborhood aggregation process. Particularly, it also incorporates edge features (e.g., link bandwidth, latency) into the GAT attention mechanism.
- *Dual GCN (DualGCN)-based policy [19].* This architecture uses two separate GCNs, i.e., one to embed the topology and features of the VN, and another for the PN. The embeddings from both graphs are then combined into an MLP to make placement.
- *Dual GAT (DualGAT)-based Policy [19]*: Similar to DualGCN, but it utilizes GATs for both VN and PN embedding.
- *Heterogeneous Graph Attention Network (HeteroGAT)-based Policy [39, 50].* Recent studies model NFV-RA instances as a heterogeneous graph, where virtual nodes, physical nodes, virtual links, and physical links are different types of nodes/edges with distinct features.

Table 5: Implemented Traditional NFV-RA Algorithms

	Core Strategy
MIP [5]	Mixed-Integer Programming (MIP)
R-Rounding [67]	Random Rounding
D-Rounding [67]	Deterministic Rounding
RW-Rank [68]	Random Walk (RW)
GRC-Rank [7]	Global Resource Control (GRC)
NRM-Rank [8]	Node Resource Management (NRM)
NEA-Rank [9]	Node Essentiality Assessment (NEA)
PL-Rank [51]	Priority of Location (RL)
GA-Meta [55]	Genetic Algorithm (GA)
PSO-Meta [57]	Particle Swarm Optimization (PSO)
ACO-Meta [54]	Ant Colony Optimization (ACO)
SA-Meta [69]	Simulated Annealing (SA)
TS-Meta [70]	Tabu Search (TS)

Additionally, HeteroGAT introduces the cross-graph connection links to represent the historical mapping status, i.e, if a virtual node is placed onto a physical node, a heterogeneous link will be created.

Additional Implementation Techniques

- *Reward Function Design*. NFV-RA is characterized by its combinatorial space and complex constraints for NFV-RA, which may present a challenging sparse reward problem. Virne allows exploration of different reward structures.
  - *No Intermediate Reward (NoIR)*. The agent only receives a terminal reward based on the final $R2C(S)$ if the entire VN is successfully embedded, and 0 otherwise.
  - *Fixed Intermediate Reward (FIR)*. The agent receives a small positive fixed reward (ImR_value) for each successful virtual node placement and link routing step. A negative reward is given for failed steps. The final $R2C(S)$ is added to the sum of intermediate rewards if the overall embedding is successful.
  - *Adaptive Intermediate Reward (AIR)*. A specialized version of FIR considers $\frac{1}{|\mathcal{N}_v|}$ as an intermediate reward value, where $|\mathcal{N}_v|$ is the number of virtual nodes in the VN. This normalizes the scale of intermediate rewards based on the complexity of the VN.
- *Feature Engineering Combinations*. Raw node/link attributes and basic topology might not be sufficient for optimal RL performance. Some studies use engineered features to augment the input to neural policies. These include:
  - *(a) Node Embedding Status*. Binary flags indicating whether a physical node is currently hosting a virtual node or whether a virtual node has already been placed.
  - *(b) Topological Features*. Standard graph centrality measures computed for both PN nodes and VN nodes, i.e., degree, closeness, betweenness, and eigenvector.
- *Action Masking Mechanism*. During training, not all actions (e.g., placing a virtual node on a physical node) are valid at every step due to resource constraints or additional service requirements. Action masking is a technique where invalid actions are explicitly disallowed by placing the selection probability with zero.

C.3.2 Implemented Traditional Algorithms

Virne implements 10+ traditional algorithms, which can be broadly categorized into three primary types: exact and rounding methods, node ranking-based approaches, and meta-heuristic methods. We summarize these key algorithms and their core strategies in Table 5. Next, we elaborate on each category and introduce implemented algorithms as follows.

Exact and rounding methods for NFV-RA either guarantee optimal solutions or provide approximate solutions through rounding techniques, mainly based on mathematical solvers.

- *MIP* [5]: Mixed-Integer Programming (MIP) is an exact method that provides optimal solutions by solving a system of linear equations with integer constraints.
- *R-Rounding* [67]: A rounding method that applies random rounding to generate approximate solutions for NFV-RA.
- *D-Rounding* [67]: This method uses deterministic rounding to map continuous variables to discrete values while providing an approximation to the optimal solution.

Node ranking-based methods for NFV-RA that first prioritize nodes using different strategies for node mapping, then execute the link mapping stage with the shortest path algorithm.

- *NRM-Rank* [8]: A heuristic method based on Node Resource Management (NRM) metric. This approach ranks virtual and physical nodes before mapping with a greedy method.
- *GRC-Rank* [7]: This method uses a Global Resource Control (GRC) strategy based on random walks for node ranking. After ranking, the nodes are mapped accordingly.
- *NEA-Rank* [9]: It ranks nodes using the Node Essentiality Assessment (NEA) metric.
- *RW-Rank* [68]: A node ranking approach based on random walks to estimate node priorities in the embedding process.
- *PL-Rank* [51]: This algorithm considers node proximity and evaluates paths comprehensively, ranking nodes based on their location priorities and the overall efficiency of the physical infrastructure.

Meta-heuristic methods for NFV-RA employ nature-inspired processes to iteratively improve solutions. Virne implements the following meta-heuristics for NFV-RA:

- *GA-Meta* [55]: A meta-heuristic method based on genetic algorithms. It models each node mapping solution as a chromosome and iteratively explores the solution space by simulating the process of genetic selection and evolution.
- *PSO-Meta* [56]: A meta-heuristic using Particle Swarm Optimization (PSO), where particles explore the NFV-RA solution space by adjusting their positions based on their own and neighbors' experiences.
- *ACO-Meta* [54]: A meta-heuristic based on Ant Colony Optimization (ACO) that simulates ant foraging behavior to explore the NFV-RA solution space and update pheromone trails to guide the search.
- *SA-Meta* [69]: A meta-heuristic using Simulated Annealing (SA) to approximate the global optimum, exploring the NFV-RA solution space with both upward and downward transitions to escape local minima.
- *TS-Meta* [70]: A meta-heuristic based on Tabu Search (TS), which uses a memory structure to store visited solutions, promoting diversity and avoiding local optima.

C.4 Evaluation Metric Definitions

We provide the definitions of key evaluation metrics commonly used to evaluate the NFV-RA algorithms as follows.

- **Request Acceptance Rate (RAC)** measures the proportion of VN requests that the system successfully accepts. Formally, it is given by

$$\mathrm{RAC} = \frac{\sum_{t=0}^{\mathcal{T}} |\tilde{\mathcal{I}}(t)|}{\sum_{t=0}^{\mathcal{T}} |\mathcal{I}(t)|} \times 100, \tag{10}$$

  where $\mathcal{T}$ denotes the total operational time of the network system. $\mathcal{I}(t)$ is the set of all VN requests arriving at time slot t, and $\tilde{\mathcal{I}}(t)$ is the subset of those requests that are accepted. The notation $|\cdot|$ denotes the cardinality of a set.

- **Long-term Revenue-to-cost Ratio (LRC)** evaluates the economic efficiency of the system by comparing revenue to resource consumption. It is formulated as

$$\mathrm{LRC} = \frac{\sum_{t=0}^{\mathcal{T}} \sum_{I \in \tilde{\mathcal{I}}(t)} \mathrm{REV}(S) \times \varpi}{\sum_{t=0}^{\mathcal{T}} \sum_{I \in \tilde{\mathcal{I}}(t)} \mathrm{COST}(S) \times \varpi} \times 100, \tag{11}$$

Table 6: Key simulation parameters and their default values.

Symbol	Description	Default Distribution
$\mathcal{X}_{\mathcal{N}_p}$	Physical Node Resources	$\mathcal{U}(50, 100)$
$\mathcal{X}_{\mathcal{L}_p}$	Physical Link Bandwidth	$\mathcal{U}(50, 100)$
$\mathcal{X}_{\mathcal{G}_v}$	VN Request Size	$\mathcal{U}(2, 10)$
$\mathcal{X}_{\mathcal{N}_v}$	Virtual Node Resources	$\mathcal{U}(0, 20)$
$\mathcal{X}_{\mathcal{L}_v}$	Virtual Link Bandwidth	$\mathcal{U}(0, 50)$
$\mathcal{X}_{\mathcal{I}}$	VN Arrival Rate	Poisson(η)

where $S = f_G(I)$ as before, and REV(S) and COST(S) denote the revenue obtained and resources consumed by the embedding, respectively.

- **Long-term Average Revenue (LAR)** quantifies the total revenue generated over a period, reflecting the economic benefits of processing VN requests. It is defined as

$$\text{LAR} = \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}} \sum_{I \in \tilde{\mathcal{I}}(t)} \text{REV}(S) \times \varpi, \tag{12}$$

where $E = f_G(I)$ represents the embedded VN request, and ϖ is the lifetime of the corresponding VN.

- **Average Solving Time (AST)** indicates the efficiency of the NFV-RA algorithm by measuring the average consumed time (in seconds) for solving one simulation run.

D Experiment Details

D.1 Experimental Setup

We summarize the key simulation parameters in Table 6, including the settings of PN and VNs.

D.1.1 Implementation Settings

We provide the details on algorithm implementation (e.g., neural networks and RL training), experimental methods on training and testing, and computing resources.

Algorithm Implementation. The implementation of the RL-based NFV-RA algorithm is based on PyTorch [71]. Regarding neural policies (e.g., MLP, CNN, GNN, etc), we generally set their layers to 3 and the hidden dimension of neural networks to 128. For RL optimization, we set the reward discount factor λ to 0.99 and use the Adam optimizer with a learning rate of 0.001 and a batch size of 128. The action section during training and testing follows a sampling strategy and a greedy strategy, respectively. Additionally, we employ the k-shortest paths algorithm with $k = 10$ for link routing, selecting the shortest physical path that meets the bandwidth requirements for each virtual link.

Training and Testing. For RL-based methods, we train policies for each average arrival rate η, with random initialization of seeds in every simulation. The number of training epochs is set to 50, which is sufficient for all algorithms to converge. Typically, training a deep RL-based NFV-RA method completes within 5 hours. During testing, we evaluate the performance of all algorithms by repeating each test with 10 different random seeds (i.e., 0, 1111, 2222, . . ., 9999) for each average arrival rate η, thereby ensuring statistical significance.

Computing Resources. We conducted all experiments on a Linux server running Ubuntu 22.04.2 LTS. This server is equipped with 8 $\times$ NVIDIA H20 GPU, 128 $\times$ AMD EPYC 9K84 96-Core Processor, and 1.2 TB of memory.

Table 7: Key Information on Evaluated Network Topologies

Network	Node Count	Link Count	Network Density
WX100	100	500	0.05
WX500	500	13,000	0.1042
GEANT	40	64	0.0821
BRAIN	161	166	0.0129

D.1.2 Evaluated Physical Topologies

To comprehensively evaluate the performance of NFV-RA algorithms, we conducted experiments using both simulated and real-world network topologies from SNDlib[7], widely adopted in existing studies. These topologies cover a wide range of network scales and densities, ensuring that the evaluation reflects diverse operational scenarios. Below, we describe the characteristics of the evaluated PN topologies.

WX100 is a medium-scale network generated by the Waxman method [36]. It consists of 100 nodes connected by approximately 500 links, resulting in a density of 0.05.

WX500 is a larger variant of the Waxman topology, extending the scale to simulate large-scale network systems. With 500 nodes and around 1300 links, it exhibits a higher density of 0.1042.

GEANT is a well-known academic research network, designed for high-speed data transfer across Europe. It consists of 40 nodes interconnected by 64 edges, a density of 0.0821.

BRAIN is a high-speed data network for scientific and cultural institutions in Berlin. It consists of 161 nodes and 166 edges, with a density of 0.0129, which is the largest topology in SDNLib.

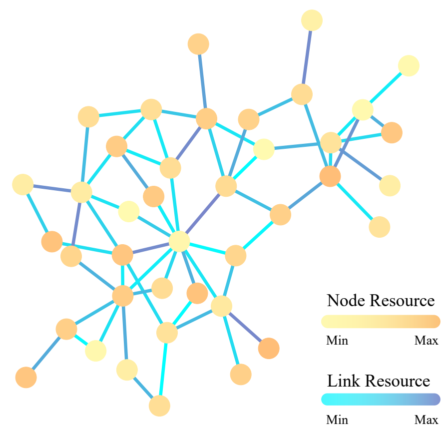


Figure 5: GEANT topology visualization.

We summarize the key characteristics of these topologies in Table 7. Furthermore, we provide a visualization of the topology of GEANT in Figure 5 to better understand the topological information.

D.2 Generalization on Network Conditions

Network conditions in real-world deployments are characterized by their inherent complexity and continuous evolution, including fluctuations in request frequencies and varying resource demands. Therefore, evaluating the generalization capabilities of trained NFV-RA policies is critical to ensure their adaptability and robust performance in diverse and dynamic network environments. Virne facilitates this by allowing pre-trained models to be tested under conditions different from their initial training setup. In this section, we investigate the generalization of various NFV-RA algorithms to two key aspects, i.e., varying traffic rates and fluctuating demand distributions. The algorithms are pre-trained under the default settings described in the main paper (Section 4.1 on Experimental Setup), and then evaluated under these new conditions. In the following experiments, to ensure figure clarity, we selected NFV-RA methods that demonstrated strong performance across various NFV-RA types in the main paper (Section 4.2.2 on Effectiveness in Online Environments).

D.2.1 Evaluation on Varying Traffic Rates

To assess how well different NFV-RA algorithms adapt to changes in network load, we evaluate their performance under a range of VN request arrival rates (η). The pre-trained policies, originally trained

[7]SNDlib is a well-known open-source library for telecommunication network design that offers a collection of realistic network system topologies. We use network topologies like GEANT and BRAIN from SNDlib. You can access this resource at https://sndlib.put.poznan.pl, though the specific licensing terms aren't clearly stated.

with specific η values for each topology (WX100 with $\eta = 0.16$, GEANT with $\eta = 0.016$, BRAIN with $\eta = 0.004$), are tested across a spectrum of η values. For WX100, this range is from 0.04 to 0.28; for GEANT, from 0.004 to 0.028; and for BRAIN, from 0.001 to 0.007. This setup simulates scenarios from low to high network congestion, while ensuring the comparability of results. The results across WX100, GEANT, and BRAIN topologies are presented in Figure 6.

As illustrated in Figure 6 (a), (d), and (g), there is a general downward trend in the Request Acceptance Rate (RAC) for all algorithms as the average arrival rate η increases across all three topologies. This is an expected outcome, as higher traffic intensity leads to increased competition for finite physical network resources, inevitably resulting in more VN request rejections. For instance, on WX100, the RAC for PPO-DualGAT decreases from 0.99 at $\eta = 0.04$ to 0.781 at $\eta = 0.16$, and further to 0.66 at $\eta = 0.28$. In comparison, MCTS drops notably.

Regarding the Long-term Revenue-to-Cost ratio (LRC), shown in Figure 6 (b), (e), and (h), we observe that ❺ *many RL-based algorithms, particularly PPO-DualGAT and PPO-DualGCN, demonstrate remarkable stability in LRC across varying traffic rates.* For example, on WX100, PPO-DualGAT's LRC remains very stable, i.e., 0.778 at $\eta = 0.04$, 0.737 at $\eta = 0.16$, and 0.746 at $\eta = 0.28$. This indicates that while they accept fewer requests under high load, the quality of accepted embeddings in terms of resource efficiency remains consistently high. In contrast, MCTS's LRC on WX100 declines more significantly from 0.543 ($\eta = 0.04$) to 0.444 ($\eta = 0.16$) and 0.410 ($\eta = 0.28$). Traditional heuristics like PL-Rank also show relatively stable LRC (e.g., on WX100, 0.662 at $\eta = 0.04$, 0.668 at $\eta = 0.16$, 0.650 at $\eta = 0.28$) but generally at a lower level than top RL performers.

The results on Long-term Average Revenue (LAR) are depicted in Figure 6 (c), (f), and (I). Generally increasing with η, most algorithms process more VN requests. Algorithms like PPO-DualGAT and PPO-DualGCN consistently achieve higher LAR across the range of η values, showcasing their superior capability in maximizing revenue even as conditions change.

These experiments highlight that ❻ *while some algorithms (often sophisticated RL policies like PPO-DualGAT) exhibit graceful performance degradation and maintain high efficiency (LRC), others are more sensitive to load changes.* This allows researchers and practitioners to select algorithms that are well-suited to the anticipated traffic dynamics of their target environments. For instance, for networks expecting highly variable loads, algorithms demonstrating flatter RAC and LRC curves across η would be preferable. The consistent outperformance of dual GNN architectures (PPO-DualGAT and PPO-DualGCN) further corroborates the findings in the main paper regarding their effectiveness.

D.2.2 Evaluation on Fluctuating Demand Distribution

In practical NFV environments, the characteristics of VN requests, such as the number of virtual nodes (VN size) or their resource requirements, can change over time due to evolving service demands. To simulate such dynamic conditions and evaluate algorithmic adaptability, we subject pre-trained policies to a sequence of VN requests where the underlying distribution of these requests' characteristics changes. Figure 7 illustrates the performance trends during a simulation run with these changing demands. The simulation comprises 1000 VN requests, divided into four distinct sub-groups (250 requests each). Compared to the default simulation settings, we modified one parameter related to the distribution of VN resource demand or VN node size for each subgroup: For the first group, the VN node and link resource distributions were changed to [0, 30] and [0, 75], respectively. For the second group, the VN node and link resource distributions were adjusted to [0, 40] and [0, 100], respectively. For the third group, the VN node size distribution was changed to [2, 15]. For the fourth group, the VN node size distribution was altered to [2, 20].

As illustrated in Figure 7 (a), (d), and (g) for RAC, most algorithms exhibit an initial sharp decrease, particularly as the simulation transitions through phases with increasing VN resource demands. This phase quickly consumes available physical resources, leading to lower acceptance rates. As the simulation progresses into phases where VN sizes increase, the RAC for many algorithms continues to decline or stabilizes at a lower level. This is because larger and more complex VNs are inherently more challenging to embed. Notably, ❼ *advanced RL methods like PPO-DualGAT and PPO-DualGCN tend to maintain a higher RAC compared to simpler MLP-based policies or traditional heuristics throughout these fluctuations, indicating better adaptation.*

The LRC metric, shown in Figure 7 (b), (e), and (h), also reflects the algorithms' adaptability. Many algorithms show a dip in LRC when the demand characteristics change, but some, like PPO-DualGAT

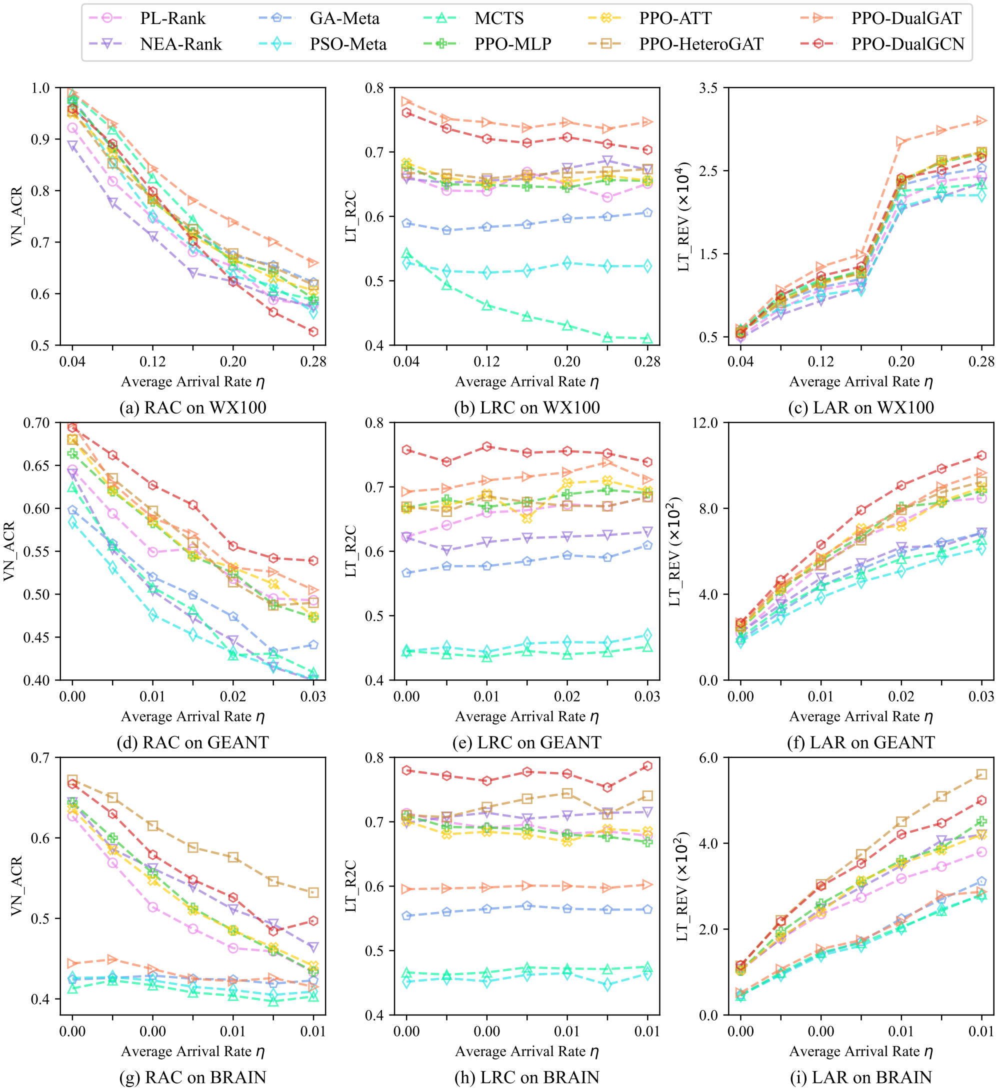
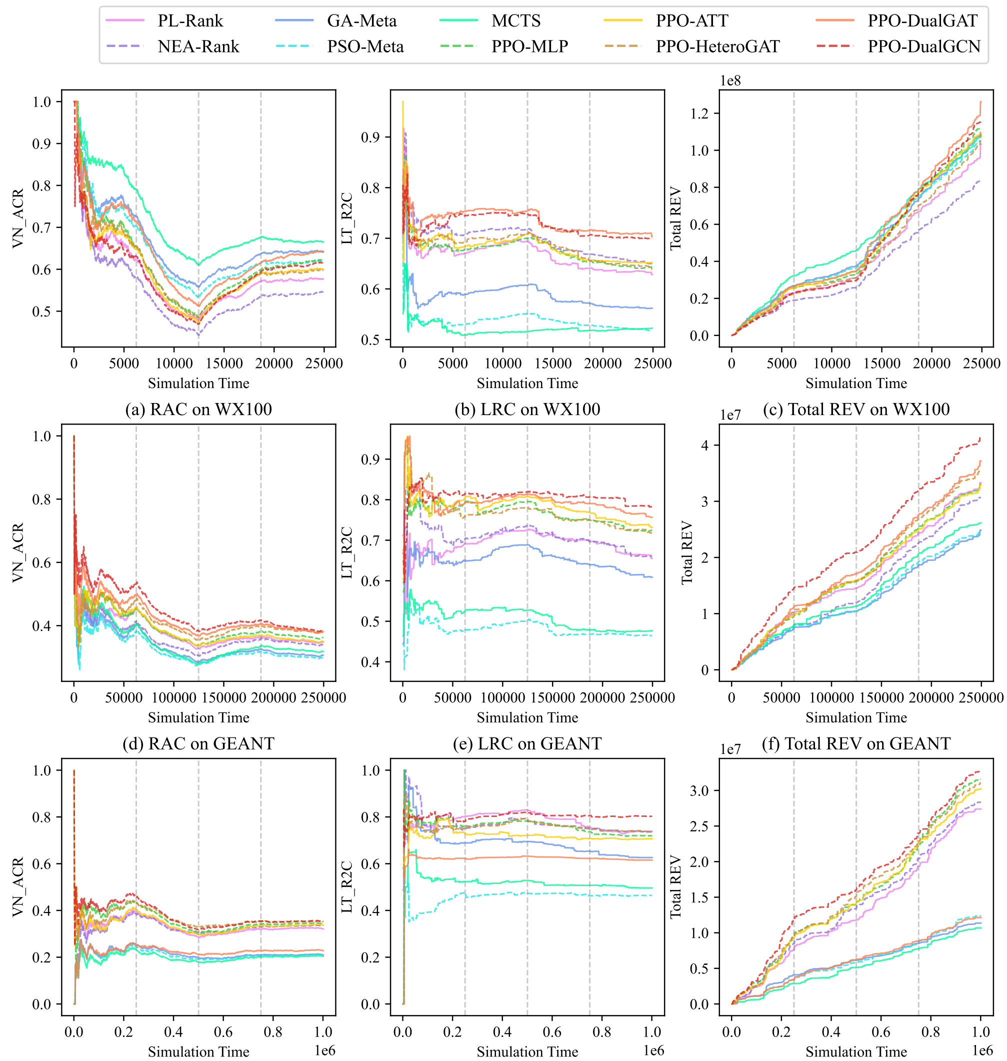


Figure 6: Results on the generalization study on varying traffic rates.

and PPO-DualGCN, manage to sustain a relatively higher LRC. For WX100 ($\eta = 0.04$) under changeable demands, PPO-DualGAT records an LRC of 0.703 and PPO-DualGCN an LRC of 0.700, whereas PPO-MLP has an LRC of 0.643. This suggests they are better at finding resource-efficient embeddings even when faced with unfamiliar or more challenging VN request types.

The Total Revenue (i.e., cumulative LAR), depicted in Figure 7 (c), (f), and (i), shows a clear differentiation in the algorithms' ability to accumulate revenue under these dynamic conditions. Algorithms that adapt well and maintain higher RAC and LRC will naturally accrue more total revenue. PPO-DualGAT and PPO-DualGCN consistently demonstrate a steeper accumulation of total revenue. For WX100 ($\eta = 0.04$) under changeable demands, PPO-DualGAT achieves a long-term average time revenue of 5080.33, compared to 4510.80 for PPO-MLP. This underscores their robustness and effectiveness in dynamic environments.

These results demonstrate that policies trained on one specific distribution of VNs may not always generalize perfectly to others. This highlights ❽ *the importance of generalization not just to varying load, but also to varying types of demand*. For practical systems where service characteristics can evolve, choosing algorithms that demonstrate robustness in such fluctuating scenarios, as identifiable through Virne, is important.



(a) RAC on WX100 (b) LRC on WX100 (c) Total REV on WX100

(d) RAC on GEANT (e) LRC on GEANT (f) Total REV on GEANT

(g) RAC on BRAIN (h) LRC on BRAIN (i) Total REV on BRAIN

Figure 7: Results on the generalization study on fluctuating demand distribution.

D.3 Solvability via Offline Evaluation

Traditional online testing of NFV-RA algorithms, while reflecting real-world scenarios, hinders direct comparisons of intrinsic algorithmic solvability due to the continuously evolving state of the PN and dynamic VN arrivals and departures. It becomes challenging to discern whether failures or poor performance stem from an algorithm's inherent limitations or from transient, potentially unsolvable, network states. To address this, Virne provides a controlled offline evaluation environment. This environment utilizes a series of static instances, each comprising a PN and a VN, to assess an algorithm's ability to find high-quality, feasible solutions under reproducible conditions. This section focuses on evaluating the solution quality, specifically the Revenue-to-Cost (R2C) ratio, achieved by different algorithms for VNs of varying sizes. The evaluation is conducted on a representative PN, WX100. The results are visualized in Figure 8, which presents a heatmap of average solution quality (R2C ratio) for VN sizes ranging from 2 to 10 nodes.

As shown in Figure 8, there is a general trend of decreasing solution quality as the size of the VN increases. This is expected, as larger VNs typically have more complex topologies and higher resource demands, making them more challenging to embed efficiently. Additionally, we can observe that ❽ *RL-based methods with advanced graph representations, particularly PPO-DualGAT and PPO-DualGCN, demonstrate superior solvability across most VN sizes.* For instance, PPO-DualGAT consistently achieves high R2C ratios. These algorithms are often highlighted in red (best-performing)

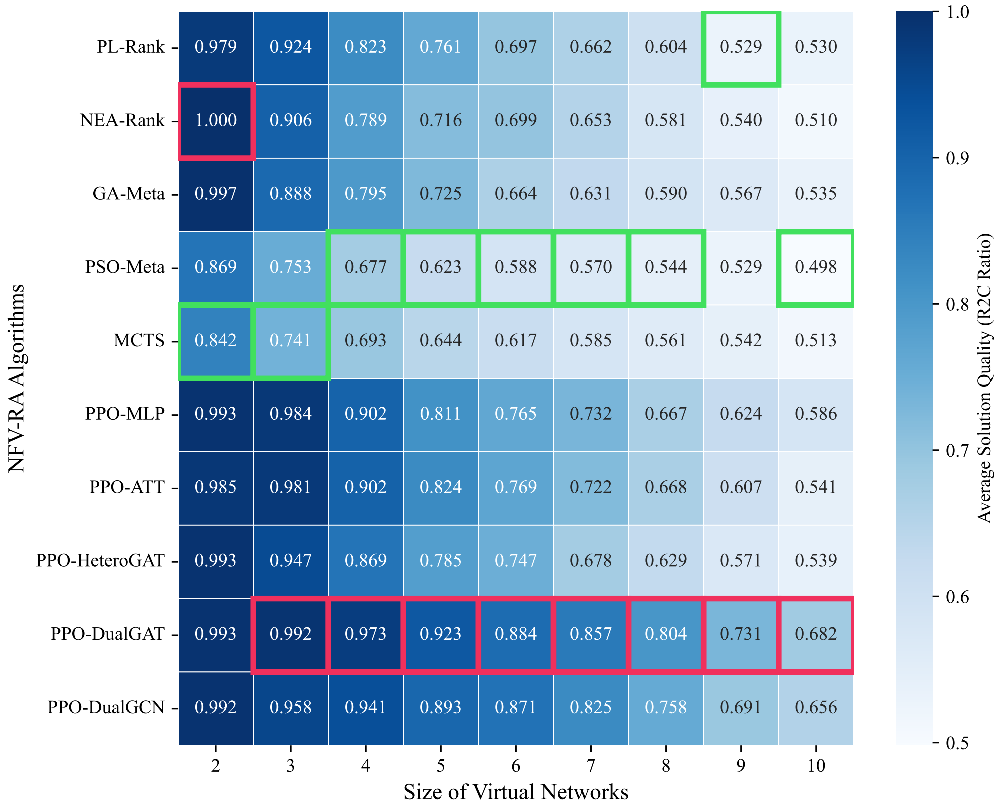


Figure 8: Results on the solvability study. For each size of VN, we highlight the worst-performing solver in green and the best-performing solver in red.

in Figure 8 for larger VN sizes. Additionally, ❾ *Simpler RL policies like PPO-MLP also show good performance, especially for smaller VNs, but degrade more noticeably for larger VNs due to their limited representation ability*. Meanwhile, traditional heuristic algorithms show mixed results. NEA-Rank performs exceptionally well for very small VNs, achieving an R2C of 1.0 for 2-node VNs. However, its performance, along with PL-Rank, tends to be lower than the top RL methods as VN size increases. Meta-heuristic algorithms like PSO-Meta tend to underperform in terms of solution quality in this offline setting, especially for larger VNs.

This study allows for a granular understanding of each algorithm's fundamental problem-solving strength (solvability) concerning VN complexity, isolated from the dynamics of online environments. The results indicate that ❿ *while some algorithms excel with smaller VNs, their performance can degrade significantly as complexity increases*. This detailed solvability analysis helps researchers identify the strengths and weaknesses of different algorithmic approaches for varying scales of the problem. For instance, the superior performance of dual GNN architectures across different VN sizes suggests their strong capability in capturing complex relationships for effective embedding. Practitioners can leverage these insights to select algorithms that are best suited for the specific types and sizes of VNs they anticipate in their networks.

D.4 Scalability on Network Sizes

The ability of an NFV-RA algorithm to scale effectively with increasing network size and complexity is paramount for its practical deployment in real-world, large-scale infrastructures. Virne facilitates a thorough assessment of scalability from two primary perspectives, i.e., (1) performance quality on large-scale network topologies, and (2) the growth trend of average solving time as both PN and VN sizes increase.

Table 8: Results on the large-scale network, WX500.

	RAC↑	**LRC↑**	**LAR↑**	**AST ↓**
PPO-MLP	69.50	0.646	**924217.96**	4.98
PPO-CNN	69.70	0.654	826532.68	5.16
PPO-ATT	68.10	0.647	923953.53	5.64
PPO-GCN	59.50	0.610	660552.71	3.80
PPO-GAT	68.80	0.698	920725.67	5.04
PPO-GCN&S2S	61.70	0.541	682516.68	4.89
PPO-GAT&S2S	58.10	0.540	699216.61	4.74
PPO-DualGCN	**69.90**	0.682	909759.61	5.10
PPO-DualGAT	69.80	**0.715**	917515.34	5.08
PPO-HeteroGAT	61.20	0.546	774076.11	7.42
MCTS	62.19	0.543	77138.32	18.13
SA-Meta	61.40	0.591	691030.70	15.09
GA-Meta	64.10	0.568	72314.48	16.49
PSO-Meta	68.90	0.566	750871.95	6.31
TS-Meta	62.10	0.614	678597.92	14.34
NRM-Rank	55.10	0.553	597325.51	**0.22**
RW-Rank	55.50	0.561	592502.79	0.24
GRC-Rank	58.50	0.559	656379.96	**0.22**
PL-Rank	64.17	0.632	79374.48	18.83
NEA-Rank	66.70	0.641	837297.93	12.97
RW-BFS	5.30	0.592	38341.41	0.27

D.4.1 Performance on Large-scale Networks

To evaluate how algorithms perform when the underlying physical infrastructure is extensive, we utilize the WX500 topology within Virne. WX500, with 500 nodes and approximately 13,000 links, represents a significantly larger and denser environment compared to WX100 (100 nodes, 500 links) or other real-world topologies like GEANT (40 nodes) and BRAIN (161 nodes) used in earlier evaluations. The VN arrival rate (η) for WX500 is appropriately adjusted to reflect its increased capacity, ensuring a comparable level of resource contention. The performance of various algorithms on WX500 is detailed in Table 8.

We observe that PPO-DualGAT and PPO-DualGCN, continue to exhibit strong performance on the large-scale WX500 network. This demonstrates the good scalability of their learned policies to more complex physical infrastructures. While other RL methods with simpler extractors exhibit limited performance, this suggests that ⓫ *the capability to effectively capture and process complex topological information becomes increasingly critical in larger networks*. Additionally, while MCTS and meta-heuristics (GA-Meta, PSO-Meta, SA-Meta, TS-Meta) can sometimes achieve competitive solution quality, their AST becomes a significant bottleneck on larger networks.

D.4.2 Analysis of Solving Time Scale

Beyond solution quality, the computational efficiency, measured by the Average Solving Time (AST), is a critical factor for scalability. Virne allows for detailed profiling of how AST evolves with increasing problem scales. We investigate this from two angles:

- The impact of increasing VN size (number of virtual nodes from 5 to 30) on AST, typically keeping the PN fixed (i.e., WX100).
- The impact of increasing PN size (number of physical nodes from 200 to 1000) on AST, for a consistent distribution of VN requests.

Figure 9 illustrates these relationships for various algorithms. Notably, we observe that ⓬ *rL-based methods generally maintain a much flatter and lower AST curve, while meta-heuristics exhibit a substantial increase.* Their inference time is primarily dictated by the forward pass through the trained neural network, which scales less dramatically with VN size compared to exhaustive search

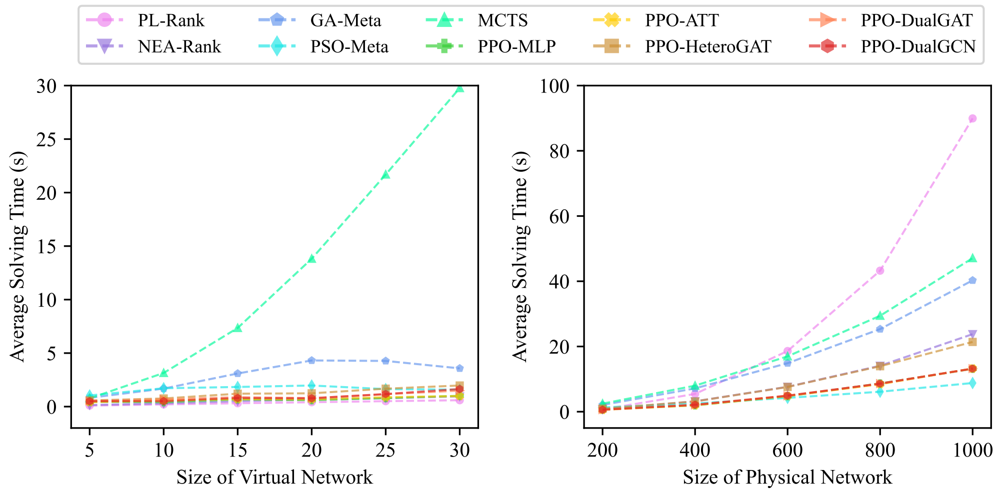


Figure 9: Results on Inference Time Scale with both VN and PN.

or numerous iterations. For other heuristics, they are typically the fastest, exhibiting very low and stable ASTs irrespective of VN size.

The Virne benchmark, through systematic AST profiling, starkly highlights the trade-off between solution quality and computational scalability. While MCTS and certain meta-heuristics might achieve competitive solution quality in some scenarios (as shown in the main paper), their substantial computational overhead, especially with increasing problem dimensions, limits their applicability in dynamic, large-scale NFV environments. ⓭ *RL-based methods, particularly those utilizing GNNs, strike a more favorable balance.* Simpler RL methods and traditional heuristics offer the highest computational speed but may compromise on optimality.

D.5 Validation on Emerging Network

NFV is a cornerstone technology for a variety of modern and emerging network paradigms, each presenting unique characteristics and operational requirements. The Virne benchmark is designed to validate the adaptability and effectiveness of NFV-RA algorithms in such specialized environments. This section focuses on two prominent emerging scenarios, i.e., (1) networks with heterogeneous computing resources, and (2) latency-aware edge networks where delay constraints are critical. Performance in these scenarios is detailed in Table 9.

D.5.1 Heterogeneous Resourcing Networks

Modern data centers and network infrastructures often deploy servers with diverse computing capabilities (e.g., varying CPU cores, GPU availability, memory capacities). Handling such resource heterogeneity is crucial for efficient VNF placement. Virne simulates this by configuring both PN nodes and VN nodes with multiple, distinct types of computing resources. For this evaluation, the WX100 topology is used with a VN arrival rate $\eta = 0.16$. An embedding is successful only if a physical node can satisfy all specified resource demands (i.e., CPU, GPU, and memory) of a virtual node simultaneously, adding a layer of complexity to the node mapping process.

As illustrated in Table 9, we observe that PPO-DualGAT and PPO-GAT&S2S demonstrate superior performance in heterogeneous resourcing environments, outperforming PPO-DualGCN. ⓮ *The attention mechanism and edge awareness in GATs, which can weigh the importance of different features and neighbors, appear particularly adept at navigating multi-dimensional resource constraints.* Simpler RL architectures like PPO-MLP and PPO-GCN, while competent, tend to show a slight dip in performance compared to scenarios with homogeneous resources. ⓯ *Effectively managing and balancing multiple, diverse resource requirements simultaneously poses a greater challenge for architectures that may not explicitly differentiate or dynamically weigh these varied resource dimensions.*

Table 9: Results on emerging networks with heterogeneous resources and latency-awareness.

	Heterogenous Resourcing WX100 ($\eta = 0.16$)				Latency-aware Edge WX100 ($\eta = 0.08$)			
	RAC↑	**LRC↑**	**LAR↑**	**AST↓**	**RAC↑**	**LRC↑**	**LAR↑**	**AST↓**
PPO-MLP	71.10	0.665	12694.51	0.14	56.70	0.609	9792.61	0.20
PPO-ATT	69.30	0.655	12477.79	0.16	57.70	0.605	9460.70	0.21
PPO-GCN	56.30	0.623	10019.26	0.15	54.50	0.653	9045.22	0.20
PPO-GAT	73.40	0.683	13214.29	0.16	53.30	0.635	9570.29	0.26
PPO-GCN&S2S	54.20	0.619	9770.58	0.16	52.10	0.652	9144.30	0.22
PPO-GAT&S2S	<u>75.00</u>	0.688	<u>13857.09</u>	0.18	63.30	0.663	10703.74	0.22
PPO-DualGCN	66.10	<u>0.715</u>	12823.80	0.24	<u>66.70</u>	<u>0.690</u>	11301.84	0.33
PPO-DualGAT	**75.10**	**0.735**	**14172.26**	0.21	**68.20**	**0.714**	**12056.58**	0.32
PPO-HeteroGAT	71.20	0.663	13225.30	0.39	64.70	0.673	<u>11563.99</u>	0.40
MCTS	73.60	0.448	12847.17	1.72	52.70	0.480	8920.53	2.42
SA-Meta	62.10	0.617	10397.84	0.72	44.80	0.557	7789.51	0.80
GA-Meta	72.30	0.589	12245.59	1.16	56.10	0.542	9193.42	1.60
PSO-Meta	67.80	0.519	10837.89	1.37	52.50	0.488	8371.48	1.46
TS-Meta	65.50	0.650	11574.57	0.68	45.90	0.579	7970.68	0.77
NRM-Rank	60.30	0.525	9713.59	0.03	46.30	0.528	7385.33	<u>0.05</u>
RW-Rank	60.20	0.543	8947.27	<u>0.02</u>	39.40	0.554	7042.06	<u>0.05</u>
GRC-Rank	57.50	0.550	9008.80	**0.01**	42.50	0.556	7313.99	**0.04**
PL-Rank	69.70	0.659	12485.17	0.14	51.60	0.617	8546.75	0.23
NEA-Rank	64.60	0.671	10304.28	0.23	45.80	0.624	7367.66	0.34
RW-BFS	33.70	0.572	6472.31	<u>0.02</u>	36.70	0.594	5944.21	<u>0.05</u>

D.5.2 Latency-aware Edge Networks

Edge computing networks bring computational resources closer to end-users, making them ideal for latency-sensitive applications. In such scenarios, NFV-RA algorithms must not only allocate resources efficiently but also strictly adhere to the delay requirements of services. This evaluation uses the WX100 topology with a VN arrival rate $\eta = 0.08$, which could represent a typical edge deployment with specific traffic characteristics. The critical factor introduced here is a latency constraint for each virtual link. When a virtual link is mapped to a physical path, the cumulative propagation delay of the physical links constituting that path must not exceed the virtual link's specified maximum tolerable latency, i.e., 100ms. This necessitates that the algorithms should possess the ability to manage constraint management. We additionally consider the latency attributes as the input features of neural networks.

As illustrated in Table 9, the introduction of latency constraints generally leads to lower RAC and LRC values across all algorithms compared to scenarios without such strict path requirements. However, PPO-DualGAT again shows the most robust performance, achieving the highest RAC (68.20%), LRC (0.714), and LAR (12056.58). In contrast, ⓰ *many traditional heuristics and MCTS find it more challenging to satisfy both resource and latency constraints simultaneously, often resulting in more pronounced performance degradation.*

E Discussion on Future Directions

The empirical analysis enabled by the Virne benchmark illuminates several promising avenues for advancing deep RL-based NFV-RA methods. While current leading algorithms demonstrate significant capabilities, addressing the following challenges could lead to even more robust, efficient, and practical solutions.

Representation Learning for Cross-graph Status Future research should focus on developing more sophisticated representation learning techniques to capture the intricate, dynamic interplay between VN requirements and PN states, including the evolving mapping status itself. While Virne shows the strength of dual-GNN architectures in processing VN and PN features separately then combining them, there's a need for models that can learn even richer cross-graph relational embeddings and path-level attribute awareness [72, 73]. This includes better capturing dependencies across multiple resource types and complex constraints, potentially leading to more nuanced and context-aware embedding decisions. Techniques that can explicitly model the partially embedded state of VNs and the resultant constraints on future placements are crucial.

Robust Learning for Complex and Hard Constraints The management of multifaceted and often conflicting constraints (e.g., resource capacity, latency, reliability, energy consumption) remains a central challenge in NFV-RA, which requires guaranteeing zero-violation. Thus, it is significant to explore constraint-aware RL frameworks that more explicitly model and navigate these hard constraints. This could involve advancing constrained policy optimization methods and developing novel reward structures that better reflect constraint satisfaction [74, 50]. The goal is to train safe policies that are not only effective in optimizing primary objectives but are also inherently robust in satisfying diverse operational constraints.

Scalability in Extremely Large-scale Networks Virne's scalability studies demonstrate that while current RL methods, particularly GNN-based ones, scale better than many traditional approaches, their computational and memory demands can still grow significantly with network size. Addressing NFV-RA in truly massive, carrier-grade networks requires further breakthroughs in algorithmic scalability. Future directions could include exploring hierarchical RL where policies operate at different levels of abstraction, building a non-autoregressive solution modeling method [60], or designing GNN architectures that are less sensitive to the global network scale [75]. Methods that can learn transferable knowledge or localizable policies that effectively stitch together solutions for very large infrastructures are highly desirable [18].

Generalizable Policy Cross Varying Scale Achieving policies that generalize effectively not only to unseen network topologies but also across varying scales (both PN and VN sizes) and highly dynamic operational conditions (e.g., non-stationary demand patterns, unexpected resource outages) is a critical frontier. Virne's generalization experiments provide a baseline, but future research should investigate advanced techniques such as meta-RL [76] to train agents that can quickly adapt to new VNR sizes or types, curriculum learning to incrementally expose agents to more complex scenarios, and domain randomization to enhance robustness against a wider array of unseen conditions [62, 19]. The ultimate aim is to develop omni-generalizable NFV-RA solutions that require minimal retraining for deployment in diverse and evolving network environments.