GPG-VNFE: Towards Foundation Models via Graph Pretraining for Generalizable VNF Embedding

Omar Houidi*, Nour-El-Houda Yellas*, Oussama Soualah†, Djamal Zeghlache*
* SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
† OS-Consulting, 79 avenue François Mitterrand, Athis-Mons, France
Email: {omar.houidi, nour-el-houda.yellas, djamal.zeghlache}@telecom-sudparis.eu, oussama.soualah@os-c.fr

***Abstract*—The placement of Virtual Network Functions (VNFs) plays a critical role in the provisioning of services across next-generation networks. However, the constantly changing conditions in network topologies, traffic demands and resource availability pose significant challenges to achieving efficient and reliable placement strategies. While Deep Reinforcement Learning (DRL) has been widely explored to address this problem, these approaches have limited generalization capabilities and require retraining when faced with unfamiliar scenarios, such as network topology changes, failure events or evolving service requests (e.g., VNF Forwarding Graphs or VNF-FGs). To address these limitations, we propose GPG-VNFE, a foundation Graph Neural Network (GNN) architecture that integrates graph contrastive learning with generative models as Learnable Priors (GraphCL-LP) within a Transformer-based DRL framework. GPG-VNFE first pretrains two domain-specific GNN encoders using GraphCL-LP, one for substrate network topologies and another for VNF-FG service graphs. Through extensive simulation on a medium-size network topology, we show that GPG-VNFE improves the acceptance ratio of incoming VNF requests by up to 57.45% compared with three baseline state-of-the-art placement approaches. We also show that our proposal lowers the power consumption at the physical layer by up to 11%, outperforming two of the baseline approaches.**

***Index Terms*—Foundation Models, Graph Neural Networks (GNNs), Deep Reinforcement Learning, Generative Models, Graph Contrastive Learning, VNF-FG Placement.**

I. Introduction

Network Function Virtualization (NFV) [1] is an emerging networking paradigm that reduces network operating expenses by decoupling network functions from the underlying dedicated hardware, commonly referred to as traditional middleboxes. This decoupling enables the hosting of network services, known as Virtualized Network Functions (VNFs), on commodity hardware (servers), which facilitates and accelerates service deployment and management by service providers, improves flexibility, leads to efficient and scalable resource usage, and reduces operational costs.

Network providers face the challenge of securing a stable power supply for their extensive large-scale networks. Due to the pivotal role of energy as an essential OPerating EXpenditure (OPEX) factor, its smart control is considered necessary to facilitate network expansion. Therefore, Communication Service Providers (CSPs) are devoting much of their efforts to reduce energy consumption of their network infrastructures. Minimizing power consumption is increasingly important, as it not only reduces operational costs but also aligns with environmental sustainability goals.

This paper addresses the critical challenge of optimally placing complex service graphs, or Virtual Network Function Forwarding Graphs (VNF-FGs), which implies the efficient placement of networking functions and their required interconnections over a shared physical infrastructure. The goal is to ensure service-level performance guarantees while optimizing the use of underlying resources.

Machine Learning (ML) has recently attracted the interest of researchers worldwide and is already widely applied in networking [2]. However, most of these applications are based on supervised learning. This approach has limitations, as many networking problems involve unlabeled data and can be seen as decision-making issues arising in dynamic and uncertain environments. For instance, creating and updating end-to-end (E2E) network slices (or VNF-FGs) for large-scale services requires robust decision-making in uncertain conditions, a niche where Reinforcement Learning (RL) offers compelling advantages in speed and accuracy. Deep Reinforcement Learning (DRL) is emerging as a powerful paradigm for tackling the NP-hard challenge of VNF-FG placement, particularly within the highly dynamic and resource-constrained environments of 5G/6G and future networks. However, existing solutions based on DRL often struggle to generalize across dynamic environments. They require extensive retraining whenever the network topology, traffic demands, service graphs, or resource availability change. This lack of adaptability undermines their scalability and impairs real-world deployment, especially in fault-prone or dynamic scenarios.

Our prior work, the Transformer-based DRL (TDRL) architecture [3], leveraged Graph Attention Networks (GAT) and Transformer models to effectively capture the intricate interplay between physical network topology and sequential service demands. However, like many DRL-based solutions, a critical limitation remains: the learned policy often lacks generalization. The TDRL model requires retraining or fine-tuning when exposed to unseen physical topologies or new service graph requests, which not only limits its scalability but also poses barriers in deploying dynamic and fast-evolving networked systems.

To address this critical limitation, we present **GPG-VNFE**, a novel framework that incorporates graph contrastive learning with generative augmentations into the TDRL pipeline to

978-3-903176-75-1 ©2025 IFIP

tackle the VNF-FG embedding problem, known to be an NP-hard problem, in large-scale network environments. GPG-VNFE builds on the intuition that pretrained graph representations can capture domain-invariant patterns, enabling generalization across heterogeneous network topologies and dynamic service requests, the same way foundation models in *Natural Language Processing* capture the underlying structure of languages.

We draw from recent advances in Graph Contrastive Learning (GraphCL), which uses perturbation-invariant objectives to learn meaningful graph representations from unlabeled data [4], [5]. The state-of-the-art graph contrastive learning framework (GraphCL) [4] emphasizes the perturbation invariance in graph neural networks (GNNs) through maximizing agreement between two augmented graph views. However, standard GraphCL relies on hand-crafted augmentations (e.g., DropEdge, NodeDrop), which may not generalize across datasets or topologies. Our approach begins by leveraging insights from *graph contrastive learning with automated augmentations (GraphCL-Auto)* [6], which formulates augmentation selection as a principled bi-level optimization problem. This enables adaptive and dynamic augmentation selection without manual tuning, improving robustness to structural variability. We pretrain Graph Neural Network (GNN) encoders for both the physical substrate and VNF-FG graphs, enabling the learning of rich, task-agnostic embeddings via contrastive self-supervised objectives.

To go beyond the limitations of automated selection from a fixed augmentation pool, which relies on a predefined set of manually crafted graph transformations, we adopt GraphCL-LP [7], which learns augmentations directly via graph generative models such as Variational Graph AutoEncoders (VGAE) [8]. This generative augmentation strategy, referred to as Learned Priors (LP), produces more informative and diverse views that better improve robustness to topological variability. Notably, GraphCL-LP has been shown to outperform GraphCL-Auto across various datasets, further motivating its integration into our framework.

In fact, causal or contrastive generative methods can improve resilience by learning invariant features that generalize across changing environments. By embedding this foundation-level knowledge into GPG-VNFE, we enable zero-shot adaptation to unseen topologies, improved transfer learning across domains, and yield faster convergence in downstream reinforcement learning tasks.

While large language models (LLMs) show impressive versatility in language and programming tasks, they cannot be easily adapted to network optimization problems such as VNFs placement. This difficulty arises because networking scenarios involve structured data, constant changes, and often limited datasets, making the typical large-scale pre-training used for LLMs impractical. Instead, we adopt a more targeted and scalable approach by combining graph-based self-supervised learning with lightweight reinforcement learning. In particular, we employ a Variational Autoencoder (VAE) as a generative backbone. This VAE learns to create concise graph representations that are independent of specific tasks, offering strong adaptability to new network structures without the heavy data requirements and computational overhead associated with LLMs.

This shift from task-specific to general-purpose GNN representations marks a key advancement toward *Graph Foundation Models* for network control, enabling scalable, self-adaptive orchestration in future 6G infrastructures.

The main contributions of this paper are as follows:

- We formulate the VNF-FG Embedding (VNF-FGE) problem as an Integer Linear Programming (ILP) model that captures physical constraints such as host capacity, bandwidth, and node separation. To handle scalability in dynamic and large-scale networks, we reformulate the problem as a Markov Decision Process (MDP), with a reward function that balances VNF-FG acceptance (i.e., service revenue) and energy consumption.
- We propose **GPG-VNFE** (Graph Pretraining for Generalizable VNF Embedding), a unified framework that combines Deep Reinforcement Learning (DRL) with contrastive graph representation learning. GPG-VNFE embeds the network topology and service graph into a latent space using graph pre-training, and performs one-shot VNF placement and routing via a Transformer-based Actor-Critic architecture.
- Within GPG-VNFE, we introduce **GraphCL-LP**, a contrastive learning module with learnable graph perturbation priors. This replaces handcrafted augmentations and enables the system to learn structure-aware, topology-conditioned representations for VNF-FGs.
- We also integrate a **graph generative augmentation module**, which learns to produce meaningful, task-specific augmentations of input service graphs. This improves the diversity and generalization of the contrastive training process, especially for complex or unseen graph structures.
- We conduct extensive simulation-based evaluation under diverse topologies and workloads. The results show that GPG-VNFE significantly outperforms state-of-the-art heuristics and DRL baselines in terms of VNF-FG acceptance rate, energy efficiency, and inference speed, while maintaining near-linear runtime scalability.

The remainder of this paper is organized as follows. Section II reviews related work on VNF-FG placement, DRL, and graph representation learning. Section III presents the formal problem definition. Section IV details the GPG-VNFE architecture, including the TDRL and GraphCL-LP modules. Section V presents our simulation setup and evaluation. Finally, Section VI summarizes the main findings.

II. Related Work

The allocation of Virtualized Network Function (VNF) resources within a single-domain has been a focal point of research since the emergence of Network Function Virtualization (NFV). Much of the existing early work concentrated on deploying simple, single-VNF services, but recent studies

have begun to tackle the more realistic challenge of placing complex service graphs (VNF-FGs), which involve multiple interconnected VNFs. Various approaches have emerged in the literature to address this intra-domain VNF-FG embedding (VNF-FGE) problem, with methods depending on the specific characteristics and objectives of the problem at hand. We present the different existing approaches by classifying them into three categories.

A. Exact Approaches

Initial efforts to find optimal solutions for the VNF-FG embedding (VNF-FGE) formulate the problem as Integer Linear Programming (ILP) [9], Mixed Integer Linear Programming (MILP) [10], or Integer Non-Linear Programming (INLP) [11]. However, due to its NP-hard nature, exact solvers are only feasible for small-scale network instances. Strategies like reducing the set of candidate servers/hosts [12] and relaxed ILP formulations [13] are proposed to improve scalability for larger instances, with the goal of optimizing resource usage, power consumption, or deployment cost.

B. Heuristic and Meta-Heuristic Approaches

Recognizing the scalability limitations of exact methods for real-world applications, heuristic and meta-heuristic solutions with lower complexity have been extensively explored to obtain approximate solutions in viable execution times [14]–[16]. Baseline Greedy algorithms, for instance, typically decompose the problem into VNF mapping and traffic routing, leading to suboptimal solutions [14]. More advanced heuristics like 'Holu' [15] address power-aware and delay-constrained problems by dividing them into centrality-based VNF placement and a Delay-Constrained Least-Cost (DCLC) shortest-path routing. Meta-heuristics, such as evolutionary algorithms like NSGA-II [16], are employed for multi-objective optimization (e.g., minimizing mapping cost and maximizing link utilization). Although meta-heuristic based approaches offer improved adaptability and performance over simple heuristics in solving complex problems, they still encounter difficulties in achieving efficient convergence.

C. Energy-aware Deep Reinforcement Learning (DRL)-based Approaches

More recently, Deep Reinforcement Learning (DRL) techniques have shown significant potential for high-complexity VNF-FGE problems in large-scale networks [17]–[21]. Many DRL approaches model VNF-FG allocation as a Markov Decision Process (MDP), employing actor-critic methods like enhanced Deep Deterministic Policy Gradient (DDPG) to maximize allocated VNF-FGs while meeting QoS requirements [17]. Although various objectives have driven the optimization of VNF-FG deployment, energy efficiency remains a critical and persistent challenge, particularly within graph-structured Service Function Chaining (SFC) environments. Recent works address this by using GNN-based DRL to minimize energy consumption through autonomous node selection [18]. Multiple efforts to reduce energy footprint in SFCs leverage heuristic approaches, such as iterative VNFs placement strategies [20], or sampling-based techniques like Markov Approximation for energy-efficient VNF deployment [21]. However, these multi-parameter solutions often suffer from limited applicability and high complexity, being more efficient in scenarios with a small number of user nodes.

D. Graph Representation Learning for Network Optimization

Graph Contrastive Learning: Beyond VNF-FGE, advancements in graph representation learning are highly relevant. Graph contrastive learning (GCL) [4], [5], [22] is a prominent technique that learns robust graph representations by maximizing agreement between augmented views of the same graph. Frameworks like GraphCL [4] utilizes hand-crafted augmentations (e.g., NodeDropping, EdgePerturbation) and train GNN encoders to map graphs into a space where similar graphs are close. Given an input graph G, and two augmentations A_1, A_2, the encoder f produces representations $h_1 = f(A_1(G))$ and $h_2 = f(A_2(G))$. The model minimizes a contrastive loss $\mathcal{L}_{\text{CL}}$ such as the one used in InfoNCE [23]:

$$\mathcal{L}_{\text{CL}} = -\log \frac{\exp(\text{sim}(h_1, h_2)/\tau)}{\sum_{h' \in \mathcal{N}} \exp(\text{sim}(h_1, h')/\tau)}$$

where $\mathcal{N}$ is a set of negative samples and τ is a temperature parameter. Despite its effectiveness, GCL often depends on manually crafted, dataset-specific augmentations, which can limit its generalization and transferability.

Learnable Priors: Learned priors have been a key component in several fields, including video prediction, compressed sensing, and Bayesian modeling. For instance, SVG-LP from [24] uses recurrent neural networks to capture the temporal dynamics of video frames, serving as an implicit uncertainty estimator. In compressed sensing, a number of works [25], [26] show that generative models can act as effective image priors, requiring fewer measurements than classical sparse recovery methods such as LASSO (Least Absolute Shrinkage and Selection Operator). Beyond that, Bayesian inference methods—such as Gaussian processes, variational autoencoders, and Bayesian neural networks—rely on learnable or adaptive priors to model uncertainty and variability.

Graph Generative Models: Graph generative modeling has long been employed in applications ranging from molecular synthesis to anomaly detection and recommendation [27]. Recent advances focus on conditional generative models, which aim to produce new graphs based on a given input structure. These approaches typically define a stochastic function $g_\phi : \mathcal{G} \to \mathcal{G}'$, parameterized by ϕ, and learn it by minimizing a generation loss $\mathcal{L}_{\text{Gen}}(G, \phi)$ that measures the divergence between the generated and target graphs [28]. This conditional framework opens up possibilities for task-adaptive generation, which can be especially useful for contrastive learning, where conventional hand-crafted augmentations may be suboptimal.

In contrast to prior methods, our approach proposes a *one-shot* strategy for VNF-FG placement, where VNFs are embedded simultaneously rather than relying on a sequential

embedding. This accelerates the response time for handling VNF-FG requests, as well as reduces inefficiencies commonly introduced by sequential embedding. Additionally, several existing works explore contrastive learning for graph representation, including GraphCL [4], InfoGraph [5], and MVGRL [22]. For instance, GraphCL applies manual augmentations, such as node dropping and subgraph sampling, to generate multiple views, using a contrastive loss to align their embeddings. However, relying on handcrafted and dataset-specific augmentations often limit the model generalization. In contrast, our approach avoids the need for fixed augmentation schemes by learning adaptive perturbations on graph structures, which enhances improved generalization across diverse graph distributions.

III. VNF-FG Embedding Problem Formulation

In this work, the objective is to efficiently embed a set of Virtual Network Function–Forwarding Graphs, representing network service requests, onto a shared physical infrastructure, with the aim of maximizing service acceptance and minimizing total power consumption. However, this optimization must be achieved while strictly satisfying the technical and topological constraints imposed by each VNF-FG. We begin by defining the physical network and the VNF-FG models, followed by a formal description of the VNF-FG embedding problem.

A. Physical layer model

The physical network is modeled as a weighted undirected graph $G^{(\mathrm{p})} = (N^{(\mathrm{p})}, L^{(\mathrm{p})})$, where $N^{(\mathrm{p})}$ is the set of physical nodes (e.g., servers, switches, or Physical Network Functions—PNFs), and $L^{(\mathrm{p})}$ is the set of physical links interconnecting these nodes. Each physical node $n^{(\mathrm{p})} \in N^{(\mathrm{p})}$ is associated with its available computational resources, such as CPU and RAM denoted by $r_{n^{(\mathrm{p})}}$. Each physical link $l^{(\mathrm{p})} \in L^{(\mathrm{p})}$ connects a pair of physical nodes and has a bandwidth capacity denoted $b_{l^{(\mathrm{p})}}$. This layer serves as the substrate over which virtualized services are embedded, with placement decisions governed by both node-level resource capacities and link-level communication capabilities.

B. Virtual layer model

The virtual layer refers to the service requests I, modeled as a Virtual Network Function–Forwarding Graph and represented by a directed graph $G^{(\mathrm{v})} = (N^{(\mathrm{v})}, L^{(\mathrm{v})})$, where $N^{(\mathrm{v})}$ denotes the set of virtual network functions (VNFs) and $L^{(\mathrm{v})}$ denotes the set of virtual links connecting them, for a given service request i with arrival time t_i.
Each virtual node $n_i^{(\mathrm{v})} \in N_i^{(\mathrm{v})}$ represents a VNF and is characterized by a resource demand vector $r_{n_i^{(\mathrm{v})}} = [r_{n_i^{(\mathrm{v})},1},..,r_{n_i^{(\mathrm{v})},k},..,r_{n_i^{(\mathrm{v})},|K|}]$, where $r_{n_i^{(\mathrm{v})},k}$ denotes the amount of resource type $k \in K$ required by VNF $n_i^{(\mathrm{v})}$. Similarly, each virtual link $l_i^{(\mathrm{v})} \in L_i^{(\mathrm{v})}$ is associated with a bandwidth demand $b_{l_i^{(\mathrm{v})}}$, specifying the required communication capacity between VNFs. The embedding of these VNF-FGs onto the physical network must ensure that all resource and bandwidth demands are satisfied, while maintaining the functional and topological integrity of the service graphs.

TABLE I: Notations.

	Parameters and variables
$r_{n^{(\mathrm{p})}}$	resources r available in physical node $n^{(\mathrm{p})}$.
$r^{\max}_{n^{(\mathrm{p})}}$	maximum capacity of resources r of physical node $n^{(\mathrm{p})}$.
$b_{l^{(\mathrm{p})}}$	bandwidth available at physical link $l^{(\mathrm{p})}$.
$b^{\max}_{l^{(\mathrm{p})}}$	maximum capacity of bandwidth of physical link $l^{(\mathrm{p})}$.
$r_{n_i^{(\mathrm{v})}}$	amount of resources r required by VNF $n_i^{(\mathrm{v})}$ in request i.
$b_{l_i^{(\mathrm{v})}}$	bandwidth requested by link $l_i^{(\mathrm{v})}$ in request i.
AR	acceptance ratio of VNF-FG requests.
PC	power consumption at the physical layer.
$PC^{(\mathrm{idle})}$	power consumption at idle state.
$PC^{(\mathrm{M})}$	maximum power consumption.
$U^{(\mathrm{p})}$	CPU utilization rate at physical node $n^{(\mathrm{p})}$.
$\sigma^{n^{(\mathrm{v})}}_{n^{(\mathrm{p})}}$	binary variables for node mapping, equal to 1, if $n^{(\mathrm{v})}$ is mapped on $n^{(\mathrm{p})}$.
$\rho^{l^{(\mathrm{v})}}_{l^{(\mathrm{p})}}$	binary variables for link mapping, equal to 1, if $l^{(\mathrm{v})}$ is mapped on $l^{(\mathrm{p})}$.

C. Problem statement and formulation

We define our VNF forwarding graph embedding problem as follows. Given the undirected graph $G^{(\mathrm{p})}$ representing the physical network, a set of requests I and a set of available VNFs denoted $N^{(\mathrm{v})}$. Our goal consists in determining the placement of VNFs on the nodes $N^{(\mathrm{p})}$, so that:
- the requests from VNF-FGs are routed in $G^{(\mathrm{p})}$;
- the requested resources from each installed VNF are satisfied;
- the power consumption is minimized;
- the request acceptance ratio is maximized;
- the system constraints imposed by the physical layer are respected.

The objective of this work is to minimize the power consumption (PC) and maximize the acceptance ratio (AR) while guaranteeing the success of VNF-FG deployment. For simplicity, we assume that the power consumption of all physical network nodes is directly proportional to the CPU utilization (i.e., depending also on the number of active servers in the physical network). The main goal is to minimize the overall power consumption in the whole physical network, by putting into sleeping mode all non-utilized physical nodes that are at idle power consumption while accommodating VNF-FGs' demands.

Let us denote the trade-off parameters as ϵ_1 and ϵ_2, where ϵ_1 represents the weight for the acceptance ratio and ϵ_2 represents the weight for energy efficiency. These parameters satisfy the condition $\epsilon_1 + \epsilon_2 = 1$, where $\epsilon_1, \epsilon_2 \in [0, 1]$.

- **Node mapping constraints**: Each VNF can be deployed at only one physical node $n^{(\mathrm{p})}$; therefore:

$$\sum_{n^{(\mathrm{p})} \in N^{(\mathrm{p})}} \sigma^{n^{(\mathrm{v})}}_{n^{(\mathrm{p})}} \leq 1, \quad \forall n^{(\mathrm{v})} \in N^{(\mathrm{v})} \tag{1}$$

where $\sigma_{n^{(\mathrm{p})}}^{n^{(\mathrm{v})}}$ is equal to 1 if VNF $n^{(\mathrm{v})}$ is mapped on node $n^{(\mathrm{p})}$.

- **Node capacity constraints**: A VNF is successfully deployed when its substrate node has sufficient resources, i.e., the total amount of the required resource k of VNF-FG request i mapped on $n^{(\mathrm{p})}$ should not exceed its residual amount of resources.

$$\sum_{n^{(\mathrm{v})} \in N^{(\mathrm{v})}} \sigma_{n^{(\mathrm{p})}}^{n^{(\mathrm{v})}} \times r_{n^{(\mathrm{v})},k} \leq r_{n^{(\mathrm{p})},k}, \quad \forall n^{(\mathrm{p})} \in N^{(\mathrm{p})}, \quad \forall k \in K \tag{2}$$

Where $r_{n^{(\mathrm{p})},k}$ is the available amount of resource k in physical node $n^{(\mathrm{p})}$.

- **Link capacity constraints**: In our case, we focus specifically on the bandwidth metric for the successful deployment of a virtual link $l_i^{n^{(\mathrm{v})}}$. This involves ensuring that the associated VNFs are deployed successfully and that the Quality of Service (QoS) requirements related to the bandwidth are met:

$$\sum_{l^{(\mathrm{v})} \in L^{(\mathrm{v})}} \rho_{l^{(\mathrm{p})}}^{l^{(\mathrm{v})}} \times b_{l^{(\mathrm{v})}} \leq b_{l^{(\mathrm{p})}}, \quad \forall l^{(\mathrm{p})} \in L^{(\mathrm{p})} \tag{3}$$

where $\rho_{l^{(\mathrm{p})}}^{l^{(\mathrm{v})}}$ is equal to 1, if the virtual link $l^{(\mathrm{v})}$ is mapped on the physical link $l^{(\mathrm{p})}$.

- **Path continuity constraint**: If the VNF-FG request i is accepted, the path mapped in the physical network should traverse through VNFs following the order specified in the request. Let $\mathcal{I}(n^{(\mathrm{p})})$ and $\mathcal{O}(n^{(\mathrm{p})})$ denote the sets of incoming and outgoing links of the physical node $n^{(\mathrm{p})}$. Additionally, $n_{i,\mathrm{src}}^{(\mathrm{v})}$ and $n_{i,\mathrm{dst}}^{(\mathrm{v})}$ represent the source and destination VNFs of the virtual link $l_i^{(\mathrm{v})}$.

$$\sum_{l^{(\mathrm{p})} \in \mathcal{I}(n^{(\mathrm{p})})} \rho_{l^{(\mathrm{p})}}^{l_i^{(\mathrm{v})}} - \sum_{l^{(\mathrm{p})} \in \mathcal{O}(n^{(\mathrm{p})})} \rho_{l^{(\mathrm{p})}}^{l_i^{(\mathrm{v})}} = \sigma_{n^{(\mathrm{p})}}^{n_{i,\mathrm{dst}}^{(\mathrm{v})}} - \sigma_{n^{(\mathrm{p})}}^{n_{i,\mathrm{src}}^{(\mathrm{v})}}, \quad \forall l_i^{(\mathrm{v})} \in L_i^{(\mathrm{v})}. \tag{4}$$

A VNF-FG is considered accepted if all its VNFs and virtual links are successfully mapped while satisfying all constraints and QoS requirements. The final objective is to maximize the number of accepted VNF-FGs and minimize power consumption.

$$\begin{aligned} &\text{Maximize:} \quad \epsilon_1 \times \mathrm{AR} - \epsilon_2 \times \mathrm{PC} \\ &\text{Subject to:} \quad (1), (2), (3), (4) \end{aligned}$$

Where AR denotes the acceptance ratio of VNF-FG requests and PC represents the total power consumption at the physical layer. The definition of these two terms is provided in the next section.

IV. System Overview: The GPG-VNFE Framework

A. MDP model for DRL

Our DRL framework models the agent–environment interaction as a Markov Decision Process (MDP), which is formalized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$. At each discrete time step t, the agent observes the state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, receives a reward $r_t \in \mathcal{R}$, and transitions to a new state s_{t+1} according to transition probability $\mathcal{P}$. In our setting, the environment is fully observable, thus the observed state aligns with the true state. The state s_t is composed of two subcomponents: the physical network state $s_t^{(\mathrm{p})}$ and the VNF-FG request state $s_t^{(\mathrm{v})}$, i.e., $s_t = (s_t^{(\mathrm{p})}, s_t^{(\mathrm{v})})$. This joint state representation allows the agent to reason over both substrate availability and service requests.

1) State: The state s_t is defined to encompass:

- Physical network state $s_t^{(\mathrm{p})}$: includes the adjacency matrix $A \in \mathbb{R}^{|N^{(\mathrm{p})}| \times |N^{(\mathrm{p})}|}$ and the feature matrix $X \in \mathbb{R}^{|N^{(\mathrm{p})}| \times F}$, where each row of X corresponds to an F-dimensional feature vector of physical node $n^{(\mathrm{p})} \in N^{(\mathrm{p})}$. Node features include the available CPU $c_{n^{(\mathrm{p})}}$, maximum CPU $c_{n^{(\mathrm{p})}}^{\max}$, total available bandwidth $b_{l^{(\mathrm{p})}}$, and maximum bandwidth $b_{l^{(\mathrm{p})}}^{\max}$. All features are normalized to the range $[0, 1]$ for consistency.
- The VNF-FG request state $s_t^{(\mathrm{v})}$: denotes the current state of the VNF-FG request i, which consists of the required amount of resources $r_{n_i^{(\mathrm{v})}}$ of VNF $n_i^{(\mathrm{v})}$, the required bandwidth $b_{l_i^{(\mathrm{v})}}$ of virtual link $l_i^{(\mathrm{v})}$, and the number of VNFs remaining to be placed.

2) Action: At time step t, the agent makes a selection of servers to simultaneously instantiate all the Virtual Network Functions (VNFs) of the current VNF-FG request. The action taken by the agent is thus represented as $A_t = \{n^{(\mathrm{p})} \in N^{(\mathrm{p})} \mid r_{n_i^{(\mathrm{v})},k} \leq r_{n^{(\mathrm{p})},k}, \ \forall k \in K\}$, indicating that the servers are responsible for hosting the VNF-FG. It is possible for a server to host multiple VNFs, as long as the available resources of the server can satisfy the deployment requirements and constraints.

3) Reward: The reward function incorporates service acceptance and energy efficiency by jointly considering the acceptance ratio (AR) and the global power consumption $PC_G(t)$. A VNF-FG is considered successfully deployed when all its VNFs and virtual links (VLs) are allocated while satisfying resource and bandwidth constraints.

Unlike prior works that rely solely on acceptance ratio (AR) as a simple reward function, we incorporate a constrained reward formulation that penalizes excessive power usage. Specifically, the power consumption of a physical node $n^{(\mathrm{p})} \in N^{(\mathrm{p})}$ at time t is estimated as:

$$PC_t(n^{(\mathrm{p})}) = PC^{(\mathrm{idle})}(n^{(\mathrm{p})}) + \Big[PC^{(\mathrm{M})}(n^{(\mathrm{p})}) - PC^{(\mathrm{idle})}(n^{(\mathrm{p})})\Big] \times U_t(n^{(\mathrm{p})}) \tag{5}$$

where $PC^{(\mathrm{idle})}(n^{(\mathrm{p})})$ and $PC^{(\mathrm{M})}(n^{(\mathrm{p})})$ denote the idle and maximum power consumption, respectively, and $U_t(n^{(\mathrm{p})}) \in [0, 1]$ is the CPU usage rate.

The global power consumption at time t is:

$$PC_G(t) = \sum_{n^{(\mathrm{p})} \in N^{(\mathrm{p})}} PC_t(n^{(\mathrm{p})}) \tag{6}$$

The final reward is expressed as:

$$\mathcal{R} = \epsilon_1 \times AR - \epsilon_2 \times PC_G(t) \tag{7}$$

where ϵ_1 and ϵ_2 are weights such that $\epsilon_1 + \epsilon_2 = 1$, balancing service acceptance and energy efficiency. This formulation promotes deployment strategies that maintain high service availability while minimizing infrastructure energy costs.

B. GPG-VNFE framework

We propose GPG-VNFE, a compositional learning framework for Virtual Network Function Embedding that integrates generative graph pre-training with reinforcement-based placement optimization. The architecture consists of two tightly coupled stages: a Graph Contrastive Learning with Learned Priors (GraphCL-LP) pre-training phase and a Deep Reinforcement Learning (DRL) agent responsible for VNF embedding decisions [3].

Figure 1 depicts the GPG-VNFE architecture. It combines graph-based pre-training with reinforcement learning to efficiently place VNFs onto physical networks. It first uses a generative contrastive learning approach to pre-train two GNN encoders: one for service request graphs and one for physical network graphs—enabling it to learn reusable graph representations. These pre-trained encoders provide rich embeddings that capture structural and functional features of the input graphs. During real-time operation, these embeddings are used by a reinforcement learning agent, which includes a Transformer and Graph Attention Network, to make intelligent placement decisions.

1) Foundational Pre-training using GraphCL with Learned Priors (GraphCL-LP): The primary contribution of GPG-VNFE lies in its pre-training phase, which generates robust and generalizable graph embeddings for both the substrate network and VNF-FGs. Unlike prior DRL approaches that learn representations from scratch during RL training, we employ the GraphCL-LP methodology from [7] to pre-train specialized GNN encoders ($f_{\theta'}$). This approach offers significant advantages over standard GraphCL by learning optimal data augmentations instead of relying on a fixed, manually selected set.

- **Learnable Augmentations via Generative Models:** Following GraphCL-LP [7], we utilize graph generative models, specifically Variational Graph Auto-Encoders (VGAE), to create diverse and informative contrastive views (augmentations) of the input graphs. This approach allows the system to discover inherent graph priors and generate rich representations directly from the data distribution.
- **Principled Bi-Level Optimization:** The training of GraphCL-LP follows a bi-level optimization framework, which is crucial for jointly learning robust graph representations and effective data augmentations. The upper-level optimizes the GNN encoder ($f_{\theta'}$) and projection head ($h_{\theta''}$) using a contrastive loss, maximizing agreement between views. The lower-level optimizes the generative models (g_ϕ) using a generative loss combined with a reward signal derived from the contrastive learning process. The bi-level optimization problem is formulated as:

$$\begin{aligned} &\min_{\theta} \ \mathbb{E}_{\mathbb{P}_G}\left[\mathcal{L}_{\text{CL}}(G, \phi_1, \phi_2, \theta)\right], \\ &\text{s.t. } \phi_1, \phi_2 \in \arg\min_{\phi_1', \phi_2'} \ \mathbb{E}_{\mathbb{P}_G}[r(G, \phi_1', \phi_2', \theta) \\ &\qquad \{\mathcal{L}_{\text{Gen}}(G, \phi_1') + \mathcal{L}_{\text{Gen}}(G, \phi_2')\}]. \end{aligned}$$

Here, $\theta = \{\theta', \theta''\}$, where $f_{\theta'}$ is the GNN Encoder and $h_{\theta''}$ is the Projection Head. The graph generators are denoted by g_{ϕ_i}.

The **contrastive loss function** ($\mathcal{L}_{CL}$) for the upper-level is defined as:

$$\begin{aligned} \mathcal{L}_{\text{CL}}(G, g_{\phi_1}, g_{\phi_2}, \theta) = &-\text{sim}(T_{\theta,\phi_1}(G), T_{\theta,\phi_2}(G)) \\ &+ \log\left(\mathbb{E}_{\mathbb{P}_{G'}} \exp\left(\text{sim}(T_{\theta,\phi_1}(G), T_{\theta,\phi_2}(G'))\right)\right), \end{aligned}$$

where $T_{\theta,\phi_i} = g_{\phi_i} \circ f_{\theta'} \circ h_{\theta''}$ transforms the input graph G into a learned view, and $\text{sim}(\cdot,\cdot)$ is the cosine similarity function.

The **reward function** (r) guides the generator optimization in the lower-level, preventing trivial solutions and model collapse by encouraging the generation of challenging yet informative views. Its general form is:

$$r(G, \phi_1, \phi_2, \theta) = \begin{cases} 1, & \text{if some condition holds} \\ \delta \ll 1, & \text{otherwise} \end{cases}$$

with the condition determined by certain principles and the reward weakens to δ if the condition is not satisfied. This reward is based on principles like Information Minimization (InfoMin) [29] and Information Bottleneck (InfoBN) [30], ensuring that the generated views are diverse and reduce information overlap.

- **Specialized Encoders:** We conduct this pre-training process twice:
  - on a large dataset of representative VNF-FG graphs to train a VNF-FG specific GNN encoder ($f_{\theta'_{\text{VNF}}}$).
  - on a large dataset of representative Physical Network graphs to train a substrate-specific GNN encoder ($f_{\theta'_{\text{Substrate}}}$).

  We use the PROTEINS and COLLAB datasets from the TUDataset collection [31], chosen for their structural similarity to VNF service chains and physical topologies, respectively. The output of this phase is two pre-trained GNN encoders ($f_{\theta'_{\text{VNF}}}$ and $f_{\theta'_{\text{Substrate}}}$), which capture foundational, task-agnostic knowledge about network and service structures.

2) DRL Placement Agent (Enhanced TDRL): The core decision-making component of GPG-VNFE is an Actor-Critic DRL agent, architecturally similar to our TDRL model. Our TDRL approach [3] integrates DRL using Graph Attention Networks (GAT), and a Transformer architecture to optimize VNF-FG placement. In this framework, the encoder component of the Transformer is responsible for encoding the VNF-FG requests. Concurrently, the GAT network is employed

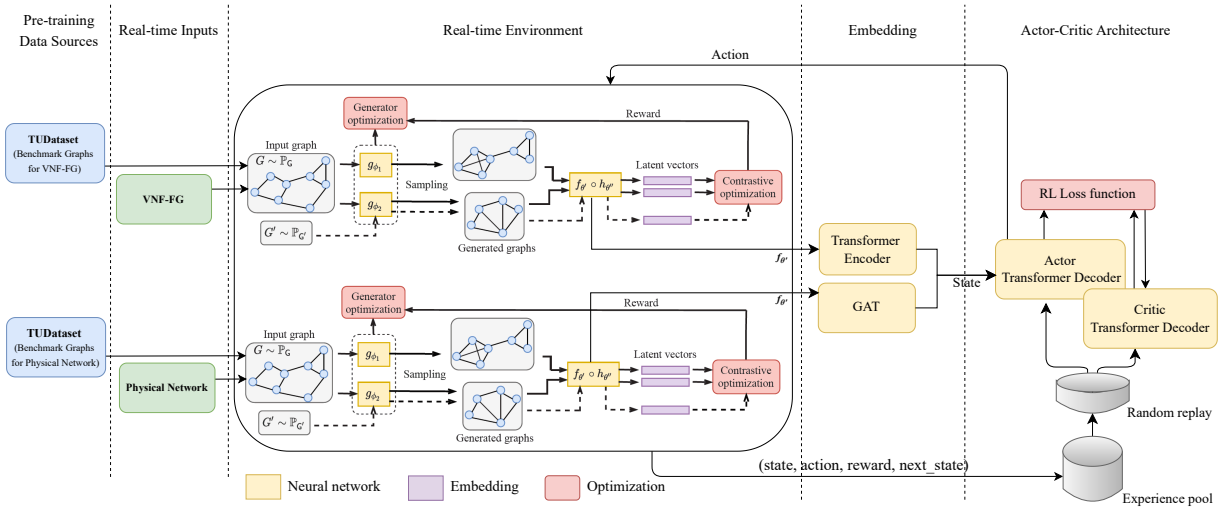

Fig. 1: Overview of the proposed GPG-VNFE architecture.

to capture the features of the physical network. These two representations are then merged to form a full description of the environment state. Then, the decoder component of the Transformer makes 'one-shot' placement decisions for the VNFs of a VNF-FG request. This approach enhances both placement efficiency and resource utilization in large-scale systems. We retain the effective Transformer-based sequence-to-sequence structure for both the Actor (policy network) and Critic (value network), as detailed in prior work [3]. However, GPG-VNFE significantly enhances the agent's perception through its *state representation*, leveraging the pre-trained encoders:

- **State Definition:** The state (s_t) at time step t includes the VNF-FG request and the current state of the physical network as already discussed.
- **Enhanced VNF-FG Embedding:** Instead of direct input, the incoming VNF-FG graph is first processed by the pre-trained $f_{\theta'_{\text{VNF}}}$ to obtain rich node embeddings for each VNF. These enhanced embeddings are then fed into the TDRL's Transformer Encoder, allowing it to capture sequential demands more effectively.
- **Enhanced Substrate Embedding:** Similarly, the physical network graph is processed by the pre-trained $f_{\theta'_{\text{Substrate}}}$. Its output provides powerful node features for the GAT within the TDRL encoder, enabling it to better understand the substrate's capabilities and current state.
- **Action & Reward:** The Actor network outputs the placement decision (e.g., node mapping for each VNF), and the Critic evaluates its quality. The reward function balances service acceptance and energy efficiency, as defined previously.

V. Performance Evaluation

This section presents the evaluation methodology and the simulation results. Note that during the testing phase, only the actor network of the trained agent works to place 1000 VNF-FG requests.

TABLE II: Simulation parameters.

learning rate of actor and critic	10^{-1}
batch size	32
delayed network update rate τ	0.1
discount factor γ	0.95
number of Transformer layers	4
number of attention heads	4

A. Simulation setup

We generate a flat network topology with 100 nodes and 500 links, following the Waxman topology model [32], which imitates a medium-sized infrastructure. The CPU and bandwidth resources are uniformly distributed between 50 to 100 units. In each episode, 1000 VNF-FG requests arrive at the system sequentially according to a Poisson process with an average arriving rate of 20 per 100 time units.

Specifically, each VNF-FG request consists of different numbers of VNFs from 5 to 15 according to the uniform distribution and has a lifetime exponentially distributed with an average of 1000. The node and link resource demands of VNF-FG requests are generated from a uniform distribution with values between 2 and 30. Our model architecture is built with PyTorch. The simulation parameter values are given in Table II. Our simulation experiments are executed on a laptop, equipped with an 11th Gen Intel Core i7-1185G7 CPU @ 3.00 GHz and 32 GB of RAM.

B. Compared approaches

To rigorously evaluate the efficiency and generalization capabilities of our proposed GPG-VNFE framework, we conduct a comprehensive comparative analysis against the following approaches.

- **TDRL-DDPG** [3]: refers to our prior work. TDRL leverages Graph Attention Networks (GAT) and a Transformer architecture within a DRL framework to address the VNF-FG embedding problem.

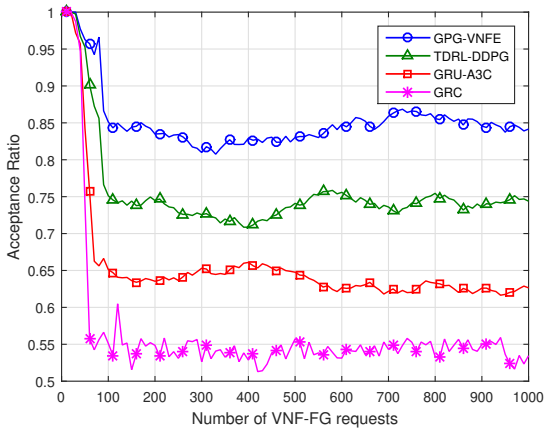

Fig. 2: Acceptance ratio for different VNF-FG requests.

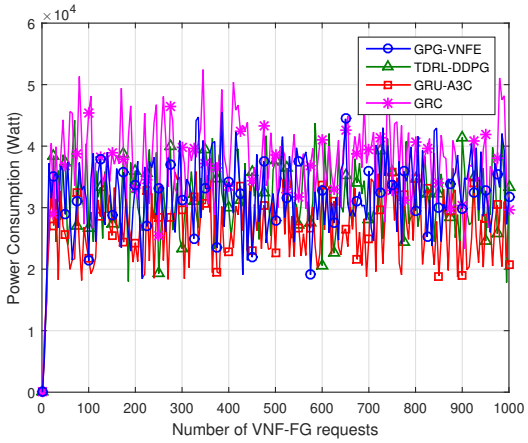

Fig. 3: Power consumption for different VNF-FG requests.

- **GRU-A3C** [33]: employs the Asynchronous Advantage Actor-Critic (A3C) algorithm for concurrent training of both policy and value function estimations. To capture the requirements of VNF-FG requests, the approach utilizes an encoder from a Seq2Seq model, implemented using a Gated Recurrent Unit (GRU) network.
- **GRC** [34]: heuristic algorithm that maps VNFs into physical nodes based on the global availability of resources.

C. Performance metrics

The comparison is done looking at the following metrics.

*1) **Acceptance ratio**:* is the ratio of incoming service requests that have been successfully deployed on the network to all incoming requests.

*2) **Power consumption**:* The global consumed power, is equal to the sum of the power consumption of the activated servers.

*3) **Robustness against node and link failure**:* Shows how versatile GPG-VNFE is, to support single node and link failures and its ability to find a solution that can satisfy the embedding of incoming VNF-FG requests, even in case one of the selected underlying physical nodes and links fails.

D. Results and discussion

Figure 2 illustrates the average acceptance rate of VNF-FG requests achieved by each algorithm. Our proposed GPG-VNFE significantly outperforms the baseline methods, achieving the highest acceptance rate, followed by TDRL-DDPG, GRU-A3C and finally GRC. Our proposed algorithm achieves acceptance ratios that exceed 81% for the entire performance evaluation compared with the three other approaches with minimum achieved acceptance rate of 71% for TDRL-DDPG, 62% for GRU-A3C, and 52% for GRC. This can be explained by the fact that GPG-VNFE introduces a more robust and generalizable architecture. By incorporating Graph Contrastive Learning with Learnable Priors (GraphCL-LP) into domain-specific GNN encoders, it pretrains on both substrate network

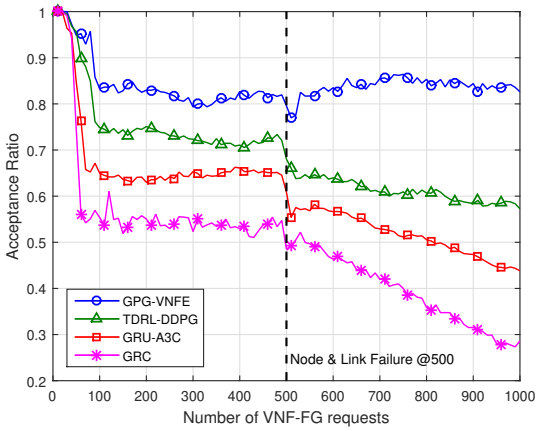

Fig. 4: Robustness against single node and link failure.

topologies and VNF-FG service graphs. By learning generalizable representations, it can handle diverse network conditions and service demands, improving placement efficiency, optimizing resource usage, and enhancing system resilience. GPG-VNFE improves the AR by 13.14% compared to TDRL-DDPG, 34.35% compared to GRU-A3C, and 57.45% compared to GRC. Figure 3 depicts the average power consumption of the four compared algorithms. The GRU-A3C has the lowest energy consumption, followed by GPG-VNFE, TDRL-DDPG and finally GRC. The proposed GPG-VNFE achieves an average power consumption of approximately 32.5×10^3 watts, compared to 31.8×10^3 watts for TDRL-DDPG and 37.7×10^3 watts for GRC. Although the baseline GRU-A3C records the lowest consumption (around 26.7×10^3 watts), it does so at the expense of a markedly reduced acceptance ratio, whereas GPG-VNFE aims to find a tradeoff between energy consumption and acceptance ratio. In fact, GPG-VNFE simultaneously maximizes the acceptance ratio and minimizes power consumption, a dual-objective optimization that is inherently more demanding than a single-metric tuning. The results, therefore, underscore GPG-VNFE ability to deliver the best overall trade-off, maintaining a high acceptance ratio while keeping energy usage at a competitive level.

Figure 4 presents the acceptance ratio when a single node and link fail. We simulate a single node and a single link

failure at the 500th incoming request. Our proposal shows the highest acceptance ratio demonstrating its rapid recovery after the failure point. It stabilizes quickly and maintains an acceptance ratio above 80%, indicating high robustness against network failure. On the other hand, TDRL-DDPG has an acceptance ratio around 70% before and slightly dropping below 70% after failure. For GRU-A3C, after the failure point, a significant drop occurs, bringing the ratio down to nearly 40% at 1000 requests. Finally, GRC is the most significantly impacted by the failure event and has the lowest overall acceptance ratio, starting just above 50% and decreasing linearly to around 25% by the end. GPG-VNFE demonstrates the highest robustness against failures due to its foundation model design that generalizes well. As already explained, the generative pre-training enables better generalization, leading to superior handling of unforeseen network failures.

VI. Conclusion

This paper proposes an energy-efficient algorithm for VNF-FG placement and chaining in virtualized infrastructures. We proposed GPG-VNFE, a Generative Graph Pretraining Transformer for VNF placement that combines DRL with contrastive graph representation learning. The proposed approach integrates a contrastive learning module with learnable graph perturbation priors to increase its robustness to unseen VNF requests or network failures. GPG-VNFE aims simultaneously to increase the quality of service in terms of VNF requests acceptance ratio and decreases the energy footprint. By extensive simulation, results show that GPG-VNFE increases the acceptance by up to 57.45% compared with state-of-the-art approaches and decreases energy consumption by up to 11%.

References

[1] Bo Yi et al. A comprehensive survey of Network Function Virtualization. *Computer Networks*, 133:212–262, 2018.

[2] Junfeng Xie et al. A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(1):393–430, 2018.

[3] Rania Sahraoui, Omar Houidi, and Fetia Bannour. Energy-Aware VNF-FG Placement with Transformer-based Deep Reinforcement Learning. In *NOMS 2024 IEEE Network Operations and Management Symposium, Seoul, Republic of Korea, May 6-10, 2024*, pages 1–9. IEEE, 2024.

[4] Yuning You et al. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[5] Fan-Yun Sun et al. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[6] Yuning You et al. Graph contrastive learning automated. In *International conference on machine learning*, pages 12121–12132. PMLR, 2021.

[7] Yuning You et al. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 1300–1309, 2022.

[8] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[9] Marcelo Caggiani Luizelli et al. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *IFIP/IEEE IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, pages 98–106, 2015.

[10] Venkatarami Reddy Chintapalli et al. RAVIN: A resource-aware VNF placement scheme with performance guarantees. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2023.

[11] Xiangqiang Gao et al. Dynamic resource allocation for virtual network function placement in satellite edge clouds. *IEEE Transactions on Network Science and Engineering*, 9(4):2252–2265, 2022.

[12] Oussama Soualah et al. Online and batch algorithms for vnfs placement and chaining. *Computer Networks*, 158:98–113, 2019.

[13] Andrea Tomassilli et al. Provably efficient algorithms for placement of service function chains with ordering constraints. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 774–782, 2018.

[14] Omar Houidi et al. An efficient algorithm for virtual network function scaling. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7, 2017.

[15] Amir Varasteh et al. Holu: Power-aware and delay-constrained vnf placement and chaining. *IEEE Transactions on Network and Service Management*, 18(2):1524–1539, 2021.

[16] Selma Khebbache et al. A multi-objective non-dominated sorting genetic algorithm for vnf chains placement. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4, 2018.

[17] Pham Tran Anh Quang et al. A deep reinforcement learning approach for vnf forwarding graph embedding. *IEEE Transactions on Network and Service Management*, 16(4):1318–1331, 2019.

[18] Siyu QI et al. Energy-efficient VNF deployment for graph-structured SFC based on graph neural network and constrained deep reinforcement learning. In *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 348–353. IEEE, 2021.

[19] Omar Houidi et al. Energy Efficient VNF-FG Embedding via Attention-Based Deep Reinforcement Learning. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, 2023.

[20] Mohammad M Tajiki et al. Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining. *IEEE Transactions on Network and Service Management*, 16(1):374–388, 2018.

[21] Chuan Pham et al. Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach. *IEEE Transactions on Services Computing*, 13(1):172–185, 2017.

[22] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[24] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pages 1174–1183. PMLR, 2018.

[25] Ashish Bora et al. Compressed sensing using generative models. In *International conference on machine learning*, pages 537–546. PMLR, 2017.

[26] Morteza Mardani et al. Deep Generative Adversarial Neural Networks for Compressive Sensing MRI. *IEEE transactions on medical imaging*, 38(1):167–179, 2018.

[27] Jiaxuan You et al. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[28] Yujia Li et al. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[29] Yonglong Tian et al. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

[30] Tailin Wu et al. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

[31] Christopher Morris et al. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[32] Zhongxia Yan et al. 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.

[33] Tianfu Wang et al. DRL-SFCP: Adaptive Service Function Chains Placement with Deep Reinforcement Learning. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6, 2021.

[34] Long Gong et al. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2014.