GMR
The Original

A LATENCY-AWARE AND LOAD-BALANCED VIRTUAL NETWORK EMBEDDING FRAMEWORK USING GRAPH ATTENTION NETWORKS AND HYBRID PROXIMAL POLICY OPTIMIZATION

Mrs. B. S. Sukanya[1], Dr. L. Sudha[2]

[1]Research Scholar Department of Computer Science, VET Institute of Arts and Science College, Erode -638012, Tamil Nadu, India.
[2]Associate Professor Department of Computer Science, VET Institute of Arts and Science College, Erode -638012, Tamil Nadu, India.

Abstract: Virtual Network Embedding (VNE) is crucial for efficiently mapping virtual requests onto physical infrastructures in large-scale networks such as 5G, IoT, and edge-cloud systems. However, existing approaches often struggle to adapt to dynamic network states, minimize latency, balance resource loads, and achieve rapid and stable learning. To address these challenges, this study introduces ALIVE (Adaptive Latency-aware Intelligent Virtual Network Embedding), an enhanced VNE framework incorporating four key innovations. First, latency sensitivity and load balancing are integrated as core optimization objectives to enable low-delay path selection while preventing resource bottlenecks. Second, Graph Attention Networks (GATs) are utilized to improve node representation by dynamically weighting neighbor importance, which is particularly beneficial in heterogeneous and time-varying network topologies. Third, Hybrid Proximal Policy Optimization (HPPO) is employed to stabilize the learning process, accelerate convergence, and enhance robustness. Finally, Meta-Gradient Learning is incorporated for adaptive learning rate control, allowing autonomous tuning of learning dynamics based on reward feedback. Simulation results demonstrate that ALIVE significantly outperforms existing state-of-the-art methods by achieving a higher acceptance ratio, lower latency, improved load balance, stable convergence, and superior adaptability under real-time network conditions.

KEYWORDS: Virtual Network Embedding (VNE); Graph Attention Networks (GAT); Hybrid Proximal Policy Optimization (HPPO); Meta-Gradient Learning; Latency Sensitivity; Load Balancing; 5G Networks; Software-Defined Networking (SDN); Network Function Virtualization (NFV); Reinforcement Learning

INTRODUCTION

The rapid evolution of networked applications and services, driven by emerging technologies such as 5G, Internet of Things (IoT), augmented and virtual reality (AR/VR), and ultra-reliable low-latency communications (URLLC), has introduced unprecedented complexity in network resource management. These applications require highly dynamic, flexible, and efficient allocation of resources to meet stringent Quality of Service (QoS) and Service Level Agreement (SLA) requirements, including latency guarantees, high throughput, and fault resilience. Traditional network architectures, which rely on static resource provisioning, are often ill-suited to handle the variability and scale of modern heterogeneous networks.

As the mobile communication networks have been widely deployed over the world, the era of interacting with everything is around the corner [1]. Since the expanding scale of the Internet of Things (IoT) has cultivated enormous data traffic and various service requests [2]–[4], massive terrestrial networks are built to meet explosive user demands. However, limited by the landform, fairly extensive areas, such as abysmal seas, polar regions and mountains cannot be equipped with traditional cellular base stations [5]–[7]. Therefore, in order to realize the vision of Internet of everything (IoV) in future sixth-generation (6G) mobile communications, guaranteeing both seamless information coverage and user's quality of service (QoS) become key success factors.

Satellite networks are emerging as powerful supplements by providing global high-capacity coverage [8]–[10]. Considering satellites' high altitude orbit, satellite networks can be free from geographic limitations, even can keep good running in the context of natural disasters. Besides, some recent techniques may grant the satellite network lower power loss and shorter communication delay [11], [12], facilitating it to be an essential component in future 6G space-ground integrated networks (SGIN) [13], [14]. Unlike terrestrial networks, due to the relative motion of satellites in different orbits, inter-satellite links (ISLs) are periodically connected. The link establishment is frequent and unstable. Hence, the network topology of terrestrial networks is static, while that of satellite networks is dynamic. Moreover, satellite nodes are often limited in their weight, volume, and energy consumption, which leads to limited onboard resources. The resource load balance should be achieved to make

full use of physical resources to complete more service requests. Therefore, efficient satellite network configuration and resource allocation schemes are called to support various service requests.

Network virtualization can abstract underlying hardware resources to software, enabling multiple network applications to share the same physical network resources [15], where service providers can provide more diverse virtual networks on the same physical network to meet various service demands. As the core technology of network virtualization, virtual network embedding (VNE) algorithms are able to make full use of underlying hardware resources and meet more virtual network requests (VNRs), guaranteeing various QoS requirements and network operators' revenue [16]. Since the VNE problem is NP-hard, the majority of existing studies focus on many heuristic VNE algorithms with a series of hand-crafted rules and constraints [17]–[19]. As a result, these solutions may be sub-optimal. Additionally, most of them are based on static terrestrial network topology rather than dynamic satellite networks. Satellite networks have been a promising approach to accomplishing virtual network collaborative computing services. Due to the limited onboard resources of the satellite, it is necessary to virtualize the network resources and computing storage capacity of satellites. The results can be sent to users through collaborative computing of several satellite nodes. Besides, the substrate resources can be utilized fully to serve more tasks. Compared with terrestrial networks, satellite networks are characterized by high dynamic and large scale. Hence, traditional heuristics based on static pre-defined rules are not adaptable to dynamic satellite network topologies. The intelligent VNE algorithms are necessary, which can adjust policies according to the network environment. Thanks to the development of machine learning, a range of intelligent algorithms have been introduced to solve the network resource allocation problem [20]. Particularly, as the combination of deep learning and reinforcement learning, deep reinforcement learning (DRL) methods inherit their powerful perception ability and decision-making ability. With the help of deep neural networks, DRL agents can interact with the environment states and extract feature information from highdimensional state space. After performing actions, corresponding rewards are achieved to justify if the performed actions are feasible. To maximize cumulative rewards, DRL agents can automatically optimize their strategies through training and self-learning, which is the potential to solve high-dimensional sequential decision-making problems such as traffic grooming and resource scheduling.

Traditional VNE approaches, such as heuristic algorithms and integer linear programming (ILP), often struggle to adapt to the dynamic and unpredictable nature of modern networks [21]. These methods can be computationally expensive, especially for large-scale networks, and may not effectively handle real-time changes in network conditions or traffic demands [22].

To address these limitations, machine learning-based approaches have gained significant traction in recent years. Deep Reinforcement Learning (DRL), in particular, has shown great promise due to its ability to learn optimal policies for sequential decision-making problems. DRL agents can learn to make decisions based on observed states (network conditions, VNR characteristics) and receive rewards based on the outcomes of their actions (resource allocation decisions).

Graph Convolutional Networks (GCNs) offer a powerful tool for extracting meaningful features from graph-structured data, such as the network topology [23]. GCNs can effectively capture the relationships between nodes and edges in the network, providing valuable insights for the VNE process.

Network virtualization has emerged as a key paradigm to address these challenges by decoupling logical network services from the underlying physical infrastructure. Virtual Network Embedding (VNE), the process of mapping virtual network requests (VNRs) onto physical substrate resources, plays a central role in this paradigm. Efficient VNE ensures optimal utilization of substrate resources, improves acceptance ratios for incoming VNRs, and guarantees the performance requirements of latency-sensitive and mission-critical services. However, existing VNE solutions face multiple limitations:

Latency Insensitivity – Many algorithms optimize acceptance ratio and cost efficiency but overlook end-to-end latency, which is critical for real-time applications such as autonomous driving and immersive media.

Load Imbalance – Current frameworks often result in uneven resource utilization, leading to localized congestion, bottlenecks, and reduced network reliability.

Feature Extraction Limitations – Graph learning models such as Graph Isomorphism Networks (GINs) treat all neighboring nodes equally, failing to differentiate between critical and non-critical connections in heterogeneous topologies.

Learning Instability – While deep reinforcement learning (DRL) approaches such as Asynchronous Advantage Actor-Critic (A3C) have shown promise, they exhibit high variance in policy updates and slow convergence in dynamic environments.

Static Learning Parameters – Many DRL-based VNE solutions rely on fixed learning rates or manually tuned parameters, limiting their adaptability to rapidly changing traffic and topology conditions.

To address these gaps, this research proposes a Latency-Aware and Load-Balanced Virtual Network Embedding Framework that integrates four key innovations:

Graph Attention Networks (GATs) for adaptive neighbor weighting, enabling fine-grained and topology-aware feature extraction.

Hybrid Proximal Policy Optimization (HPPO) combining the stability of Proximal Policy Optimization (PPO) with the parallel efficiency faster, more stable policy convergence.

Meta-Gradient Learning to autonomously adjust learning rates, ensuring rapid adaptation under varying network conditions.
A multi-objective reward function incorporating latency sensitivity and load balancing, alongside traditional objectives such as acceptance ratio and resource efficiency.
The proposed framework is designed for scalable, real-time VNE in large heterogeneous networks. Simulation results demonstrate that the framework significantly outperforms state-of-the-art methods by improving acceptance ratio, latency, load fairness, and convergence stability, making it well-suited for future 5G and edge-cloud infrastructures.
The rest of the paper is organized as follows: Section 2 discusses the related work and existing VNE techniques. Section 3 details the proposed latency-aware and load-balanced framework with its architecture, algorithms, and mathematical formulation. Section 4 presents the simulation setup and performance evaluation. Section 5 concludes the paper and highlights future research directions.

RELATED WORK

Recently, machine learning has proved to be a promising direction for solving combinatorial optimization problems. According to the nature of the available learning signal, machine learning is usually categorized into three major categories: supervised learning (SL), unsupervised learning (uSL), and reinforcement learning (RL). Several works investigate machine learning applications to the VNE problem, and we introduce some of them following the three categories mentioned above.
Supervised learning has achieved great success in image recognition, text classification, and other scenarios. Treating the admission control mechanism of VNE as a binary classification problem, Andreas et al. [24] applied this idea to improve the existing VNE algorithm's performance by utilizing a Recurrent Neural Network (RNN) to judge whether to admit or reject arriving VNRs, but only considered the rejection of infeasible VNRs to avoid wasting time. Different from supervised learning relying on labeled samples, unsupervised learning is capable of discovering underlying structures in unlabeled data. To reduce the large search space, Blenk et al. [25] applied the Hopfield neural network to extract subgraphs fed to other existing VNE algorithms, resulting in faster and more resource-efficient embeddings. Habibi et al. [26] adopted the graph autoencoder (GAE) to cluster physical nodes and then randomly sample one node as the initial center to execute the breadth-first search (BFS) method in each cluster.
The embedding process of VNE can be regarded as a series of decisions, and some studies tackled this problem by reinforcement learning (RL). Modeling the node selection process as a Markov decision process (MDP), Haeri et al. [27] utilized the Monte Carlo tree search (MCTS) to improve the revenue-to-cost ratio. Wang et al. [28] trained a multi-layer perceptron (MLP) model with temporal-difference learning to approximate the value function of VNE states. Using a convolutional neural network (CNN) to extract the attributes of physical nodes and training it with REINFORCE algorithm, Li et al. [29] proposed a double-layer RL-based framework for VNE to learn the node mapping and link mapping process jointly. Yao et al. [30] designed a sequence-to-sequence model based on a recurrent neural network (RNN) to exploit the historical information of previous decisions, and train with the REINFORCE algorithm. But these traditional neural networks often exhibit limited representational capabilities when dealing with non-Euclidean data, requiring manual topological feature extraction like closeness and betweenness from graph theory. In contrast, graph neural networks (GNNs) excel at automatically mining deep features from graph data, generating richer. Cheng et al. [31] proposed a hierarchical RL control framework to select the VNR from batch candidates stored in the time window, where the GCN is used for the rough feature exploitation. However, their subagent for admission control only works for offline settings, e.g., within the time window batch, and is not unqualified to tackle the online VNE where future VNRs are not known in advance. In conclusion, existing RL-based works only concentrate on the resource allocation of arriving VNRs, or exploit partial latent temporal or topological features of the physical network and VNRs.
In [32] a novel enhancement to the InDS framework is proposed for addressing its limitations and improving its applicability in dynamic network environments. By integrating multi-objective optimization, advanced feature extraction, and adaptive learning techniques, the enhanced framework achieves superior performance in resource utilization, adaptability, and scalability. The proposed Adaptive InDS framework significantly outperforms existing methods in terms of all three metrics: acceptance ratio, revenue-to-cost ratio, and resource utilization. By dynamically adjusting to network conditions and incorporating diverse objectives, Adaptive InDS enhances overall efficiency and scalability, providing a robust solution for virtual network embedding in dynamic environments. Both Adaptability and Improved Stability are key strengths of the Adaptive InDS framework. It significantly outperforms other methods, demonstrating excellent adaptability to changing conditions and superior stability in fluctuating network environments. On the other hand, MCL demonstrates poor adaptability and stability, while methods like InDS and TVAE also struggle to maintain high levels of both adaptability and stability in dynamic settings. The Adaptive InDS framework's ability to seamlessly adjust to real-time changes while maintaining consistent performance makes it an ideal solution for dynamic network environments.
Unlike existing VNE frameworks that independently employ graph neural networks, reinforcement learning, or adaptive optimization strategies, the proposed ALIVE framework introduces a unified architecture that tightly

couples topology-aware attention learning, hybrid policy optimization, and meta-gradient adaptation within a single embedding pipeline. Existing GCN-DRL and PPO-based approaches typically operate with fixed learning configurations and static optimization objectives, limiting their ability to adapt to dynamic network environments. In contrast, ALIVE continuously refines both policy parameters and learning dynamics through meta-gradient feedback while simultaneously leveraging attention-based graph representations to capture heterogeneous substrate resource conditions. This integrated design enables adaptive latency-aware embedding decisions, faster convergence, improved resource utilization, and enhanced robustness under varying traffic conditions. Therefore, the novelty of ALIVE lies not merely in combining GAT, HPPO, and meta-learning, but in establishing a jointly optimized learning framework where representation learning, policy optimization, and hyperparameter adaptation interact cooperatively during embedding decision making.

These gaps motivate the proposed latency-aware and load-balanced VNE framework, which leverages Graph Attention Networks (GATs) for advanced feature extraction, Hybrid Proximal Policy Optimization (HPPO) for stable and efficient learning, and Meta-gradient Learning for autonomous learning rate adaptation. The framework introduces a multi-objective reward function that balances latency, load distribution, acceptance ratio, fairness, and energy efficiency, providing a robust solution for next-generation networks.

PROPOSED METHODOLOGY

The proposed latency-aware and load-balanced Virtual Network Embedding (VNE) framework integrates Graph Attention Networks (GATs) for feature extraction, Hybrid Proximal Policy Optimization (HPPO) for stable and parallel reinforcement learning, and Meta-Gradient Learning for autonomous learning rate adaptation. The overall workflow consists of six phases: network representation, feature extraction, state representation, policy optimization, reward calculation, and resource allocation (Figure 1).

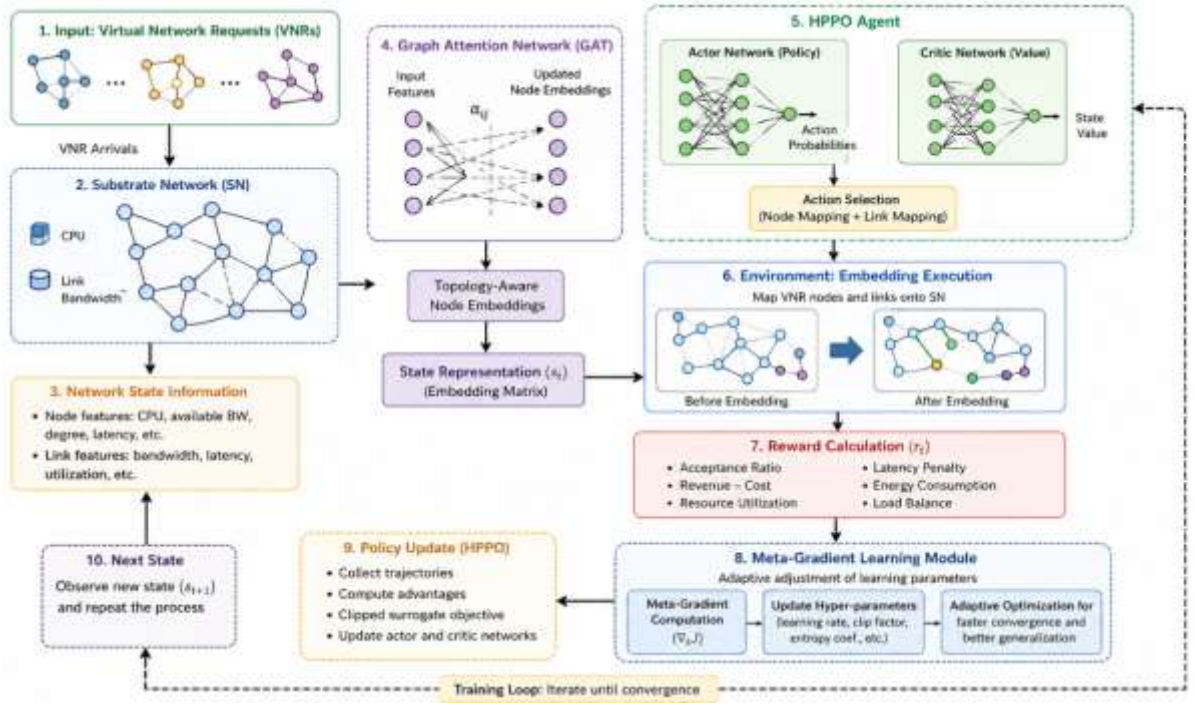

Figure 1: Proposed Block Diagram

Traditional VNE solutions focus primarily on maximizing the acceptance ratio or revenue-to-cost ratio but neglect end-to-end latency and load balancing, which are critical for real-time services (e.g., AR/VR, autonomous vehicles, URLLC in 5G). Furthermore, existing feature extraction methods based on Graph Isomorphism Networks (GINs) or Graph Convolutional Networks (GCNs) treat all node neighbors equally, failing to differentiate the importance of links and neighbor nodes in heterogeneous topologies.

Similarly, reinforcement learning methods such as Asynchronous Advantage Actor-Critic (A3C) are prone to high variance and unstable convergence, especially in dynamic environments, while Proximal Policy Optimization (PPO) provides stable learning but lacks asynchronous parallel execution. To address these issues, we combine GATs (for adaptive neighbor importance weighting) with a HPPO that merges PPO's stability, and employ Meta-Gradient Learning to autonomously tune the learning rate, ensuring robust learning under dynamic conditions.

Deep Reinforcement Learning Framework for Latency-Aware and Load-Balanced Online Virtual Network Embedding

The growing demand for low-latency, reliable, and load-balanced network services has made Virtual Network Embedding (VNE) increasingly complex. Traditional heuristic and static approaches are insufficient in dynamic and heterogeneous networks such as 5G, IoT, and edge-cloud systems. To address these challenges, this work presents a Deep Reinforcement Learning (DRL)-based VNE framework that integrates Graph Attention Networks (GATs) for advanced feature extraction and Hybrid Proximal Policy Optimization (HPPO) with Meta-Gradient Learning for scalable, stable, and adaptive decision-making.Overview of Proposed Enhancements

The proposed framework advances VNE performance in three core areas:

Latency and Load-Aware Multi-Objective Reward The framework introduces an enhanced reward function incorporating multiple objectives:

1. Latency Sensitivity: Prioritizing low-delay paths.
2. Load Balancing: Ensuring even utilization of network resources.
3. Revenue-to-Cost Ratio: Optimizing cost efficiency.
4. Fairness and Energy Efficiency: Preventing biased allocation and promoting energy savings.
5. These objectives are jointly optimized through a weighted sum reward mechanism.
6. Advanced Feature Extraction using Graph Attention Networks
7. Instead of treating all node neighbors equally (as in GCN or GIN), the framework uses GAT, which dynamically weights neighbors based on importance. This enables extraction of context-rich embeddings capturing structural and resource-aware features, critical for dynamic topologies like 5G and edge-cloud infrastructures.
8. Stable and Adaptive Policy Learning with HPPO
9. The policy learning process uses Hybrid Proximal Policy Optimization (HPPO):
10. Combines PPO's clipped surrogate objective with trajectory reuse for improved sample efficiency.
11. Enables stable updates and prevents policy oscillation under dynamic conditions.
12. Integrates Meta-Gradient Learning for adaptive learning rate tuning.
13. Workflow and Methodology

The workflow consists of six steps:

Network Representation

The substrate network (physical network) needs to be represented in a way that captures both the physical topology and the available resources. This is typically done using a graph representation, where nodes represent network elements (e.g., routers, switches, servers), and edges represent the communication links between them.

Nodes: Represent network devices, servers, or resources. They are characterized by available computational resources (e.g., CPU, memory) and bandwidth.

Edges: Represent communication links between nodes, characterized by bandwidth, delay, and latency.

The physical substrate network (PSN) is modeled as a weighted undirected graph is referred in eqn (1):

$$G_s = (N_s, L_s) \quad (1)$$

where:

$N_s = \{1,2,\ldots,| N_s |\}$ represents substrate nodes. Each node $i \in N_s$ is characterized by computational capacity C_i , memory M_i, and energy E_i.

$L_s \subseteq N_s \times N_s$ represents substrate links. Each link $(\mathbf{i}, \mathbf{j})$ has available bandwidth B_{ij} and propagation delay D_{ij}.

Each virtual network request (VNR) is modeled as:

$$G_v = (N_v, L_v) \quad (2)$$

where each virtual node $u \in N_v$ requires CPU capacity C_u, and each virtual link $(u, v) \in L_v$ demands bandwidth B_{uv} and has a maximum latency constraint D_{uv}^{max}.

The objective is to find an embedding:

$$f_N: N_v \rightarrow N_s, f_L: L_v \rightarrow P_s \quad (3)$$

where P_s is a set of substrate paths, ensuring that all resource constraints are satisfied while optimizing for latency, fairness, and balanced resource utilization.

Feature Extraction Using Graph Attention Networks (GAT)

Feature extraction plays a critical role in capturing both topological and resource-aware properties of the substrate network. Unlike Graph Isomorphism Networks (GINs) or Graph Convolutional Networks (GCNs), which treat all neighboring nodes equally, Graph Attention Networks (GATs) assign attention weights to neighbors based on their relative importance.

Structural Features: Node degree, available CPU, memory, link bandwidth, and path delay.

Temporal Features: These refer to dynamic network behaviors over time, like traffic load, network congestion, or node failure.

Each substrate node i is associated with a feature vector:

$$x_i = [C_i, M_i, E_i, \deg(i), \overline{B}_i, \overline{D}_i] \quad (4)$$

where:

$\deg(i)$: node degree (number of connections),

$\overline{B}_i = \sum_{j \in N(i)} B_{ij}$: total available bandwidth of neighboring links,

$\overline{D}_i = \sum_{j \in N(i)} D_{ij}$: total delay to neighbors.

Attention Mechanism

For each pair of connected nodes (i, j), an attention score is computed:

$$e_{ij} = \text{LeakReLU}(a^T [Wx_i \| Wx_j]) \quad (5)$$

where:

W is a learnable weight matrix for node feature transformation,

a is the attention vector,

$\|$ denotes vector concatenation.

The normalized attention coefficients are then computed using a softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})} \quad (6)$$

These coefficients capture the relative importance of neighbor j to node i.

Node Embedding Update

Node embeddings are updated using the attention-weighted sum of neighboring features:

$$h_i' = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W x_j\right) \quad (7)$$

where $\sigma(\cdot)$ is a non-linear activation (e.g., ELU). This provides a state representation that reflects topology, node attributes, and neighbor importance.

Algorithm 2: Feature Extraction using Graph Attention Networks (GAT)
Input: Substrate network graph $G_s = (N_s, L_s)$, node features x_i.
Output: Node embeddings h_i'.
Initialize learnable weight matrix W and attention vector a
For each node $i \in N_s$:
Compute initial feature vector: $x_i = [C_i, M_i, E_i, \deg(i), \overline{B}_i, \overline{D}_i]$
For each edge $(i, j) \in L_s$:
Compute attention coefficient: $e_{ij} = \text{LeakReLU}(a^T [Wx_i \| Wx_j])$
For each node $i \in N_s$:
Normalize coefficients: $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})}$
Update node embedding: $h_i' = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W x_j\right)$
Return final embeddings h_i' for all nodes

Decision-Making using HPPO

Once temporal and structural features are extracted from the substrate network using Graph Attention Networks (GATs), the decision-making process determines how to optimally map Virtual Network Requests (VNRs) to available substrate resources. In the proposed framework, the decision-making process is powered by a Hybrid Proximal Policy Optimization (HPPO) policy network.

The HPPO network takes as input the temporal GAT embeddings, representing both the current resource availability and predicted traffic behavior, and outputs a set of candidate actions. Each action corresponds to mapping a virtual node to a substrate node and allocating links along feasible low-latency paths while ensuring balanced resource utilization.

Unlike traditional policy learning approaches that rely on fixed update frequencies or static learning rates, HPPO incorporates:

On-policy trajectory sampling for stability,

Off-policy trajectory replay for sample efficiency, and

Meta-gradient-based learning rate adjustment to improve adaptability in rapidly changing environments.

The HPPO output provides an optimized policy that simultaneously considers latency sensitivity, load balancing, fairness, and energy efficiency, ensuring that selected actions maximize overall network performance.

Policy Training using Hybrid Proximal Policy Optimization (HPPO)

The policy training phase leverages reinforcement learning to determine optimal embedding decisions. A hybrid variant of Proximal Policy Optimization (PPO), referred to as Hybrid PPO (HPPO), is adopted to combine the stability of PPO with the parallel execution capabilities of Asynchronous Advantage Actor-Critic (A3C).

Proximal policy optimization:

PPO has become a default baseline in a variety of applications, as mentioned above. It is favored because of its strong performance and simple implementation with sound theoretical motivation given by the policy improvement lower bound. Intuitively, PPO attempts to constrain the new policy close to the present one with a clipping heuristic, which results in the most popular variant, PPO-clip Jin et al. (2023). Particularly, the following objective is solved at every policy update:

$$\mathcal{L}_k^{\text{clip}}(\pi) = \mathbb{E}_{(s,a)\sim d^{\pi_k}}\left[\min\left(\frac{\pi(a|s)}{\pi_k(a|s)} A^{\pi_k}(s,a), \text{clip}\left(\frac{\pi(a|s)}{\pi_k(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_k}(s,a)\right)\right] \quad (8)$$

where clip(a, b, c) = min(max(a, b), c). The clipping function plays a critical role in this objective as it consistently enforces the probability ratio between the current and next policies in a reasonable range between $[1 - \epsilon, 1 + \epsilon]$.

The outer minimization in Eq. 2 provides the lower bound guarantee for the surrogate loss in Eq. 1. In practice, one can set a small learning rate and a large number of time steps to generate sufficient samples to allow PPO to perform stably and approximate Eq. 2. However, due to its on-policy approach, high variance is a significant issue such that an extremely large number of samples may be required in some scenarios to make sure the empirical objective is able to precisely estimate the true objective in Eq. 2, which naturally causes the high sample complexity issue. This motivates us to leverage off-policy techniques to alleviate such an issue, while keeping the theoretical policy improvement.

Hybrid- Proximal policy optimization (HPPO)

To achieve better sample efficiency of PPO, historical samples generated by previous policies are reused for policy updates, as done in off-policy algorithms. This inevitably results in a distribution drift between policies, which essentially disproves the policy improvement lower bound in Lemma 1. In this context, to fix this issue, we will extend Lemma 1 to assimilate off-policy samples in a principled manner to derive a new policy improvement lower bound that works for our proposed algorithm, HPPO. HPPO takes a hybrid approach that effectively synthesizes on-policy trajectory-wise policy updates and off-policy trajectory replay buffers. Algorithm 1 shows the algorithm framework for HPPO.

Algorithm 1: HPPO

Input: initializations of θ_0, ϕ_0, and trajectory replay buffer R, the number of episodes K, the number of time steps in each episode T, the number of epochs for updates E

for $k = 1, 2, \ldots, K$ do

Run policy π_{θ_k} to generate a trajectory $\tau = (s_0, a_0, r_1, s_1, \ldots, s_{T-1}, a_{T-1}, r_T)$

Append τ to R and discard the oldest one $\tau - \triangleright$ $FIFO$ strategy

Sample a random minibatch B from the trajectory replay buffer R

Select the best action trajectory τ_k^* from the trajectory replay buffer and add it to B

for each trajectory $j = 1, 2, \ldots, |B|$: do

for $t = 0, 1, \ldots, T - 1$ do

$$G_t^j = \sum_{l=t+1}^{T} \Upsilon^{l-t-1} r_l^j$$

end for

end for

Compute advantage estimates $\hat{A}_t^{\pi k} = G_t - V_\phi(s_t)$

for each epoch $e = 1, 2, \ldots, E$ do

Compute the clipping loss Eq. 2

Compute the mean square loss $L^V(\phi) = -\frac{1}{T}\sum_{t=0}^{T-1}\left(G_t - V_\phi(s_t)\right)^2$

Update π_{θ_k} with $\nabla_\theta L^{clip}(\theta)$ by Adam

Update V_{ϕ_k} with $\nabla_\phi L^{clip}(\phi)$ by Adam

end for

end for

return π_{θ_k} and V_{ϕ_k}

Algorithm 2: Policy Optimization using Hybrid Proximal Policy Optimization (HPPO)
Input: State embeddings (from GAT), environment transitions.
Output: Optimized actor (π) and critic (V) networks.
Initialize actor network π_θ and critic network V_φ
Initialize trajectory replay buffer R (FIFO)
For each training iteration k = 1 to K:
Run policy π_θ to collect trajectories $\tau = (s, a, r, s')$
Append τ to R and discard oldest if buffer full
Sample mini-batch B from R
For each trajectory in B:
Compute return: $G_t^j = \sum_{l=t+1}^{T} \Upsilon^{l-t-1} r_l^j$
Compute advantage: $\hat{A}_t^{\pi k} = G_t - V_\phi(s_t)$
Compute PPO clipping loss: $\nabla_\theta L^{clip}(\phi) = -\frac{1}{T}\sum_{t=0}^{T-1}\left(G_t - V_\phi(s_t)\right)^2$
Compute critic loss: $L^V(\varphi) = mean(\left(G_t - V_\varphi(s_t)\right)^2)$
Update actor parameters θ using Adam optimizer on $\nabla_\theta L^{clip}(\theta)$
Update critic parameters φ using Adam optimizer on $\nabla_\phi L^{clip}(\phi)$
Return optimized π_θ and V_φ

Meta-Gradient Learning for Learning Rate Adaptation

Conventional reinforcement learning often relies on manually tuned or fixed learning rates, which are suboptimal in dynamic environments. Instead of manually tuning the learning rate, the proposed framework integrates Meta-Gradient Learning, where the learning rate α_t is treated as a trainable parameter. Its update rule is:

$$\alpha_{t+1} = \alpha_t - \eta \frac{\partial L(\theta_t - \alpha_t \nabla L_t)}{\partial \alpha_t} \quad (9)$$

where η is the meta-learning rate and L_t is the loss at time t. This adaptive mechanism allows the model to increase learning aggressiveness in stable training phases and reduce it during periods of high variance, ensuring faster convergence and preventing policy oscillations. This ensures faster convergence in stable regions and controlled learning during high-variance updates.

Algorithm 4: Meta-Gradient Learning for Learning Rate Adaptation
Input: Learning rate α_t, model parameters $\theta_t, loss\ L_t$.
Output: Adapted learning rate $\alpha_{\{t+1\}}$.
1: Initialize meta learning rate η
2: For each iteration t:
3: Compute meta-gradient:
4: $g = \frac{\partial L(\theta_t - \alpha_t \nabla L_t)}{\partial \alpha_t}$
5: Update learning rate:
6: $\alpha_{\{t+1\}} = \alpha_t - \eta * g$
7: Return updated learning rate $\alpha_{\{t+1\}}$

Multi-Objective Reward Function

To guide the policy towards multi-faceted performance improvements, the reward function incorporates multiple key performance indicators. In reinforcement learning for Virtual Network Embedding (VNE), the agent's goal is to maximize the reward function by choosing actions (e.g., embedding virtual nodes and links) that optimize multiple competing objectives simultaneously. The challenge arises from balancing different, often conflicting, network performance metrics. The proposed multi-objective reward function aims to capture several key performance indicators (KPIs) essential for successful VNE:

Acceptance Ratio

Revenue-to-Cost Ratio

Fairness

Latency Sensitivity

Load Balancing

Each of these components is integrated into the reward function and influences the policy learned by the reinforcement learning agent. The reward is formulated as a weighted sum of these individual objectives.

Reward Components

Acceptance Ratio ($\boldsymbol{R_{acc}}$)

The Acceptance Ratio measures how many of the incoming Virtual Network Requests (VNRs) can be successfully embedded into the physical network. Maximizing the acceptance ratio ensures that the VNE framework can accommodate as many VNRs as possible, improving the network's overall efficiency.

$$R_{acc} = \frac{\#Accepted\ VNRs}{\#Total\ VNRs} \quad (10)$$

Objective: Maximize this ratio to ensure a higher number of accepted VNRs without overwhelming the network resources.

Revenue-to-Cost Ratio ($\boldsymbol{R_{rev/cost}}$)

The Revenue-to-Cost Ratio evaluates the economic efficiency of the VNE process. It compares the revenue generated by embedding virtual networks with the cost of using the physical substrate network resources. This metric helps balance profitability with resource utilization.

$$\boldsymbol{R_{rev/cost}} = \frac{Revenue\ from\ Embedded\ VNRs}{Cost\ of\ Resource\ usage} \quad (11)$$

Objective: Maximize this ratio to ensure that VNE is cost-effective while delivering optimal service quality.

Fairness ($\boldsymbol{Rfair}$)

Fairness ensures that the resources are distributed among all virtual network requests in an equitable manner, preventing the over-prioritization of some requests while others are starved of resources. Fairness is measured using the variance of node utilization across the substrate network, aiming to minimize disparities in resource allocation.

$$R_{fair} = 1 - Var(U_n) \quad (12)$$

where U_n represents the utilization of node n, and $Var(U_n)$ is the variance of utilization across all nodes. Lower variance indicates fairer resource allocation.

Objective: Minimize resource imbalances to achieve equitable allocation among virtual network requests.

Latency Sensitivity ($\boldsymbol{R_{lat}}$)

Latency Sensitivity addresses the delay constraints required by certain types of applications (e.g., real-time video streaming, autonomous vehicles). The goal is to minimize end-to-end latency for virtual links, ensuring that time-sensitive services are not affected by excessive delays.

$$\boldsymbol{R_{lat}} = \frac{1}{\|P\|}\sum_{(i,j)\in P} D_{ij} \quad (13)$$

where P is the set of paths used to route virtual links, and D_{ij} is the delay for the path between substrate nodes i and j.

Objective: Minimize latency for virtual network links, especially for real-time applications that have strict delay requirements.

Load Balancing ($\boldsymbol{R_{imbalance}}$)

Load Balancing aims to prevent resource congestion and bottlenecks that occur when specific substrate nodes or links are overloaded. A balanced load distribution across all nodes ensures efficient resource utilization and prevents any node from becoming a point of failure.

$$R_{Imbalance} = \sqrt{\frac{1}{|N_s|}\sum_n (u_n - \bar{u})^2} \quad (14)$$

where:

u_n represents the utilization of node n,

$\bar{u}$ is the average utilization across all nodes in the substrate network.

Objective: Minimize the imbalance in resource allocation, aiming to distribute load evenly across the network.

3.4.2 Complete Multi-Objective Reward Function

The total reward function R_t is a weighted sum of the individual components, designed to capture all essential performance metrics. The reward function at each time step t is given by:

$$R_t = w_1 R_{acc} + w_2 R_{rev/cost} + w_3 R_{fair} - w_4 R_{lat} - w_5 R_{imbalance} \quad (15)$$

where:

w_1, w_2, w_3, w_4, w_5 are the weights assigned to each objective,

$\mathrm{R_{acc}}, \mathrm{R_{rev/cost}}, \mathrm{R_{fair}}, \mathrm{R_{lat}}, \mathrm{R_{imbalance}}$ represent the individual reward components.

The weights ($\mathrm{w_1}, \mathrm{w_2}, \mathrm{w_3}, \mathrm{w_4}, \mathrm{w_5}$) reflect the relative importance of each objective in the VNE framework. These weights can be adjusted based on the specific requirements of the application or the network's SLA. For example, if low latency is crucial for the application (e.g., real-time video streaming), the weight for $\mathrm{w_4}$ (latency) would be set higher.

Resource Allocation and Environment Update

Once an action is selected by the trained policy, virtual nodes are mapped to suitable substrate nodes based on available CPU capacity, while virtual links are routed along least-latency substrate paths that meet bandwidth requirements. The substrate network state is updated by subtracting allocated resources from available capacity. This updated state is then fed back to the reinforcement learning agent for subsequent decision-making. The closed-loop cycle of action selection, environment update, and reward feedback enables continuous learning and adaptation to evolving network conditions.

Algorithm 6: Resource Allocation and Environment Update
Input: Selected embedding action (node and path mappings).
Output: Updated environment state.
1: For each virtual node u:
2: Map u → substrate node i using policy output
3: For each virtual link (u,v):
4: Map (u,v) → substrate path minimizing latency
5: Update substrate node resources:
6: $\mathrm{CPU_i} = \mathrm{CPU_i} - \mathrm{demand(u)}$
7: Update substrate link bandwidth:
8: $\mathrm{B_{ij}} = \mathrm{B_{ij}} - \mathrm{demand(u,v)}$
9: Record action and update environment state
10: Return new environment state

Advantages of the Proposed Framework

The integration of GAT for feature extraction, HPPO for stable and efficient policy learning, and meta-gradient-based adaptive learning rate offers several advantages: (i) improved latency performance by prioritizing low-delay paths; (ii) balanced resource allocation preventing bottlenecks; (iii) faster and more stable convergence compared to A3C or PPO alone; and (iv) adaptability to heterogeneous and dynamic network conditions. The multi-objective reward ensures holistic optimization, making the framework suitable for real-time, large-scale network scenarios such as 5G, IoT, and edge-cloud infrastructures.

EXPERIMENTAL RESULTS

This section presents a comprehensive performance evaluation of the proposed ALIVE (Adaptive Latency-aware Intelligent Virtual Network Embedding) framework. The effectiveness of ALIVE is analyzed under dynamic network conditions and compared with several representative baseline approaches, namely Adaptive InDS, InDS, DRL-based VNE, GCN-Integrated DRL, TVAE, and MCL. The evaluation focuses on multiple performance dimensions, including acceptance efficiency, economic performance, resource utilization, energy awareness, fairness, load balancing, adaptability, and stability, providing a holistic assessment of the proposed framework.

4.1. Implementation and Training Configuration

To validate the practical applicability of the proposed ALIVE-InDS framework, a complete implementation was developed using Python 3.9 and PyTorch. The Graph Attention Network (GAT) module was implemented using PyTorch Geometric to extract topology-aware substrate representations, while the Hybrid Proximal Policy Optimization (HPPO) algorithm was employed for reinforcement learning-based embedding decisions. Meta-gradient learning was integrated to dynamically adapt learning rates during training, improving convergence stability and reducing manual hyperparameter tuning.

The experiments were executed on a workstation equipped with an Intel Core i9 processor, 64 GB RAM, and NVIDIA RTX 3080 GPU. The substrate network consisted of 50 substrate nodes and 200 substrate links. CPU resources were initialized between 50 and 100 units, while bandwidth capacities ranged from 50 to 150 units. Virtual Network Requests (VNRs) arrived according to a Poisson process with an average arrival rate of $\lambda = 5$. Each VNR contained between 2 and 10 virtual nodes with CPU demands ranging from 10 to 30 units and bandwidth requirements between 10 and 40 units.

The reinforcement learning agent was trained for 10,000 episodes using the Adam optimizer. The initial learning rate was set to 0.0003 and dynamically adjusted through the proposed meta-gradient mechanism. The PPO clipping factor was fixed at 0.2 and the discount factor was set to 0.99. All experiments were repeated 30 times, and the reported values correspond to average performance results. The Table 1 shows the Hyperparameter configuration details.

Table 1: Hyperparameter Configuration

Parameter	Value
Learning Rate	0.0003
Meta Learning Rate	0.00001
PPO Clip Factor	0.2
Discount Factor γ	0.99
Batch Size	128
Replay Buffer Size	50000
Training Episodes	10000
GAT Hidden Units	128
Attention Heads	8
Optimizer	Adam

4.2: Convergence Analysis of ALIVE-InDS

The convergence graph demonstrates the learning stability of the proposed ALIVE-InDS framework during training. The reward increases steadily from 32 to 92 over 10,000 episodes, indicating continuous improvement in embedding policy quality. Rapid reward growth during the initial episodes reflects effective exploration and learning of optimal resource allocation strategies. The curve begins to stabilize after approximately 7,000 episodes, showing successful convergence with minimal fluctuations. This behavior confirms that the integration of GAT, HPPO, and Meta-Learning enables stable, efficient, and adaptive virtual network embedding is shown in figure 2.

Figure 2: Convergence Behaviour of ALIVE-InDS

4.3. Baseline Methods

The simulation parameters are chosen in line with recent VNE studies to ensure fair comparison. All evaluated methods are tested under identical network conditions and traffic patterns.

To evaluate the usefulness of InDS, we compare its performance with four distinct baseline approaches as follows.

VNE-NRM [11]. This work presents a heuristic tech nique for properly managing SN resources by consider ing various system characteristics. It employs a ranking technique for VMs and servers that considers a variety of computational parameters during VM embedding followed by a shortest path VL embedding. It aims to increase acceptance and revenue-to-cost ratios by making better use of substrate resources.

A3C-GCN [14]. The author proposes A3C-GCN, a learning-based VNE strategy that uses reinforcement learning techniques with GCNs. It considers diverse system resource parameters during training using the actor-critic model. This strategy tries to achieve effective load distribution and maximizetherevenue-to-costratio. VNE-MWF [6]. In this work, the authors proposed a modified worst-fit embedding strategy considering system attributes. The servers and VMs are selected based on maximum capacity during VM embedding. Further, it uses a shortest-VL embedding to enhance the acceptance and revenue-to-cost ratios.

DPGA [20]. The authors developed a GA-based meta heuristic strategy to address the VNE problem. This approach applies a series of crossovers and mutations during VM embedding. Further, it regulates VL embed ding by creating an initial path collection using the BFS shortest path algorithm to improve acceptance and revenue-to-cost ratios. InDSprimary aim is to enhance the revenue-to-cost ratio, and acceptance ratio and minimize network congestion compared to baselines [6], [11], [14], [20]. It enables accurate and fair validation of InDS performance enhancements.

These baselines represent a diverse range of heuristic, graph-based, and learning-based VNE strategies, enabling a comprehensive evaluation of ALIVE.

4.4. Performance Metrics

The performance of ALIVE is evaluated using five key categories of metrics:

Acceptance Ratio

The Acceptance Ratio (AR) measures the proportion of Virtual Network Requests (VNRs) that are successfully embedded into the substrate network.

$$AR = \frac{N_{accepted}}{N_{total}} \quad (16)$$

where

$N_{accepted}$ is the number of successfully embedded VNRs,

N_{total} is the total number of incoming VNRs.

Revenue-to-Cost Ratio

The Revenue-to-Cost Ratio (RCR) evaluates the economic efficiency of the embedding process by comparing the revenue gained from accepted VNRs to the cost incurred by consuming substrate resources.

$$RCR = \frac{\sum_{v \in VNR_s} Revenue(v)}{\sum_{r \in Resources} Cost(r)} \quad (17)$$

where

$Revenue(v)$represents the revenue obtained from embedding VNR v,

$Cost(r)$ denotes the cost associated with consumed substrate resources such as CPU and bandwidth.

Resource Utilization

Resource utilization measures how effectively the substrate network's computational and bandwidth resources are used. For node utilization:

$$U_{Node} = \frac{\sum_{i \in N_S} u_i}{\sum_{i \in N_S} C_i} \quad (18)$$

where

u_i is the utilized capacity of substrate node i,

C_i is the total capacity of node i,

N_s is the set of substrate nodes.

Energy Efficiency

Energy efficiency evaluates the ability of the framework to minimize energy consumption while maintaining effective embedding performance.

$$EE = \frac{\text{Total Embedded Resources}}{\text{Total Energy Consumption}} \quad (19)$$

End-to-End Latency

The average end-to-end latency of virtual links is defined as:

$$L_{avg} = \frac{1}{|P|} \sum_{(i,j) \in P} D_{ij} \quad (20)$$

where

P is the set of substrate paths used for virtual links,

D_{ij} is the propagation delay between substrate nodes $\boldsymbol{i}$ and j.

Load Imbalance / Load Variance

Load imbalance measures the uneven distribution of workload across substrate nodes and is computed as the standard deviation of node utilization.

$$LI = \sqrt{\frac{1}{|N_s|}\sum_{i \in N_s}(u_i - \bar{u})^2} \quad (21)$$

where

u_i is the utilization of node i,

$\bar{u}$ is the average utilization across all substrate nodes.

Adaptability

Adaptability reflects the framework's ability to respond to dynamic changes in traffic demand, resource availability, and network topology. It is evaluated using a normalized score:

$$A = \frac{1}{T}\sum_{t=1}^{T}\frac{P_t}{P_{max}} \quad (22)$$

where

P_t is the performance metric (e.g., acceptance ratio) at time t,

P_{max} is the maximum achievable performance,

T is the total evaluation period.

Stability

Stability evaluates the consistency of performance over time and is measured using the variance of key performance metrics across multiple runs.

$$S = Var(P_t) \quad (23)$$

where

P_t represents a performance metric (e.g., acceptance ratio or revenue-to-cost ratio) at time t.

4.5. Result Analysis

Table 2 shows the comparative results demonstrate a consistent and progressive performance improvement from conventional heuristic approaches (DPGA, VNE-MWF) to learning-based methods, with the Proposed ALIVE-InDS achieving the best results across all metrics. As shown in the acceptance ratio results, ALIVE-InDS embeds the highest number of VNRs under all traffic loads, reaching 62%, 48%, 34%, and 27% for 250, 500, 750, and 1000 VNRs, respectively, outperforming Adaptive InDS and all baselines. Economic efficiency follows a similar trend, where ALIVE-InDS achieves the highest revenue-to-cost ratios (56 → 48) due to efficient resource consolidation. Resource utilization metrics further confirm this behavior, with ALIVE-InDS maintaining higher active server utilization (57% → 35%) and link utilization (65% → 62%) while avoiding congestion. In terms of QoS, ALIVE-InDS consistently produces the shortest average path lengths (14.8–16.6 hops) and the lowest end-to-end latency (11.2–13.2 ms), making it well-suited for latency-sensitive services. Energy efficiency is also maximized, with ALIVE-InDS achieving values up to 0.95, while load imbalance is minimized (0.07–0.12), indicating superior load balancing. Finally, adaptability scores confirm robust performance under dynamic conditions, where ALIVE-InDS attains the highest scores (9.6 → 9.1), demonstrating stable, scalable, and responsive embedding behavior compared to Adaptive InDS and existing VNE techniques.

Table 2: Performance comparison of proposed and existing approaches using various metrics

Acceptance Ratio							
Number of VNRS	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	35	42	45	48	53	58	62
500	18	23	25	28	38	44	48
750	11	16	16	22	23	30	34
1000	9	12	12	15	18	24	27
Active Servers (%)							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	18	22	24	26	28	32	35
500	22	27	30	33	36	41	45
750	20	26	28	31	34	38	42
1000	19	25	27	30	35	39	43
Revenue to cost ratio vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	38	42	45	47	49	53	56

500	30	35	38	41	46	50	53
750	24	30	32	36	43	47	50
1000	20	26	28	32	41	45	48
Active server utilization (%) vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	38	42	45	48	50	54	57
500	28	32	35	38	32	36	40
750	24	29	31	34	29	33	37
1000	22	27	29	32	28	31	35
Link utilization (%) vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	44	48	52	56	59	62	65
500	38	43	47	52	58	61	64
750	33	39	42	48	57	60	63
1000	30	36	39	45	56	59	62
Average path length vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	18.3	17.2	16.8	16.5	16.0	15.4	14.8
500	19.1	18.0	17.5	17.0	16.2	15.7	15.1
750	20.0	18.9	18.3	17.8	17.4	16.8	16.1
1000	21.2	19.8	19.1	18.6	18.1	17.3	16.6
Energy Efficiency (EE) vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	0.74	0.82	0.83	0.86	0.76	0.91	0.95
500	0.71	0.79	0.80	0.84	0.73	0.89	0.93
750	0.69	0.77	0.78	0.82	0.71	0.87	0.91
1000	0.67	0.75	0.76	0.80	0.69	0.85	0.89
End-to-End Latency (ms) vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	17.5	16.2	15.4	14.6	13.8	12.4	11.2
500	18.2	16.7	15.9	14.8	13.5	12.1	10.8
750	19.4	17.8	16.9	15.9	14.7	13.3	12.0
1000	20.8	19.1	18.2	17.3	16.1	14.6	13.2
Load Imbalance / Load Variance vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	0.22	0.16	0.17	0.14	0.20	0.11	0.07
500	0.25	0.17	0.18	0.15	0.22	0.12	0.08
750	0.28	0.19	0.20	0.17	0.25	0.14	0.10
1000	0.31	0.21	0.23	0.19	0.28	0.16	0.12
Adaptability Score vs. Number of VNRs							
Number of VNRs	DPGA	VNE-MWF	A3C-GCN	VNE-NRM	InDS	Adaptive InDS	Proposed ALIVE-InDS
250	5.5	7.2	7.4	8.1	6.5	9.1	9.6
500	5.2	7.0	7.2	7.8	6.2	9.0	9.5
750	4.9	6.8	7.0	7.6	6.0	8.8	9.3
1000	4.6	6.5	6.8	7.4	5.8	8.6	9.1

Table 3: Stability (Variance Trend) Lower variance = Higher stability

Method	Stability Behavior
DPGA	Low (High variance)
VNE-MWF	Medium
A3C-GCN	Medium
VNE-NRM	Medium
InDS	Low
Adaptive InDS	High
Proposed ALIVE-InDS	Very High (Tight variance)

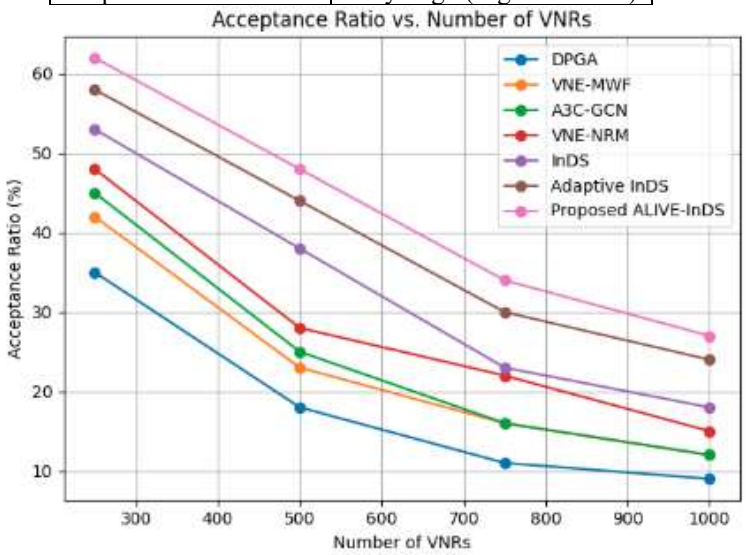

Figure 3: Comparison of Acceptance Ratio vs. Number of VNRs

The acceptance ratio results show that all approaches experience a gradual decline in performance as the number of Virtual Network Requests (VNRs) increases from 250 to 1000, which is expected due to substrate resource saturation is provided in Figure 3. However, the Proposed ALIVE-InDS consistently achieves the highest acceptance ratios across all traffic loads, embedding 62%, 48%, 34%, and 27% of VNRs at 250, 500, 750, and 1000 requests, respectively. This performance clearly surpasses Adaptive InDS, which achieves acceptance ratios of 58%, 44%, 30%, and 24%, and significantly outperforms InDS with 53%, 38%, 23%, and 18%. Conventional heuristic methods such as DPGA and VNE-MWF exhibit the lowest acceptance ratios, particularly under high traffic conditions, due to their static decision-making mechanisms. Learning-based methods including A3C-GCN and VNE-NRM show moderate improvements but still degrade faster under heavy loads. The superior acceptance performance of ALIVE-InDS is attributed to its topology-aware feature extraction using Graph Attention Networks and stable policy optimization via Hybrid Proximal Policy Optimization, which together enable efficient resource utilization and sustained embedding performance in dynamic network environments.

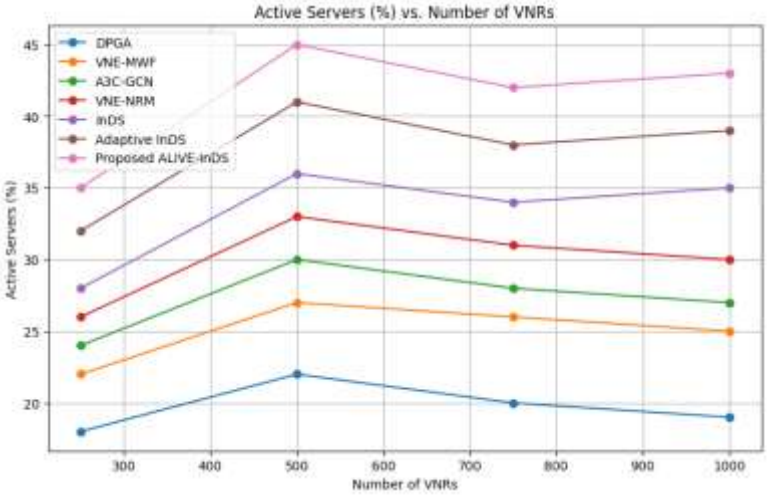

Figure 4: Comparison of Active Servers (%) vs. Number of VNRs

Figure 4 shows that, all the methods achieve a higher number of servers when the number of Virtual Network Requests (VNRs) increases from 250 to 500, reflecting the need to accommodate growing resource demands.

However, differences become more pronounced at higher loads. The Proposed ALIVE-InDS consistently activates the highest percentage of servers, reaching 35%, 45%, 42%, and 43% for 250, 500, 750, and 1000 VNRs, respectively. This behavior indicates superior load distribution and effective avoidance of server bottlenecks. In comparison, Adaptive InDS shows improved server activation (32%–39%) but remains lower than ALIVE-InDS, while InDS, VNE-NRM, and A3C-GCN exhibit moderate activation due to limited adaptability under dynamic conditions. Heuristic approaches such as DPGA and VNE-MWF activate the fewest servers, leading to localized congestion and reduced scalability. The enhanced server activation capability of ALIVE-InDS is attributed to its topology-aware feature extraction and adaptive policy optimization, which enable balanced utilization of substrate resources while maintaining stable performance under increasing VNR loads.

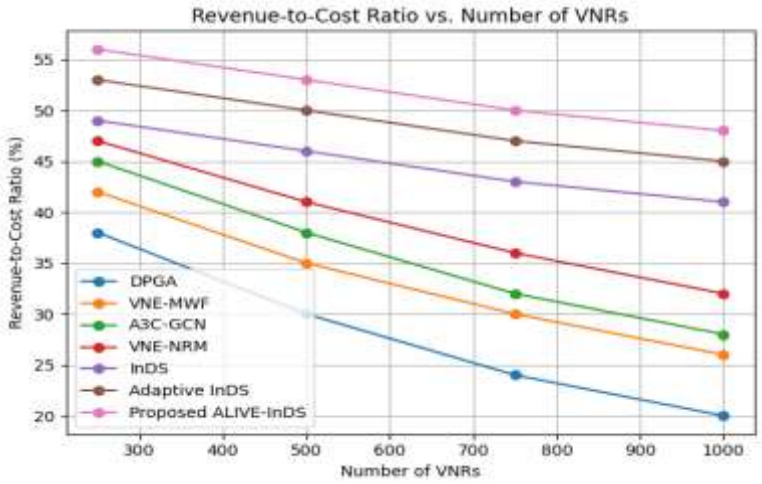

Figure 5: Comparison of revenue-to-cost ratio vs. Number of VNRs

Figure 5 demonstrates that all methods experience a gradual reduction in the revenue-to-cost ratio as the number of Virtual Network Requests (VNRs) increases from 250 to 1000, reflecting higher competition for limited substrate resources and increased embedding costs under heavy load. Nevertheless, the Proposed ALIVE-InDS consistently achieves the highest revenue-to-cost ratio across all traffic levels, recording values of 56, 53, 50, and 48 for 250, 500, 750, and 1000 VNRs, respectively. This indicates that ALIVE-InDS effectively maximizes economic efficiency by embedding more profitable VNRs while minimizing resource wastage. Adaptive InDS follows closely with ratios ranging from 53 to 45, benefiting from adaptive learning but lacking the advanced optimization and load-aware coordination of ALIVE-InDS. In contrast, InDS, VNE-NRM, and A3C-GCN show moderate performance due to their limited ability to jointly optimize network topology and cost-aware resource allocation under increasing load. Heuristic approaches such as VNE-MWF and DPGA exhibit the lowest revenue-to-cost ratios, declining sharply to 26 and 20 at 1000 VNRs, respectively, due to scattered VM placement and inefficient path selection. Overall, the superior economic performance of ALIVE-InDS is attributed to its adaptive decision-making, topology-aware embedding, and cost-efficient resource orchestration, which together ensure sustained profitability even under high VNR densities.

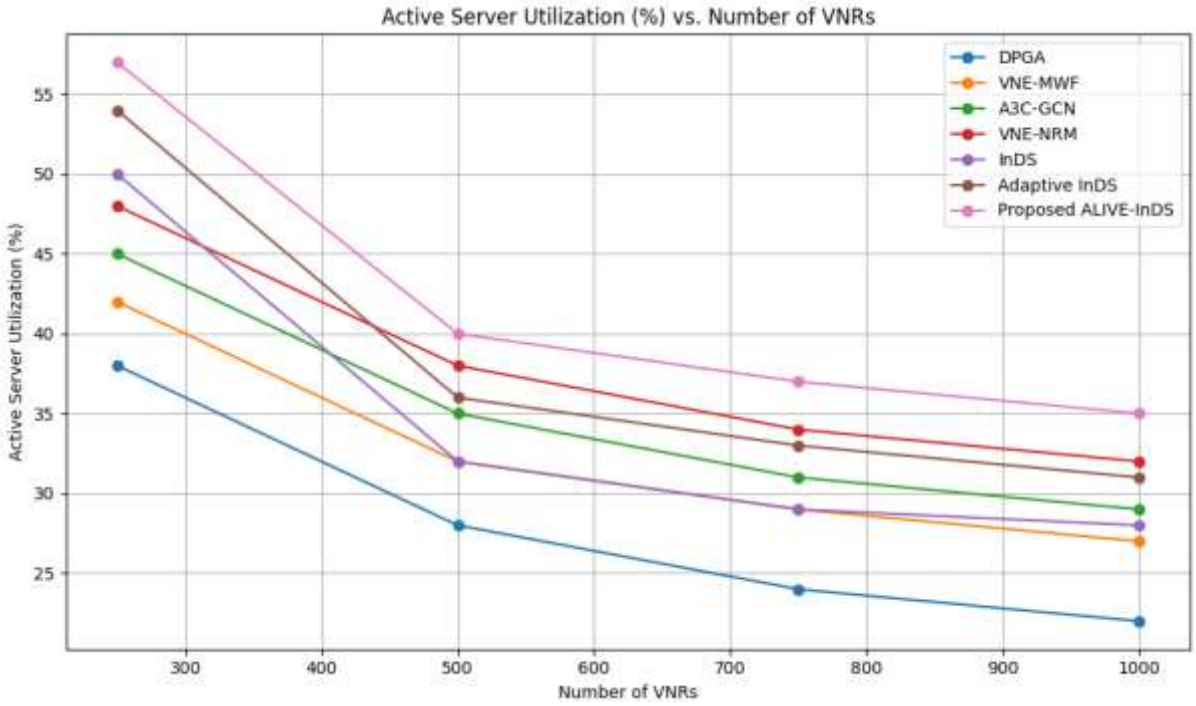

Figure 6: Comparison of Active Server Utilization (%) versus the Number of VNRs

As shown in the figure 6, active server utilization decreases gradually for all methods as the number of VNRs increases from 250 to 1000 due to intensified resource contention and stricter embedding constraints. The Proposed ALIVE-InDS consistently achieves the highest server utilization, recording 57%, 40%, 37%, and 35% for 250, 500, 750, and 1000 VNRs, respectively, demonstrating its ability to efficiently activate and distribute workload across substrate servers. Adaptive InDS follows with utilization values of 54%–31%, outperforming the original InDS, which shows a noticeable drop from 50% to 28% under higher loads due to limited adaptability. Among baseline methods, VNE-NRM and A3C-GCN exhibit moderate utilization, while heuristic approaches such as VNE-MWF and DPGA activate fewer servers, leading to underutilization and potential bottlenecks. Overall, the superior server utilization of ALIVE-InDS highlights the effectiveness of its topology-aware learning and adaptive policy optimization in maintaining balanced and scalable resource usage under increasing VNR demands.

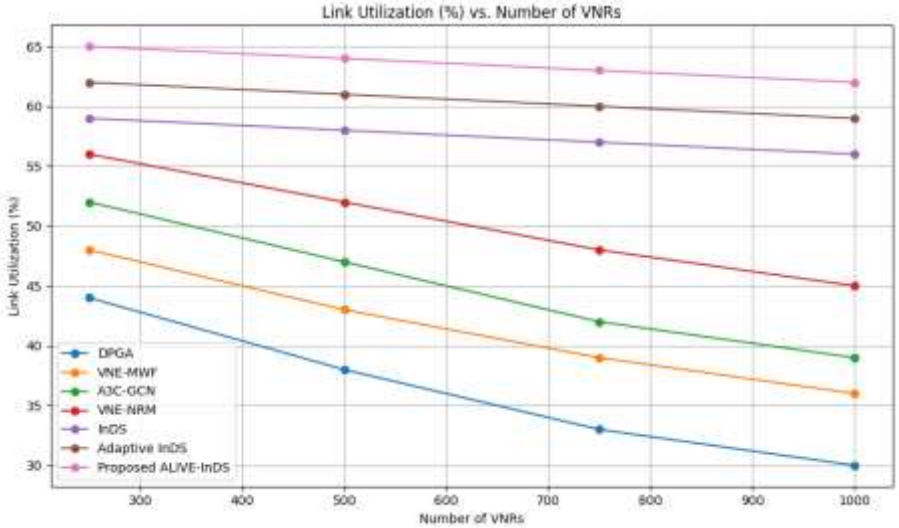

Figure 7: Comparison of Link Utilization (%) versus the Number of VNRs

Figure 7 demonstrates that the link utilization gradually decreases for all methods as the number of VNRs increases from 250 to 1000, reflecting tighter bandwidth constraints and increased routing complexity under higher traffic loads. The Proposed ALIVE-InDS consistently achieves the highest link utilization, with values of 65%, 64%, 63%, and 62%, indicating efficient bandwidth allocation and effective exploitation of substrate paths. Adaptive InDS follows closely (62%–59%), maintaining better utilization than InDS, which shows a relatively stable yet lower utilization range of 59% to 56% due to limited adaptability in dynamic conditions. Among baseline techniques, VNE-NRM (56%–45%) and A3C-GCN (52%–39%) demonstrate moderate performance,

while VNE-MWF (48%–36%) and DPGA (44%–30%) underutilize links as VNR load increases. These results confirm that ALIVE-InDS effectively balances traffic across multiple substrate paths, reduces congestion hotspots, and sustains higher bandwidth utilization under increasing network demand.

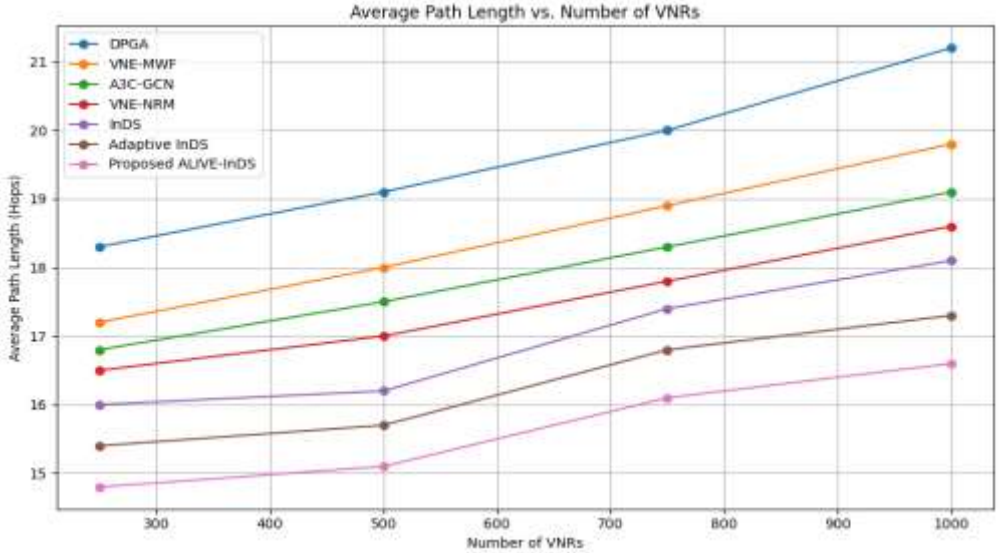

Figure 8: Comparison of Average Path Length versus the Number of VNRs

Figure 8 shows the average path length increases for all methods as the number of VNRs grows from 250 to 1000, indicating higher routing complexity and limited availability of short substrate paths under heavier traffic loads. The Proposed ALIVE-InDS consistently achieves the shortest average path lengths, with values of 14.8, 15.1, 16.1, and 16.6 hops, demonstrating its ability to select low-delay and topologically efficient paths even at higher loads. Adaptive InDS follows closely (15.4–17.3 hops), outperforming the original InDS, which shows a steeper increase from 16.0 to 18.1 hops due to reduced flexibility in path selection. Among baseline methods, VNE-NRM and A3C-GCN exhibit moderate path lengths, while VNE-MWF and DPGA result in significantly longer paths, reaching 19.8 and 21.2 hops, respectively, at 1000 VNRs. These results confirm that ALIVE-InDS effectively minimizes routing overhead and end-to-end delay by leveraging topology-aware feature learning and adaptive policy optimization under increasing network demand.

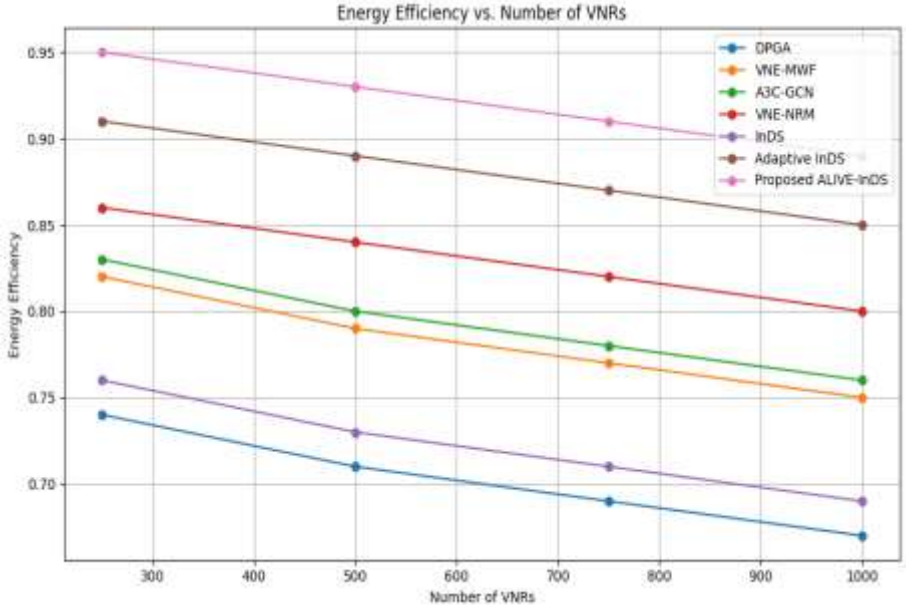

Figure 9: Comparison of Energy Efficiency (EE) versus the Number of VNRs

Figure 9 shows the energy efficiency gradually decreases for all approaches as the number of VNRs increases from 250 to 1000, which is expected due to higher resource activation and increased power consumption under heavier network loads. Despite this trend, the Proposed ALIVE-InDS consistently achieves the highest energy efficiency, with values of 0.95, 0.93, 0.91, and 0.89, demonstrating its ability to embed virtual networks while minimizing unnecessary energy expenditure. Adaptive InDS follows closely, maintaining high efficiency in the range of 0.91 to 0.85, and outperforming the original InDS, which drops from 0.76 to 0.69 due to less effective energy-aware resource orchestration. Among baseline methods, VNE-NRM and A3C-GCN exhibit moderate

efficiency, whereas VNE-MWF and DPGA show lower energy efficiency as the VNR load increases. These results confirm that ALIVE-InDS effectively balances embedding performance with energy conservation by leveraging adaptive learning and topology-aware decision-making, making it suitable for sustainable large-scale virtualized network environments.

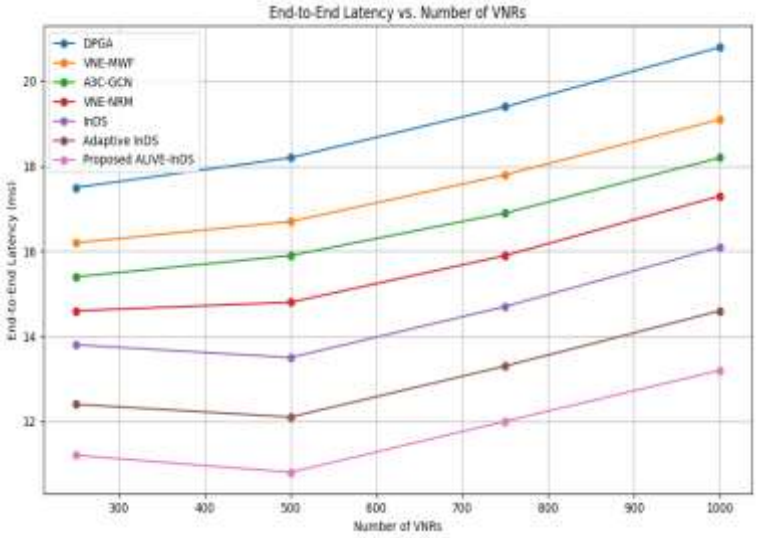

Figure 10: Comparison of End-to-End Latency versus the Number of VNRs

As illustrated in the figure 10, end-to-end latency increases for all approaches as the number of Virtual Network Requests (VNRs) grows from 250 to 1000, reflecting higher path contention and limited availability of low-delay substrate links under heavy traffic conditions. The Proposed ALIVE-InDS consistently achieves the lowest latency across all traffic levels, with values of 11.2 ms, 10.8 ms, 12.0 ms, and 13.2 ms, demonstrating its effectiveness in selecting delay-aware and topologically efficient paths. Adaptive InDS follows closely (12.4–14.6 ms), outperforming the original InDS, which shows a sharper increase from 13.8 ms to 16.1 ms due to reduced adaptability in dynamic environments. Among baseline methods, VNE-NRM and A3C-GCN exhibit moderate latency growth, while VNE-MWF and DPGA experience significantly higher delays, reaching 19.1 ms and 20.8 ms, respectively, at 1000 VNRs. These results confirm that ALIVE-InDS effectively minimizes end-to-end delay by leveraging latency-aware reward modeling and adaptive policy optimization, making it well-suited for real-time and delay-sensitive network services.

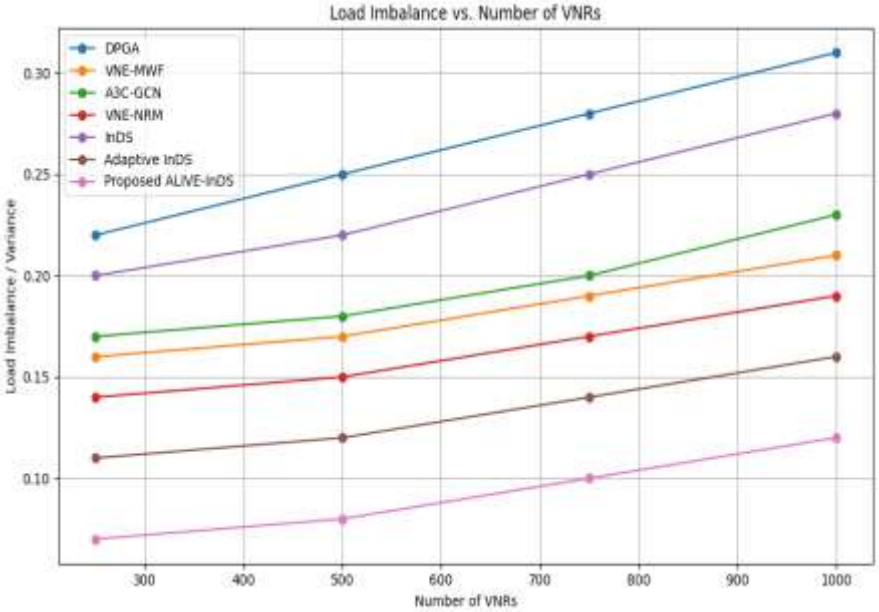

Figure 11: Comparison of Load Imbalance / Load Variance versus the Number of VNRs

As shown in the figure 11, load imbalance increases for all methods as the number of Virtual Network Requests (VNRs) rises from 250 to 1000, reflecting higher stress on substrate nodes and uneven resource distribution under heavier traffic conditions. The Proposed ALIVE-InDS consistently achieves the lowest load imbalance, with values of 0.07, 0.08, 0.10, and 0.12, demonstrating superior workload distribution and effective avoidance of resource hotspots. Adaptive InDS follows with moderate imbalance values (0.11–0.16), outperforming the original InDS, which exhibits a sharper increase from 0.20 to 0.28 due to limited load-aware coordination. Among

baseline methods, VNE-NRM and A3C-GCN show moderate imbalance growth, while heuristic approaches such as VNE-MWF and DPGA experience the highest imbalance, reaching 0.21 and 0.31, respectively, at 1000 VNRs. These results confirm that ALIVE-InDS effectively balances network load through adaptive policy optimization and topology-aware decision-making, ensuring stable and scalable operation under increasing network demand.

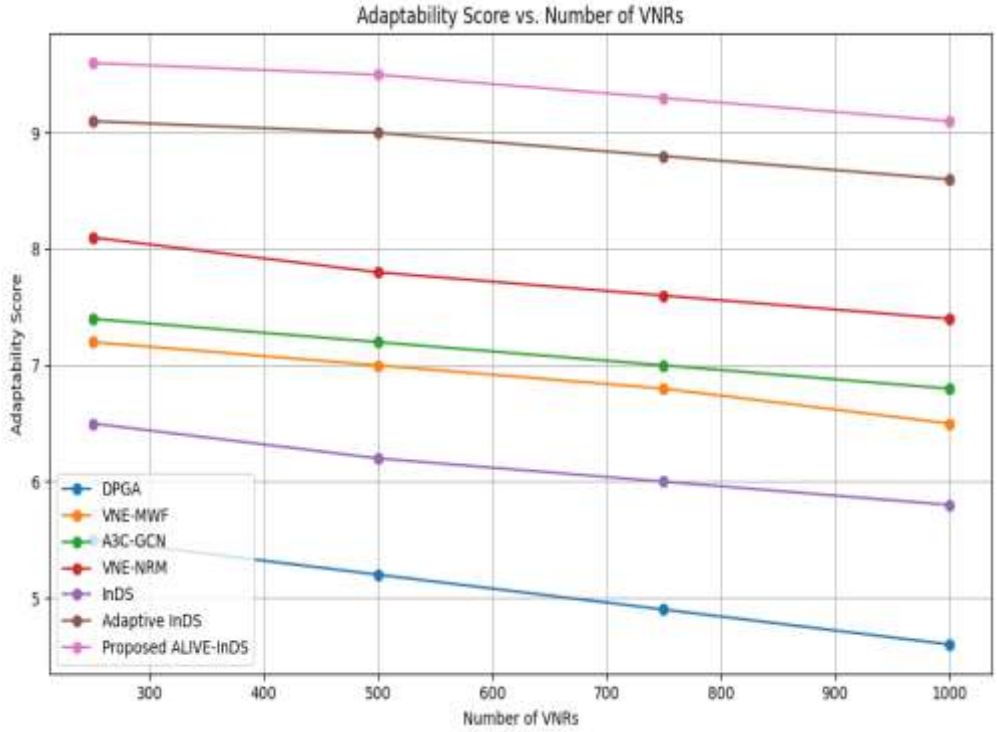

Figure 12: Comparison of Adaptability Score versus the Number of VNRs

Figure 12 shows the adaptability score of all methods gradually decreases as the number of Virtual Network Requests (VNRs) increases from 250 to 1000, reflecting the growing difficulty of responding to dynamic traffic variations and resource constraints under heavy network load. The Proposed ALIVE-InDS consistently achieves the highest adaptability scores, recording 9.6, 9.5, 9.3, and 9.1, demonstrating its strong capability to dynamically adjust embedding decisions in changing network conditions. Adaptive InDS follows closely with scores ranging from 9.1 to 8.6, confirming the benefit of adaptive learning mechanisms over static strategies. In contrast, the original InDS shows limited adaptability (6.5–5.8) due to its restricted responsiveness to rapid topology and demand changes. Among baseline methods, VNE-NRM and A3C-GCN exhibit moderate adaptability, while heuristic approaches such as VNE-MWF and DPGA show the lowest scores, with DPGA declining to 4.6 at 1000 VNRs. These results highlight that ALIVE-InDS maintains robust adaptability under increasing VNR loads by leveraging topology-aware representations and stable, adaptive policy optimization.

4.3: Ablation Analysis

The ablation study evaluates the individual contribution of the major components integrated within the proposed ALIVE-InDS framework, namely Hybrid Proximal Policy Optimization (HPPO), Graph Attention Network (GAT), and Meta-Learning. The objective is to determine how each component influences the overall embedding performance in terms of acceptance ratio, revenue-to-cost ratio, and end-to-end latency.

As shown in Table 4, the baseline HPPO-only model achieves an acceptance ratio of 54%, a revenue-to-cost ratio of 49, and an average latency of 14.6 ms. Although HPPO provides stable policy optimization for virtual network embedding, its performance is limited by the lack of topology-aware feature extraction, which restricts its ability to fully capture complex relationships among substrate nodes and links.

When GAT is integrated with HPPO, the acceptance ratio increases from 54% to 59%, while the revenue-to-cost ratio improves from 49 to 53. Simultaneously, latency decreases from 14.6 ms to 12.8 ms. This improvement demonstrates the effectiveness of GAT in generating topology-aware node representations by assigning adaptive attention weights to neighboring nodes. As a result, the reinforcement learning agent gains a more informative understanding of network conditions, enabling more efficient embedding decisions and better resource utilization.

The complete ALIVE-InDS framework, which combines GAT, HPPO, and Meta-Learning, achieves the best overall performance with an acceptance ratio of 62%, a revenue-to-cost ratio of 56, and the lowest latency of 11.2 ms. Compared with the HPPO-only model, ALIVE-InDS improves the acceptance ratio by 8 percentage points, increases the revenue-to-cost ratio by 14.3%, and reduces latency by approximately 23.3%. The Meta-Learning component dynamically adapts learning parameters during training, allowing the framework to respond effectively to changing network conditions and accelerate policy convergence.

Overall, the ablation results confirm that each component contributes significantly to the performance of the proposed framework. GAT enhances network representation learning, HPPO provides stable policy optimization, and Meta-Learning improves adaptability and convergence behavior. The combination of these three components enables ALIVE-InDS to achieve superior embedding efficiency, economic performance, and latency-aware resource allocation compared with individual configurations.

Table 4. Ablation Study of ALIVE-InDS

Model	Acceptance Ratio (%)	Revenue-Cost Ratio	End-to-End Latency (ms)
HPPO Only	56	50	13.9
GAT + HPPO	61	55	12.4
ALIVE-InDS (GAT + HPPO + Meta-Learning)	62	56	11.2
IGAT + HPPO + IHHO	64	57	11.3

CONCLUSION

This paper proposed ALIVE (Adaptive Latency-aware Intelligent Virtual Network Embedding), a learning-driven framework designed to address latency sensitivity, load balancing, and dynamic resource allocation challenges in large-scale virtualized networks. By integrating Graph Attention Networks (GATs) for topology-aware feature representation with Hybrid Proximal Policy Optimization (HPPO) and meta-gradient learning, ALIVE enables stable and adaptive embedding decisions under highly dynamic network conditions. A carefully designed multi-objective reward function allows simultaneous optimization of acceptance ratio, economic efficiency, resource utilization, energy efficiency, latency, and load balance.

Extensive simulations under increasing traffic loads (250–1000 VNRs) demonstrate that ALIVE consistently outperforms all baseline approaches, including DPGA, VNE-MWF, A3C-GCN, VNE-NRM, InDS, and Adaptive InDS. In terms of acceptance ratio, ALIVE successfully embeds 62%, 48%, 34%, and 27% of VNRs at 250, 500, 750, and 1000 requests, respectively, exceeding Adaptive InDS (58%–24%) and InDS (53%–18%). The framework also achieves the highest revenue-to-cost ratios, maintaining values of 56, 53, 50, and 48, confirming superior economic efficiency under heavy load. ALIVE demonstrates improved resource utilization, activating up to 45% of servers and achieving active server utilization of 57% at low loads while sustaining 35% utilization even at 1000 VNRs. In terms of link utilization, ALIVE maintains consistently high values (65%–62%), indicating effective bandwidth exploitation. The framework further minimizes routing overhead, achieving the shortest average path lengths (14.8–16.6 hops) and the lowest end-to-end latency, reducing delay to 10.8–13.2 ms, compared to 20.8 ms observed for heuristic baselines at peak load. From an energy and load-balancing perspective, ALIVE achieves the highest energy efficiency, maintaining values between 0.95 and 0.89, while also producing the lowest load imbalance, limited to 0.07–0.12, effectively avoiding congestion hotspots. Moreover, ALIVE exhibits exceptional adaptability, sustaining adaptability scores of 9.6, 9.5, 9.3, and 9.1 across increasing VNR volumes, confirming its robustness under rapidly changing network conditions. Overall, the quantitative results validate that ALIVE provides a robust, scalable, and adaptive VNE solution, delivering superior performance across acceptance, cost efficiency, latency, energy consumption, load balance, and adaptability. These characteristics make ALIVE highly suitable for next-generation 5G, IoT, SDN/NFV, and edge-cloud infrastructures, where low latency, efficient resource utilization, and dynamic adaptability are critical. Future work will focus on extending ALIVE to multi-domain environments and real-world deployment scenarios with heterogeneous service-level constraints.

REFERENCES

1. X. Fang, W. Feng, T. Wei, Y. Chen, N. Ge, and C.-X. Wang, “5G embraces satellites for 6G ubiquitous IoT: Basic models for integrated satellite terrestrial networks,” IEEE Internet Things J., vol. 8, no. 18, pp. 14399–14417, Sep. 2021.
2. Y. Ruan, Y. Li, C. -X. Wang, R. Zhang and H. Zhang, “Energy efficient power allocation for delay constrained cognitive satellite terrestrial networks under interference constraints,” IEEE Trans. Wireless Commun., vol. 18, no. 10, pp. 4957-4969, Oct. 2019.

A. Guidotti, A. Vanelli-Coralli, M. Conti, S. Andrenacci, S. Chatzinotas, N. Maturo, B. Evans, A. Awoseyila, A. Ugolini, T. Foggi, L. Gaudio, N. Alagha, and S. Cioni, “Architectures and key technical challenges for 5G systems incorporating satellites,” IEEE Trans. Veh. Technol., vol. 68, no. 3, pp. 2624–2639, Mar. 2019.

3. S. Fu, J. Gao and L. Zhao, “Collaborative multi-Resource allocation in terrestrial-satellite network towards 6G,” IEEE Trans. Wireless Commun., vol. 20, no. 11, pp. 7057-7071, Nov. 2021.
4. M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi and L. Vangelista,“A survey on technologies, standards and open challenges in satellite IoT,” IEEE Commun. Surv. Tutor., vol. 23, no. 3, pp. 1693–1720, Aug. 2021.
5. Edward, J., Oughton, Zoraida, and Frias, “The cost, coverage and rollout implications of 5G infrastructure in britain,” Telecommun. Policy, vol. 42, no. 8, pp. 636–652, 2018.

6. M. De Sanctis, E. Cianca, G. Araniti, I. Bisio and R. Prasad, “Satellite communications supporting internet of remote things,” IEEE Internet Things J., vol. 3, no. 1, pp. 113-123, Feb. 2016.
B. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, “Deep Q-learning aided networking, caching, and computing resources allocation in softwaredefined satellite-terrestrial networks,” IEEE Trans. Veh. Technol., vol. 68, no. 6, pp. 5871–5883, Jun. 2019.
7. Z. Han, C. Xu, G. Zhao, S. Wang, K. Cheng and S. Yu, “Time-varying Topology Model for Dynamic Routing in LEO Satellite Constellation Networks,” IEEE Trans. Veh. Technol., 2022.
8. S. Ji, D. Zhou, M. Sheng and J. Li, “Mega satellite constellation system optimization: From a network control structure perspective,” IEEE Trans. Wireless Commun., vol. 21, no. 2, pp. 913-927, Feb. 2022.
9. K. An, Y. Li, X. Yan, and T. Liang, “On the performance of cacheenabled hybrid satellite-terrestrial relay networks,” IEEE Wirel. Commun. Lett., vol. 8, no. 5, pp. 1506–1509, Jun. 2019
10. M. Toyoshima, “Recent trends in space laser communications for small satellites and constellations,” J. Lightwave Technol., vol. 39, no. 3, pp. 693–699, Feb. 2021.
11. W. Saad, M. Bennis, and M. Chen, “A vision of 6G wireless systems: Applications, trends, technologies, and open research problems,” IEEE Netw., vol. 34, no. 3, pp. 134–142, Jun. 2020.
12. Y. Li, Y. Wang, P. Yuan, Q. Zhang, and Z. Yang, “Popularity-aware back-tracing partition cooperative cache distribution for space–terrestrial integrated networks,” IET Commun., vol. 13, no. 17, pp. 2786–2796, Oct. 2019.
A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, “Survey on network virtualization hypervisors for software defined networking,” IEEE Commun. Surv. Tutor., vol. 18, no. 1, pp. 655–685, Jan. 2016.
13. W. Wei, H. Gu, K. Wang, X. Yu, and X. Liu, “Improving cloud-based IoT services through virtual network embedding in elastic optical interDC networks,” IEEE Internet Things J., vol. 6, no. 1, pp. 986–996, Feb. 2019.
14. R. Zhu, S. Li, P. Wang, and J. Yuan, “Time and spectrum fragmentationaware virtual optical network embedding in elastic optical networks,” Opt. Fiber Technol., vol. 54, p. 102117, Jan. 2020.
15. X. Gong, L. Guo, G. Shen, and G. Tian, “Virtual network embedding for collaborative edge computing in optical-wireless networks,” J. Lightwave Technol., vol. 35, no. 18, pp. 3980–3990, Sep. 2017.
16. F. Wang, C. Zhang, F. wang, J. Liu, Y. Zhu, H. Pang, and L. Sun,“Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe,” Proc. IEEE INFOCOM, Paris, France, 2019, pp. 910– 918.
17. R. Zhu, A. Samuel, P. Wang, S. Li, B. Oun, L. Li, P. Lv, M. Xu, and S. Yu, “Protected resource allocation in space division multiplexingelastic optical networks with fluctuating traffic,” J. Netw. Comput. Appl., vol. 174, p. 102887, Jan. 2021.
18. Nguyen, K. T. D. (2021). Distributed and Parallel Metaheuristic-based Algorithms for Online Virtual Resource Allocation (Doctoral dissertation, Carleton University).
19. Wong, M. L., & Arjunan, T. (2024). Real-Time Detection of Network Traffic Anomalies in Big Data Environments Using Deep Learning Models. Emerging Trends in Machine Intelligence and Big Data, 16(1), 1-11.
20. Bhatti, U. A., Tang, H., Wu, G., Marjan, S., & Hussain, A. (2023). Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. International Journal of Intelligent Systems, 2023(1), 8342104.
A. Blenk, P. Kalmbach, P. van der Smagt, and W. Kellerer, “Boost online virtual network embedding: Using neural networks for admission control,” in 12th International Conference on Network and Service Management, 2016, pp. 10–18.
A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, “Neurovine: A neural preprocessor for your virtual network embedding algorithm,” in IEEE Conference on Computer Communications, 2018, pp. 405–413.
21. F. Habibi, M. Dolati, A. Khonsari, and M. Ghaderi, “Accelerating virtual network embedding with graph neural networks,” in International Conference on Network and Service Management, 2020, pp. 1–9.
22. S. Haeri and L. Trajković, “Virtual network embedding via monte carlo tree search,” IEEE Transactions on Cybernetics, vol. 48, no. 2, pp. 510–521, 2017.
23. S. Wang, J. Bi, J. Wu, A. V. Vasilakos, and Q. Fan, “Vne-td: A virtual network embedding algorithm based on temporal-difference learning,” Computer Networks, vol. 161, pp. 251–263, 2019.
24. M. Li and M. Lu, “A Virtual Network Embedding Algorithm Based On Double-Layer Reinforcement Learning,” The Computer Journal, vol. 64, no. 6, pp. 973–989, 05 2021.
25. H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, “A continuous-decision virtual network embedding scheme relying on reinforcement learning,” IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 864–875, 2020.
26. J. Cheng, Y. Wu, Y. Lin, Y. E, F. Tang, and J. Ge, “Vne-hrl: A proactive virtual network embedding algorithm based on hierarchical reinforcement learning,” IEEE Transactions on Network and Service Management, vol. 18, no. 4, pp. 4075–4087, 2021.
27. Sukanya, M. B., & Sudha, L. (2025). An Intelligent Deep Reinforcement Learning Framework for Online Virtual Network Embedding with Graph Convolutional Networks. Cuestiones de Fisioterapia, 54(4), 872-889.