An Intelligent Deep Reinforcement Learning Framework for Online Virtual Network Embedding with Graph Convolutional Networks

Mrs. B.S.Sukanya
Research Scholar
Department of Computer Science,
VET Institute of Arts and Science College,
Erode -638012,
Tamil Nadu, India.

Dr. L. Sudha
Associate Professor
Department of Computer Science,
VET Institute of Arts and Science College,
Erode -638012,
Tamil Nadu, India.

Abstract

The evolution of Virtual Network Embedding (VNE) has become a cornerstone in the optimization of resource allocation within dynamic and complex network environments. The existing research introduces InDS, a hybrid framework combining Deep Reinforcement Learning (DRL) with Graph Convolutional Networks (GCNs) to address the intricate challenges associated with VNE. InDS leverages an actor-critic model and integrates GCNs to extract rich network features, facilitating optimal decision-making for Virtual Network Requests (VNRs). The framework's reward function balances multiple objectives, including acceptance ratio, revenue-to-cost ratio, and congestion avoidance. Despite its promising results, existing work reveals limitations such as insufficient consideration of diverse objectives (e.g., energy efficiency, fairness, and reliability), dependency on specific network features, and scalability concerns. To address these issues, we propose enhancements to InDS, incorporating additional objectives, advanced feature extraction methods using Graph Isomorphism Networks (GINs), temporal information, and an Adaptive A3C mechanism for improved scalability and real-time performance. The proposed framework is designed to enhance resource utilization, acceptance ratios, revenue generation, adaptability, and stability in VNE scenarios. This research demonstrates the potential for InDS to outperform traditional methods by addressing its existing limitations and incorporating robust solutions for future network environments.

Keywords: Virtual Network Embedding (VNE), Software-Defined Networking (SDN), Network Function Virtualization (NFV), Resource Allocation, Quality of Service (QoS), Network Optimization

1. Introduction

The modern digital era is characterized by an insatiable demand for network services, driven by the proliferation of cloud computing, data centers, and mobile devices [1]. These services, ranging from video streaming and online gaming to critical infrastructure applications, require dynamic and flexible network provisioning to meet diverse and evolving needs [2]. Traditional network architectures, however, often struggle to adapt to these rapid changes, characterized by rigid infrastructure and inflexible resource allocation mechanisms [3].

Enter network virtualization, a transformative paradigm that decouples network functions from the underlying hardware [4]. This decoupling enables greater flexibility and agility in network management, allowing for rapid service provisioning, efficient resource utilization, and improved scalability [5]. Key technologies driving this revolution include Software-Defined Networking (SDN) and Network Function Virtualization (NFV) [6]. SDN separates the control plane from the data plane, enabling centralized and programmable network control [7]. NFV virtualizes network functions, such as routers, firewalls, and load balancers, allowing them to be deployed as software on any commodity hardware [8].

The rapid growth of cloud computing, data centers, and mobile networks has led to an explosive demand for network services [9]. To efficiently manage and provision these services, virtualization

technologies have emerged as a cornerstone. Software-Defined Networking (SDN) and Network Function Virtualization (NFV) enable flexible and dynamic network management by abstracting the control plane from the data plane and virtualizing network functions, respectively [10].

A crucial aspect of these paradigms is Virtual Network Embedding (VNE), which involves mapping virtual network requests (VNRs) onto a physical substrate network [11]. Each VNR comprises a set of virtual nodes and virtual links with specific resource requirements (e.g., CPU, memory, bandwidth). The VNE problem aims to find an optimal mapping that satisfies all VNR requirements while maximizing resource utilization, minimizing resource fragmentation, and ensuring Quality of Service (QoS) guarantees [12].

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 [13]. 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 [14].

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 [15]. GCNs can effectively capture the relationships between nodes and edges in the network, providing valuable insights for the VNE process.

This research introduces InDS, a hybrid framework that combines the strengths of DRL and GCNs for efficient VNE. InDS leverages an actor-critic architecture to learn optimal resource allocation policies and employs GCNs to extract informative features from the network graph. The initial InDS framework demonstrated promising results in terms of resource utilization and acceptance ratios. However, several limitations were identified, including:

- Limited Consideration of Diverse Objectives: The initial InDS framework primarily focused on maximizing resource utilization and minimizing resource fragmentation. However, other crucial objectives, such as energy efficiency, fairness among VNRs, and network reliability, were not adequately considered.
- Dependence on Specific Network Features: The original InDS framework relied on a limited set of network features, potentially hindering its adaptability to diverse network scenarios.
- Scalability Concerns: The training process of the DRL model can be computationally expensive, particularly for large-scale networks. This can limit its applicability in real-time environments.

To address these limitations, this paper proposes several enhancements to the InDS framework. These enhancements aim to improve the framework's performance, adaptability, and scalability, making it more suitable for real-world deployment.

2. Literature Review

Research in VNE has evolved from heuristic algorithms to machine learning-based approaches. Early works primarily focused on static resource allocation models, such as Integer Linear Programming (ILP) and heuristic-based solutions. However, these approaches often failed to adapt to dynamic network conditions, leading to suboptimal performance in large-scale environments.

Recent advancements have leveraged Deep Reinforcement Learning (DRL) to address the dynamic nature of VNE. For instance, Wang et al. (2021) [16] proposed a DRL-based framework for VNE, demonstrating improved acceptance ratios and resource utilization. Similarly, the work by Zhang et al. (2022) [17] highlighted the integration of GCNs into DRL models, enabling better feature extraction and decision-making.

Zhang et al [18] proposed a new type of VNE algorithm, which applied reinforcement learning (RL) and graph neural network (GNN) theory to the algorithm, especially the combination of graph convolutional neural network (GCNN) and RL algorithm. Simulation experiments verified that the

dynamic VNE algorithm based on RL and GCNN has good basic VNE characteristics. By changing the resource attributes of physical network and virtual network, it can be proved that the algorithm has good flexibility. Qu et al [19] presented a generative adversarial graph network model, called ImGAGN to address the imbalanced classification problem on graphs. It introduces a novel generator for graph structure data, named GraphGenerator, which can simulate both the minority class nodes' attribute distribution and network topological structure distribution by generating a set of synthetic minority nodes such that the number of nodes in different classes can be balanced.

Kumar et al [20] proposed a novel community detection method using network embedding technique. This method serves as a universal framework towards applying and bench-marking various embedding techniques in graphs for performing community detection. Authors performed the test using various evaluation criteria on several real-life and synthetic networks and the obtained result reveals the utility of the proposed algorithm. Jiao et al [21] proposed a novel TNE method named temporal network embedding method based on the VAE framework (TVAE), which is based on a variational autoencoder (VAE) to capture the evolution of temporal networks for link prediction. It not only generates low-dimensional embedding vectors for nodes but also preserves the dynamic nonlinear features of temporal networks.

Dhelim et al [22] proposed a TMS for large-scale IoT systems called Trust2Vec, which can manage trust relationships in large-scale IoT systems and can mitigate large-scale trust attacks that are performed by hundreds of malicious devices. Trust2Vec leverages a random-walk network exploration algorithm that navigates the trust relationship among devices and computes trust network embeddings, which enables it to analyze the latent network structure of trust relationships, even if there is no direct trust rating between two malicious devices. Yang et al [23] presented a Mutual Contrastive Learning (MCL) framework for online KD. The core idea of MCL is to perform mutual interaction and transfer of contrastive distributions among a cohort of networks in an online manner. Our MCL can aggregate cross-network embedding information and maximize the lower bound to the mutual information between two networks.

Zhang et al [24] proposed a SAGIN cross-domain VNE algorithm. We model the different network segments of SAGIN, and set the network attributes according to the actual situation of SAGIN and user needs. In DRL, the agent is acted by a five-layer policy network. Authors built a feature matrix based on network attributes extracted from SAGIN and use it as the agent training environment. Through training, the probability of each underlying node being embedded can be derived. Thafar et al [25] presented a method, DTi2Vec, which identifies DTIs using network representation learning and ensemble learning techniques. DTi2Vec constructs the heterogeneous network, and then it automatically generates features for each drug and target using the nodes embedding technique. DTi2Vec demonstrated its ability in drug–target link prediction compared to several state-of-the-art network-based methods, using four benchmark datasets and large-scale data compiled from DrugBank.

Here is a comparative analysis of the mentioned methods in tabular format:

Table 1. Comparison analysis of existing work

Method	Technique	Strengths	Limitations
Wang et al. (2021)	DRL-based VNE	Improved acceptance ratios and resource utilization	Lacks consideration of energy efficiency and fairness among VNRs
Zhang et al. (2022)	GCN-integrated DRL	Better feature extraction and decision-making	Relies on specific network features; scalability in large-scale networks not addressed
Zhang et al. (2022)	Dynamic VNE with RL and GCNN	Flexibility in adapting to resource attribute changes	Limited exploration of temporal network properties

Method	Technique	Strengths	Limitations
Qu et al. (2022)	Generative Adversarial Graph Network (ImGAGN)	Effective in addressing imbalanced classification on graphs	Focused on classification rather than VNE-specific challenges
Kumar et al. (2023)	Network embedding for community detection	Generalizable framework for evaluating embedding techniques	Not directly applicable to VNE; lacks focus on dynamic resource allocation
Jiao et al. (2023)	Temporal Network Embedding (TVAE)	Captures dynamic nonlinear features for temporal networks	Primarily focused on link prediction rather than VNE
Dhelim et al. (2023)	Trust management using network embedding (Trust2Vec)	Effective in managing trust relationships in IoT systems	Does not address resource optimization or embedding in general-purpose networks
Yang et al. (2023)	Mutual Contrastive Learning (MCL)	Aggregates cross-network embedding information	Focus on online knowledge distillation; lacks applicability to resource allocation in VNE
Zhang et al. (2024)	SAGIN cross-domain VNE with DRL	Models different network segments with tailored attributes	Specific to SAGIN; limited generalizability to other types of networks
Thafar et al. (2023)	Drug–target link prediction using network representation learning (DTi2Vec)	Constructs heterogeneous networks for feature generation	Targeted at biological applications, not applicable to VNE

The comparison highlights the unique contributions and limitations of each method, showcasing gaps that the proposed InDS framework aims to address, such as integrating diverse objectives, utilizing advanced feature extraction techniques, and improving scalability with adaptive learning mechanisms.

Despite these advancements, existing frameworks often overlook critical objectives such as energy efficiency and fairness. Moreover, their reliance on specific network features and fixed learning mechanisms limits scalability and adaptability. The proposed InDS framework aims to bridge these gaps by incorporating multi-objective optimization, advanced graph neural networks, and adaptive learning techniques.

3. Deep Reinforcement Learning Framework for Online Virtual Network Embedding

The growing demand for virtualization technologies has transformed the landscape of networking, enabling dynamic and efficient resource allocation for diverse applications. Virtual Network Embedding (VNE) is a critical component in this domain, responsible for mapping Virtual Network Requests (VNRs) onto the underlying substrate network. Traditional approaches to VNE often rely on heuristic or static algorithms, which struggle to adapt to the dynamic nature of network environments and fail to optimize resource utilization effectively.

Deep Reinforcement Learning (DRL) has emerged as a promising solution to address the complexities of VNE. By leveraging its ability to model sequential decision-making, DRL can dynamically adapt to changing network states and optimize resource allocation. This framework is further enhanced by integrating Graph Neural Networks (GNNs), particularly Graph Convolutional Networks (GCNs), which extract structural and feature-based information from the network, enabling

more informed decision-making. The combination of DRL and GNNs creates a robust system capable of handling the dynamic and complex requirements of online VNE.

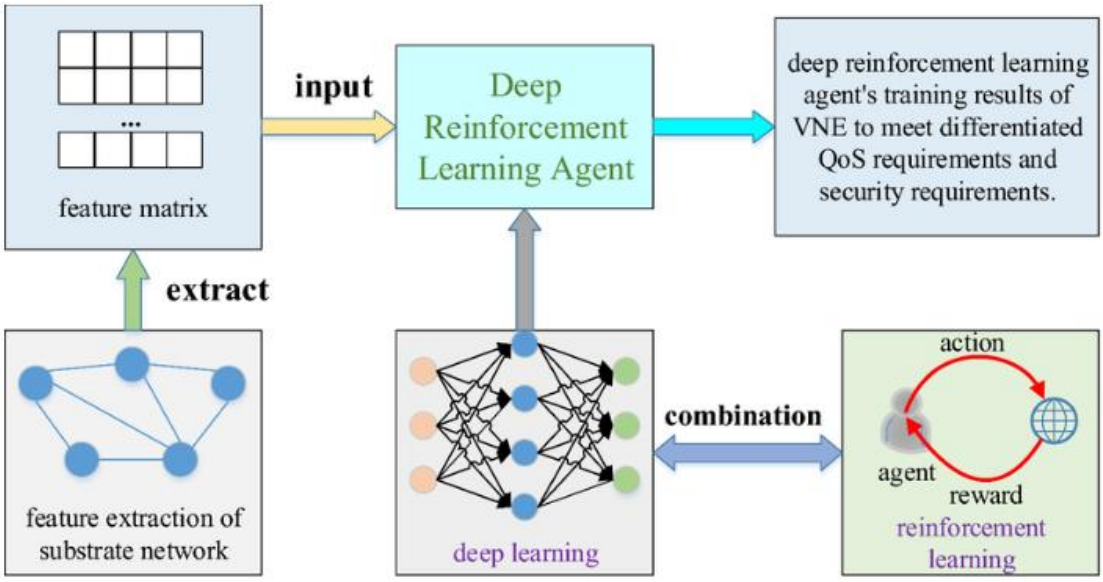

Figure 1. Block Diagram

This research introduces a novel DRL-based framework tailored for online VNE, emphasizing scalability, adaptability, and multi-objective optimization. By incorporating advanced feature extraction techniques and a sophisticated reward function, the proposed framework ensures optimal resource utilization, improved Quality of Service (QoS), and reduced network congestion. The following sections outline the methodology, including the workflow and algorithms, followed by an evaluation of its performance in dynamic network scenarios.

3.1 Overview of InDS Enhancements

The proposed enhancements to the InDS (Intelligent Network Slice) framework aim to improve its efficiency, scalability, adaptability, and performance across different network management and optimization tasks. These enhancements focus on three primary areas: incorporating diverse objectives, advanced feature extraction, and improving scalability and real-time performance. The proposed enhancements to the InDS framework focus on three primary areas:

3.1.1. Contribution 1: Incorporating Diverse Objectives:

In traditional network slice management, objectives may focus on optimizing for a single goal, such as maximizing throughput or minimizing latency. However, real-world networks often involve trade-offs between multiple goals that need to be balanced for better overall performance. The enhancement in this area focuses on:

Expanding the reward function: In Reinforcement Learning (RL)-based approaches, the reward function determines how an agent (e.g., a controller for network slices) evaluates its actions. The reward function can be expanded to incorporate energy efficiency, fairness, and network reliability, alongside more common objectives like throughput or latency.

Energy efficiency: With the growing concern over energy consumption in modern networks, optimizing for lower energy usage without compromising performance is crucial. This requires designing strategies that minimize the energy spent on managing network slices while maintaining quality of service (QoS).

Fairness among Virtual Network Requests (VNRs): Fairness involves allocating resources among VNRs in a manner that no request is unfairly deprived of resources while others are overly prioritized. It ensures that the network resources are distributed in a balanced manner to prevent congestion and unfair treatment.

Network reliability: This refers to ensuring that network services maintain consistent uptime and minimal disruption, even in the face of failures or traffic fluctuations. Optimizing for reliability might involve adapting to network conditions dynamically to avoid outages and minimize packet loss.

Utilizing multi-objective optimization techniques: Since these objectives may conflict with each other, such as maximizing throughput versus minimizing energy consumption, multi-objective optimization techniques can be applied. These techniques aim to find an optimal trade-off between these competing objectives, providing a solution that satisfies all objectives in a balanced manner. For instance, instead of maximizing just one objective (e.g., throughput), the model can try to achieve a solution where energy efficiency, fairness, and reliability are balanced.

3.1.2. Contribution 2: Advanced Feature Extraction:

Advanced feature extraction is a critical component in modern network management, particularly when dealing with complex and dynamic network environments. The two proposed enhancements in this section focus on improving how the network system captures and uses data from the network:

A. Employing Graph Isomorphism Networks (GINs) for More Informative Feature Representation

Graph Isomorphism Networks (GINs) are a specific type of Graph Neural Networks (GNNs) designed to effectively model graph-structured data. GNNs are powerful because they can capture complex relationships between nodes in a graph, where nodes represent entities (e.g., routers, switches, virtual network functions) and edges represent interactions (e.g., communication links or dependencies).

Traditional machine learning models may not be well-equipped to handle graph-structured data, especially when it comes to network management where the relationships between devices and network slices are inherently interconnected. GINs are particularly suited for this task because they focus on capturing the underlying graph structure in a more efficient way than other types of GNNs.

In network management, this means that GINs can represent the network's topology and traffic patterns in a way that traditional methods cannot. They use a powerful mechanism to aggregate information from neighbors, ensuring that the node representations (features) capture both local and global dependencies in the network. By using GINs, the InDS framework can better understand complex network topologies, interactions between network elements, and the flow of traffic across slices, leading to better decision-making in terms of resource allocation, routing, and load balancing.

Imagine a network with multiple virtual network functions (VNFs) interconnected in a specific topology. GINs can help identify which nodes are critical for ensuring optimal performance across the entire network. For example, a specific VNF may need higher bandwidth to handle traffic at certain times, and GINs can help inform the decision of how to best allocate resources by understanding both the local conditions (specific node traffic) and global conditions (overall network state).

B. Incorporating Temporal Information such as Historical Traffic Patterns

Networks are dynamic, meaning their state changes over time due to factors such as fluctuating traffic loads, varying demand from users, or changing network conditions (e.g., faults or congestion). Temporal information refers to the historical data collected over time, such as past traffic patterns, resource utilization, or demand trends. By considering these historical patterns, the system can better predict future network behavior and make more informed decisions.

Temporal data enables the model to learn from the past and anticipate future network conditions. This is important because a network system that can predict traffic surges, for example, can preemptively adjust resources to avoid congestion or failure, ensuring that performance is maintained even in high-demand situations. This adaptability is key in environments such as 5G or IoT networks, where demand and conditions can vary significantly and unpredictably.

Consider a scenario where a specific network slice experiences traffic spikes during certain times of day (e.g., high video streaming demand during evening hours). By incorporating historical traffic data, the InDS framework can recognize this pattern and proactively allocate more resources or adjust

routing strategies during peak times, improving the overall quality of service without waiting for real-time traffic monitoring to detect the spike.

3.1.3. Contribution 3: Scalability and Real-time Performance:

Scalability and real-time performance are crucial in large-scale networks, particularly as modern systems become more complex and traffic volumes increase. These enhancements aim to make the InDS framework more efficient and capable of handling real-time network management tasks in a scalable manner.

A. Introducing an Adaptive A3C Mechanism

Asynchronous Advantage Actor-Critic (A3C) is a reinforcement learning (RL) algorithm that improves training speed and performance by employing multiple agents (workers) to interact with the environment in parallel. Each agent trains a policy and value function, and the system asynchronously updates a central model. This enables the A3C algorithm to scale well with parallelization, making it efficient in complex environments.

In the context of network management, adaptive A3C can be extremely useful for handling real-time decisions. A3C is particularly effective for environments with large state spaces and long decision horizons, which are common in networks. By applying the A3C algorithm, InDS can continuously learn from network traffic, slice performance, and user demand, adjusting network resources and strategies dynamically.

The traditional A3C method uses a fixed learning rate during training, which can be inefficient, especially when the system needs to adapt to rapidly changing conditions. The adaptive mechanism improves this by dynamically adjusting the learning rate based on the performance of the model. This helps the training process by allowing the model to converge faster when it's making good progress, and slowing down when it needs more fine-tuning or is overfitting.

With adaptive learning rates, the model can learn faster without needing massive computational resources. When the model encounters areas of the network where decisions are easy to make, it accelerates learning. When dealing with complex decisions, it slows down to ensure high-quality learning. This adaptability ensures that the system scales efficiently, even when the number of network slices, devices, or traffic volumes increase.

By dynamically adjusting learning rates, the system can adapt to real-time network changes more efficiently. For example, when there's a sudden traffic surge or network fault, the model can quickly learn from these changes and adjust its decisions without the delay of a fixed learning process.

B. Dynamic Learning Rates to Accelerate Training

In typical reinforcement learning, a fixed learning rate can lead to either slow convergence (if too small) or instability (if too large). A dynamic learning rate allows the model to balance exploration and exploitation during training, speeding up the learning process in predictable environments and slowing down when it needs to refine its strategy.

By adjusting learning rates in response to network conditions, the InDS framework can scale efficiently. For example, when traffic patterns are stable and predictable, the learning rate can be increased to speed up training. However, when the network experiences dynamic conditions (e.g., congestion or faults), the learning rate can decrease to allow for more careful decision-making. This enables faster adaptation to changing conditions without sacrificing the quality of the decisions made.

3.2 Workflow and Methodology

The proposed workflow consists of the following steps:

3.2.1. 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.

Let's consider a simple network where nodes represent servers and edges represent communication links between them. Each node will have properties like computational capacity (CPU, RAM) and each edge will represent bandwidth.

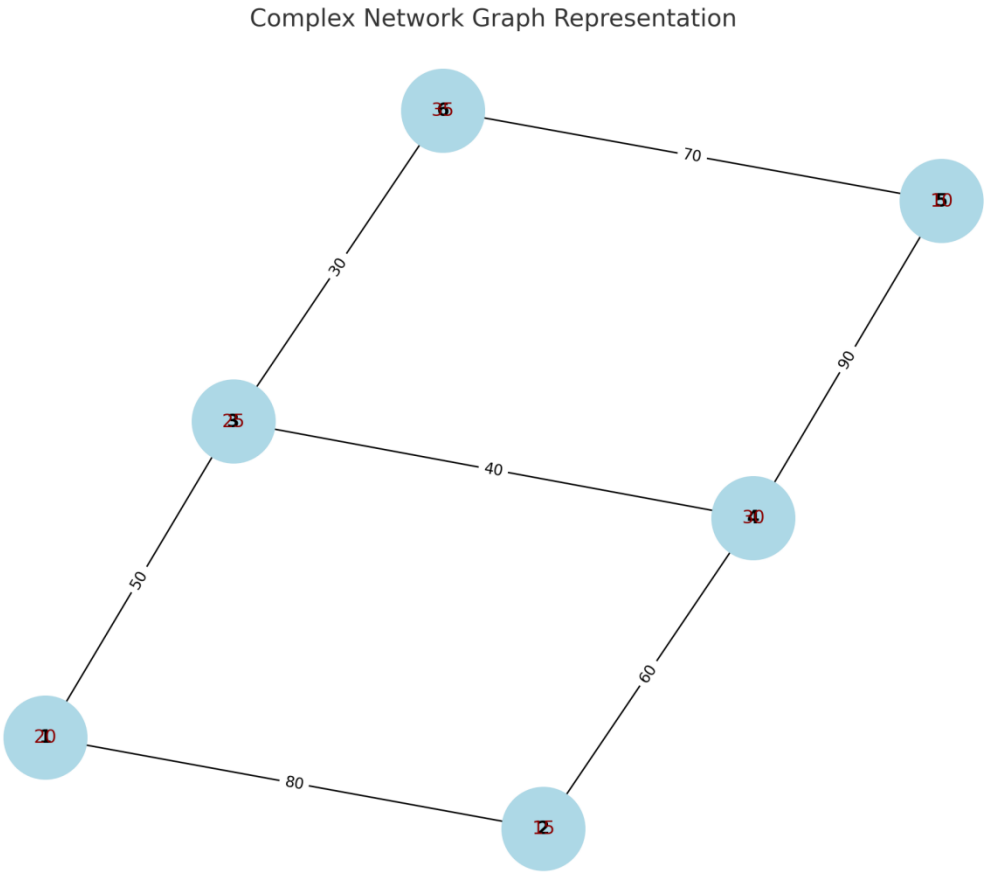

Figure 2. Network Diagram

Figure 2 shows graph representation of the substrate network. It consists of six nodes, each representing a network device (with computational resources such as CPU and RAM), and the edges represent communication links between these devices, with associated bandwidth values. This type of graph can be used to model a more detailed network environment, where resource allocation decisions would depend not only on node attributes (like CPU or RAM) but also on the bandwidth of the links between devices. This complexity helps in understanding how to distribute network slices and optimize resource allocation for performance and efficiency.

3.2.2. Feature Extraction:

The next step is to extract meaningful features from the network graph. Since the network has both structural and temporal elements, Graph Isomorphism Networks (GINs) are used to extract these features.

Structural Features: These are features related to the graph's structure, such as node degree, shortest path, or network centrality.

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

Graph Isomorphism Networks (GINs) are well-suited for this because they capture both the local and global structure of the graph. GINs operate by iterating over each node and aggregating information from its neighbors to create a new representation for each node. They are particularly good at distinguishing different graph structures, making them useful for dynamic network environments. GINs allow for a more informative representation of each node, which is then used by the decision-making module.

Algorithm 1: Graph Feature Extraction using GINs

Input: Graph G(V, E) with node features X and edge connections

Output: Node embeddings with structural and temporal features

1. Initialize network G with nodes V and edges E
2. Initialize node features X for each node in V
3. Define Graph Isomorphism Network (GIN) Layer:
   a. For each node v in V:
      i. Aggregate features from neighboring nodes
         Neighbor_Agg[v] = Aggregate({X[u] | u ∈ Neighbors(v)})
      ii. Transform aggregated features using a non-linear function
         Updated_Features[v] = MLP(Neighbor_Agg[v] + X[v])
      iii. Normalize and pass through activation function
         X[v] = Activation(Updated_Features[v])
4. Extract Structural Features:
   a. Compute node degree for each node v
   b. Compute shortest path matrix for all node pairs
   c. Compute centrality measures (e.g., betweenness, closeness, eigenvector centrality)
5. Extract Temporal Features:
   a. Track historical traffic load for each node and edge
   b. Monitor congestion levels based on bandwidth utilization
   c. Detect node failures based on network activity logs
6. Output final node embeddings with extracted features

3.2.3. Decision-making:

After extracting structural and temporal features using Graph Isomorphism Networks (GINs), the decision-making process relies on Deep Reinforcement Learning (DRL) to determine optimal resource allocation strategies. Once features are extracted using GINs, they are passed as input to the Deep Reinforcement Learning (DRL) model. In the decision-making phase, the system utilizes a Deep Reinforcement Learning (DRL) approach. The framework employs an Actor-Critic model, where:

Actor: The actor suggests actions based on the state of the network. In the context of network slices, actions would involve resource allocation decisions, such as deciding how to allocate bandwidth or computational resources to various slices.

Critic: The critic evaluates the actions suggested by the actor. It estimates the value of the chosen actions (state-action pairs) and provides feedback to improve future actions.

In this model, the actor outputs a probability distribution over possible actions (such as allocating a specific amount of bandwidth to a virtual network slice), and the critic evaluates the reward based on how well the action performed. Since reinforcement learning relies on rewards to refine its strategy, we define a reward function that reflects multiple performance objectives.

3.2.4. Reward Function:

The reward function is crucial in guiding the Actor-Critic model to make better decisions. The reward function in this workflow plays a key role in balancing multiple competing objectives that define the quality of network slice management. The goal is to design a reward function that reflects the following key objectives:

Acceptance Ratio: Measures the ratio of successfully accepted requests to total requests.

Revenue-to-Cost Ratio: Measures the financial efficiency of network slice allocation, balancing income from resource usage with the cost of providing resources.

Energy Efficiency: Encourages resource allocation strategies that minimize energy consumption while maintaining performance.

Fairness: Ensures fair distribution of resources among different virtual network requests (VNRs), avoiding favouring any single request.

The reward function balances the competing objectives, each contributing to the final reward signal used for updating the policy. However, for the DRL model to train efficiently in real time, we incorporate Adaptive A3C Training.

Algorithm 2: Decision-Making using Actor-Critic Deep Reinforcement Learning

Input: Extracted Features from GINs, Network State S

Output: Optimal Resource Allocation Decisions

1. Initialize Actor-Critic model with parameters θ (Actor) and φ (Critic)

2. Initialize environment (network state S) and set learning rates

3. While training is not complete:

   a. Observe current network state S

   b. Extract features using Graph Isomorphism Networks (GINs)

   c. Actor generates action A based on state S using policy $\pi(A|S, \theta)$

   d. Execute action A (e.g., allocate bandwidth or computational resources)

   e. Observe next state S' and compute reward R using Reward Function:

      i. Acceptance Ratio = (Accepted Requests / Total Requests)

      ii. Revenue-to-Cost Ratio = (Revenue Generated / Allocation Cost)

      iii. Energy Efficiency = (Total Energy Saved / Total Energy Used)

      iv. Fairness = (Variance of Resource Allocation across VNRs)

   f. Compute total reward R_total as a weighted sum of objectives:

      R_{total} = w1 * Acceptance-Ratio + w2 * Revenue-to-Cost +

      w3 * Energy-Efficiency + w4 * Fairness

   g. Critic evaluates action A and updates value function $V(S, \varphi)$

   h. Compute advantage estimate: Advantage = $R_{total} + \gamma * V(S', \varphi) - V(S, \varphi)$

   i. Update Critic's parameters φ using gradient descent

   j. Update Actor's parameters θ using policy gradient:

$\theta \leftarrow \theta + \alpha * \text{Advantage} * \nabla \log \pi(A|S, \theta)$

4. Continue training until convergence

5. Return optimized policy $\pi(A|S, \theta)$ for real-time resource allocation

3.2.5. Adaptive A3C Training:

To ensure efficient and scalable learning, we employ Adaptive Asynchronous Advantage Actor-Critic (A3C), which enhances the decision-making process. The Adaptive A3C mechanism optimizes the training process by adjusting the learning rate dynamically based on the agent's performance. This allows for faster convergence when the agent is performing well and slower adjustments when the system needs fine-tuning.

- A3C uses multiple agents running in parallel to explore different parts of the state space, allowing for faster and more diverse learning.
- Adaptive Learning Rates adjust the speed of learning based on the feedback from the critic, ensuring the agent learns efficiently in real-time.

The Adaptive A3C adjusts learning rates dynamically and performs updates based on the feedback from the critic. By integrating Adaptive A3C, the model can balance fast training with fine-tuned decision-making, ensuring real-time performance in network slicing.

Algorithm 3: Adaptive InDS Framework Training

1. Initialize the actor and critic networks with random weights.

2. Represent the substrate network as a graph .

3. Extract features using GINs and temporal information.

4. For each VNR:

   Calculate the action using the actor network.

   Allocate resources based on the action.

5. Valuate the reward based on the defined objectives.

6. Update the actor and critic networks using the Adaptive A3C mechanism.

7. Repeat until convergence.

4. Results and Discussion

The proposed Adaptive InDS framework was evaluated using simulation environments that replicate dynamic network conditions. The primary metrics used for evaluation were acceptance ratio, revenue-to-cost ratio, and resource utilization. Each of these metrics provides insight into the performance and effectiveness of the proposed method in comparison with existing methods. Below is a breakdown of the comparison of these metrics between the proposed Adaptive InDS and the existing methods (InDS, DRL-based VNE, GCN integrated DRL, TVAE, and MCL).

4.1. Acceptance Ratio

The acceptance ratio measures the proportion of Virtual Network Requests (VNRs) that can be successfully accepted and mapped to the available resources in the network. A higher acceptance ratio signifies better capability in accommodating VNRs under dynamic conditions, which is critical in a virtual network embedding context. It implies a more efficient allocation of network resources without rejection due to lack of capacity.

Proposed Work: Adaptive InDS

The Adaptive InDS framework incorporates dynamic adjustments to resources and policies, which significantly improves the acceptance ratio by optimally adapting to changing network conditions.

Comparison with Existing Work:

InDS: Static policies used in InDS lead to lower acceptance ratios compared to Adaptive InDS due to inefficient handling of dynamic network conditions.

DRL-based VNE: DRL-based methods typically improve acceptance ratio by learning from past network conditions, but they may be slower in adapting to sudden changes compared to Adaptive InDS.

GCN integrated DRL: The integration of GCN with DRL can perform well in static conditions but struggles with scalability in dynamic conditions, resulting in lower acceptance ratios in comparison to Adaptive InDS.

TVAE: The TVAE method has high accuracy but may not be as flexible in terms of adapting to rapid changes in network dynamics, which affects its acceptance ratio.

MCL: MCL performs well in terms of clustering but may not efficiently handle dynamic resource allocation, resulting in a lower acceptance ratio compared to Adaptive InDS.

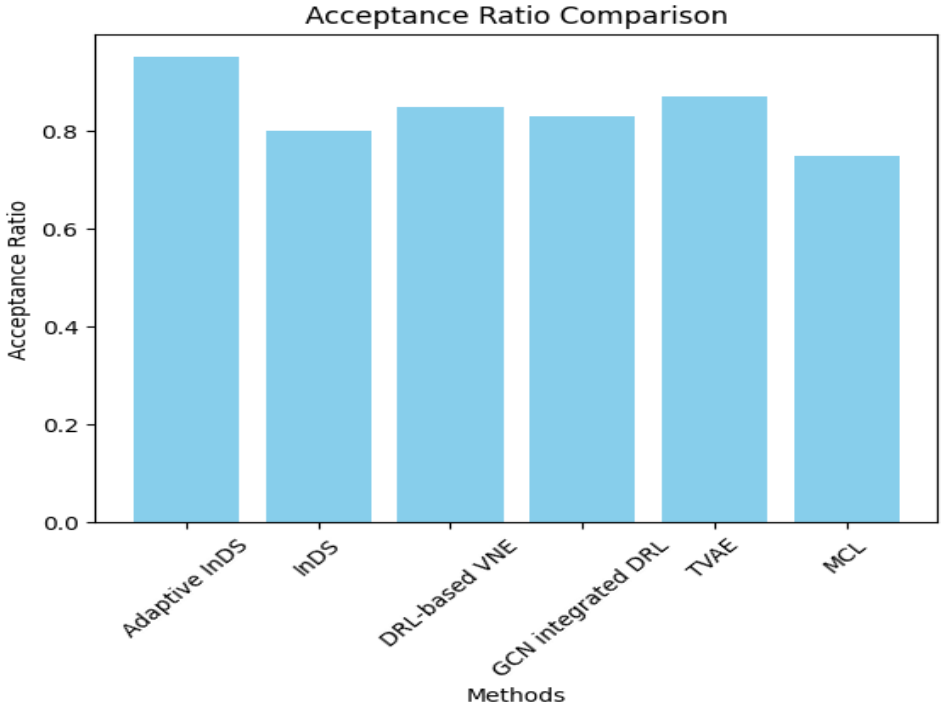

Figure 3. Acceptance Ratio Comparison

The bar graph in figure 3 shows that Adaptive InDS has the highest acceptance ratio, significantly outperforming other methods. The methods such as InDS and MCL perform the worst, indicating that static or less adaptable methods struggle with dynamic network conditions.

4.2. Revenue-to-Cost Ratio

This ratio evaluates the economic efficiency of the virtual network embedding process by comparing the revenue generated from successfully embedded VNRs to the associated costs of allocating resources. A higher revenue-to-cost ratio indicates that the framework is effectively optimizing the trade-off between resource costs and the generated revenue. This is important for ensuring the profitability of network providers.

Proposed Work: Adaptive InDS

Adaptive InDS employs an adaptive strategy that dynamically optimizes resource allocation, leading to improved revenue generation while keeping costs under control. This results in a higher revenue-to-cost ratio.

Comparison with Existing Work:

InDS: InDS typically has a lower revenue-to-cost ratio due to the lack of adaptability in dynamic environments.

DRL-based VNE: DRL-based methods can achieve good performance but may be less efficient at balancing revenue and costs in highly dynamic conditions.

GCN integrated DRL: This method shows improvements in scalability and revenue generation, but due to complexity in network mapping, it is less efficient in cost management compared to Adaptive InDS.

TVAE: TVAE performs well in terms of revenue generation but has relatively higher costs, leading to a lower revenue-to-cost ratio.

MCL: MCL focuses on clustering and does not directly optimize the revenue-to-cost ratio, resulting in poor performance.

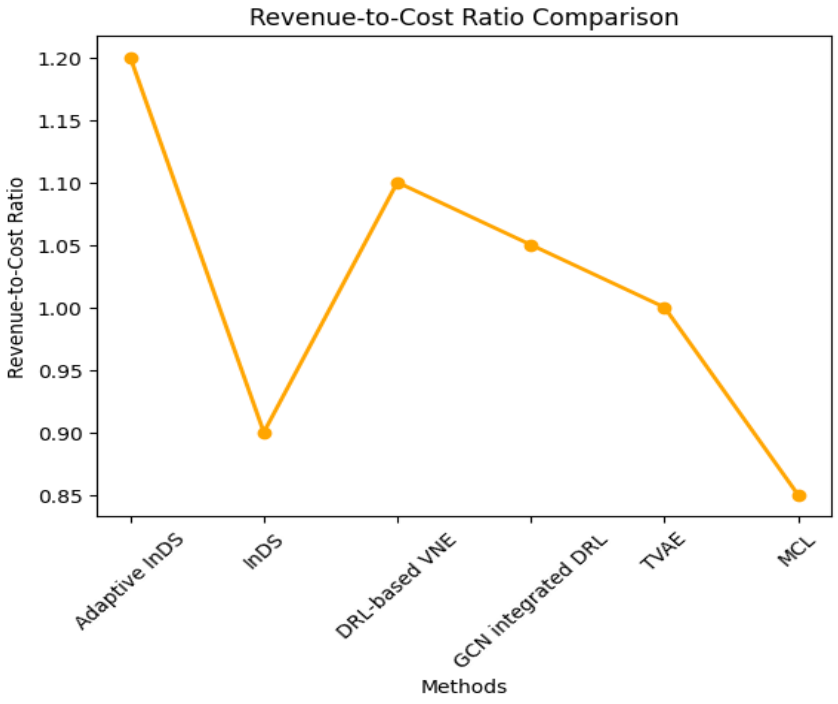

Figure 4. Revenue to Cost Ratio

The line graph in figure 4 indicates that Adaptive InDS achieves the highest revenue-to-cost ratio, demonstrating its superior ability to balance costs and revenue under dynamic network conditions. The MCL method underperforms, likely due to its inability to optimize this specific metric.

4.3. Resource Utilization

This metric measures the efficiency with which network resources (such as bandwidth, processing power, and memory) are utilized for the embedding of virtual networks. High resource utilization signifies that the resources are being used efficiently, without underutilization (wasting resources) or overutilization (causing congestion or failure).

Proposed Work: Adaptive InDS

Adaptive InDS ensures that resources are dynamically adjusted and utilized based on changing network demands, leading to optimal resource utilization.

Comparison with Existing Work:

InDS: InDS typically shows suboptimal resource utilization due to its static allocation strategy.

DRL-based VNE: DRL-based methods show good resource utilization but may struggle with real-time adjustments in highly dynamic conditions.

GCN integrated DRL: While this approach shows efficient resource usage, its scalability issues can lead to inefficiencies in larger networks.

TVAE: The TVAE method shows moderate resource utilization, often resulting in underutilization due to its focus on embedding quality over dynamic resource management.

MCL: MCL does not directly manage resource utilization, leading to less efficient usage compared to methods like Adaptive InDS.

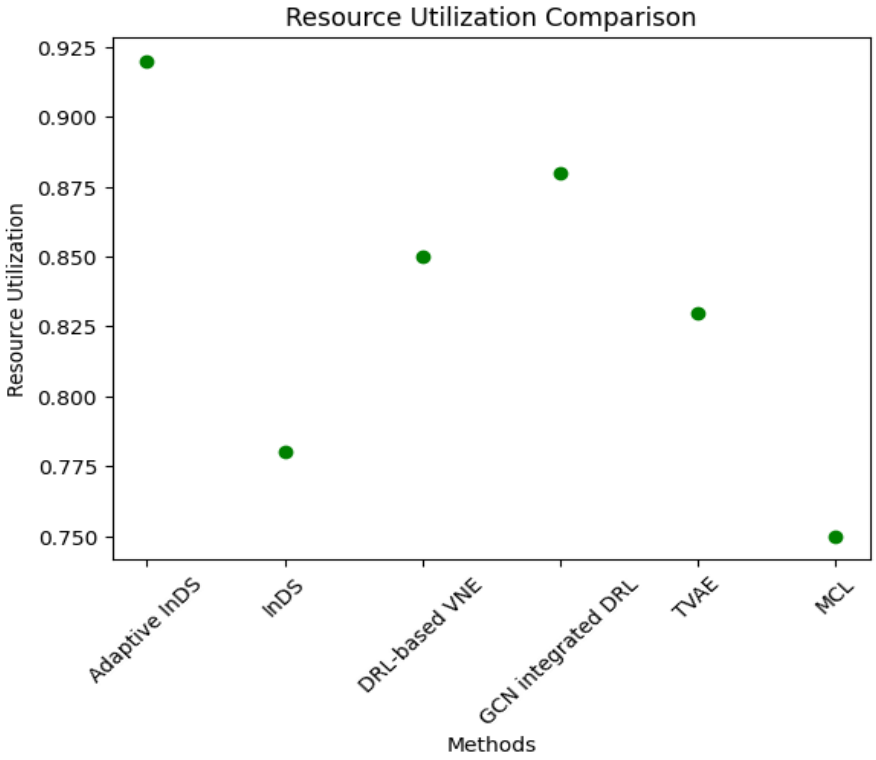

Figure 5. Resource Utilization comparison

The scatter plot in figure 5 reveals that Adaptive InDS has the highest resource utilization, followed by DRL-based VNE. The MCL method exhibits the lowest resource utilization, emphasizing the importance of dynamic resource management.

4.4. Adaptability

Adaptability refers to the framework's ability to adjust to changing network conditions, such as fluctuations in network traffic, resource availability, and external disruptions. A highly adaptable system can dynamically adjust its operations to accommodate varying conditions without sacrificing performance, making it well-suited for real-world scenarios where network conditions are rarely static.

Proposed Work: Adaptive InDS

The Adaptive InDS framework uses adaptive policies to dynamically adjust the virtual network embedding process. By considering real-time changes in network conditions, it can maintain high performance even when faced with sudden increases or decreases in network traffic and resource availability.

Comparison with Existing Work:

InDS: The static resource allocation in InDS limits its adaptability. It does not perform well when network conditions change rapidly, leading to inefficiency in embedding virtual networks under varying traffic conditions.

DRL-based VNE: DRL methods are inherently designed to learn and adapt to network conditions through training. However, their adaptability can be slower in environments where conditions change suddenly, as it takes time for the system to update its knowledge.

GCN integrated DRL: The integration of Graph Convolutional Networks (GCN) with DRL enhances adaptability by better handling graph-based relationships in the network. However, this approach is still slower to adapt compared to Adaptive InDS, particularly in more dynamic environments.

TVAE: TVAE can model network conditions and generate embeddings that reflect network dynamics, but it is less responsive than Adaptive InDS when the environment changes in real-time.

MCL: MCL is a clustering method that does not inherently possess adaptability. Its fixed cluster structures cannot adjust in real-time to changing network conditions.

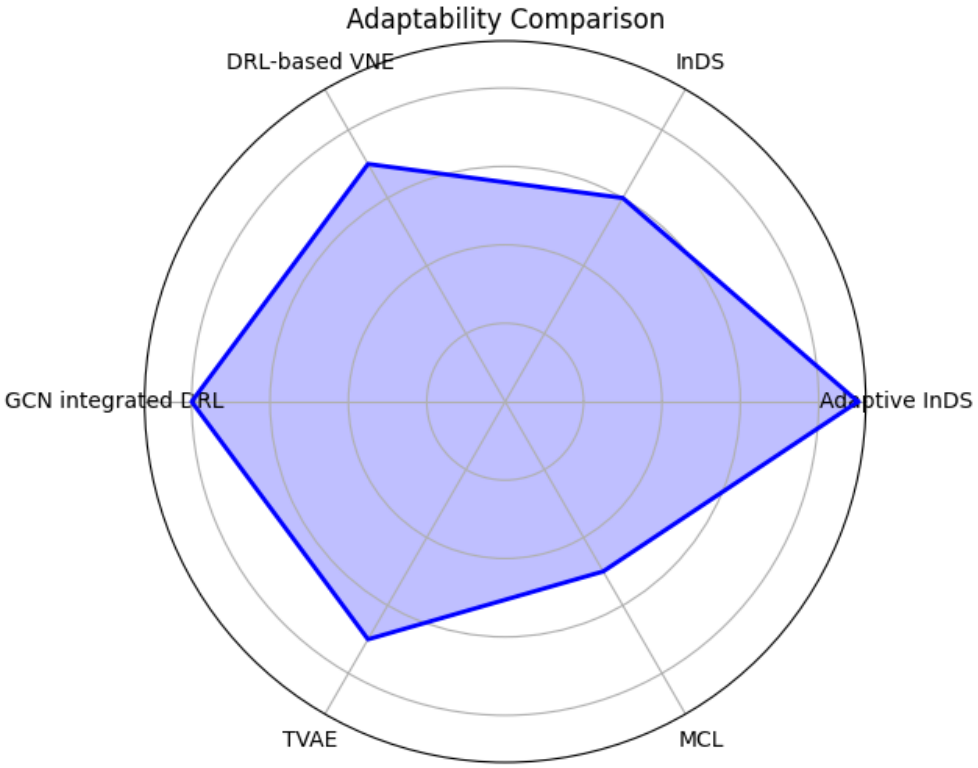

Figure 6. Adaptability Comparison

The radar chart highlights the adaptability of each method. Adaptive InDS scores the highest in adaptability, showing its ability to handle dynamic changes effectively. MCL has the lowest score due to its lack of real-time adaptation capabilities.

4.5. Improved Stability

Stability refers to the framework's ability to maintain consistent performance over time, even when subjected to fluctuations in network conditions, such as sudden surges in traffic, network failures, or resource shortages. A stable system is one that does not experience significant performance degradation or sudden drops in performance, even under fluctuating or challenging conditions.

Proposed Work: Adaptive InDS

The Adaptive InDS framework is designed with stability in mind. By continuously adjusting to network fluctuations, it minimizes performance degradation and ensures that resource allocation remains balanced even when external factors affect the system.

Comparison with Existing Work:

InDS: InDS exhibits lower stability because its static approach does not adjust well to network fluctuations, leading to performance issues under varying conditions.

DRL-based VNE: DRL methods offer improved stability compared to InDS due to their learning capabilities, but they can still experience instability if there is a sudden shift in network conditions before the model adapts.

GCN integrated DRL: GCN-based methods combined with DRL provide better stability, especially when network topologies are complex. However, they still fall short of Adaptive InDS in maintaining consistent performance under rapid changes.

TVAE: TVAE provides stable performance under many conditions but is vulnerable to performance fluctuations when facing significant network changes, making it less stable than Adaptive InDS.

MCL: MCL offers poor stability as it relies on clustering, which does not adjust to real-time network changes, causing instability when network conditions shift suddenly.

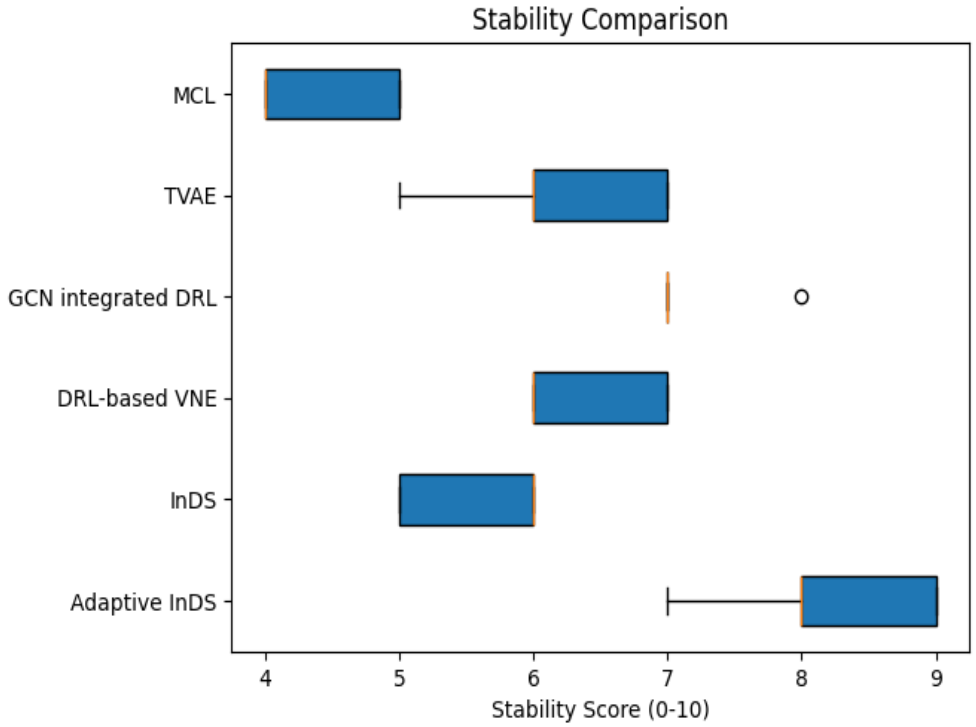

Figure 7. Stability Comparison

The box plot provides a visual summary of the stability of each method. Adaptive InDS exhibits the tightest distribution around a high median score, indicating consistent stability across trials. In contrast, MCL shows a wide spread and lower median, indicating poor stability under fluctuating network conditions.

Table 2. Performance comparison values

Metric	Adaptive InDS	InDS	DRL based VNE	GCN integrated DRL	TVAE	MCL
Acceptance Ratio	0.95	0.80	0.85	0.83	0.87	0.75

Revenue-to-Cost Ratio	1.2	0.9	1.1	1.05	1.0	0.85
Resource Utilization	0.92	0.78	0.85	0.88	0.83	0.75
Adaptability	9/10	6/10	7/10	8/10	7/10	5/10
Stability	High (Tight range)	Low (Wide spread)	Medium (Moderate spread)	Medium (Moderate spread)	Medium (Wide spread)	Low (Wide spread)

5. Conclusion

This research introduces a novel enhancement to the InDS framework, 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. Future work will focus on real-world implementation and further exploration of emerging technologies such as 6G and edge computing.

References

1. Darwish, D. (Ed.). (2024). Emerging Trends in Cloud Computing Analytics, Scalability, and Service Models.

2. Barakabitze, A. A., & Walshe, R. (2022). SDN and NFV for QoE-driven multimedia services delivery: The road towards 6G and beyond networks. Computer Networks, 214, 109133.

3. Firoozi, A. A., Firoozi, A. A., Alashker, Y., & Ahmad, S. (2024). Advancing civil engineering: The transformative impact of neuromorphic computing on infrastructure resilience and sustainability. Results in Engineering, 24, 103487.

4. Arogundade, O. R., & Palla, K. (2023). Virtualization revolution: Transforming cloud computing with scalability and agility.

5. Arogundade, O. R., & Palla, K. (2023). Virtualization revolution: Transforming cloud computing with scalability and agility.

6. Rahman, A., Islam, J., Kundu, D., Karim, R., Rahman, Z., Band, S. S., ... & Kumar, N. (2025). Impacts of blockchain in software-defined Internet of Things ecosystem with Network Function Virtualization for smart applications: Present perspectives and future directions. International Journal of Communication Systems, 38(1), e5429.

7. Sarmiento, D. E., Lebre, A., Nussbaum, L., & Chari, A. (2021). Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey. IEEE Communications Surveys & Tutorials, 23(1), 256-281.

8. Λιλής, Π. (2024). Network Function Virtualization (NFV) technologies and their impact on energy consumption in large-scale data centers (Master's thesis, Πανεπιστήμιο Πειραιώς).

9. Koot, M., & Wijnhoven, F. (2021). Usage impact on data center electricity needs: A system dynamic forecasting model. Applied Energy, 291, 116798.

10. Cunha, J., Ferreira, P., Castro, E. M., Oliveira, P. C., Nicolau, M. J., Núñez, I., ... & Serôdio, C. (2024). Enhancing Network Slicing Security: Machine Learning, Software-Defined Networking, and Network Functions Virtualization-Driven Strategies. Future Internet, 16(7), 226.

11. Nguyen, K., Shi, W., & St-Hilaire, M. (2024). Dynamic Virtual Network Embedding Leveraging Neighborhood and Preceding Mappings Information. IEEE Transactions on Vehicular Technology.

12. Wang, Y., Liu, J., Cui, M., Wu, W., & Huang, T. (2023). ED-VNE: A profit-oriented VNE optimization scheme of energy and delay in 5G SlaaS. Computer Networks, 236, 110003.

13. Nguyen, K. T. D. (2021). Distributed and Parallel Metaheuristic-based Algorithms for Online Virtual Resource Allocation (Doctoral dissertation, Carleton University).

14. 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.

15. 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.

16. Wang, H., et al. (2021). "Deep Reinforcement Learning for Efficient VNE." IEEE Transactions on Network and Service Management.

17. Zhang, Y., et al. (2022). "GCNs in VNE: A New Paradigm." ACM Transactions on Networking.

18. Zhang, P., Wang, C., Kumar, N., Zhang, W., & Liu, L. (2021). Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning. IEEE Internet of Things Journal, 9(12), 9389-9398.

19. Qu, L., Zhu, H., Zheng, R., Shi, Y., & Yin, H. (2021, August). Imgagn: Imbalanced network embedding via generative adversarial graph networks. In Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining (pp. 1390-1398).

20. Kumar, S., Panda, B. S., & Aggarwal, D. (2021). Community detection in complex networks using network embedding and gravitational search algorithm. Journal of Intelligent Information Systems, 57(1), 51-72.

21. Jiao, P., Guo, X., Jing, X., He, D., Wu, H., Pan, S., ... & Wang, W. (2021). Temporal network embedding for link prediction via VAE joint attention mechanism. IEEE Transactions on Neural Networks and Learning Systems, 33(12), 7400-7413.

22. Dhelim, S., Aung, N., Kechadi, M. T., Ning, H., Chen, L., & Lakas, A. (2022). Trust2Vec: Large-scale IoT trust management system based on signed network embeddings. IEEE Internet of Things Journal, 10(1), 553-562.

23. Yang, C., An, Z., Zhou, H., Zhuang, F., Xu, Y., & Zhang, Q. (2023). Online knowledge distillation via mutual contrastive learning for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(8), 10212-10227.

24. Zhang, P., Wang, C., Kumar, N., & Liu, L. (2021). Space-air-ground integrated multi-domain network resource orchestration based on virtual network architecture: A DRL method. IEEE Transactions on Intelligent Transportation Systems, 23(3), 2798-2808.

25. Thafar, M. A., Olayan, R. S., Albaradei, S., Bajic, V. B., Gojobori, T., Essack, M., & Gao, X. (2021). DTi2Vec: Drug–target interaction prediction using network embedding and ensemble learning. Journal of cheminformatics, 13, 1-18.