CS363 - Cloud Computing

Towards learning-based energy-efficient online coordinated virtual network embedding framework

Team Members:

1. N Yaswanth - 221CS232
2. Syed Farhan - 221CS254
3. Vishruth S Kumar - 221CS262
4. Yashas - 221CS265

Table of Contents

Contents

Introduction

In today's data-driven world, network virtualization enables efficient resource sharing in large data centers and IoT networks, where multiple virtual networks (VNs) operate over a shared physical infrastructure. The efficient allocation of resources in these virtual networks, a process known as **Virtual Network Embedding (VNE)** [1], is critical to supporting diverse, dynamic applications. This section introduces the key concepts and methods that are central to understanding VNE, focusing on recent advancements and their role in network management [2].

1. Virtual Network Embedding (VNE)

VNE is the process of mapping virtual network components, such as nodes and links, onto physical resources in a substrate network. This enables multiple virtual networks to coexist on a single physical network infrastructure. The VNE problem is NP-hard [3–5], meaning that finding an optimal solution is computationally infeasible in large-scale network environments. Effective VNE algorithms must maximize resource utilization, minimize costs, and maintain efficient performance across dynamic network conditions.

2. Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) combines deep learning with reinforcement learning to tackle decision-making in complex environments [6]. In DRL, an autonomous agent interacts with an environment and learns from the feedback (rewards) received from each action. By learning through trial and error, DRL [7] can optimize complex tasks such as VNE, where the agent must adapt to changing network demands and optimize the placement of virtual nodes and links in real-time [8].

3. Proximal Policy Optimization (PPO)

One of the standout algorithms in DRL is Proximal Policy Optimization (PPO) [9], which provides a stable and efficient policy update mechanism. PPO prevents drastic changes in policy by clipping updates, ensuring smoother learning. This makes PPO particularly suitable for handling the real-time and dynamic nature of VNE, where small adjustments in policies can lead to more reliable and adaptive embedding strategies over time [10].

4. Graph Convolutional Networks (GCN)

Graph Convolutional Networks (GCN) are designed to process data with complex, graph-like relationships, such as the topologies of virtual and physical networks in VNE. GCNs capture intricate structural relationships between nodes and edges, allowing them to extract valuable features for network embedding [11]. This enables more informed decision-making when mapping virtual networks onto physical infrastructure.

5. Hybrid Feature Extraction

In the PPO-based VNE framework, feature extraction is handled by a Hybrid Feature Extraction approach, which integrates both manually designed features and automatically extracted GCN features [12]. This hybrid method enhances the model's ability to recognize network patterns, leading to improved performance in embedding decisions. Manually crafted features provide control and precision, while GCN features offer adaptability, making this approach both versatile and efficient.

6. Multi-Objective Optimization

To address the diverse objectives of VNE, the PPO-based framework incorporates a Multi-Objective Optimization strategy that aims to maximize network revenue while minimizing energy consumption [13]. The reward function used in this framework is designed to balance these goals, guiding the agent toward actions that optimize both financial and resource efficiency. This multi-objective approach enhances the framework's ability to handle large-scale network environments.

7. Performance Metrics

Key performance metrics are used to evaluate the effectiveness of VNE solutions. Two critical metrics include the **Acceptance Rate** (the proportion of virtual networks successfully embedded onto the physical network) and the **Revenue-to-Cost Ratio** (a measure of how efficiently resources are utilized in the network) [14]. These metrics help determine the quality of VNE algorithms and their ability to handle large-scale and dynamic networks.

8. Energy Consumption and Efficiency

In VNE, energy efficiency is a significant concern, especially in large-scale networks. Metrics such as **Energy Consumption** (the maximum energy consumed per unit time) and the **Revenue-to-Energy Consumption Coefficient** (financial return relative to energy use) provide insights into the energy performance of the VNE solution [15]. Minimizing energy consumption while maximizing performance is essential for sustainable network operations.

9. Software-Defined Networking (SDN) and Network Function Virtualization (NFV)

Software-Defined Networking (SDN) and **Network Function Virtualization (NFV)** are foundational technologies that support flexible, scalable, and energy-efficient network infrastructure [16]. SDN separates the network control plane from the data plane, allowing centralized management of network resources. NFV virtualizes essential network functions, enabling efficient allocation of resources. Together, these technologies enable adaptable network solutions that can support the real-time demands of modern applications [17].

10. Service Function Chain Placement (SFCP)

In addition to VNE, Service Function Chain Placement (SFCP) is an optimization problem relevant in NFV environments [18]. SFCP involves placing linked virtual network functions (VNFs) as a sequence, or "service function chain" (SFC), in the physical network to ensure optimal performance and resource use. Integrating DRL into SFCP enables dynamic and efficient solutions that balance resource demands with service quality requirements.

11. The Role of DRL in Future Network Management

As VNE frameworks evolve, the integration of DRL with NFV and SDN systems opens new avenues for dynamic, energy-efficient network management [19]. DRL-based solutions provide powerful tools for balancing performance with sustainability, making them ideal for applications such as load balancing, resource management, and real-time optimization in next-generation networks.

In conclusion, the integration of advanced technologies such as DRL, GCNs, and multi-objective optimization strategies into the VNE process presents significant opportunities for enhancing the performance and efficiency of network management [20]. As the demand for dynamic and scalable network solutions continues to grow, the evolution of VNE frameworks will be pivotal in addressing the challenges posed by resource allocation and energy consumption in modern networking environments.

The subsequent sections will delve into the detailed methodologies and experimental results that underscore the effectiveness of the proposed approaches in optimizing virtual network embedding.

Understanding

This section details the modeling and structure of the Virtual Network Embedding (VNE) problem within the Reinforcement Learning (RL) framework, particularly focusing on the elements of state, action, and reward.

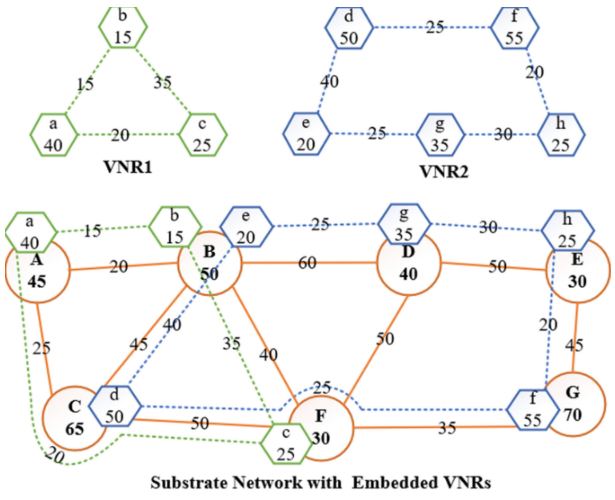

Figure 1: A typical example of virtual network embedding

1. State Representation

The state in an RL context represents the current condition of the network and encapsulates the information necessary for the RL agent to make informed decisions. In the VNE framework, the state is composed of two primary components:

- **Substrate Network State**: This aspect of the state provides a detailed view of the physical network's resources and configurations. It encompasses crucial metrics such as the availability of CPU resources, the degree of nodes (i.e., the number of connections each node has), and the mapping status of these nodes [21]. Additionally, it includes available bandwidth and other network features that are extracted via a Graph Convolutional Network (GCN). The GCN aids in identifying the structural and topological characteristics of the substrate network, enabling the RL agent to better understand how these features influence the embedding process.

- **Virtual Network State**: The virtual network state reflects the specific requirements and configurations of the virtual networks that are being requested for embedding [22]. This includes detailed information about virtual nodes and links, as well as attributes related to their performance and resource demands. The virtual network state may also integrate features that enhance the agent's understanding of incoming virtual network requests, such as the anticipated traffic patterns and resource utilization profiles. By capturing both the substrate and virtual network states, the RL agent gains a holistic view that is essential for effective decision-making in the embedding process.

The combination of these state representations enables the RL agent to assess the current network situation accurately, facilitating more informed embedding decisions [23].

2. Action Space

In the context of VNE, actions refer to the decisions made by the RL agent regarding the embedding of virtual network components onto the physical substrate. To effectively manage the complexity of the embedding task, the action space is structured to break down the overall embedding process into a series of manageable decisions:

- Each action corresponds to either the mapping of a virtual node to a specific substrate node or the connection of virtual links to substrate links [24]. This decomposition into discrete actions allows the RL agent to explore the vast action space systematically. Instead of making a single, complex decision for an entire virtual network, the agent makes a series of simpler decisions for each individual virtual node and link.
- For each virtual node present in a network request, the RL agent evaluates potential substrate nodes and selects the most suitable one for mapping. This decision process considers various factors, such as resource availability, current node loads, and the overall network topology, thus enabling more efficient utilization of the underlying physical infrastructure.

By structuring the action space in this manner, the RL model can effectively navigate and optimize the embedding process, significantly enhancing the agent's ability to respond to diverse and dynamic virtual network requests.

3. Reward Function

The reward function plays a critical role in guiding the learning process of the RL agent by providing feedback on the quality of its embedding actions [25]. This function is meticulously designed to encapsulate the objectives of the VNE problem:

- The reward function consists of both positive and negative rewards. Positive rewards are assigned when the agent successfully maps virtual nodes and links to substrate resources while adhering to the constraints of resource availability and network policies [26]. This encourages the agent to pursue actions that lead to effective and efficient embeddings.
- Conversely, negative rewards are imposed when the agent's actions result in resource violations, such as attempting to map a virtual node to a substrate node that is already over-committed or failing to allocate sufficient bandwidth for a virtual link [27]. By penalizing such actions, the reward function fosters a preference for resource-efficient strategies that align with the overarching goals of maximizing revenue and minimizing energy consumption.

The careful calibration of the reward function is crucial, as it balances the competing objectives within the VNE framework [28]. The RL agent learns to optimize its decision-making over time by maximizing cumulative rewards, ultimately achieving effective embedding strategies that enhance both performance and sustainability within the network [29].

Workflow

The workflow of the PPO-VNE approach is designed to achieve efficient Virtual Network Embedding (VNE) by dynamically coordinating the mapping of virtual nodes and links onto the substrate network. The goal of this workflow is to allow the RL agent to make adaptive, real-time decisions that maximize resource efficiency while minimizing energy consumption. Below is an in-depth outline of each step in this workflow:

1. **Feature Extraction:**

- Feature extraction is critical for providing the RL agent with a detailed understanding of both the substrate (physical) network and incoming Virtual Network (VN) requests [30]. The PPO-VNE framework uses a hybrid approach to feature extraction, combining manually selected features with features derived automatically using Graph Convolutional Networks (GCNs).
- **Manually Extracted Features:**
  - Manually extracted features provide essential information on substrate network resources and configurations. These include:
    - **CPU Resources**: Available CPU capacity for each substrate node.
    - **Node Degree**: The connectivity level of each node, indicating how integrated each node is within the network [31].
    - **Bandwidth Availability**: The available bandwidth on each link in the substrate network.
  - These features are computationally efficient and provide a foundation for understanding resource constraints, making them essential for the initial decision-making process.
- **GCN-Extracted Features:**
  - Graph Convolutional Networks (GCNs) are utilized to capture complex structural relationships within the network. GCNs analyze the connectivity patterns and topological structures within the network graph, revealing:
    - **Network Clustering Patterns**: The extent to which nodes form clusters within the network.
    - **Resource Bottlenecks**: Potential areas where resources may become scarce due to high traffic or demand.
  - These deeper, automatically extracted features complement the manually selected features by capturing nuanced patterns and dependencies in the network that may not be apparent from basic metrics alone.
- The combination of manually selected and GCN-derived features gives the RL agent a "hybrid" feature set, empowering it to make well-informed, context-aware embedding decisions that account for both resource metrics and structural patterns.

2. **Action Selection:**

- Action selection refers to the RL agent's decision-making process in mapping virtual nodes and links onto specific substrate nodes and paths. Each action corresponds to a discrete mapping decision, making it possible to handle the VNE problem incrementally.
- For each virtual node in a VN request, the RL agent selects an action that maps it to a specific substrate node [32]. This decision is guided by the agent's current policy, which is continually optimized through the Proximal Policy Optimization (PPO) algorithm.
- The agent evaluates possible substrate nodes based on several factors:
  - **Resource Availability**: Ensures that the chosen substrate node has sufficient resources (e.g., CPU and memory) to meet the requirements of the virtual node.
  - **Load Balancing**: Distributes the load across the network to avoid congestion and improve performance.
  - **Network Topology**: Considers the position of nodes and links in the network to minimize link lengths and maximize efficient bandwidth usage.
- This breakdown of the embedding process into a sequence of manageable actions enables the RL agent to systematically explore a vast action space without becoming overwhelmed by the complexity of the entire VN request.

3. **State Update and Reward Calculation:**

- Following each mapping decision, the state of the network is updated to reflect the new resource allocations and link usage, providing the RL agent with an accurate, up-to-date view of the substrate network [33].

- The agent receives feedback in the form of rewards or penalties based on the success of the action:
  - **Positive Rewards**: Awarded for successful mappings that meet resource and network policy constraints. These rewards encourage the agent to take actions that lead to efficient embeddings and high acceptance rates.
  - **Negative Rewards**: Penalties are applied when actions result in resource violations (e.g., mapping a virtual node to an over-utilized substrate node) or other constraint violations. Negative rewards discourage the agent from inefficient or unsustainable mapping actions.
- This reward function is carefully calibrated to strike a balance between competing objectives, such as maximizing resource efficiency, maintaining high VN acceptance rates, and minimizing energy consumption.
- Over time, the agent learns to optimize its decisions by maximizing cumulative rewards, leading to more efficient VNE strategies that balance multiple network objectives.

4. **Iteration and Policy Optimization:**
   - The Proximal Policy Optimization (PPO) algorithm is used to iteratively refine the policy that guides the agent's actions. The PPO algorithm includes a "clipping" mechanism that prevents abrupt policy changes, ensuring a stable learning process.
   - **Clipping Mechanism**: By constraining the range of policy updates, PPO reduces the risk of drastic behavior shifts in the agent's decision-making [34]. This results in smoother learning, allowing the agent to refine its strategy gradually.
   - **Cumulative Reward Maximization**: Through successive training episodes, the agent learns to maximize cumulative rewards, developing an embedding strategy that balances VN acceptance rates, revenue, and energy efficiency.
   - This iterative optimization process is essential for adapting the RL agent's policy to the evolving conditions of the network environment. As the agent's experience grows, it becomes increasingly adept at embedding virtual networks effectively, even under varying network conditions and resource constraints [35].

Through this workflow, the PPO-VNE approach enables the RL agent to learn and apply efficient VNE strategies in real-time. By dynamically managing the embedding of virtual networks onto the substrate network, the RL agent achieves a balance between resource utilization, energy consumption, and network performance, making it suitable for large-scale, adaptive network environments.

Proposed Methodology

The proposed methodology, named PPO-VNE, introduces a deep reinforcement learning-based approach to efficiently solve the Virtual Network Embedding (VNE) problem. By leveraging the Proximal Policy Optimization (PPO) algorithm, the PPO-VNE framework dynamically coordinates both node and link mapping decisions, aiming to maximize resource utilization and minimize energy consumption. The core components of this methodology are detailed as follows:

1. Reinforcement Learning Framework

The PPO-VNE framework models the VNE problem as a reinforcement learning (RL) task, structured around the concepts of states, actions, and rewards:

- **States**: The state representation captures the current status of both the substrate network and virtual network requests. Key elements of the state include:
  - **Substrate Network State**: Detailed information on substrate nodes and links, such as CPU capacity, node degree, available bandwidth, and extracted structural features from Graph Convolutional Networks (GCNs).

  - **Virtual Network Request State**: Specific requirements of incoming virtual network requests, such as resource demands for each virtual node and link, which enable the agent to assess compatibility with available substrate resources.

  This comprehensive state representation allows the RL agent to make informed embedding decisions by considering both the physical and virtual network conditions.

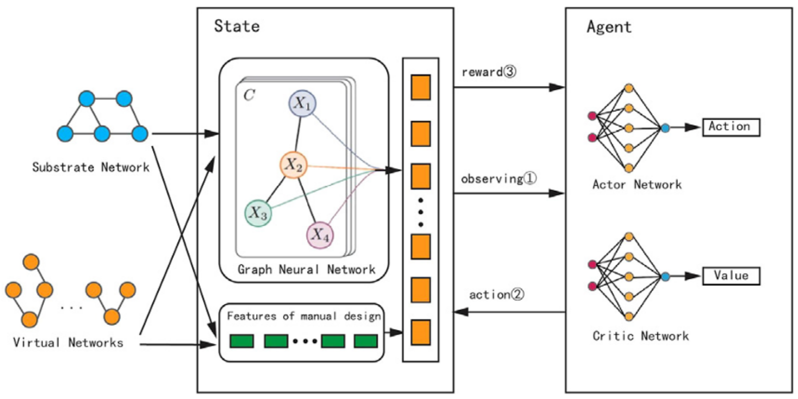

Figure 2: The learning algorithm used by the agent.

- **Actions**: The action space defines the decisions available to the RL agent for embedding virtual network components onto the substrate network. Instead of making a single, large decision, the VNE process is decomposed into sequential actions:
  - Each action involves selecting a specific substrate node for a virtual node or mapping a virtual link to a substrate path. This action space decomposition enables the RL agent to explore possible mappings incrementally, reducing the complexity of the embedding task.
- **Rewards**: The reward function is carefully designed to guide the RL agent's actions toward achieving a balance between maximizing revenue and minimizing energy consumption:
  - Positive rewards are assigned for successful embeddings that comply with resource constraints and optimize resource utilization.
  - Negative rewards are applied for actions that result in resource violations or inefficiencies, discouraging the agent from taking suboptimal or unsustainable actions.

  This reward structure drives the RL agent to focus on embedding strategies that are both resource-efficient and energy-conscious, promoting sustainable network operations.

2. Hybrid Feature Extraction

To effectively capture network characteristics and enhance decision-making, the PPO-VNE framework employs a hybrid feature extraction approach, integrating:

- **Manually Engineered Features**: Basic network metrics such as CPU availability, node degree, and bandwidth, which provide foundational insights into the substrate network's resource status.
- **GCN-Derived Features**: Graph Convolutional Networks (GCNs) extract complex structural patterns and dependencies, such as node centrality and clustering tendencies, that offer deeper insights into the network's topology.

This hybrid approach enables the RL agent to leverage both simple resource metrics and intricate network patterns, supporting more context-aware and effective embedding decisions.

3. Multi-Objective Optimization

Given the multi-faceted goals of VNE, the PPO-VNE framework incorporates a custom reward function to achieve a balanced trade-off between competing objectives:

- The reward function considers metrics like **Acceptance Rate** (proportion of successful VN embeddings), **Revenue** (financial gains from resource utilization), and **Energy Efficiency** (minimizing resource consumption).
- By weighting these factors, the RL agent is guided to prioritize embedding actions that maximize network performance while minimizing energy consumption, aligning with sustainable network management goals.

4. Policy Optimization via PPO

The Proximal Policy Optimization (PPO) algorithm is used to train the RL agent's policy. PPO offers stable learning and adaptability in dynamic environments, with key benefits:

- **Policy Clipping Mechanism**: PPO's clipping mechanism constrains the degree of policy updates, preventing drastic policy shifts that could destabilize learning. This ensures controlled, incremental improvements in the policy network.
- **Adaptation to Network Changes**: By iteratively optimizing the policy based on cumulative rewards, the RL agent refines its embedding strategy over time, improving its ability to handle fluctuating network conditions and varying VN requests.

Through the PPO algorithm, the RL agent achieves an optimal policy for VNE, balancing efficiency and resource conservation, thus enabling robust performance in large-scale, real-time network environments. This proposed methodology provides a comprehensive solution to the VNE problem, leveraging the strengths of deep reinforcement learning, graph-based feature extraction, and multi-objective optimization. The PPO-VNE framework demonstrates an ability to dynamically adapt to network conditions, efficiently embedding virtual networks with minimal resource usage and energy consumption, making it highly suitable for modern network infrastructure.

Assumptions

To effectively model and address the Virtual Network Embedding (VNE) problem, the proposed PPO-VNE approach is based on the following assumptions:

- **Network Representation**: Both the substrate (physical) network and virtual networks are represented as weighted, undirected graphs:
  - **Substrate Nodes** possess specific CPU capacities, reflecting the processing resources available at each physical node.
  - **Substrate Links** have designated bandwidth capacities, indicating the maximum data transfer rate between connected nodes.
- **Virtual Network Request (VNR) Arrival Process**: Virtual network requests are assumed to arrive according to a Poisson process, simulating real-time demand in a network environment [36]. Each request has a predefined duration (or lifetime) after which the resources allocated to it are released.
- **Mapping of Virtual Nodes and Links**:
  - Each virtual node within a VNR is mapped to a substrate node with sufficient resources to meet its CPU requirements.

  - Virtual links are mapped onto substrate paths based on available bandwidth and shortest-path criteria, optimizing for minimal path length while ensuring resource constraints are met.

- **Agent's Access to Network State Information**: The Reinforcement Learning (RL) agent has access to detailed state information for both the substrate network and virtual networks:
  - This includes both manually crafted features (e.g., CPU capacity, bandwidth availability) and Graph Convolutional Network (GCN)-derived features [37], enabling the agent to make informed, context-aware decisions.

- **Action Constraints and Embeddability**: Maximum allowable actions per mapping task are predefined to maintain computational efficiency:
  - If the RL agent exceeds these predefined limits during the embedding of a VN request, the request is marked as unembeddable and is rejected [38].

These assumptions establish a structured environment that enables the PPO-VNE model to operate effectively in dynamic, large-scale networks, balancing resource demands with computational constraints for optimal performance [39].

Implementation

The implementation of the PPO-VNE framework leverages deep reinforcement learning to address the Virtual Network Embedding (VNE) problem effectively. Key components of the implementation are outlined below:

1. Simulation Environment Setup

The PPO-VNE framework is evaluated within a simulated network environment designed to mimic real-world network conditions:

- The **substrate network** is constructed with 100 nodes and approximately 500 links, where the links are created with a probability of 0.1 between any two nodes.
- Each node in the substrate network is assigned CPU capacities, and each link has specific bandwidth capacities. These resource attributes are drawn from uniform distributions to introduce variation.

2. RL Agent and Feature Extraction

- The RL agent employs a **hybrid feature extraction** approach, combining manually designed features (e.g., CPU, bandwidth, degree vectors) with features derived using Graph Convolutional Networks (GCNs). This enables the agent to capture both basic resource metrics and complex structural patterns in the network.
- For each virtual node in a VNR, the RL agent selects an action that maps it to a suitable substrate node. This decision is guided by the Proximal Policy Optimization (PPO) algorithm, which considers resource availability, network topology, and load distribution.

3. Training Process and Reward Mechanism

The RL agent is trained with a reward function that balances objectives, including resource efficiency, energy consumption, and acceptance rate:

- Training is performed iteratively, with the PPO algorithm optimizing the policy using a clipped objective function, which stabilizes learning by limiting abrupt policy updates.
- Over time, the agent refines its policy to maximize cumulative rewards, evolving an optimal strategy for the VNE problem.

Results and Analysis

Key performance metrics used in the evaluation are described below:

1. Acceptance Rate

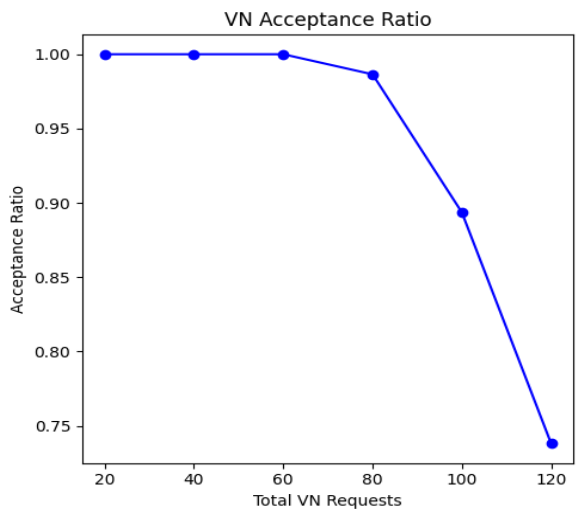

Figure 3: VN Acceptance Ratio

- The PPO-VNE framework achieved a high acceptance rate, especially at lower VNR volumes, demonstrating its capacity to embed VNRs efficiently within available resource constraints.
- For example, with 60 VNRs, PPO-VNE achieved a 100% acceptance rate.%.

2. Overall Revenue

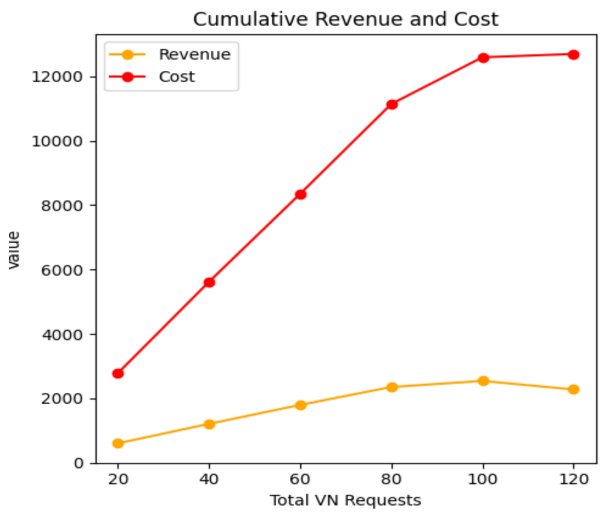

Figure 4: Overall Revenue

- PPO-VNE produced a high overall revenue , owing to its effective utilization of substrate resources and adaptive embedding strategies.
- Revenue gains were consistent across various VNR volumes, underscoring PPO-VNE's suitability for high-demand network environments.

3. Revenue-to-Cost Ratio

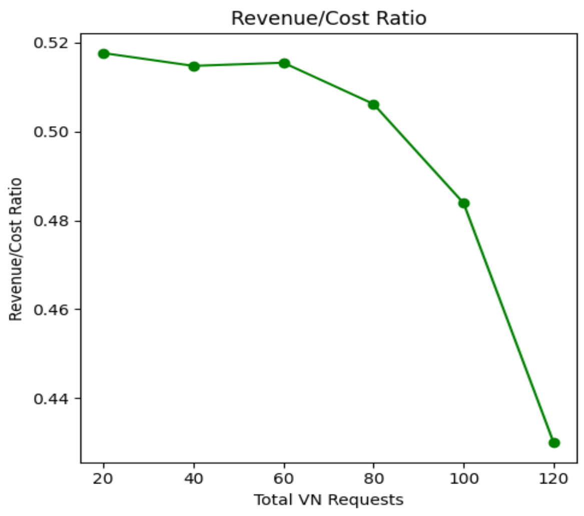

Figure 5: Revence to Cost ratio

- The framework maintained a high revenue-to-cost ratio. This result indicates the framework's efficiency in generating revenue relative to its resource costs.

4. Maximum Unit Time Energy Consumption

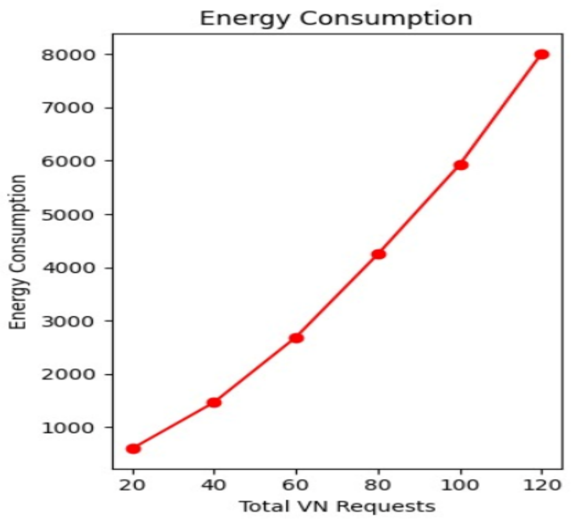

Figure 6: Energy Consumption

- Despite achieving higher revenue, PPO-VNE demonstrated lower maximum energy consumption per unit time.

- This result illustrates PPO-VNE's capability to manage energy resources efficiently, which is critical for sustainable network operations.

5. Summary of Results

Overall, PPO-VNE performed well across most metrics, demonstrating robustness and scalability. Its ability to handle varying network conditions with high acceptance rates and low energy consumption suggests that it has strong potential for deployment in dynamic, large-scale network environments.

Conclusion

In this report, the PPO-VNE framework demonstrated a high level of effectiveness in addressing the VNE problem through the use of deep reinforcement learning and the Proximal Policy Optimization (PPO) algorithm. By dynamically coordinating node and link mappings, PPO-VNE achieves a balance between revenue generation and energy consumption. Simulation results indicate that PPO-VNE surpasses traditional VNE approaches across various performance metrics, making it a promising solution for dynamic and large-scale network environments.
Future Work: The PPO-VNE framework offers a foundation for further advancements, including exploring more refined multi-objective optimization techniques and expanding the framework's application to emerging areas such as IoT and cloud-based networks.

References

[1] T. W. Zhonglei Duan, Towards learning-based energy-efficient online coordinated virtual network embedding framework, `https://www.sciencedirect.com/science/article/abs/pii/S1389128623005844` (2024).

[2] S. S. J. T. T. Anderson, L. Peterson, Overcoming the internet impasse through virtualization, Computer 38 (4) (2005) 34–41 (2005).

[3] J. R. M. C. M. Yu, Y. Yi, Rethinking virtual network embedding: Substrate support for path splitting and migration, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 17–29 (2008).

[4] R. B. M.R. Rahman, I. Aib, Survivable virtual network embedding, in: NETWORKING 2010, Springer, 2010, pp. 40–52 (2010).

[5] M. H. T. Wang, B. Qin, An efficient framework for online virtual network embedding in virtualized cloud data centers, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 159–164, `http://dx.doi.org/10.1109/CloudNet.2015.7335299` (2015).

[6] M. B. H. d. M. X. H. A. Fischer, J.F. Botero, Virtual network embedding: A survey, IEEE Commun. Surv. Tutor. 15 (4) (2013) 1888–1906 (2013).

[7] Z. Y. Q. L. W. Fu, Y. Li, Decision making for autonomous driving via multi modal transformer and deep reinforcement learning, in: 2022 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Guiyang, China, 2022, pp. 481–486, `http://dx.doi.org/10.1109/RCAR54675.2022.9872180` (2022).

[8] H. B. N.E. Houda Ouamane, Deep reinforcement learning applied to nlp: A brief survey, in: 2022 2nd International Conference on New Technologies of Information and Communication (NTIC), Mila, Algeria, 2022, pp. 1–5, `http://dx.doi.org/10.1109/NTIC55069.2022.10100477` (2022).

[9] M. H. T. Wang, Presto: Towards efficient online virtual network embedding in virtualized cloud data centers, Comput. Netw. 106 (sep.4) (2016) 196–208 (2016).

[10] N. A. e. a. O. Soualah, I. Fajjari, A reliable virtual network embedding algorithm based on game theory within cloud's backbone, in: 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 2014, pp. 2975–2981, `http://dx.doi.org/10.1109/ICC.2014.6883777` (2014).

[11] S. S. e. a. Z. Zhang, C. Xiang, A unified enhanced particle swarm optimization based virtual network embedding algorithm, Int. J. Commun. Syst. 26 (8) (2013) 1054–1073 (2013).

[12] A. O. P.T.A. Quang, Y. Hadjadj-Aoul, A deep reinforcement learning approach for vnf forwarding graph embedding, IEEE Trans. Netw. Serv. Manag. 16 (4) (2019) 1318–1331, `http://dx.doi.org/10.1109/TNSM.2019.2947905` (2019).

[13] N. K. W. Z. L. L. P. Zhang, C. Wang, Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning, IEEE Internet Things J. 9 (12)

(2022) 9389–9398, http://dx.doi.org/10.1109/JIOT.2021.3095094 (2022).

[14] M. L. P. Z. L. W. H. Yao, X. Chen, A novel reinforcement learning algorithm for virtual network embedding, Neurocomputing 284 (2018) 1–9 (2018).

[15] M. S. M. H. C. F. R. T. E. Amiri, N. Wang, Deep reinforcement learning for robust vnf reconfigurations in o-ran, in: IEEE Transactions on Network and Service Management, http://dx.doi.org/10.1109/TNSM.2023.3316074 (2023).

[16] M. S. R. T. E. Amiri, N. Wang, Energy-aware dynamic vnf splitting in o-ran using deep reinforcement learning, IEEE Wirel. Commun. Lett. 12 (11) (2023) 1891–1895, http://dx.doi.org/10.1109/LWC.2023.3298548 (2023).

[17] R. B. M. Chowdhury, M.R. Rahman, Vineyard: Virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Netw. 20 (1) (2011) 206–219 (2011).

[18] C. H. Q. Lu, K.T. Nguyen, A novel one-stage distributed parallel embedding for virtualized network environment, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2020 (2020).

[19] S. V. R. T. E. Amiri, N. Wang, Dynamic anchor point selection in software defined distributed mobility management, in: 2022 IEEE Symposium on Computers and Communications (ISCC), Rhodes, Greece, 2022, pp. 1–7, http://dx.doi.org/10.1109/ISCC55528.2022.9912766 (2022).

[20] C. L. X. Chen, Energy efficient virtual network embedding for path splitting, in: The 16th Asia-Pacific Network Operations and Management Symposium, Hsinchu, Taiwan, 2014, pp. 1–4, http://dx.doi.org/10.1109/APNOMS.2014.6996550 (2014).

[21] M. W. T.N. Kipf, Semi-supervised classification with graph convolutional networks, 2016, arXiv:1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907 (2016).

[22] P. D. e. a. J. Schulman, F. Wolski, Proximal policy optimization algorithms, 2017, http://dx.doi.org/10.48550/arXiv.1707.06347 (2017).

[23] M. L. M. Li, A virtual network embedding algorithm based on double-layer reinforcement learning, Comput. J. 64 (1) (2021) 973–989, http://dx.doi.org/10.1093/comjnl/bxab040 (2021).

[24] e. a. T. Wang, Drl-sfcp: Adaptive service function chains placement with deep reinforcement learning, in: ICC 2021- IEEE International Conference on Communications, Montreal, QC, Canada, 2021, pp. 1–6, http://dx.doi.org/10.1109/ICC42927.2021.9500964 (2021).

[25] M. L. L. H. J. W. X. H. W. Fan, F. Xiao, Node essentiality assessment and distributed collaborative virtual network embedding in datacenters, IEEE Trans. Parallel Distrib. Syst. 34 (4) (2023) 1265–1280, http://dx.doi.org/10.1109/TPDS.2023.3242952 (2023).

[26] L. Y. H. Cao, H. Zhu, Collaborative attributes and resources for single-stage virtual network mapping in network virtualization, J. Commun. Netw. 22 (1) (2020) 61–71, http://dx.doi.org/10.1109/JCN.2019.000045 (2020).

[27] Q. L. I. Sutskever, O. Vinyals, Sequence to sequence learning with neural networks, Adv. Neural Inf. Process. Syst. 27 (2014) (2014).

[28] M. M. A. G. T. L. T. H. D. S. K. K. V. Mnih, A.P. Badia, Asynchronous methods for deep reinforcement learning, in: Proc. ICML, 2016, pp. 1928–1937 (2016).

[29] H. P. V. V. Q. N. P. Rahimi, C. Chrysostomou, Joint radio resource allocation and beamforming optimization for industrial internet of things in software-defined networking-based virtual fog-radio access network 5g-and-beyond wireless environments, IEEE Trans. Ind. Inform. 18 (6) (2022) 4198–4209, http://dx.doi.org/10.1109/TII.2021.3126813 (2022).

[30] e. a. L. Liu, Scaleflux: Efficient stateful scaling in nfv, IEEE Trans. Parallel Distrib. Syst. 33 (12) (2022) 4801–4817, http://dx.doi.org/10.1109/TPDS.2022.3204209 (2022).

[31] A. M. A. E. N. N. M. G. A. Awad Abdellatif, A. Abo-Eleneen, Intelligent-slicing: An ai-assisted network slicing framework for 5g-and-beyond networks, IEEE Trans. Netw. Serv. Manag. 20 (2) (2023) 1024–1039, http://dx.doi.org/10.1109/TNSM.2023.3274236 (2023).

[32] A. B. R. N. H. Babbar, S. Rani, Lbsmt: Load balancing switch migration algorithm for cooperative communication intelligent transportation systems, IEEE Trans. Green Commun. Netw. 6 (3) (2022) 1386–1395, http://dx.doi.org/10.1109/TGCN.2022.3162237 (2022).

[33] K. R. M. A. T. W. X. F. S.G. Kulkarni, G. Liu, Reinforce: Achieving efficient failure resiliency for network function virtualization-based services, IEEE/ACM Trans. Netw. 28 (2) (2020) 695–708, http://dx.doi.org/10.1109/TNET.2020.2969961 (2020).

[34] H. Z. J. Chen, J. Chen, Drl-qor: Deep reinforcement learning-based qos/qoe-aware adaptive online orchestration in nfv-enabled networks, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1758–1774, http://dx.doi.org/10.1109/TNSM.2021.3055494 (2021).

[35] e. a. Y. Zeng, Ruledrl: Reliability-aware sfc provisioning with bounded approximations in dynamic

environments, IEEE Trans. Serv. Comput. 16 (5) (2023) 3651–3664, http://dx.doi.org/10.1109/TSC.2023.3281759 (2023).

[36] e. a. C. Fang, Deep-reinforcement-learning-based resource allocation for content distribution in fog radio access networks, IEEE Internet Things J. 9 (18) (2022) 16874–16883, http://dx.doi.org/10.1109/JIOT.2022.3146239 (2022).

[37] K. R. R. B. S. Tuli, S. Ilager, Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks, IEEE Trans. Mob. Comput. 21 (3) (2022) 940–954, http://dx.doi.org/10.1109/TMC.2020.3017079 (2022).

[38] S. K. V. M. I. Sarkar, M. Adhikari, Deep reinforcement learning for intelligent service provisioning in software-defined industrial fog networks, IEEE Internet Things J. 9 (18) (2022) 16953–16961, http://dx.doi.org/10.1109/JIOT.2022.3142079 (2022).

[39] J. W. Y. L. H. Z. W. Yang, Y. Li, An efficient approach for network function virtualization using deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. 19 (2) (2022) 1086–1100, http://dx.doi.org/10.1109/TNSM.2022.3160897 (2022).