

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 23 October 2022

I. Ullah
Y-H. Han
KOREATECH
TY. Kim
ETRI
21 April 2022

Reinforcement Learning-Based Virtual Network Embedding: Problem
Statement
draft-ihsan-nmrg-rl-vne-ps-02

Abstract

In Network virtualization (NV) technology, Virtual Network Embedding (VNE) is an algorithm used to map a virtual network to the substrate network. VNE is the core orientation of NV which has a great impact on the performance of virtual network and resource utilization of the substrate network. An efficient embedding algorithm can maximize the acceptance ratio of virtual networks to increase the revenue for Internet service provider. Several works have been appeared on the design of VNE solutions, however, it has becomes a challenging issues for researchers. To solved the VNE problem, we believe that reinforcement learning (RL) can play a vital role to make the VNE algorithm more intelligent and efficient. Moreover, RL has been merged with deep learning techniques to develop adaptive models with effective strategies for various complex problems. In RL, agents can learn desired behaviors (e.g, optimal VNE strategies), and after learning and completing training, it can embed the virtual network to the subtract network very quickly and efficiently. RL can reduce the complexity of the VNE algorithm, however, it is too difficult to apply RL techniques directly to VNE problems and need more research study. In this document, we presenting a problem statement to motivate the researchers toward the VNE problem using deep reinforcement learning.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 23 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction and Scope	2
2. Reinforcement Learning-based VNE Solutions	5
3. Terminology	8
4. Problem Space	9
4.1. State Representation	9
4.2. Action Space	9
4.3. Reward Description	10
4.4. Policy and RL Algorithms	11
4.5. Training Environment	12
4.6. Sim2Real Gap	13
4.7. Generalization	14
5. IANA Considerations	14
6. Security Considerations	14
7. Informative References	14
Authors' Addresses	18

1. Introduction and Scope

Recently, Network virtualization (NV) technology has received a lot of attention from academics and industry. It allows multiple heterogeneous virtual networks to share resources on the same substrate network (SN) [RFC7364], [ASNVT2020]. The current large-size fixed substrate network architecture is no longer efficient and not extendable due to network ossification. To overcome this limitations, traditional Internet Service Providers (ISPs) are

divided into two independent parts which work together. One is the Service Providers (SPs) who create and own the different number of the VNs, and the other one is the Infrastructure Providers (InPs) who own the SN devices and links as underlying resources. SPs generate and construct the customized Virtual Network Requests (VNRs), and lease the resources from InPs based on that requests. In addition, two types of mediators can enter into the industry domain for better coordination of SPs and InPs. One is the Virtual Network Providers (VNPs) who assemble and coordinate diverse virtual resources from one or more InPs, the other one is the Virtual Network Operators (VNOs) who create, manage, and operate the VN according to the demand of the SPs. VNPs and VNOs could enable efficient use of the physical network and increase the commercial revenue of both SPs and InPs. NV can increase network agility, flexibility and scalability while creating significant cost savings. Greater network workload mobility, increased availability of network resources with good performance, and automated operations, are all the benefits of NV.

Virtual Network Embedding (VNE) [VNESURV2013] is one of the main technique and strategy which used to map a virtual network to the substrate network. VNE algorithm has two main parts, Node embedding: where virtual nodes of VN have to be mapped to the SN nodes, and Link embedding: where virtual links between the VNs have to be mapped to the physical paths in the substrate network. It has been proven to be NP-Hard, and both node and link embeddings have become challenging for the researchers. A virtual node and link should be efficiently embedded into a given SN, so that more VNR can be accepted with minimum cost. The distance of the virtual nodes from each other in a given SN is a big contribution to the link failures and causes the rejection of VNRs. Hence, an efficient and intelligent technique is required for VNE problem to reduce VNRs rejection [ENViNE2021]. In the perspective of the InPs, the efficient VNE performs better mostly in terms of revenue, acceptance ratio, and revenue-to-cost ratio.

Figure 1 shows the the example of two virtual network request VNR1 and VNR2 to embed them in the given substrate network. VNR1 contain three virtual nodes (a, b, and c) with cpu demands (15, 30, and 10) respectively, and the link between virtual the nodes a-b, b-c, and c-a with bandwidth demands 15, 20, and 35 respectively. Similarly, VNR2 contains virtual nodes and links with cpu and bandwidth demand respectively. The purpose of the VNE algorithm to map the virtual nodes and links of the VNRs to the physical nodes and links of the given substrate as shown in Figure 1. [ENViNE2021].

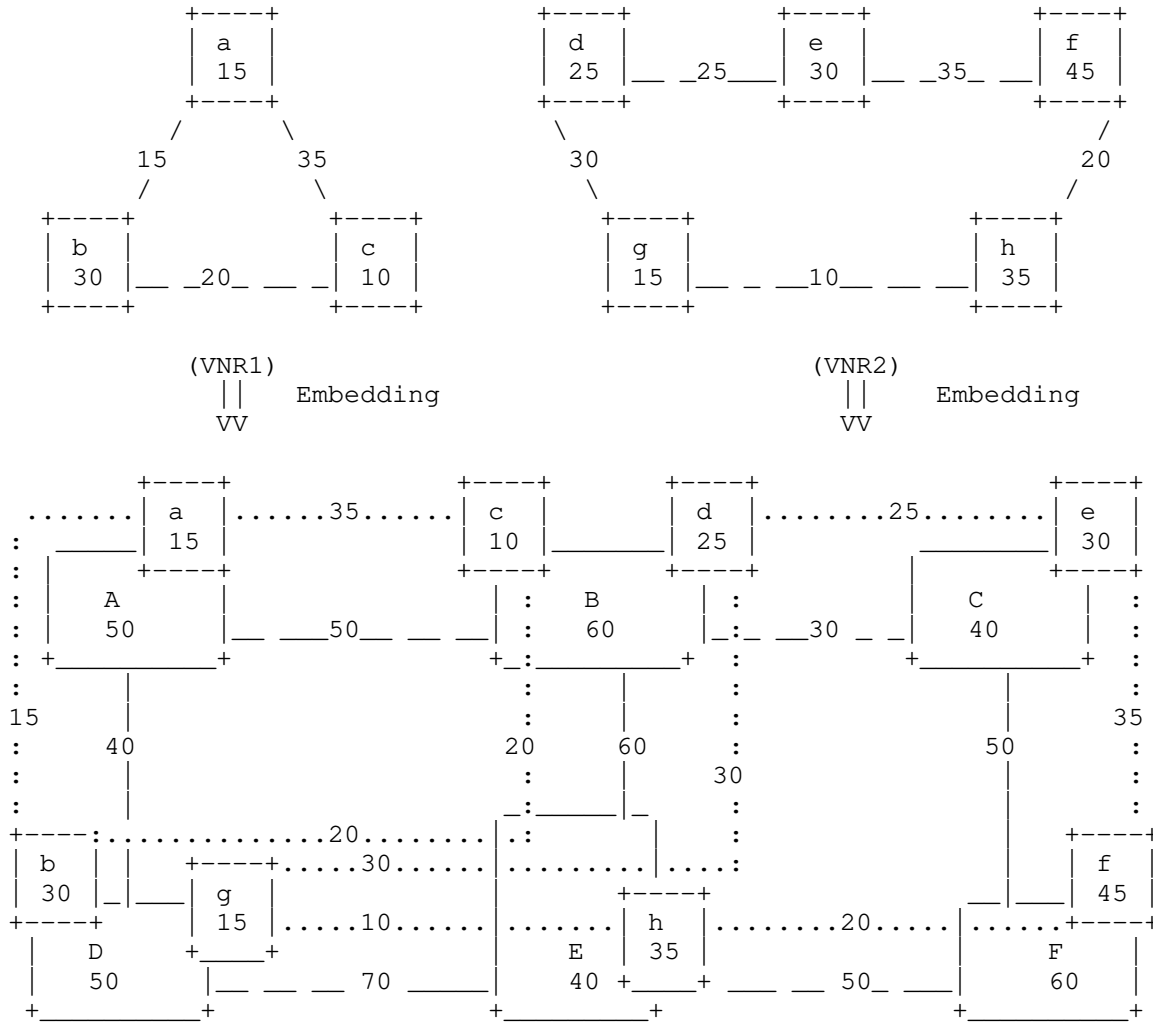


Figure 1: Substrate network with embedded virtual network, VNR1 and VNR2

Recently, artificial intelligence and machine learning technologies have been widely used to solve networking problems [SUR2018], [MLCNM2018], [MVNNML2021]. There has been a surge in research efforts, specially, reinforcement learning (RL) which has been contributed much more in the many complex tasks, e.g. video games and auto-driving etc. The main goal of an RL to learn better policies for sequential decision making problems (e.g., VNE) and solve them very efficiently.

Problems such as node classification, pattern matching, and network feature extraction, can be simplified by graph-related theories and techniques. Graph neural network (GNN) is a new type of ML model architecture that can aggregate graph features (degrees, distance to specific nodes, node connectivity, etc.) on nodes [DVNEGCN2021]. Graph convolution neural network (GCNN) is a natural generalization form of GNN which is used to automatically extract the features of underlying network, which optimizes the selection of VNE decision. The model can be used to cluster nodes and links according to the physical nodes and physical links attribute characteristics (CPU, storage, bandwidth, delay, etc.), and it is highly suitable for graph structures of any topological form. Hence, GNN is useful to find the best VNE strategy by intelligent agent training, and the organic combination of VNE and GCN has a good prerequisite.

Designing and applying RL techniques directly into VNE problems is not yet trivial, but may face several challenges. Several works have been appeared on the design of VNE solutions using RL, which focuses on how to interact with the environment to achieve maximum cumulative return [VNEQS2021], [NRRL2020], [MVNE2020], [CDVNE2020], [PPRL2020], [RLVNEWSN2020], [QLDC2019], [VNFFG2020], [VNEGCN2020], [NFVDeep2019], [DeepViNE2019], [VNETD2019], [RDAM2018], [MOQL2018], [ZTORCH2018], [NeuroViNE2018], [QVNE2020]. This document outlines the problems encountered when designing and applying RL-based VNE solutions. Section 2 describes how to design RL-based VNE solutions. Section 3 gives terminology, and Section 4 describes the problem space details.

2. Reinforcement Learning-based VNE Solutions

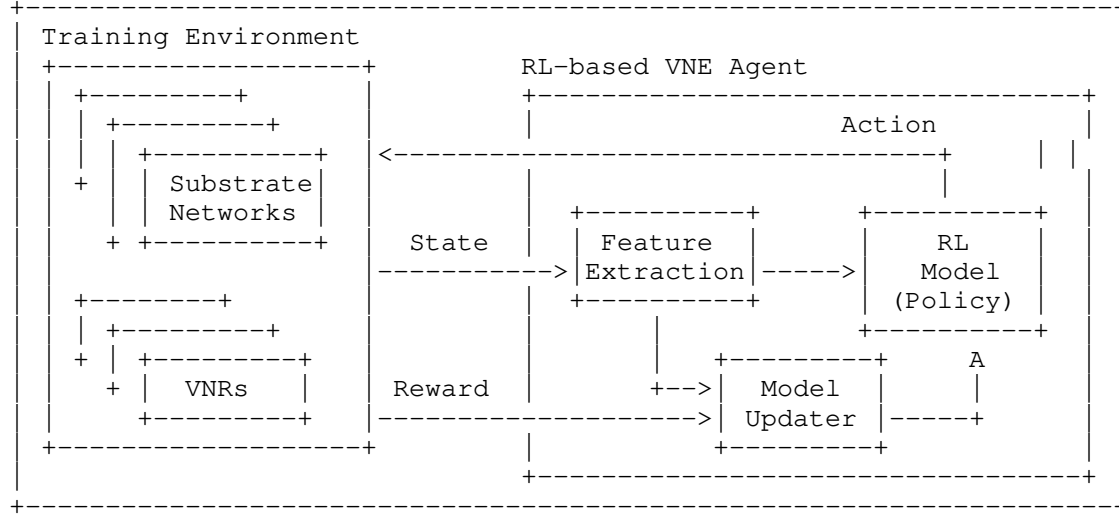
As we discussed that RL has been studied in various fields (such as game, control system, operation research, information theory, multi-agent system, network system, etc.) and shows better performance than humans. Unlike deep learning, RL trains a policy model by receiving rewards through interaction with the environment without training label data.

Recently, there have been several attempts to solve VNE problems using RL. When applying RL-based algorithms to solve VNE problems, the RL agent automatically learns through the environment without human intervention. Once the agent completed the learning process, it can generate the most appropriate embeddings decision (action) based on the his knowledge and network state. For single embedding or action at each time step the agent get reward from the environments to adaptively train its policy for future action. The RL agent gets the most optimized model based on the reward function defined according to each objective (revenue, cost, revenue to cost ratio and acceptance ratio). The optimal RL policy model provides the VNE strategy appropriately according to the objective of the network operator.

Figure 2. shows the virtual network embedding solution based on RL algorithm. The RL strategy is divided into two main parts training process and an inference process. In the training process, state information is composed of various substrate networks and VNRS (Environment), which are used as suitable inputs for RL models through feature extraction. After that, the RL model is updated by model updater using a feature extracted state and reward. In the inference process, using the trained RL model, the embedding result is provided to the operating network in real time.

The following figure shows the detail about RL method based virtual networks embedding solutions.

RL Model Training Process



Inference Process

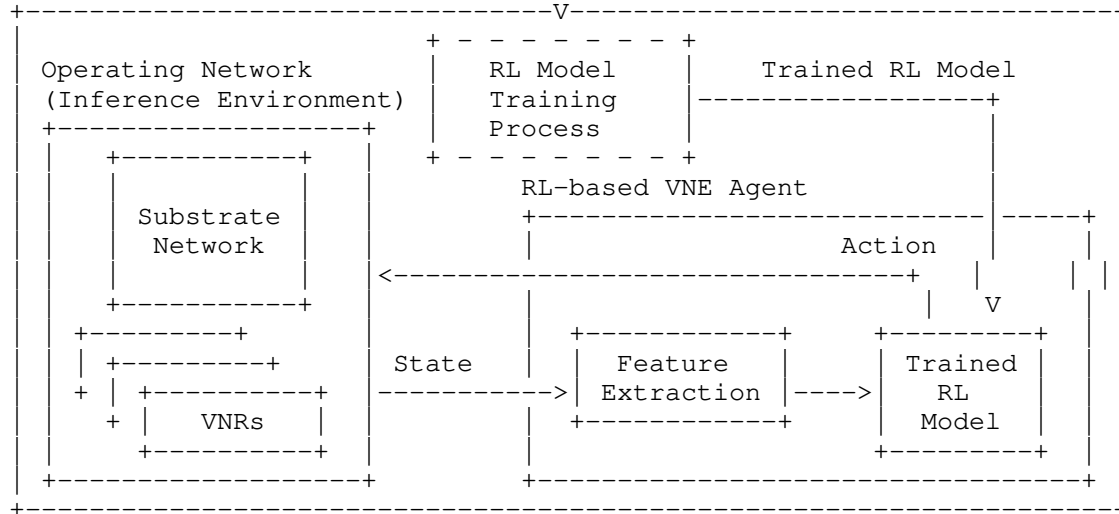


Figure 2: Two processes for RL method based VNE

3. Terminology

Network Virtualization

Network virtualization is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a virtual network [RFC7364].

Virtual Network Embedding (VNE)

Virtual Network Embedding (VNE) [VNESURV2013] is one of the main techniques used to map a virtual network to the substrate network.

Substrate Network (SN)

The underlying physical network which contains the resources such as CPU and bandwidth for virtual networks is called substrate network.

Virtual Network Request (VNR)

Virtual Network Request is a complete single Virtual network containing virtual nodes and virtual links.

Agent

In RL, an agent is the component that makes the decision and take action (i.e., embedding decision).

State

State is a representation (e.g., remaining SN capacity and requested VN resource) of the current environment, and it tells the agent what situation it is in currently.

Action

Actions (i.e., node and link embedding) are behavior an RL agent can do to change the states of the environment.

Policy

A policy defines an agent's way of behaving at a given time. It is a mapping from perceived states of environment to actions to be taken when in those states. It is usually implemented as a deep learning model because the state and action spaces are too large to be completely known.

Reward

A reward is the feedback which provides an agent to the agent for taking actions that lead to good outcomes (i.g., achieve the objective of the network operator).

Environment

An environment is the agent's world in which it lives and interacts. The agent can interact with the environment by performing some action but cannot influence the rules of the environment by those actions.

4. Problem Space

RL contains three main components: state representation, action space, and reward description. For solving a VNE problem, we need to consider how to design the three main RL components. In addition, a specific RL algorithm, training environment, sim2real gap, and generalization are also important issues that should be considered and addressed. We will describe each one in detail as follows.

4.1. State Representation

The way to understand and observe the VNE problem is crucial for an RL agent to establish a thorough knowledge of the network status and generate efficient embedding decisions. Therefore, it is essential to firstly design the state representation that serves as the input to the agent. The state representation is the information which an agent can receive from the environment, and consists of a set of values representing the current situation in the environment. Based on the state representation, the RL agent selects the most appropriate action through its policy model. In the VNE problem, an RL agent needs to know the information of the overall SN entities and their current status in order to use the resources of the nodes and links of the substrate network. Also it must know the requirements of the VNR. Therefore, in the VNE problem, the state usually should represent the current resource state of the nodes and links of the substrate network (ie, CPU, memory, storage, bandwidth, delay, loss rate, etc.) and the requirements of the virtual node and link of the VNR. The collected status information is used as raw input, or refined status information through the feature extraction process is used as input for the RL agent. The state representation may vary depending on the operator's objective and VNE strategy. The method of determining such feature extraction and representation greatly affects the performance of the agent.

4.2. Action Space

In RL, an action represents a decision that an RL agent can take based on current state representation. The set of all possible actions is called an action space. In the VNE problems, actions are generally divided into node embedding and link embedding. The action for node embedding means the VNR's nodes are assigned to which nodes in the SN. Also, for link embedding, the action represents the

selected paths between the selected substrate network nodes from the node embedding result. If the policy model of the RL agent is well trained, it will select the embedding result to maximize the reward appropriate for the operator's objectives. The output actions generated from the agent will indicate the adjustment of allocated resources. It is noted that, at each point of time step, an RL algorithm may decide to 1) embed each virtual node onto substrate nodes and then embed each virtual link onto substrate paths separately, or 2) embed the given whole VNR onto substrate nodes and links in the SN at once. In the former case, at every single step, a learning agent focuses on exactly one virtual node from the current VNR, and it generates a certain substrate node to host the virtual node. Link embedding is then performed separately in the same time step. To solve the VNE problem efficiently, mapping of virtual nodes and links are considered together, although they are performed separately. Link mapping is considering more complex than node mapping, because a virtual link can be mapped onto a physical path with different hops. On the other hand, at every single step, a learning agent can try to embed the given whole VNR, i.e., all virtual nodes and links in the given VNR, onto a subset of SN components. The whole VNR embedding should be handled as a graph embedding, so that the action space is huge and the design of the RL algorithm is usually more difficult than the one with each node and link embedding.

4.3. Reward Description

Designing rewards is an important issue for an RL algorithm. In general, the reward is the benefit that an RL agent follows when performing its determined action. Reward is an immediate value that evaluates only the current state and action. The value of reward depends on success or failure of each step. In order to select the action that gives the best results in the long run, an RL agent needs to select the action with the highest cumulative reward. The reward is calculated through the reward function according to the objective of the environment, and even in the same environment, it may be different depending on the operator's objective. Based on the given reward the agent can evaluate the effectiveness to improve the policy. Hence, the reward function play a important rules in the training process of RL. In the VNE problem, the overall objectives are to reduce the VNE rejection, embed them with minimum cost, maximize the revenue, and increase the resource utilization of physical resources. Reward function should be designed to achieve one or multiple ones of these objectives. Each objective and its correspondent reward design are outlined as follows:

Revenue

Revenue is the sum of the virtual resources requested by the VN, and calculated to determine the total cost of the resources. Typically, a successful action (e.g., VNR is embedded without violation) is treated to be a good reward which also increases the revenue. Otherwise, a failed action (e.g., VNR is rejected) leads that the agent will receive a negative reward as well as decreasing the revenue.

Cost

Cost is the expenditure incurred when VNR is embedded as a substrate network. It's not a good embedding result to pursue only high revenue. It is important for the network operator and SP to spend less. The lower the cost, the better the agent will be rewarded.

Acceptance Ratio

Acceptance ratio is the ratio measured by the number of successfully embedded virtual network requests divided by total number of virtual network requests. To achieve a high acceptance ratio, the agent is trying to embed maximum VNR and get a good reward. Getting a good reward is usually proportional to the acceptance ratio.

Revenue-to-cost ratio

To balance and compare the cost of resources for embedding VNR, the revenue is divided by cost. Revenue-to-cost ratio compares the embedding algorithms with respect to their embedding results in terms of the cost and revenue. Since most VNOs are most interested in this objective, a reward function should be made to relate to this performance metric.

4.4. Policy and RL Algorithms

The policy is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions that promise the highest reward. Therefore, an RL agent updates its policy repeatedly in the learning phase to maximize the expected cumulative reward. Unlike supervised learning, in which each sample has a corresponding label indicating the preferred output of the learning model, an RL agent relies on reward signals to evaluate the effectiveness of actions and further improve the policy. From the perspective of RL, the goal of VNE is to find an optimal policy to embed an VNR onto the given SN in any state at any time. There are two types of RL algorithms: on-policy and off-policy. In on-policy RL algorithms, the (behaviour) policy of the exploration step to select an action and the policy to learn are the same. On-policy algorithms work with a single policy, and require any observations

(state, action, reward, next state) to have been generated using that policy. Representative on-policy algorithms include A2C, A3C, TRPO, and PPO. On the other hand, off-policy RL algorithms work with two policies. These are a policy being learned, called the target policy, and the policy being followed that generates the observations, called the behaviour policy. In off-policy RL algorithms, the learning policy and the behaviour policy are not necessarily the same. It allows the use of exploratory policies for collecting the experience, since learning and behavior policies are separated. In the VNE problem, various experiences can be accumulated by extracting embedding results using various behavior policies. Representative off-policy algorithms include Q-learning, DQN, DDPG, and SAC. There are different classifications for RL algorithms: model-based and model-free. In model-based RL algorithms, an RL agent learns its optimal behavior indirectly by learning a model of the environment by taking actions and observing the outcomes that include the next state and the immediate reward. The models predict the outcomes of actions. The model is used instead of the environment or in addition to interaction with it to learn optimal policies. This becomes, however, impractical when the state and action space is large. Unlike model-based algorithms, model-free RL algorithms learn directly by trial and error with the environment and do not require the relatively large memory. Since data efficiency or safety is very important even in VNE problems, the use of model-based algorithms can be actively considered. However, since it is not easy to build a good model that mimics a real network environment, a model-free RL algorithm may be more suitable for VNE problems. In conclusion, a good RL algorithm selection plays an important role in solving the VNE problem, and VNE performance metrics vary depending on the selected RL algorithm.

4.5. Training Environment

Simulation is the use of software to simulate an interacting environment that is difficult to actually execute and test. An RL algorithm learns by iteratively interacting with the environment. However, in the real environment, various variables such as failure and component consumption exist. Therefore, it is necessary to learn through a simulation that simulates the real environment. In order to solve the VNE problem, we need to use a network simulator similar to the real environment because it is difficult to repeatedly experiment with real network environments using an RL algorithm, and it is very challenging and overwhelming to directly apply an RL algorithm to real-world environments. When solving VNE problems, a network simulation environment similar to a real network is required. The network simulation environment should have a general SN environment and VNR required by the operator. The SN has nodes and links between nodes, and each has capacity such as CPU and Bandwidth.

In the case of VNR, there are virtual nodes and links required by the operator, and each must have its own requirements.

As described in [DTwin2022], a digital twin network is a virtual representation of the physical network environment and can be built by applying digital twin technologies to the environment and creating virtual images of diverse physical network facilities. The digital twin for networks is an expansion platform of network simulation. In [DTwin2022], Section 8.2 describes that a digital twin network provides the complete machine learning lifecycle development by providing a realistic network environment, including network topologies, etc. Hence, RL algorithms to solve the VNE problem can be trained and verified on a digital twin network upfront before deployed to the physical networks, and the verification accuracy will be generally high when the digital twin network reproduces network behaviors well under various conditions. On the other hand, two placeholders marked as [DTwin2022] in the above new paragraph should be replaced with the right reference number after inserting the following new Internet-Draft, which introduces the definition, architecture, and use-cases of digital twin network, into "Section 7. Informative References" of our Internet Draft.

4.6. Sim2Real Gap

Sim-to-real is a very comprehensive concept and applied in many fields including robotics and classic machine vision tasks. An RL algorithm iteratively learns through a simulation environment to train a model of the desired policy. The trained model is then applied to the real environment and/or tuned more for adapting to the real one. However, when the trained model is applied in the simulation to the real environment, sim2real gap problem arises. Closing the gap between simulation and reality gap in terms of actuation requires simulators to be more accurate, and to account for variability in agent dynamics. Obviously, the simulation environment does not match perfectly to the real environment which mostly fails in the tuning process and gives poor performance in the model because of the Sim2Real gap. The sim2real gap is caused by the difference between the simulation and the real environment. It is because the simulation environment cannot perfectly simulate the real environment, and there are many variables in the real environment. In a real network environment for VNE, the SN's nodes and links may fail due to external factors, or capacity such as CPU may change suddenly. In order to solve this problem, the simulation environment should be more robust or the trained RL model should be generalized. To reduce the gap between sim and real network environments we need to train our model with an efficient and large number of VNR and keep learning the agent not only depend on previous memorization.

4.7. Generalization

Generalization refers to the trained model's ability to adapt properly to previously unseen new observations. An RL algorithm tries to learn a model that optimizes some objective with the purpose of performing well on data that has never been seen by the model during training. In terms of VNE problems, the generalization is a measure of how the agent's policy model performs on predicting unseen VNR. The RL agent not only has to memorize all the previous variance of the VNR but also to learn and explore more possible variance. It is important to have good and efficient training data for VNR with good variance and train the model with all possible VNRs.

5. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

6. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

7. Informative References

[ASNVT2020]

Sharif, Kashif., Li, Fan., Latif, Zohaib., Karim, MM., and Sujit. Biswas, "A Survey of Network Virtualization Techniques for Internet of Things using SND and NFV", DOI 10.1145/3379444, April 2020, <<https://doi.org/10.1145/3379444>>.

[CDVNE2020]

"A Continuous-Decision Virtual Network Embedding Scheme Relying on Reinforcement Learning", DOI 10.1109/TNSM.2020.2971543, February 2020, <<https://ieeexplore.ieee.org/document/8982091>>.

[DeepViNE2019]

Dolati, M., Hassanpour, S. B., Ghaderi, M., and A. Khonsari, "DeepViNE: Virtual Network Embedding with Deep Reinforcement Learning", BCP 72, RFC 3552, DOI 10.1109/INFCOMW.2019.8845171, September 2019, <<https://ieeexplore.ieee.org/document/8845171>>.

[DTwin2022]

Yang, H., Zhou, C., Duan, X., Lopez, D., Pastor, A., and Q. Wu, "Digital Twin Network: Concepts and Reference Architecture", DOI <https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/>, March 2022, <<https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/>>.

[DVNEGCN2021]

Zhang, Peiying., Wang, Chao., Kumar, NeeraJ., Zhang,, Weishan., and Lei. Liu, "Dynamic Virtual Network Embedding Algorithm based on Graph Convolution Neural Network and Reinforcement Learning", DOI 10.1109/JIOT.2021.3095094, July 2021, <<https://ieeexplore.ieee.org/document/9475485>>.

[ENViNE2021]

ULLAH, IHSAN., Lim, Hyun-Kyo., and Youn-Hee. Han, "Ego Network-Based Virtual Network Embedding Scheme for Revenue Maximization", DOI 10.1109/ICAIIIC51459.2021.9415185, April 2021, <<https://ieeexplore.ieee.org/document/9415185>>.

[MLCNM2018]

Ayoubi, Sara., Noura, Limam., Salahuddin, Mohammad., Shahriar, Nashid., Boutaba, NRaouf., Estrada-Solano, Felipe., and Oscar. M. Caicedo, "Machine Learning for Cognitive Network Management", DOI 10.1109/MCOM.2018.1700560, January 2018, <<https://ieeexplore.ieee.org/document/8255757>>.

[MOQL2018] "Multi-Objective Virtual Network Embedding Algorithm Based

on Q-learning and Curiosity-Driven", DOI 10.1109/TETC.2018.2871549, June 2018, <<https://jwcn-urasipjournals.springeropen.com/articles/10.1186/s13638-018-1170-x>>.

[MVNE2020] "Modeling on Virtual Network Embedding using Reinforcement

Learning", DOI 10.1002/cpe.6020, September 2020, <<https://doi.org/10.1002/cpe.6020>>.

[MVNNML2021]

Boutaba, Raouf., Shahriar, Nashid., A, Mohammad., and Noura. Limam, "Managing Virtualized Networks and Services with Machine Learning", DOI 48b8fc73c1609d4632d7db5e67e373a62a3cc1f6, January 2021, <<https://www.semanticscholar.org/paper/Managing-Virtualized-Networks-and-Services-with-Boutaba-Shahriar/48b8fc73c1609d4632d7db5e67e373a62a3cc1f6>>.

[NeuroViNE2018]

"NeuroViNE: A Neural Preprocessor for Your Virtual Network Embedding Algorithm", DOI 10.1109/INFOCOM.2018.8486263, June 2018, <<https://ieeexplore.ieee.org/document/8486263>>.

[NFVDeep2019]

Xiao, Y., Zhang, Q., Liu, F., Wang, J., Zhao, M., Zhang, Z., and J. Zhang, "NFVdeep: Adaptive Online Service Function Chain Deployment with Deep Reinforcement Learning", RFC 1129, DOI 10.1145/3326285.3329056, June 2019, <<https://doi.org/10.1145/3326285.3329056>>.

[NRRL2020]

"Network Resource Allocation Strategy Based on Deep Reinforcement Learning", DOI 10.1109/OJCS.2020.3000330, June 2020, <<https://ieeexplore.ieee.org/document/9109671>>.

[PPRL2020]

"A Privacy-Preserving Reinforcement Learning Algorithm for Multi-Domain Virtual Network Embedding", DOI 10.1109/TNSM.2020.2971543, September 2020, <<https://ieeexplore.ieee.org/document/8982091>>.

[QLDC2019]

"A Q-Learning-Based Approach for Virtual Network Embedding in Data Center", DOI 10.1007/s00521-019-04376, July 2019, <<https://link.springer.com/article/10.1007/s00521-019-04376-6>>.

[QVNE2020]

Yuan, Y., Tian, Z., Wang, C., Zheng, F., and Y. Lv, "A Q-learning-Based Approach for Virtual Network Embedding in Data Center", DOI 10.1007/s00521-019-04376-6, July 2020, <<https://link.springer.com/article/10.1007/s00521-019-04376-6>>.

[RDAM2018]

"RDAM: A Reinforcement Learning Based Dynamic Attribute Matrix Representation for Virtual Network Embedding", DOI 10.1109/TETC.2018.2871549, September 2018, <<https://ieeexplore.ieee.org/document/8469054>>.

- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC7364] Thomas, P.T., Eric, Y., David, A., Luyuan, A., Larry, A., and A. Maria Napierala, "Problem Statement: Overlays for Network Virtualization", October 2015, <<https://datatracker.ietf.org/doc/rfc7364/>>.
- [RLVNEWSN2020]
"Reinforcement Learning for Virtual Network Embedding in Wireless Sensor Networks", DOI 10.1109/WiMob50308.2020.9253442, October 2020, <<https://ieeexplore.ieee.org/document/9253442>>.
- [SUR2018] Boutaba, Raouf., Salahuddin, Mohammad., Limam, Noura., Ayoubi, Sara., Shahriar, Nashid., Estrada-Solano, Felipe., and Oscar. M. Caicedo, "A Comprehensive survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities", DOI 10.1186/s13174-018-0087-2, June 2018, <<https://link.springer.com/article/10.1186/s13174-018-0087-2>>.
- [VNEGCN2020]
Yan, Z., Ge, J., Wu, Y., Li, L., and T. Li, "Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks", RFC 1129, DOI 10.1109/JSAC.2020.2986662, April 2020, <<https://ieeexplore.ieee.org/document/9060910>>.
- [VNEQS2021]
Wang, Chao., Batth, Ranbir Singh., Zhang, Peiying., Aujla, Gagangeet., Duan, Youxiang., and Lihua. Ren, "VNE Solution for Network Differentiated QoS and Security Requirements: From the Perspective of Deep Reinforcement Learning", DOI 10.1007/s00607-020-00883-w, January 2021, <<https://link.springer.com/article/10.1007/s00607-020-00883-w>>.
- [VNESURV2013]
Fischer, Fischer., Botero, Juan Felipe., Till Beck, Michael;., Karim, MM., De Meer, Hermann., and Xavier.

Hesselbach, "Virtual Network Embedding: A Survey",
DOI 10.1109/SURV.2013.013013.00155, April 2020,
<<https://doi.org/10.1109/SURV.2013.013013.00155>>.

[VNETD2019]

Wang, S., Bi, J., V.Vasilakos, A., and Q. Fan, "VNE-TD: A
Virtual Network Embedding Algorithm Based on Temporal-
Difference Learning", BCP 72, RFC 3552,
DOI 10.1016/j.comnet.2019.05.004, October 2019,
<<https://doi.org/10.1016/j.comnet.2019.05.004>>.

[VNFFG2020]

Anh Quang, P.T., Hadjadj-Aoul, Y., and A. Outtagarts,
"Evolutionary Actor-Multi-Critic Model for VNF-FG
Embedding", RFC 1129, DOI 10.1109/CCNC46108.2020.9045434,
January 2020, <<https://www.rfc-editor.org/info/rfc2629>>.

[ZTORCH2018]

Sciancalepore, V., Chen, X., Yousaf, F. Z., and X. Costa-
Perez, "Z-TORCH: An Automated NFV Orchestration and
Monitoring Solution", BCP 72, RFC 3552,
DOI 10.1109/TNSM.2018.2867827, August 2018,
<<https://ieeexplore.ieee.org/document/8450000>>.

Authors' Addresses

Ihsan Ullah
KOREATECH
1600, Chungjeol-ro, Byeongcheon-myeon, Dongnam-gu
Cheonan
Chungcheongnam-do
31253
Republic of Korea
Email: ihsan@koreatech.ac.kr

Youn-Hee Han
KOREATECH
1600, Chungjeol-ro, Byeongcheon-myeon, Dongnam-gu
Cheonan
Chungcheongnam-do
31253
Republic of Korea
Email: yhhan@koreatech.ac.kr

TaeYeon Kim
ETRI
218 Gajeong-ro, Yuseong-gu
Daejeon
34129
Republic of Korea
Email: tykim@etri.re.kr