An Isolation-Aware Online Virtual Network Embedding via Deep Reinforcement Learning

Ali Gohar, Chunming Rong and Sanghwan Lee

March 15, 2023

An Isolation-aware Online Virtual Network Embedding via Deep Reinforcement Learning

Ali Gohar*†, Chunming Rong†, Sanghwan Lee‡

†Department of Electrical Engineering and Computer Science, University of Stavanger, Stavanger, Norway

‡College of Computer Science, Kookmin University, Seoul, Korea

*Corresponding author: Ali Gohar

{ali.gohar, chunming.rong}@uis.no, sanghwan@kookmin.ac.kr

Abstract—Virtualization technologies are the foundation of modern ICT infrastructure, enabling service providers to create dedicated virtual networks (VNs) that can support a wide range of smart city applications. These VNs continuously generate massive amounts of data, necessitating stringent reliability and security requirements. In virtualized network environments, however, multiple VNs may coexist on the same physical infrastructure and, if not properly isolated, may interfere with or provide unauthorized access to one another. The former causes performance degradation, while the latter compromises the security of VNs. Service assurance for infrastructure providers becomes significantly more complicated when a specific VN violates the isolation requirement.

In an effort to address the isolation issue, this paper proposes isolation during virtual network embedding (VNE), the procedure of allocating VNs onto physical infrastructure. We define a simple abstracted concept of isolation levels to capture the variations in isolation requirements and then formulate isolation-aware VNE as an optimization problem with resource and isolation constraints. A deep reinforcement learning (DRL)-based VNE algorithm ISO-DRL_VNE, is proposed that considers resource and isolation constraints and is compared to the existing three state-of-the-art algorithms: NodeRank, Global Resource Capacity (GRC), and Mote-Carlo Tree Search (MCTS). Evaluation results show that the ISO-DRL_VNE algorithm outperforms others in acceptance ratio, long-term average revenue, and long-term average revenue-to-cost ratio by 6%, 13%, and 15%.

Index Terms—Internet of Things, Isolation, Reinforcement Learning, Resource Allocation, Smart City, Virtual Network Embedding, Vertical Industries

I. INTRODUCTION

Network Function Virtualization (NFV) technology enables network operators to deploy, scale, and manage Virtual Network Functions (VNFs) more flexibly and efficiently. NFV technology is frequently used in conjunction with network slicing to support the diverse needs of 5G mobile networks. Network slicing allows partitioning a physical network infrastructure into multiple logical, end-to-end, isolated Virtual Networks (VNs), each with its own characteristics and performance parameters. A VN can be composed of a series of VNFs, and if a set of VNFs is chained in the order in which traffic flows through the network, it is called a Service Function Chain (SFC). This novel approach in the mobile network ecosystem allows for a role separation between Infrastructure Providers (InP) and Service Providers (SPs) [1].

To create VNs, InP deploys the necessary VNFs on the InP infrastructure, which comprises heterogeneous resources such as computing, storage, and network, also known as a Substrate Network (SN). Efficient mapping of the VN towards the SN primarily deals with the VNF placement or is a typical Virtual Network Embedding (VNE) problem. VNE has been extensively researched in the literature from various perspectives, including resource allocation, power consumption, and security [2]. Due to its NP-Completeness [3], various solutions for solving the VNE problem have been proposed and are broadly classified into the following categories [4] [2];

(i) Mathematical optimization-based: [5] model the VNE problem using MIP, but to obtain the solution in polynomial time, a rounding-based approach is used to relax the integer constraints. [6] use ILP to model the VNE problem as a set cover problem and propose algorithms with a logarithmic approximation factor to place the VNFs. Due to the high computational complexity, these offline approaches are challenging to apply to online placement scenarios;

ii) Heuristic-based: To increase the long-term average revenue and acceptance ratio, a PageRank-inspired VN embedding algorithm is proposed to compute ranking values using the Random Walk (RW) in [7]. [8] propose an algorithm based on the Global Resource Capacity (GRC) metric to maximize the revenue-to-cost ratio. [9] presents an abstracted numerical concept of security levels and demands followed by the formulating a security-aware VNE problem that considers takes security constraints. These algorithms cannot adapt to a various environments and may result in local optimal solutions.

iii) Reinforcement learning (RL)-based: [10] formalize the VNE problem using the Markov Decision Process (MDP) and use the Monte-Carlo Tree Search (MCTS) algorithm to maximize the revenue-to-cost and acceptance ratio. To minimize power consumption, [11] employs a policy gradient approach combined with the Lagrange relaxation technique. Their evaluations, however, are limited to using a grid or star topology to represent the SN. The RL algorithms offer significant advantages over traditional methods, including adaptation to changing network conditions, optimization for a global solution, handling uncertainty, and minimizing human intervention through continual learning.

SPs use a variety of third-party InP SNs for data computa-

tion and/or storage, which leads to inherent confidentiality, integrity, and availability vulnerabilities for VNs [11]. However, as an essential dimension, researchers have only discussed VN isolation in this context from theoretical perspectives and challenges that VNs face concerning isolation, which is the property that one VN operates independently of other VNs using the same infrastructure [12].

In this paper, we present an abstracted numerical concept of isolation levels and requirements, followed by the formulation of an isolation-aware VNE problem that considers isolation constraints. Furthermore, we propose a Deep RL (DRL)-based algorithm ISO-DRL_VNE for adaptive VN placement. ISO-DRL_VNE maximizes long-term average revenue by learning to make VN placement decisions based on past decision performance rather than a hypothetical environment. The main contributions of this paper are:

- a concept of abstractions to represent isolation levels and requirements,
- a model and description of the VNE problem constrained by isolation,
- a solution based on DRL for adaptive VN placement,
- a performance comparison of the proposed solution with three existing works: NodeRank [7], GRC [8], and MCTS [10].

The rest of this paper is organized as follows: Section II presents isolation levels. Section III describes the problem model. Section IV describes the solution algorithm. Section V compares the performance of the proposed algorithm with the three existing algorithms, and finally, Section VI concludes this study.

II. ISOLATION-AWARE VIRTUAL NETWORK EMBEDDING

The 3GPP highlights several important issues that need to be addressed to maintain an isolated environment for future mobile networks [13]. Previous studies have discussed the challenges of achieving end-to-end isolation, identifying enabling technologies and the need for standardized approaches for designing isolated VNs or slices [14] [15]. The isolation types and parameters are further explored in [15]. It is recognized that defining the proper isolation parameters and points is a critical issue [15]. Based on available technologies and current literature, this paper identifies three major isolation points that can be implemented to ensure VN and SN separation, as follows (i) isolation between different SN nodes, (ii) VN nodes deployed on a SN node, and (iii) VN nodes co-deployed on the same SN node.

To address VN node isolation issues, each SN node is assumed to have an isolation level that allows each VN node on the shared infrastructure to operate independently of other VN nodes. The variation in isolation levels stems from the realization that, depending on the service type, not all VN nodes require the same isolation level. We identify the following abstracted isolation between an SN node and a VN node based on the assumptions stated above.

- The isolation level of a SN node should be equal to or greater than the isolation requirement level of the VN node deployed on the SN node.
- The isolation level of a VN node should be equal to or less than the isolation level of the SN node hosting the VNF.
- The isolation level of a VNF should be equal to or less than the isolation requirement level of the VNFs co-deployed on the same SN node.

III. PROBLEM MODEL AND DESCRIPTION

Two actors are considered in the VNE problem: InPs and SP. The InP receives a VN Request (VNR) from SP representing a specific VN and must be allocated on SN. The following two subsections describe the types of resources provided by InP SN and the requirements for VNRs.

A. SN Model

The SN is composed of the following two elements:
1) *Computation platforms (CPs):* We indicate N^S as the set of CPs where A_N^S represents a configuration for each CP. For each CP configuration, we consider the following resources:
   - the amount of processing $CPU(n^s)$ in [vCPU];
   - isolation level available $ISA(n^s)$ in [Units]; $\forall \, n^s \in N^S$.

2) *Logical connections (LCs):* The LCs connect CPs and are provided by network operators. We indicate L^S as the set of LCs where A_L^S represents a configuration for each LC. We consider the following resources for each LC configuration:
   - the amount of data rate $BW(l^s)$ in [Mb/s]; $\forall \, l^s \in L^S$

The SN model can now be formalized using graph theory and can be represented as a weighted undirected graph by $G^S = \{N^S, L^S, A_N^S, A_L^S\}$.

B. VNR Model

A VNR represents a specific VN provided by SP for its users, and has the following requirements:
1) *Computing requirements:* We indicate N^V as the set of virtual nodes (i.e., VNFs) where A_V^S represents a configuration for each virtual node. For each virtual node configuration, we consider the following resources:
   - computing resource requirement $CPU(n^v)$, in [vCPU];
   - isolation level requirement $ISR(n^v)$ in [Units]; $\forall \, n^v \in N^V$.

2) *Networking requirement:* Since, virtual nodes are connected together. We consider virtual links that connect various virtual nodes. We indicate L^V as the set of virtual links where A_L^V represents a configuration for each virtual link. We consider the following resources for each virtual link configuration:
   - data rate requirement $BW(l^v)$ in [Mb/s]; $\forall \, l^v \in L^V$

The VNR model is formalized using graph theory and is represented as $G^V = \{N^V, L^V, A_N^V, A_L^V\}$.
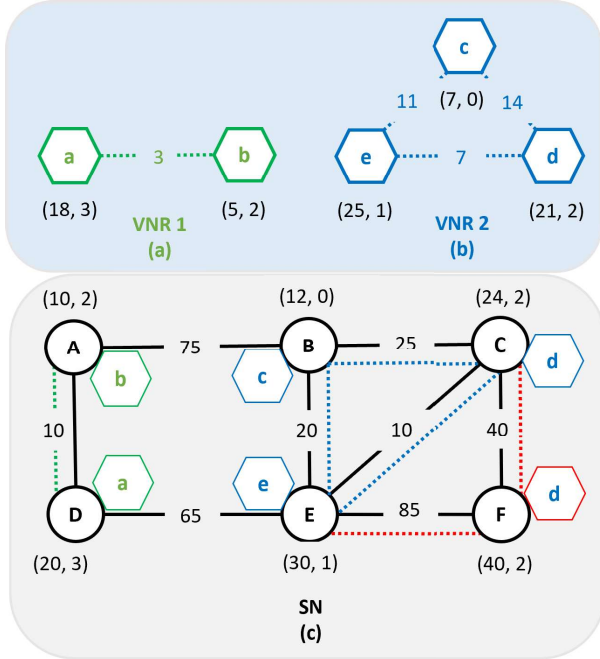
C. VNE Problem Description

Fig. 1: SN & VN model with possible embedding examples

The VNE problem can be express as mapping G^V to $G^V(N^V, L^V) \rightarrow G^S(N^{S_i}, L^{S_i})$ where $N^{S_i} \in N^S, L^{S_i} \in L^S$. Figure 1 shows two different embedding scenarios of VNRs. Moreover, (a) and (b) depict two distinct VNRs, (VNR 1 and VNR 2), while (c) depicts an SN. A Hexagon represents the VN nodes, and a dotted line represents their VL demand. A circle represents an SN node, and a solid line connecting two circles represents an SN link. The numbers in brackets next to the virtual nodes indicate the computing and security level requirements for two VNRs. The rest numbers on those links indicate the bandwidth needs of the virtual links. Similarly, the numbers in brackets in SN represent the available computing resources, and security levels, respectively. The rest numbers represent the available bandwidth resources.

We highlight three possible scenarios in which VNRs are embedded in the SN. The virtual nodes a and b in VNR 1 are mapped to the A and D. However, the virtual nodes c, d, and e in VNR 2 can be mapped to the B, C, and E or the C, E, and F, respectively. It should be noted that the virtual node d of VNR 2 can be embedded in either C or F; however, embedding virtual node d of VNR 2 in F will consume more SN node and link resources and is not an optimal allocation.

D. Evaluation Metrics

This paper assumes that the InP charges SPs by the "pay-as-you-go" pricing model. As a result, the revenue generated by an InP accepting a VNR is defined as follows:

$$REV(G^V) = \alpha \sum_{i=1}^{|N^V|} CPU(n_i^v) + \beta \sum_{j=1}^{|L^V|} BW(l_j^v) \quad (1)$$

Eq. 1, shows the revenue of InP, where $CPU(n_i^v)$ represents the computing resources occupied by the virtual node (n_i^v). $BW(l_j^v)$ represents the bandwidth resource occupied by the virtual link l_i^v. where α is the unit price of resource for SN nodes (CPs), and β is the unit price of bandwidth for SN links (LCs)

Eq. 2, shows the cost to InP for providing SN resource services to the SP and is formulated as follows:

$$CST(G^V) = \sum_{i=1}^{|N^V|} CPU(n_i^v) + \sum_{j=1}^{|L^V|} \sum_{k=1}^{|L^S|} BW(l_{jk}^{vs}) \quad (2)$$

where $CPU(n_i^v)$ represents the computing resources occupied by the virtual node (n_i^v) after successful embedding. $BW(l_{jk}^{vs})$ represents the bandwidth resource occupied by the virtual link l_{jk}^{vs} with link splitting.

VNRs arrive with varying time distributions and resource requirements. It is preferable to have a VNE algorithm that accepts more VNRs. The ratio of accepted VNRs to the total number of VNR that arrived is defined as:

$$ACR = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} accepted\left(G^V, t\right)}{\sum_{t=0}^{T} arrive\left(G^V, t\right)} \quad (3)$$

Eq. 3 shows the acceptance rate where $accepted(G^V, t)$ denotes the total number of VNRs that are successfully embedded and $arrive(G^V, t)$ represents the total number of VNRs that make resource requests.

The long-term average revenue (LAR), is used to assess the overall effectiveness of a VNE algorithm, is defined as:

$$LAR = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} REV\left(G^V, t\right) \quad (4)$$

Eq. 4 shows (G^V, t) the revenue of successful VNR allocation arrived at time t where T is the total elapsed time. A higher LAR leads to a higher profit for the InP.

A higher long-term average revenue-to-cost ratio (LRC) indicates a higher resource utilization efficiency to meet the resource needs of more VNRs. LRC is defined in Eq. 5 as:

$$LRC = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} REV\left(G^V, t\right)}{\sum_{t=0}^{T} CST\left(G^V, t\right)} \quad (5)$$

The VNE process needs to meet the following constraints:

$$R_{CPU}(n^s) = CPU\left(n^s\right) - \sum_{i=1, n^v \to n^s}^{|VNR|} CPU\left(n_i^v\right) \quad (6)$$

$$R_{BW}(l^s) = BW(l^s) - \sum_{i=1, l^v \to l^s}^{|VNR|} BW(l_i^v) \quad (7)$$

$$R_{cpu}(n^s) \geq CPU(n^v) \tag{8}$$

$$R_{BW}(l^s) \geq BW(l^v) \tag{9}$$

$$ISA(n^s) \geq ISR(n^v) \tag{10}$$

Eq. 6 and Eq. 7 represent the remaining computing and bandwidth resources for the SN nodes and SN links. Eq. 8 and Eq. 9 represent the computing and bandwidth constraints of a VNR. Eq. 10 represent the security constraints of VNR. Eq. 8 and Eq. 10 are followed during the node embedding stage, and Eq. 9 is followed during the link embedding stage.

In this paper, ACR, LAR, and LRC are all optimized together. The goal is to maximize following joint optimization, which is defined as:

$$maximize\ Objective =\ E_1 + E_2 + E_3 \tag{11}$$

$where\ E_1 = ACR,\ E_2 = LAR,\ E_3 = LRC.$

IV. EMBEDDING ALGORITHM

The basic elements of RL are agent, environment, state, action, and reward. The Markov Decision Process (MDP) is used to describe the environment in RL. We model the placement of each VNR as a finite-horizon MDP in which the agent selects SN nodes to place VNFs of VNR at discrete time step $t \in (1, 2, 3, \ldots, T)$. We define three key elements: state, action, and reward as follows:

1) State: To represent the state information of SN, the SN features are extracted. Extracting all SN features would be unrealistic due to the increased computational complexity, and insufficient feature extraction would result in an inadequate representation of the environment. For each SN node, the following features are extracted:

- $CPU(n^s)$: The remaining CPU capacity of an SN node has a significant impact on allocation.
- $DEG(n^s)$: Degree represents the number of LC directly connected to the SN node. Higher degree results in easier path finding between other SN nodes.
- $SUM(n^s)$: Represents the sum of the bandwidth of all LC connected to each SN node. A higher SUM is more likely to complete the link embedding in this SN node.

Following feature extraction, the values are normalized and represented as a feature vector. The above properties of the $i-th$ SN node are represented as a 3-dimensional vector, as follows:

$$v_i = (CPU(n_i^s), DEG(n_i^s), SUM(n_i^s))^T \tag{12}$$

All SN node features are represented as vectors and are combined into a feature matrix M, which is provided as input to the agent and updated as the SN state changes.

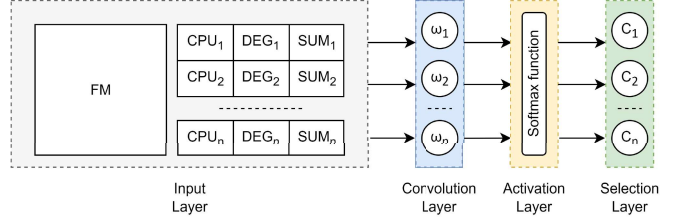$$M = (v_1, v_2 \ldots v_n)^T \tag{13}$$

Fig. 2: Policy network

The feature matrix M is expressed as follows:

$$\begin{bmatrix} CPU(n_1^s) & DEG(n_1^s) & SUM(n_1^s) \\ CPU(n_2^s) & DEG(n_2^s) & SUM(n_2^s) \\ CP\ddot{U}(n_k^s) & DE\ddot{G}(n_k^s) & SU\ddot{M}(n_k^s) \end{bmatrix} \tag{14}$$

The purpose of normalization is to accelerate the training process and enable the agent to converge quickly.

2) Action: The action at time t selects an SN node from the set of all SN nodes whose available resources exceed the requirements requested by VNF.

3) Reward: A reward signal is required to help the agent learn better actions. As a reward, the LRC generated by a successful VNR embedding is used; otherwise, the agent receives 0 as a reward. Reward based on LRC reflects the utilization of network resources. High LRC values indicate that the selected actions by the agent are effective in realizing high VNE revenue, while low LRC values necessitate adjustments to the actions by the agent.

A. Policy Network

In this paper, we use a four-layer ANN to construct a policy network as the learning agent, which takes the M as input and outputs the probabilities of mapping VNR nodes to SN nodes. The policy network comprises four layers: the input layer, the convolution layer, the probability layer, and the selection layer as shown in Fig. 2.

The convolution layer receives a M and transfers it to the input layer (CNL). CNL applies the convolution operation on M to create a vector representing each SN node available resources and is given by Eq. 15

$$z_i = \omega.v_i + d \tag{15}$$

z_i is the ith output of the convolution layer, ω is the weight of the convolution kernel vector, and d is the deviation.

Softmax is mainly used to measure the probability of each SN node based on the available resource in the probability layer and is given by Eq. 16.

$$P_i = \frac{e^{r_i}}{\sum_k e^{r_n}} \tag{16}$$

where P_i represents the probability that the ith physical node is embedded. n is the number of feature vectors. The selection layer filters the set of SN nodes with sufficient resource capacities, and the SN node with the highest probability is chosen for embedding.

B. Training and Testing

The gradient update method is used to learn policy network parameters. The gradient update procedure updates the loss function in the direction of gradient descent to optimize the parameters of the policy network. It optimizes both the reward mechanism and the policy network at the same time, and the result is as follows:

$$\mathcal{J} = \mathcal{J} - \mathcal{J}' \times \alpha \times Reward \qquad (17)$$

α is an important factor influencing the gradient update and training speed. A small value will slow down the training process. A large value will lead to poor decisions, and a value too large will produce unstable training. A batch updating approach is used to ensure learning stability and speed up updating the policy network parameters.

Algorithm 1 shows the DRL-based training process for the VNE algorithm. All policy network parameters are initialized randomly in line 1 and trained for several epochs in line 2 during the training phase. Each VNR is mapped on SN in two stages. In the first stage, lines 5-8 map the VNR nodes onto SN. The virtual links of VNR nodes are mapped using Breadth-First Search (BFS) in the second stage in line 10 if all VNR nodes are mapped.

The feature matrix (M) of SN is passed to the policy network in line 3 along with the VNR that is chosen from the *trainingSet*. Lists of potential SN nodes and the likelihood that they will map to VNR nodes are produced by the policy network in line 7. If no SN nodes or SN links with enough resources are present, the mapping fails. Line 8 includes the gradient calculation. Following the successful mapping of VNR nodes and links, line 11 determines the reward. Before calculating the gradient update in line 12, we provide LRC as a *Reward* which is used in Eq. 17.

During testing of the policy network, a greedy strategy is employed, selecting the SN node with the highest probability as the mapping node for the VNR virtual node. The testing process of the VNE algorithm based on DRL is demonstrated in Algorithm 2.
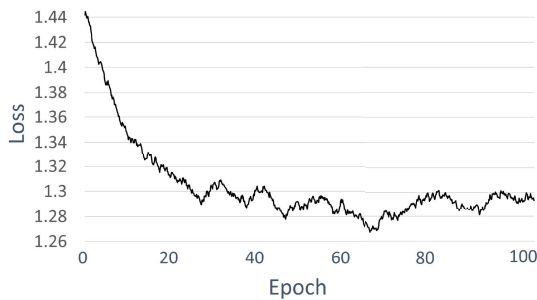

Fig. 3: The loss variation during the training process.

V. EVALUATION

In this section, we first describe the evaluation parameters used to generate SN topology, VNRs, and comparison algo-

---

Algorithm 1: Training Policy Network

Input : SN, trainingSet, epochNumber, α, batchSize
Output: Policy network parameters
1 Initialize all parameters of the policy network
2 **while** $iterationNumber \leq epochNumber$ **do**
3   **for** $VNR \in trainingSet$ **do**
4       $counter = 0$
5   **for** $VNR\_node \in VNR$ **do**
6   receive feature matrix (M)
7   get probability distribution of SN nodes
8   compute the gradient
9   **if** *isMapped ($\forall$ VNR_nodes $\in$ VNR)*
10   **if** *linkMappingUsingBFS(VNR)*
11   reward = rewardCalculation(VNR)
12   gradientUpdate(reward, α)
13   **else**
14   clearGradient()
15       $counter+ = 1$
16   **if** $counter == batchSize$
17          $counter == 0$
18    $iterationNumber+ = 1$

Algorithm 2: Testing Policy Network

Input : SN, testingSet
Output: Results for three evaluation objectives
1 Initialize all policy network parameters
2 **for** $VNR \in testingSet$ **do**
3   **for** $VNR\_node \in VNR$ **do**
4   get feature matrix (M)
5   get mapping probability of SN node
6   **if** *isMapped ($\forall$ VNR_nodes $\in$ VNR)*
7   **if** *linkMappingUsingBFS(VNR)*
8   return $embeddingSuccess$

TABLE I: Evaluation parameters

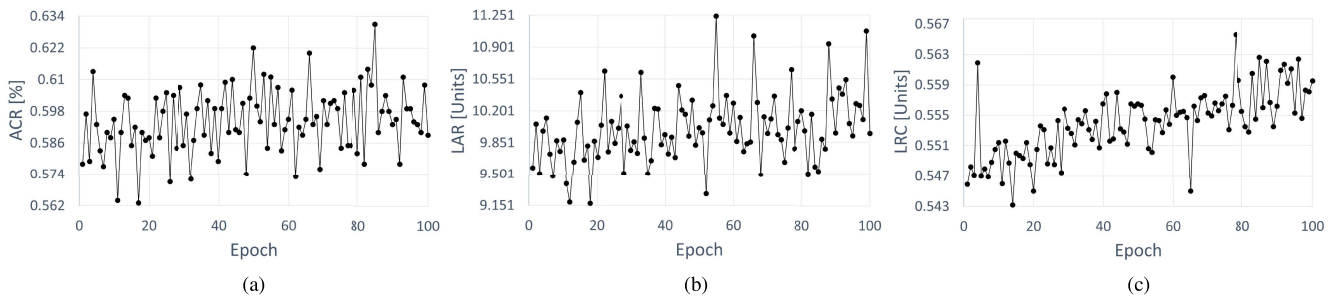| Configuration Attribute	Attribute Value
Number of SN nodes	100
Computing resources	U[50, 100]
Bandwidth resources	U[50, 100]
ISA	U[1, 3]
VNRs	2000
VNR nodes	U[2, 10]
Computing requirements	U[0, 50]
Bandwidth requirements	U[0, 50]
ISR	U[1, 3]
VNRs for training	1000
VNRs for testing	1000

Fig. 4: Training results. (a) Performance of the VNRs acceptance ratio (ACR) on the training set. (b) Performance of long-term average revenue (LAR) on the training set. (c) Performance of long-term revenue to cost ratio (LRC) on the training set.

TABLE II: parameters of the DRL-based learning agent

Parameters	Value
Learning rate (α)	0.001
Discount factor (γ)	0.998
Batch Size ($batchSize$)	64
Number of epoch ($epochNumber$)	100

rithms. The main simulation parameters are also presented in Table. I.

A. Evaluation Settings

1) Generating SN: Using the GT-ITM tool, we create an SN that is roughly the size of a middle-sized ISP and has 100 nodes [7] [8] [9] and about 500 links [7] [9]. The computing capacity of every SN node and the bandwidth of every SN node link is a positive real number that follows a uniform distribution and is between 50 and 100 [5] [7] [8] [9].

2) Generating VNRs: The VNR arrive with an average of 5 VNRs per 100-time units according to a Poisson distribution, and they stay for the duration with an average of 500-time units according to an exponential distribution [7] [8] [9]. There are 2–10 virtual nodes in each VNR [5]. Every virtual node has a uniformly distributed computing, and bandwidth requirement between 0 and 50 units [7] [8] [9]. 2000 VNRs are generated and are equally divided into two sets: a training set and a evaluation set.

3) Comparison Algorithms: Our algorithm is written in Python, and we use Pycharm as an IDE. For comparison, we chose the following algorithms: NodeRank [7]: Computes ranking values based on RW and sorts SN nodes based on their importance. GRC [8]: Considers global resource capacity to map VNFs onto SN nodes. MCTS [10]: RL-based algorithm that uses Monte Carlo Tree Search (MCTS) to make the VNF placement decision. For link mapping, the shortest path algorithm is used by all evaluation algorithms.

4) ISO-DRL_VNE algorithm: TensorFlow is used to build the policy network, and Xavier [16] is used to initialize the policy network parameters using the normal distribution. The gradient of each training iteration is calculated using a stochastic gradient descent optimizer with a learning rate of 0.001 over 100 epochs. The key parameters of the DRL-based learning agent to reproduce our results are listed in Table. II.

5) ISO-DRL_VNE Complexity: The algorithm complexity of the ISO-DRL_VNE algorithm is expressed as a function of the number of incoming VNRs, C_{VNR}, the number of substrate nodes, C_{n^s}, the dimension of the feature matrix, d, and the number of successfully embedded virtual nodes, C_{n^v}, and virtual links, C_{l^v}. The general form of the algorithm complexity is $O\left(C_{VNR}\left(C_{n^s} \cdot d + C_{n^v} + C_{l^v}\right)\right)$. This means that the algorithm's running time will increase linearly with the number of incoming VNRs, as well as with the size of the substrate network, the dimension of the feature matrix, and the number of successfully embedded virtual nodes and virtual links.

B. Training Results

Fig.3 depicts the performance of the DRL based VNE algorithm (DRL-ISO_VNE), which shows the variation of the loss of the policy network with increasing number of training requests. The loss reaches a minimum value, a local optimum, after about 50 epochs (400 VNRs). This implies that the approximation utilized in the algorithm is effective. At the beginning of the training process, the loss value is relatively high due to the random initial parameter values, but as training continues, the learning algorithm updates the Artificial Neural Network (ANN) based on the reward signal. The decreasing trend in the loss suggests that the loss approximation performance of the DRL-based VNE algorithm is satisfactory.

Fig.4 shows the performance of ISO-DRL_VNE during the training process. Initially, the parameters of the policy network are randomly initialized and it takes time for the algorithm to adapt to the environment, causing the performance to be unstable. However, as training continues, the agent becomes more familiar with the environment and starts to take actions that lead to a larger reward, causing the performance to become more stable. By the end of the training period, the performance has reached its limit and the policy network has fully adapted to the learning environment.

C. Evaluation Results

The comparison algorithms are heuristic, and they do not consider the long-term impact of resources and isolation constraints when allocating VNRs.

Figure 5 shows the acceptance ratio (ACR) of all algorithms. The ACR is high initially because, as time passes, the SN nodes become increasingly congested, making it more difficult to allocate additional VNRs. ISO-DRL_VNE achieves the highest acceptance rate of 0.66, up to 4.3%, 6.4%, and 6.0 better than NodeRank, MCTS and GRC.

Figure6 shows the long-term average revenue (LAR), which declines over time and has a similar trend as in Figure 5. This is because both metrics are affected by the number of resources available in SN. The ACR is high before 200 VNRs due to the abundance of SN resources that can satisfy more VNR requirements. However, due to the continued use of SN resources in the later period, the trend began to decline gradually. Overall, ISO-DRL_VNE has a LAR of 10.6, up to 10%, 13%, and 7% better than NodeRank, MCTS, and GRC.

Figure 7 shows the long-term revenue-to-cost ratio (LRC). An upward trend is observed because the LRC is independent of available SN resources. This evaluation metric depends on the efficiency of the algorithm. The LRC of the ISO-DRL_VNE algorithm is 13%, 15%, and 15.3% higher than NodeRank, MCTS, and GRC.

The run time of ISO-DRL_VNE algorithm is 29%, 9%, and 52% lower than NodeRank, MCTS, and GRC.
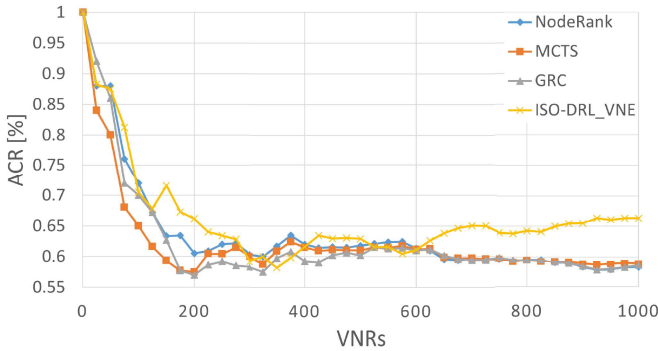

Fig. 5: Acceptance ratio (ACR)

VI. Conclusion

The VNE problem with an isolation levels is introduced and investigated in this study. First, this paper explains why VN VNFs need different levels of isolation requirements and proposes the VNE problem with isolation constraints. Second, to maximize the InP long-term average revenue (LAR) and long-term average revenue-to-cost ratio (LRC), an isolation-aware VNE problem is presented. ISO-DRL_VNE, a DRL-based algorithm, is proposed and evaluated in comparison to three state-of-the-art algorithms. Based on the outcomes of the experiments, we found that the ISO-DRL_VNE could enhance VN isolation while offering a good VNE strategy with high performance and practical applicability in online scenarios.

References

[1] D. Bega *et al.*, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.
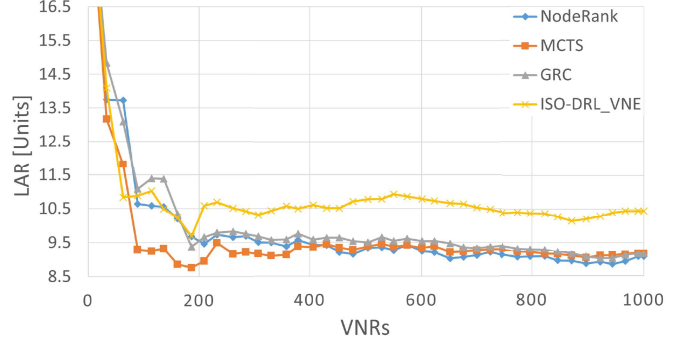
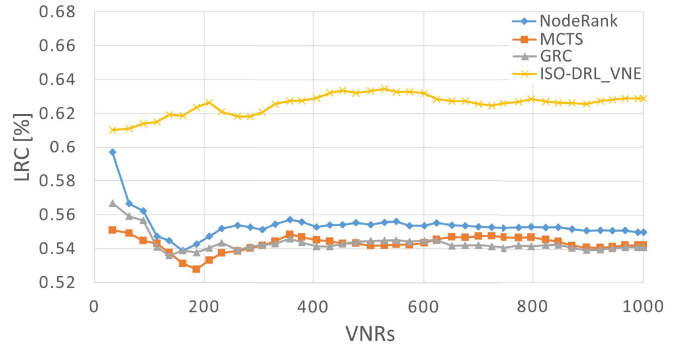Fig. 6: Long-term average revenue (LAR)

Fig. 7: Long-term average revenue to cost ratio (LRC)

[2] J. Sun *et al.*, "A survey on the placement of virtual network functions," *Journal of Network and Computer Applications*, vol. 202, p. 103361, 2022.

[3] D. G. Andersen, "Theoretical approaches to node assignment," in *TANA*. Carnegie Mellon University, 2002.

[4] H. Cao *et al.*, "A survey of embedding algorithm for virtual network embedding," *China Communications*, vol. 16, no. 12, pp. 1–33, 2019.

[5] M. Chowdhury *et al.*, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[6] Y. Sang *et al.*, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[7] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, p. 38–47, apr 2011.

[8] L. Gong *et al.*, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 1–9.

[9] S. Liu *et al.*, "Towards security-aware virtual network embedding," *Computer Networks*, vol. 91, pp. 151–163, 2015.

[10] S. Haeri *et al.*, "Virtual network embedding via Monte Carlo tree search," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 510–521, 2018.

[11] R. Solozabal *et al.*, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2020.

[12] A. J. Gonzalez *et al.*, "The isolation concept in the 5G network slicing," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 12–16.

[13] "3GPP TR 23.799 – Study on architecture for next generation system," Dec 2016.

[14] A. A. Barakabitze *et al.*, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[15] Z. Kotulski *et al.*, "Towards constructive approach to end-to-end slice isolation in 5G networks - eurasip journal on information security," Mar 2018.

[16] J. Guo *et al.*, "AI notes: Initializing neural networks," 2019.