Enhancing 5G Network Slicing: Slice Isolation via Actor-Critic Reinforcement Learning with Optimal Graph Features

Amir Javadpour*‡**, Forough Ja'fari†§, Tarik Taleb*¶, and Chafika Benzaïd*‖
*Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, 90570, Finland
**ICTFICIAL Oy, Espoo, Finland
†Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
‡a.javadpour87@gmail.com §azadeh.mth@gmail.com ¶tarik.taleb@oulu.fi ‖chafika.benzaid@oulu.fi

***Abstract*—Network slicing within 5G networks encounters two significant challenges: catering to a maximum number of requests while ensuring slice isolation. To address these challenges, we present an innovative actor-critic Reinforcement Learning (RL) model named 'Slice Isolation based on RL' (SIRL). This model employs five optimal graph features to construct the problem environment, the structure of which is adapted using a ranking scheme. This scheme effectively reduces feature dimensionality and enhances learning performance. SIRL was assessed through a comparative analysis with nine state-of-the-art RL models, utilizing four evaluation metrics. The average results demonstrate that SIRL outperforms other models with a 70% higher coverage rate of requests and an 8% reduction in damage resulting from DoS/DDoS attacks.**

***Index Terms*—5G, beyond 5G, network slice, slice isolation, Reinforcement Learning (RL), Distributed Denial of Service (DDoS), and security.**

I. Introduction

Network slices are independent virtual networks that share physical infrastructures in 5G. Different users use these virtual networks to serve their requests for specific resources and services [1, 2]. Denial of Service (DoS) and Distributed DoS (DDoS) attacks [3, 4] are some of the security threats targeting these slices [5, 6]. In these attacks, the adversary sends flooded traffic to one of the network nodes, and that node cannot provide its services. As a result, we can say that the network slicing process faces two main challenges: (1) *How to maximize the number of slices, which are supported by a specific physical infrastructure* and (2) *How to reduce the impact of DoS/DDoS attacks against the slices*.

Mapping the virtual networks on the shared physical network is a kind of Virtual Network Embedding (VNE) problem, which tries to map a set of weighted graphs (i.e., virtual network) on another weighted graph (i.e., substrate network) considering the capacity of the nodes and links [7, 8]. Several researchers have proposed optimal mapping strategies for the first challenge. However, according to the authors' best knowledge, none considered the second one. Sattar and Matrawy [9] considered slice isolation for protecting the slices from DDoS attacks. However, the first challenge is not considered in this work. As a result, in this paper, we aim to provide slice isolation for 5G networks, wherein both challenges are addressed [10].

We propose an actor-critic Reinforcement Learning (RL) model called Slice Isolation based on Reinforcement Learning (SIRL) to cover the two above-mentioned challenges. RL is a machine-learning approach wherein an agent explores the problem environment and interacts with all its different states. In each state, the agent observes the features of that state and can act. Then the agent will be rewarded for helping it find the optimal solution [11, 1]. The environment of some problems is too broad, and some actions are not possible in certain states. Actor-critic RL models define some policies to help the agent improve its learning performance [12]. This paper makes significant contributions, outlined as follows:

- Introducing an enhanced actor-critic RL model that incorporates innovative features to isolate the slices effectively.
- Devising a technique to decrease the environmental footprint of the RL model, thereby enhancing its efficiency.
- Assessing the performance of the proposed model through a comprehensive comparison with various established RL models using four distinct evaluation metrics.

The remaining sections of this paper are organized as follows: In section II, a comprehensive review of various RL approaches proposed for addressing VNE problems is provided. section III introduces our novel RL model and outlines the algorithms used for establishing the problem environment. The performance of our proposed model is assessed in section IV through a comparative analysis with multiple RL models. The paper is concluded in section V.

II. Related Work

This section provides an overview of various RL models proposed for mapping virtual nodes onto substrate nodes. Notably, in recent work by Yao et al. [13], a novel RL model named CDRL was introduced to address VNE problems. CDRL incorporates three critical features for each substrate node: CPU capacity, cumulative bandwidth of adjacent links, and node degree. Similarly, Yao et al. [14] introduced an alternative model named RDAM, which also leverages the aforementioned features of CDRL. However, RDAM introduces an additional feature: the average distance to other nodes, enhancing the model's mapping capabilities. Moreover, Cao et al. [15] contributed to this field by presenting an RL model built upon the foundation of CDRL. This model, while retaining the core features of CDRL, integrates an extra feature about security levels. This additional feature enhances the protection of both network slices and substrate nodes within 5G networks.

Jiang and Zhang [16] proposed an RL model that considers each virtual node's security level in solving the VNE problem. We call this model VNEQS, short for VNE for Quality of Service and Security. The features representing a substrate node in VNEQS are the CPU capacity, the sum of adjacent links bandwidth, the delay, and the safety level.

Lu et al. [17] proposed a RL model, MLRL, for software-defined networks. MLRL finds the optimal mapping solution for two-layer virtual requests. The first layer request acts as the substrate network for the second layer request in a two-layer request. Three topological features are considered in MLRL for modeling the environment: degree centrality, closeness centrality, and betweenness centrality. Then, a vector consisting of the CPU capacity, the sum of adjacent links bandwidth, and the topological features is considered for each substrate node. Li and Lu [18] also proposed a similar RL model, called DRLVNE, with two extra features: the average distance to the other nodes and eigenvector centrality.

In a novel approach, Zhang et al. [19] introduced an actor-critic RL model that employs a graph convolutional neural network for automatic feature extraction from the environment. This model is aptly named GCNNRL (Graph Convolution Neural Network with RL). Within the context of GCNNRL, features of interest encompass the CPU capacity of individual substrate nodes, the available bandwidth for substrate links, the requisite CPU capacity of virtual nodes, the required bandwidth for virtual links, and the successful mappings of virtual nodes and links.

Similarly, Yan et al. [20] contributed an innovative deep RL model, A3CGCN, designed to tackle the VNE problem autonomously. Notably, A3CGCN leverages a graph convolutional network to extract both substrate and virtual network features. The core features shaping the model's environment include the maximum CPU capacity of substrate nodes, the maximum cumulative adjacent link bandwidth among substrate nodes, the remaining capacity and cumulative adjacent link bandwidth for each substrate node, a binary indicator reflecting whether a previous virtual node of the current request has been mapped onto a substrate node, the required CPU capacity and cumulative adjacent link bandwidth for each virtual node, and the count of virtual nodes within the ongoing request that remain unmapped.

Dolati et al. [21] proposed a deep RL model called DeepViNE, wherein the substrate network graph is encoded as an image considering the spatial locality property. Some features of the virtual and substrate nodes are considered to create this image. The required CPU and a binary value, which indicates whether or not the virtual node is mapped, are the features considered for each virtual node. The required bandwidth between every two virtual nodes is also considered. Moreover, the CPU capacity of each substrate node and a binary sequence that indicates which virtual nodes are mapped on are also considered in DeepViNE.

Wang et al. [22] proposed a RL model, called PNVNE, for solving VNE problems, which utilizes an attention mechanism that selects the most suitable substrate nodes; hence, the model can focus on them. The remaining CPU capacity, the sum of the bandwidth of the adjacent links, and the minimum and maximum available bandwidth of each substrate node are considered in PNVNE. The requested CPU capacity, the requested bandwidth of the virtual links, and the minimum and maximum requested bandwidth are the features for each virtual node.

The A2CRL model presented by Troia et al. [23] focuses on mapping slice requests in 5G networks, emphasizing acceptance decisions and incorporating various features. However, it lacks consideration for DoS/DDoS damage and faces challenges with suboptimal state sizes. In response, we propose a novel RL model that addresses these limitations, enhancing security measures and optimizing state size.

III. Proposed Model for Slice Isolation

Providing slice isolation has two aspects: (1) covering an optimal number of slice requests and (2) protectively mapping the slices. These aspects are shown in Figure 1. The circles in this figure are the virtual nodes, and the colors specify the requester. For example, the blue circles are the virtual nodes that belong to the fourth user's request. The numbers inside the circles and on the links specify the required CPU and bandwidth capacities, respectively. In Figure 1a, the fourth user is completely unsatisfied. This is because one of the virtual nodes related to its request is not mapped due to resource limitations, the other mapped virtual node (the crossed circle) cannot handle the whole request. As a result, we can say that the fourth user does not receive the service. The other three users are completely covered. However, they do not get the service when the adversary launches an attack against PM1. The virtual nodes mapping in Figure 1b is more acceptable because the fourth user is also covered, and when PM2 is under attack, the nodes related to that user are not involved. Finally, Figure 1c shows a better situation, where only two of the users are affected by the attack.

Considering the definition of slice isolation suggested by Sattar and Matrawy [9], we define slice isolation as any security mechanism that can protect a slice from the threats targeting the other slices. Therefore, the meaning of slice isolation is not limited to securing the links. Preventing the impact of a DDoS attack against slice A on slice B can also be considered slice isolation. For example, assume that slice A and slice B are mapped on the core network so that they have a shared substrate node C. If an adversary launches a DDoS attack against C to cause damage to slice A, slice B is also indirectly affected. This paper focuses on the slice isolation mechanism to separate the slices to reduce the DoS/DDoS attack damage.

In our proposed SIRL model, we aim to map as many slice requests as possible while keeping their availability at an

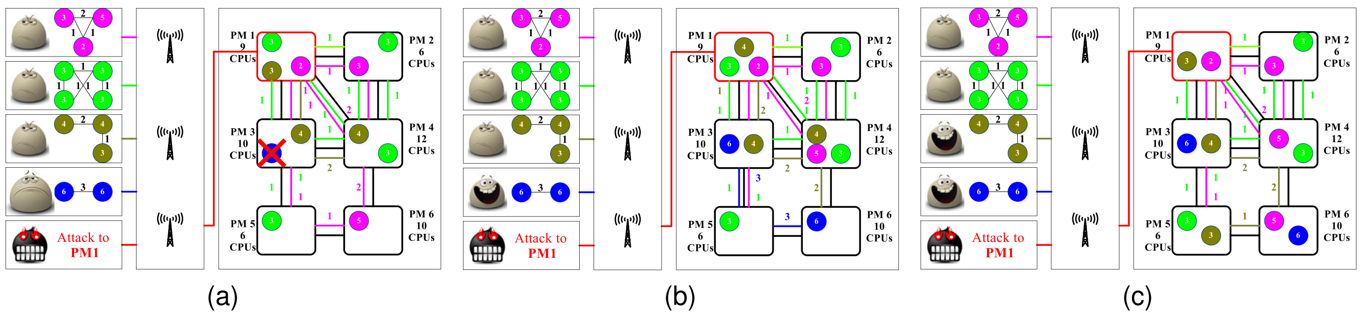

Fig. 1. An example of considering two aspects of slice isolation (1) covering a maximum number of slice requests and (2) securely mapping the slices: (a) Considering none of the aspects, (b) Considering the first aspect, (c) Considering both aspects.

acceptable level. Our proposed model works based on RL concepts. We have chosen actor-critic RL as our learning model because the learning environment is too wide, and we need to limit the valid actions according to our constraints. Actor-critic models are a good candidate for such environments.

The main strategy in SIRL is (1) considering appropriate network features to help the agent find optimal mapping solutions that lead to both the maximum possible number of successfully mapped requests and the minimum possible DoS/DDoS damage and (2) reducing the state size of the environment to improve the performance. The details of SIRL are described in what follows.

A. Action space

The agent in a RL model interacts with the environment to learn all its different conditions. In other words, the agent explores the environment to realize its state and then performs an action that transfers the environment from the current state to another valid state. In our case, the substrate network is the environment. The agent receives a single virtual node and is responsible for selecting one of the substrate nodes to be the host of that virtual node. Selecting one of the substrate nodes to host the current virtual node is the action. The agent chooses a number from 1 to N that indicates the index of a substrate node, on which the current virtual node must be mapped.

Considering this action space, whenever the number of passed steps reaches the total number of virtual nodes, one episode of the learning phase will be finished.

B. Environment states

The agent in a RL model interacts with the environment to learn all its different conditions. In our case, the substrate network is a weighted graph that must be provided to the agent. A graph has many different features, such as the nodes and link weights and the way they are connected. However, the dimensions of the environment states must be fixed. For example, if a network consists of N nodes and we represent it to the agent by an $N \times N$ array, the trained model is not suitable for suggesting the solutions for a network consisting of N' ($N \neq N'$) nodes. Therefore, we have to specify the features and the dimension of data that represents a state in our environment exactly. Moreover, we cannot pass the whole parameters of the network to the agent because it is just helpful for a single network that receives specific requests at specific times. As a result, there is a need to extract appropriate network features that can generally model the network.

Our first strategy is to represent the network with an optimal set of features. Five initial features are considered in this regard:

- $\boldsymbol{F_1}$: The remained CPU capacity of each substrate node.
- $\boldsymbol{F_2}$: The sum of the adjacent links remained bandwidth of each substrate node.
- $\boldsymbol{F_3}$: The number of virtual nodes that are currently mapped on each substrate node.
- $\boldsymbol{F_4}$: A binary value for each substrate node that indicates whether or not that substrate node can host the current virtual node.
- $\boldsymbol{F_5}$: The importance value of each substrate node.

Some existing RL models in this field have considered the first two features (i.e., F_1 and F_2) [17, 19]. However, the last three features (i.e., F_3, F_4, and F_5) are suggested by SIRL. The number of currently mapped virtual nodes on each substrate node helps the agent determine the DDoS vulnerability of that substrate node. Assume that there are two substrate nodes with the same features, but one has 3, and the other has 5 mapped virtual nodes. In this case, when the agent selects the latter substrate node and receives a low reward according to the DDoS damage, it can determine the result based on this feature. We have also considered the state of validity of a substrate node for hosting a virtual node. It helps the agent to determine which substrate nodes are currently valid for hosting. The last feature is the importance value of a substrate node. We define the importance value of a substrate node as the sum of the essential values of its adjacent links. The importance value of a substrate link is also defined as the ratio of the remaining CPU capacities of its endpoints to the number of substrate links that have a shared substrate node with that substrate link.

The second strategy SIRL employs involves optimizing the state space size, essentially focusing on the count of potential states, to enhance learning performance. Let's consider an illustrative scenario to underscore the state size's significance. Imagine an environment with four distinct states and two possible actions. In this setup, the agent needs to grasp and adapt to eight unique situations within this environment. Let's presume the environment modeling isn't optimal and is represented with eight states instead. Consequently, the agent is tasked with comprehending and navigating through 16 scenarios. The training duration for the latter model significantly exceeds that of the former. This example vividly illustrates the pivotal role played by the number of states in determining the learning efficacy of an RL model. In instances where models possess a continuous state space, the potential states an agent might encounter are closely tied to the size of the state space. For instance, discretizing a continuous state into 5 bits results in 32 distinct states within that environment. As a result, our approach revolves around minimizing the state size by introducing a novel representation of the environmental states. This strategic reduction in state size serves to enhance the efficiency of the RL model's learning process To the best of our knowledge, in the current RL models in the field of mapping virtual nodes on the substrate node, the environment is passed to the agent with the numeric form of the features. For example, suppose that the capacity of the substrate nodes in a network is represented as $\{5, 4, 4, 10, 2, 3\}$. This representation has some weaknesses. Since different networks have different capacities, one must consider the greatest capacity to dedicate enough memory for the environment state. In other words, each substrate node may vary from 1 to 10; hence, we may have 10^6 different states. This weakness also exists in the normalized features. The normalized features are float numbers that vary from 0 to 1. Assuming that the float numbers are presented with 64 bits, there may be 64^6 different states in this environment.

When a single model is trained, it can be used for different networks. It would be useless if we do not consider the model's generality. Now we explain the new form of representing these numbers. We can rank the nodes by CPU capacity in a network with six substrate nodes. So, the substrate nodes can be represented as $\{4, 2, 2, 5, 0, 1\}$. So, each substrate node may vary from 0 to 5, and only 6^6 different states may be available for the agent. As a result, SIRL uses the ranking of the substrate nodes for each initial feature to generate the final features and create the environment state. This procedure is shown in Algorithm 1, where F'_i is the set of the i^{th} feature values, which is generated based on F_i. A simple sorting algorithm is presented in Algorithm 1 for making it readable. However, in the implementation phase of SIRL, we can use any sorting algorithm instead.

C. Reward function

When the agent takes action within a given state, it subsequently receives a reward that serves as an evaluative measure for its generated solution. The reward function is paramount in any RL model, as it offers the agent a suitable guidance to navigate the problem-solving process. In alignment with our specific objectives of enhancing the coverage of slice requests and mitigating the adverse effects of DoS/DDoS attacks, it becomes imperative to integrate both the count of successfully mapped requests and the extent of DDoS damage into the reward function. By incorporating these factors, the reward function aligns with our overarching goals and steers the agent

Algorithm 1 The procedure of generating the set of final features in SIRL

```
Require: $F_i$, the set of the values of the $i^{th}$ initial feature
Ensure: $F'_i$, the set of the values of the $i^{th}$ final feature
  $S \leftarrow$ an empty array
  for $1 \le j \le N$ do
      $S \leftarrow S + \{\{j, F_i[j]\}\}$
  for $1 \le j \le N$ do
      $min \leftarrow j$
      for $j \le k \le N$ do
          if $S[k][2] < S[min][2]$ then
              $min \leftarrow k$
      $temp \leftarrow S[j]$
      $S[j] \leftarrow S[min]$
      $S[min] \leftarrow temp$
  $F'_i \leftarrow$ an empty array
  for $1 \le j \le N$ do
      $F'_i \leftarrow F'_i + \{j - 1\}$
      if $j \neq 1$ and $S[j][2] = S[j-1][2]$ then
          $F'_i[j] \leftarrow F'_i[j-1]$
  return $F'_i$
```

toward actions that lead to optimal solutions. This holistic approach captures the dual essence of expanding the network's utility by accommodating more requests and bolstering its robustness by minimizing the impact of malicious attacks, thus empowering the RL model to address the intricacies of the problem at hand effectively.

Since SIRL cannot predict future requests due to their random times and features, the remaining resources can serve as a metric for mapping ability. The sum of the remaining bandwidth of the substrate links is an excellent metric for estimating the remaining resources. We excluded the remaining CPU capacity of the substrate nodes from the reward function because mapping solutions consume the same amount of CPU capacity. However, the link bandwidth may be different in different mapping solutions. According to this definition, a network's remaining resource capacity can be calculated as Equation 1, where N is the number of substrate nodes and $rlc(i, j)$ is the remained capacity of the link between the i^{th} and the j^{th} substrate nodes.

$$\text{Re} = \sum_{i=1}^{N} \sum_{j=1}^{N} rlc(i, j) \tag{1}$$

The other metric for defining the reward function in SIRL is the maximum number of virtual nodes mapped on the substrate nodes. This metric helps the agent avoid mapping a large number of virtual nodes on a single substrate node. This metric must be calculated by Equation 2, and only when the whole virtual nodes of a request are mapped. In this equation, N is the number of substrate nodes, and $smv(i)$ is the number of successfully mapped virtual nodes on the i^{th} substrate node.

$$\text{Ma} = \max_{1 \le i \le N} smv(i) \tag{2}$$

If the agent maps the virtual node on an invalid substrate node, it receives an infinitely negative reward. According to these definitions, we defined the reward function of mapping the j^{th}

Algorithm 2 The procedure of training the agent in SIRL

```
Require: $S$, the substrate network
Require: $episodes$, the number of training episodes
Ensure: $ac$, the trained model
  $ac \leftarrow$ initialize the actor-critic model
  for $1 \le e \le episodes$ do
      $moves \leftarrow$ The number of virtual nodes
      $move \leftarrow 0$
      while $move < moves$ do
          $state \leftarrow$ the environment state from Algorithm 1
          $action \leftarrow$ the optimal action from $ac$
          $s \leftarrow action$
          $v \leftarrow move$
          Map the $v^{th}$ virtual node on the $s^{th}$ substrate node
          $reward \leftarrow$ Reward$(s, v)$                ▷ Equation 3
          Update $ac$ based on $state$, $action$, and $reward$
          $move \leftarrow move + 1$
  return $ac$
```

virtual node on the i^{th} substrate node as Equation 3, where α is the condition of having the last virtual node of a request.

$$\text{Reward}(i, j) = \begin{cases} -\infty, & \text{If mapping is invalid} \\ \text{Re} - \text{Ma}, & \text{Else if } \alpha \\ \text{Re}, & \text{Otherwise} \end{cases} \tag{3}$$

An agent's complete process during the training phase is described in Algorithm 2.

IV. Evaluation

To evaluate the performance of our proposed RL model (i.e. SIRL), we simulated different network topologies in Python. PyTorch is used for implementing and training the RL models. We compared the performance of SIRL against nine existing RL models, namely CDRL [13], RDAM [14], VNEQS [16], MLRL [17], DRLVNE [18], GCNNRL [19], A3CGCN [20], DeepViNE [21], PNVNE [22], and A2CRL [23]. Since we focus on the features and their ability to model the environment, we considered the same reward function for all the simulated models to remove the impact of the rewards on the model's performance. Moreover, all the models are trained with the same number of episodes (i.e., 2000). For VNEQS, we have considered the number of virtual nodes mapped on each substrate node as the security level.

Eight different network scenarios are considered in the simulations, and we tried to make them cover different topological features. For example, in one of the scenarios, the network topology is a complete graph, or in another one, we have a leaf node and a node connected to more than half of the nodes. In the simulated scenarios, several random slice requests arrive at random times and last for a random time. However, not more than 20 requests are active at a time. Each request contains at least 2 virtual nodes, which are connected randomly together. Each virtual node and link require a random resource capacity of less than 7 and 5, respectively. While all the features of the requests are random, the exact requests are passed to the models for a fair comparison.

We analyzed four metrics in the simulation results to fairly evaluate the performance of SIRL compared with the other models. The analysis is presented as follows.

A. Requests acceptance ratio

The acceptance ratio is the proportion of successfully mapped requests to the total number of arrived requests. Higher values indicate better performance and goal achievement for maximizing the number of covered slice requests.

The reported results in Figure 2a are the average acceptance ratios of the different scenarios. They are categorized based on the maximum number of requests in the network at a time. We can see a big difference between the number of successfully mapped requests in SIRL and the other models. The acceptance ratio of SIRL is about 70% greater than the average acceptance ratio of the other models. SIRL's acceptance ratio is also about 100% and 19% greater than the other models' minimum and maximum acceptance ratios. This means that SIRL can map more requests on the substrate nodes than the other models. It is worth noting that A3CGCN has the best performance in terms of the acceptance ratio among the other models, except SIRL. One of the reasons could be the appropriate modeling of the environment in A3CGCN with a low state size. The other reason is that A3CGCN, like SIRL, focuses on general network features instead of complex details.

B. Required memory

The required memory for modeling an environment plays an important role in RL models in two aspects. The first point is that the models consuming more memory have space complexity, and they are not appropriate for being implemented on hardware with limited resources. The other important aspect is the agent learning rate. Modeling the environment with a large state size leads to more possible cases that an agent can learn. Assume that the state size is s bits; hence, the number of possible states is 2^s. In each state, the agent can perform a different actions. So, the number of conditions an agent must consider is $a \times 2^s$. When a is fixed, one can improve the learning performance of the agent by reducing s. In other words, modeling the environment states with the lowest number of bits is preferable.

Figure 2b shows the state size of different models in bits. Since the simulations are performed in Python, the float numbers require 64 bits. In all the cases, the number of required bits for modeling the environment state in SIRL is lower than in the other models. The state size of SIRL varies from 65 bits to 187 bits, while the other models require much more space. The difference between the state size of different models is much more tangible as the number of substrate nodes grows. Considering the total scale of the results, we can say that the required memory of SIRL and A2CRL is almost fixed, while that of the other models grows linearly or exponentially. The state sizes of GCNNRL and DeepViNE grow rapidly as the substrate network gets larger, and this is not suitable. The models considering just substrate nodes features have linear growth, and on the other hand, the models that consider

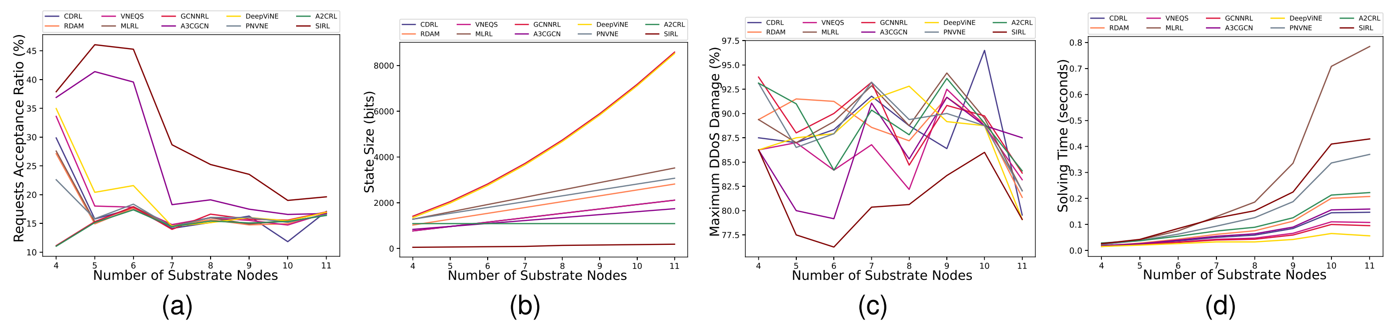

Fig. 2. The performance of SIRL compared with the other RL models. (a) regarding the mapped slice requests. (b) regarding the required memory for modeling the environment. (c) regarding the security level of the slices against a DDoS attack. (d) regarding the required time for finding the optimal solution.

both virtual nodes and links in the state representation have exponential growth. The state size of SIRL is nearly 95% lower than the average state size of the other models. This positive point claims that SIRL can be implemented on devices with limited resources, especially for large-scale networks.

C. DDoS damage

When the adversary performs a Dos/DDoS attack against the substrate nodes, all the mapped slices on that substrate node will be affected. Since all the virtual nodes of a request must collaborate, losing one of them leads to the failure of the whole request [3, 24]. As a result, a good mapping solution must map the virtual nodes so that attacking one of the substrate nodes causes the lowest possible number of slices to be unavailable. We define DDoS damage as the ratio of the average number of requests that are affected by a DDoS attack to the total number of requests. This metric is related to the goal of reducing the impact of DoS/DDoS attacks. In the simulations, we have considered that each substrate node crashes when at least 10 end-hosts send flooded traffic toward it. In each simulation, the adversary selects a random substrate node in the network and commands the compromised hosts to launch the attack against it. The average results of the attacks and their maximum effect on the network are reported in Figure 2c. We can see that in all the scenarios, the average number of slices that crashed after the attack utilizing SIRL is lower than that of the other cases. The results show that the attacks cause about 8% lower damage to the slices when they are mapped on the substrate network using the proposed SIRL scheme. This is because while the reward function is the same for all the models, the features representing the environment in SIRL are much more related to the impact of DDoS attacks on the network than the other features. We can see that the reported maximum DDoS damage does not have a specific trend. This is because of the different features of the eight scenarios. For example, in one scenario, the agent has to map most of the requests on the substrate nodes with a high capacity greater than that of the other nodes. Hence, this node is so vulnerable to attacks.

D. Time to solve

To maintain the users' quality of experience, when a request arrives, it must be mapped on the network as soon as possible. Therefore, a mapping solution has to be found in an acceptable time, and the models that solve the problem of VNE with extravagant time are unsuitable. As a result, we also measured the time a trained model spends solving the mapping problem. The results are shown in Figure 2d. We can see that the solving time grows as the number of slice requests increases. This is because checking the mapping constraints is time-consuming, and more conditions must be checked with more requests. SIRL is the first or the second time-consuming model in all the scenarios. The reason is that SIRL requires some sorting processes to create the states in each step. The required time in SIRL is only 0.07 seconds higher than the average solving time of the other models. It is also worth noting that the solving time of SIRL is about 35% lower than the solving time of MLRL. Since MLRL considers some complex features related to the shortest path, it requires much time. It is reasonable to ignore the solving time of SIRL according to the satisfying results of the acceptance ratio, state size, and DDoS damage.

V. Conclusion

In conclusion, this study introduces the SIRL actor-critic RL model to address the intricate challenge of efficiently mapping slice requests on 5G networks while upholding high acceptance ratios and ensuring slice security. By incorporating novel network features and strategically reducing environmental states, SIRL demonstrates remarkable performance improvements compared to existing models. The simulation results highlight its ability to enhance acceptance ratios and significantly mitigate DoS/DDoS damage. As we move forward, exploring more robust topological features and developing a post-training algorithm stand to further elevate SIRL's effectiveness in optimizing network slicing in 5G environments.

Acknowledgment

This research work is partially supported by the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022, the Research Council of Finland (former Academy of Finland) IDEA-MILL project (Grant No. 352428), the Research Council of Finland (former Academy of Finland) 6G Flagship program (Grant No. 346208), and the European Union's Horizon Europe research and innovation programme HORIZON-JU-SNS-2022 under the RIGOUROUS project

(Grant No. 101095933) and the 6G-SANDBOX project (Grant No. 101096328). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains.

References

[1] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *to appear in IEEE Transactions on Network and Service Management*, 2023.

[2] C. Benzaid, T. Taleb, P. Cao-Thanh, T. Christos, and T. George, "Distributed ai-based security for massive numbers of network slices in 5g & beyond mobile systems," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2021, pp. 401–406.

[3] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: an sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2022.

[4] C. Benzaïd and T. Taleb, "ZSM Security: Threat Surface and Best Practices," *IEEE Network Magazine*, vol. 34, no. 3, pp. 124 – 133, May/June 2020.

[5] C. Benzaïd, T. Taleb, and J. Song, "Ai-based autonomic and scalable security management architecture for secure network slicing in b5g," *IEEE Network*, vol. 36, no. 6, pp. 165–174, Dec. 2022.

[6] C. Benzaïd, M. Boukhalfa, and T. Taleb, "Robust self-protection against application-layer (d) dos attacks in sdn environment," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[7] A. Javadpour and G. Wang, "ctmvsdn: improving resource management using combination of markov-process and tdma in software-defined networking," *The Journal of Supercomputing*, pp. 1–23, 2022.

[8] A. Javadpour, F. Ja'fari, P. Pinto, and W. Zhang, "Mapping and embedding infrastructure resource management in software defined networks," *Cluster Computing*, vol. 26, no. 1, pp. 461–475, 2023.

[9] D. Sattar and A. Matrawy, "Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices," in *in Proc. IEEE Conf. on Communications and Network Security (CNS)*. IEEE, 2019.

[10] J. Ortiz et. al., "Inspire-5gplus: Intelligent security and pervasive trust for 5g and beyond networks," in *in Proc. 15th Int'l Conf. on Availability, Reliability and Security*, Nov. 2020.

[11] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[12] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[13] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, "A continuous-decision virtual network embedding scheme relying on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 864–875, 2020.

[14] H. Yao, B. Zhang, P. Zhang, S. Wu, C. Jiang, and S. Guo, "Rdam: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 901–914, 2018.

[15] H. Cao, G. S. Aujla, S. Garg, G. Kaddoum, and L. Yang, "Embedding security awareness for virtual resource allocation in 5g hetnets using reinforcement learning," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 20–27, 2021.

[16] C. Jiang and P. Zhang, "Vne solution for network differentiated qos and security requirements from the perspective of deep reinforcement learning," in *QoS-Aware Virtual Network Embedding*. Springer, 2021, pp. 61–84.

[17] M. Lu, Y. Gu, and D. Xie, "A dynamic and collaborative multi-layer virtual network embedding algorithm in sdn based on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2305–2317, 2020.

[18] M. Li and M. Lu, "A virtual network embedding algorithm based on double-layer reinforcement learning," *The Computer Journal*, vol. 64, no. 6, pp. 973–989, 2021.

[19] P. Zhang, C. Wang, N. Kumar, W. Zhang, and L. Liu, "Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning," *IEEE Internet of Things Journal*, 2021.

[20] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[21] M. Dolati, S. B. Hassanpour, M. Ghaderi, and A. Khonsari, "Deepvine: Virtual network embedding with deep reinforcement learning," in *in Proc. IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019.

[22] C. Wang, F. Zheng, G. Zheng, S. Peng, Z. Tian, Y. Guo, G. Li, and Y. Yuan, "Modeling on virtual network embedding using reinforcement learning," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 23, p. e6020, 2020.

[23] S. Troia, A. F. R. Vanegas, L. M. M. Zorello, and G. Maier, "Admission control and virtual network embedding in 5g networks: A deep reinforcement-learning approach," *IEEE Access*, vol. 10, pp. 15 860–15 875, 2022.

[24] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *in Proc. IEEE GLOBECOM'22, Rio De Janeiro, Brazil*. IEEE, Dec. 2022.