Reinforcement Learning-based Slice Isolation against DDoS Attacks in Beyond 5G Networks

Amir Javadpour, Forough Ja'fari, Tarik Taleb, and Chafika Benzaïd

***Abstract*—Network slicing in 5G networks can be modeled as a Virtual Network Embedding (VNE) problem, wherein the slice requests must be efficiently mapped on the core network. This process faces two major challenges: covering the maximum number of requests and providing slice isolation. Slice isolation is a mechanism for protecting the slices against Distributed Denial of Service (DDoS) attacks. To overcome these two challenges, we have proposed a novel actor-critic Reinforcement Learning (RL) model, called Slice Isolation-based Reinforcement Learning (SIRL), using five optimal graph features to create the problem environment, the form of which is changed based on a ranking scheme. The ranking procedure reduces the dimension of the features and improves learning performance. We evaluated SIRL by comparing it against four non-RL and nine state-of-the-art RL models. The average results show that the ratio of the covered requests and the damage caused by a DDoS attack of SIRL is 54% higher and 23% lower than that of the other models, respectively. It also has an acceptable learning performance and generality, regarding the reported results that show SIRL agents trained and tested with different networks outperform the other agents by 97%.**

***Index Terms*—5G networks, Slice isolation, Reinforcement Learning (RL), and Distributed Denial of Service (DDoS)**

I. Introduction

THE rapid evolution of cellular networks toward 5G and beyond reveals that a various range of services, including proprietary and personalized ones, are expected to be supported by the currently available resources. 5G networks handle the massive number of service providing requests by network slicing, which is the process of sharing a substrate network between multiple independent virtual networks, called network slices [1]. Supporting the requested services, considering resource limitations, is not the only challenge that the 5G networks are facing. Emerging services, such as the Internet of Things, autonomous transportation, telemedicine, and smart cities, have always been an attractive target for the attackers. Hence, 5G networks have to deal with the security threats against the supported services.

Distributed Denial of Service (DDoS) attacks are one of the security threats targeting 5G slices. The adversary recruits an army of compromised devices, and then commands them to send flooded traffic toward the substrate network hosting the slices. These attacks degrade the performance of the services, and in the intense cases, the service becomes completely unavailable [2]. A DDoS attack becomes more destructive when multiple slices are sharing a common substrate node, which is a common situation due to the nature of network slicing. In such a situation, all the services running on the target substrate node are affected [3].

As most of the existing security researches on 5G networks [4, 5] have stated, slice isolation is a mechanism for protecting the communication channel between the different slices. However, this concept can be extended to a wider area. Considering the definition of slice isolation suggested by Sattar and Matrawy [3], we define slice isolation as any security mechanism that can protect a slice from the threats targeting the other slices. Slice isolation faces two main challenges: (1) covering an optimal number of slice requests, and (2) making the slices as isolated as possible. These challenges are shown in Figure 1. There are four users (on the left side of the figure) and each of them has a custom service that must be embedded on the substrate network (on the right side of the figure). There is also an adversary who launches a DDoS attack against one of the Substrate Nodes (SNs). The circles are the Virtual Nodes (VNs) to be embedded, and the colors specify the requester. The numbers inside the circles and on the links specify the required CPU and bandwidth capacities of the VNs and Virtual Links (VLs), respectively. The Substrate Links (SLs) are considered to have enough resources. In Figure 1(a), all the black VNs are not mapped on the SNs due to CPU resource limitation. Hence, the fourth user is unsatisfied with the service providing performance. The requests of the other three users are completely mapped on the substrate network; However, since at least one of the VNs of their request is mapped on the adversary's target SN, their services become partially available. The embedding scenario of Figure 1(b) is much more acceptable, because the request of the fourth user is also covered, and its VNs are not affected by the attack. Finally, Figure 1(c) shows a better situation, where all the requests are covered, and only two of the users are affected by the attack.

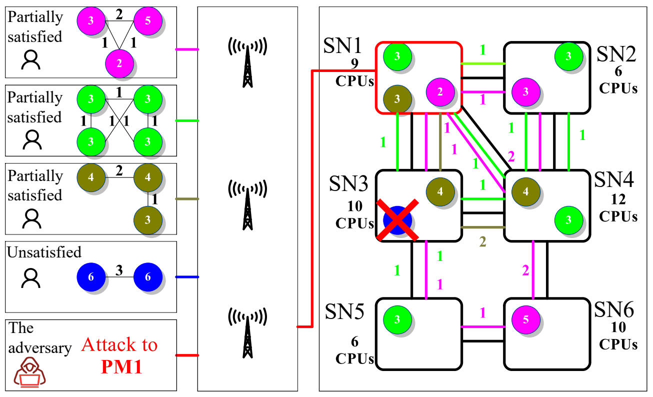
To the best knowledge of the authors, the current solutions for embedding virtual networks on a shared substrate network do not consider slice isolation (i.e., second challenge). Sattar and Matrawy [3] have focused on slice isolation for securing the services, but, on the other hand, the embedding performance (i.e., first challenge) is not discussed. In this paper, we aim to provide slice isolation for beyond 5G networks, wherein the number of covered slices is maximized. To achieve this goal, we have proposed an actor-critic Reinforcement Learning (RL) model, called Slice Isolation based on Reinforcement

Amir Javadpour is with the faculty of Information Technology and Electrical Engineering, University of Oulu (e-mail: a.javadpour87@gmail.com).

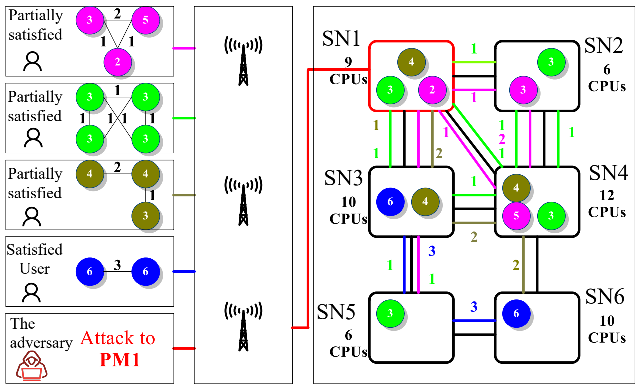
Forough Ja'fari is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: azadeh.mth@gmail.com).

Tarik Taleb and Chafika Benzaïd are with the faculty of Information Technology and Electrical Engineering, Oulu University, Oulu, 90570 Finland (e-mail: tarik.taleb@oulu.fi, and e-mail: and chafika.benzaid@oulu.fi)

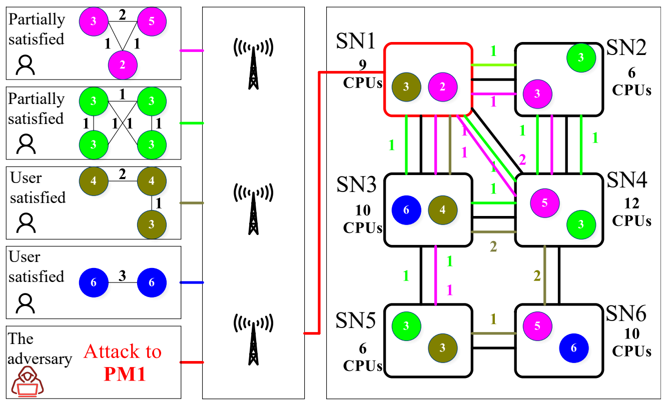
Corresponding authors: Amir Javadpour and Tarik Taleb



(a) Dealing with none of the challenges



(b) Dealing with the first challenge



(c) Dealing with both of the challenges

Fig. 1. A sample scenario of slice isolation challenges, which are (1) covering an optimal number of slice requests, and (2) making the slices as isolated as possible.

Learning (SIRL) designed for solving Virtual Network Embedding (VNE) problems. RL is a type of machine learning approach, in which an agent interacts with the problem environment, and finds the optimal solution based on the achieved reward or punishment [6].

Presenting the problem space to the agent is challenging, due to the various number of features that may influence the final solution. Moreover, leading the agent toward the optimal solution requires a subtle methodology. The proposed method tries it best to overcome these challenges. The key contributions of this paper are as follows:

- Providing a mathematical model for VNE problems that considers time in its representation, defining two mapping objectives for increasing the number of successfully mapped requests, while keeping their security and isolation at an acceptable level, and then proposing an objective function, maximizing which will help in finding the optimal solution based on these objectives;
- Introducing novel features for presenting a network as the problem environment, that better cope with the challenges in maximizing the number of accepted requests and improving slice isolation against DDoS attacks;
- Proposing an optimized actor-critic RL model, that processes the extracted features from the network in a way that improves its generality and learning performance.

The remainder of this paper is structured as follows. Section II reviews different RL models that are proposed for solving VNE problems. In Section III, we have described the problem and represent it with a mathematical model. Section IV presents the proposed RL model, and the algorithms for creating the environment. In Section V, we have evaluated the performance of our proposed model by comparing it with multiple RL models. Finally, Section VI provides a summary of our work and the future directions.

II. Related Work

In this section, we review the different solutions, especially the RL models, for effectively mapping the VNs on the SNs. It is worth noting that several types of research have been conducted in this field. Hence due to the space limitation, we will focus on the most recent ones.

Different types of solutions are currently proposed for solving the problem of VNE. Some of the optimization-based researches use heuristic algorithms in order to rank the SNs and/or VNs regarding their competency to host the VNs and/or their competency to be first mapped on the SNs, respectively. A heuristic function in these algorithms calculates the score of the items that must be ranked. Cao et al. [7] have proposed a ranking algorithm for solving VNE problems, that sorts the SNs based on a heuristic function with number of links, sum of bandwidth, and shortest distance to other nodes as its parameters. This sorting process considers both direct and stable outputs of the heuristic function to rank the nodes. In this research, both the SNs and VNs are sorted, and then, the VN with the highest required resources is mapped on the SN with the highest embedding power. Cao et al. [8] have considered the same topological features as well as node centrality and link interference. Another heuristic ranking method is proposed by Hashmi and Gupta [9], where the parameters involved in the sorting function are the number of links, the available bandwidth, and the CPU capacity of the SNs. Only the SNs are sorted and the VNs are mapped one by one. Shi et al. [10] have focused on solving VNE problems in software-defined networks, where the embedding process can be dynamically managed in real-time. The closeness aggregation of the SNs and the central aggregation of the VNs are the novel topological features considered in this method.

Even though the heuristic ranking techniques are fast, the found solution is not always the optimal one.

Metaheuristic algorithms are another optimization-based solutions, that can seek larger problem spaces for finding the optimal solution rather than ranking approaches, by defining an objective function and attempting to maximize or minimize it. Aguilar-Fuster and Rubio-Loyola [11] have focused on defining an efficient objective function, which leads the problem space searching process with the help of the SNs and VNs degrees. Another metaheuristic approach is proposed by Song et al. [12], in which the data records of the most repeated situations are stored to improve the performance of VNE. Particle swarm optimizer is used for exploring the problem space. There are also other types of metaheuristic solutions that focus on genetic algorithms, in which some random solutions (i.e., the chromosomes that are made up of genes) are generated, and then the poor ones are omitted, and the others are improved by crossover and mutation phases. The improvement phases try to minimize/maximize the output of a fitness function. The fitness function and the structure of the chromosomes in these algorithms are designed based on the VNE problem. Zhang et al. [13] have proposed a genetic solution for multi-domain VNE problems, where two or more infrastructure providers share the same substrate network, while managing their own users and requests. The number of genes of each chromosome in this research is equal to the total number of SNs, and different genes indicated different VNs that are mapped on that SN. They have defined the fitness function as the amount of available CPU and bandwidth of the substrate network. The goal is to maximize the remained resources after mapping the VNs by maximizing the fitness function. The method proposed by Zhuang et al. [14] is another example of genetic-based solutions of VNE. In this method, the number of chromosomes are equal to the total number of VNs, and the genes indicated which of the SNs are hosting that VN. This solution considers the amount of consumed bandwidth of the substrate network as the fitness function, which has to be minimized. The genetic algorithms suffer from high complexity when the problem space becomes larger, and this why RL solutions are suggested.

Yuan et al. [15] proposed a RL model for solving VNE problems with multiple agents, each of which finds the optimal mapping solution for a single request. Since the focus of this model is to optimize the agent strategies, it is not mentioned which features are considered for learning the environment. Thakkar et al. [16] also proposed a RL model that solves the problem of VNE in two stages. The first stage assigns a binary value to each of the SNs, indicating whether the current VN can be mapped on them. In the second stage, a probability is assigned to the SNs that indicates their suitability for hosting the VN. In this research, the main focus is on the learning process, and therefore, the exact features of the SNs and VNs are not mentioned.

Yao et al. [17] proposed a RL model, called CDRL (Continuous-Decision virtual network embedding scheme relying on Reinforcement Learning), for solving VNE problems, in which three main features are considered for each of the SNs: the CPU capacity, the sum of the bandwidth of the adjacent link, and the node degree (i.e., the number of connected links). Yao et al. [18] also presented a model, RDAM (Reinforcement learning based Dynamic Attribute Matrix representation), that uses the same features of CDRL, with an extra one: the average distance to the other nodes. Cao et al. [19] also deployed a RL model based on CDRL with an extra security level feature to protect the slices and the substrate nodes of the 5G network.

Jiang and Zhang [20] proposed a RL model that considers both quality of service and the security level of each VN in the process of solving the VNE problem. We call this model VNEQS, short for Virtual Network Embedding for Quality of service and Security. The nodes safety level and the delay are notable features of this model. Lu et al. [21] proposed a RL model, MLRL (Multi-Layer virtual network embedding algorithm based on Reinforcement Learning), 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, degree centrality, closeness centrality, and betweenness centrality, are considered in MLRL for modeling the environment. Li and Lu [22] also proposed a similar RL model, called DRLVNE (Double-layer Reinforcement Learning-based VNE algorithm), with two extra features: the average distance to the other nodes and eigenvector centrality.

Zhang et al. [23] proposed an actor-critic RL model that uses a graph convolution neural network to extract the environment features automatically. We call this model GCNNRL, short for Graph Convolution Neural Network with Reinforcement Learning. The number of VNs and VLs that are successfully mapped are considered as mapping state features. Yan et al. [24] proposed a deep RL model, A3CGCN (Asynchronous Advantage Actor-Critic Graph Convolutional Network), to automatically solve the problem of VNE, in which the features of the substrate and virtual network are extracted using a graph convolutional network. The notable features that represent the environment in A3CGCN are a binary value that indicates whether one of the previous VNs of the current request is mapped on a SN, and the number of VNs in the current request that is not still mapped.

Dolati et al. [25] proposed a deep RL model, called DeepViNE (Deep reinforcement learning Virtual Network Embedding), in which the substrate network graph is encoded as an image considering the spatial locality property. The features of the SNs and VNs construct the processes image. Wang et al. [26] proposed a RL model, called PNVNE (Pointer Network VNE), for solving VNE problems, which utilizes an attention mechanism that selects the most suitable substrate nodes, hence, the model can focus on them. Troia et al. [27] proposed an Advantage Actor-Critic RL (A2CRL) model for mapping slice requests in a 5G network on the SNs, as a special form of VNE problems. The main focus of this model is to decide whether to accept a request or not. The notable feature considered by A2CRL is the percentage of leaf nodes in the network.

None of these RL models, which are summarized in Table I, focus on the damage caused by a DDoS attack against the slices. Moreover, generality of the models are not considered, and this is challenging in large-scale networks. As a result, we propose a novel RL model to cope with these weaknesses.

III. Problem Definition

In this section, we describe the network model and the symbolic presentation of each element in a 5G network, which are summarized in Table II and Table III. We also define the VNE problem using three constraints, and then explain our two objectives in solving such problems. 5G networks commonly consist of user devices, radio access networks, core networks, network slice managers, and controllers [28]. Due to the scope of this paper, we are focusing on the core network, and hence, modeling the other components is not taken into consideration. Since in VNE problems, the physical network is called the substrate network, we refer to the core network as the substrate network in the remaining of this paper. It is worth noting that all the sets that are presented in this section are ordered sets.

The substrate network, $\mathcal{S}$, can be written as a tuple like $(\mathcal{N}, \mathcal{L}, \mathcal{R})$, where $\mathcal{N}$ is the set of substrate network nodes capacity, $\mathcal{L}$ is the matrix of substrate network links capacity, and $\mathcal{R}$, is the set of requested slices that arrive over the time.

$\mathcal{N}$ is written as $\{n_1, n_2, \ldots, n_N\}$, where N is the total number of nodes in the substrate network and n_i is the i^{th} node capacity. L is a square $N \times N$ matrix, in which the element in the i^{th} row and j^{th} column is shown by $l_{(i,j)}$. $l_{(i,j)}$ is the capacity of the link that connects the i^{th} and j^{th} nodes in the substrate network. Since the substrate network is modeled as an undirected graph, we have $l_{(i,j)} = l_{(j,i)}$.

$\mathcal{R}$ is written as $\{r_1, r_2, \ldots, r_R\}$, where R is the total number of requests and r_i is the i^{th} arriving request. If we are focusing on a limited time interval, R is an integer, otherwise, it is ∞. Each of the requests arrives at and lasts for a specific time, and also asks for a VN to be mapped on the SNs and links in the substrate network. Hence, we can show r_i with a tuple like $(t_i, d_i, \mathcal{N}_i, \mathcal{L}_i, \mathcal{M}_i)$, where t_i and d_i are the arrival time of the i^{th} request and its duration, respectively, $\mathcal{N}_i$ is the set of VNs capacity of the i^{th} request, $\mathcal{L}_i$ is the matrix of VLs capacity of the i^{th} request, and $\mathcal{M}_i$ is the set of substrate network nodes, on which the VNs of the i^{th} request are mapped. We have $\mathcal{N}_i = \{n_1^i, n_2^i, \ldots, n_{N_i}^i\}$, where N_i is the total number of VNs and n_j^i is the j^{th} VN capacity in the i^{th} request. $\mathcal{L}_i$ is also a square $N_i \times N_i$ matrix, in which the element in the j^{th} row and k^{th} columns is shown by $l_{(j,k)}^i$. $l_{(j,k)}^i$ is the capacity of the link that connects the j^{th} and k^{th} VNs in the i^{th} request. Like the substrate network, the requests are also modeled as an undirected graph, and therefore, we have $l_{(j,k)}^i = l_{(k,j)}^i$. Finally, we can write $\mathcal{M}_i$ as $\{(m_1^i, t_1^i), (m_2^i, t_2^i), \ldots, (m_{N_i}^i, t_{N_i}^i)\}$, where m_j^i is the substrate network node index, on which the j^{th} VN in the i^{th} request is mapped, and t_j^i is the time that this mapping is performed. If the VN is not still mapped, the value of m_j^i and t_j^i are zero and ∞, respectively. If all the VNs of a request are mapped at once, the mapping time is the same. It must be noted that the value of t_j^i for all valid i and j is always lower than $t_i + d_i$. Because if a request is not mapped until its deadline, the mapping is failed. Moreover, on the departure of a request, the related m_j^i becomes zero again.

In our proposed model, the process of mapping the VLs starts after all the VNs of a single request are mapped. Moreover, we map the VLs on the shortest path between the SNs, on which the VNs of that link is mapped. If two paths have the same number of edges (i.e., hops), we select the path with the highest minimum weight of the edges.

We define a binary function, called $\alpha(t, i, j, k)$, that returns one if the k^{th} VN of the j^{th} request is mapped on the i^{th} SN at time t, and otherwise it returns zero. In other words, we can calculate this function as defined in Equation 1. Based on this equation, the value of α is one, only if the request is still active in the current time, and the value of m is equal to the SNs index.

$$\alpha(t, i, j, k) = \begin{cases} 0, & \text{if } t < t_k^j \\ 0, & \text{else if } t > t_j + d_j \\ 1, & \text{else if } m_k^j = i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We also define a similar binary function, $\beta(t, i, i', j, k, k')$, that returns one if the link between the k^{th} and k'^{th} VNs of the j^{th} request is mapped on the link between the i^{th} and i'^{th} SNs at time t. Assuming that the shortest path between the i^{th} and i'^{th} SNs is $p^{(i,i')}$, and it contains $P^{(i,i')}$ edges, in which the e^{th} edge is between $p_e^{(i,i')}$ and $p_{e+1}^{(i,i')}$ SNs, the β function can be calculated as Equation 2, where π is the condition of $p_e^{(m_k^j, m_{k'}^j)} = i \land p_{e+1}^{(m_k^j, m_{k'}^j)} = i'$ or $p_e^{(m_k^j, m_{k'}^j)} = i' \land p_{e+1}^{(m_k^j, m_{k'}^j)} = i$. According to this equation, the value of β is one, only if the related request is still active, the endpoints of the VL are successfully mapped, and the endpoint SNs are in the shortest path.

$$\beta(t, i, i', j, k, k') = \begin{cases} 0, & \text{if } t < t_k^j \text{ or } t < t_{k'}^j \\ 0, & \text{else if } t > t_j + d_j \\ 0, & \text{else if } m_k^j = 0 \text{ or } m_{k'}^j = 0 \\ 1, & \text{else if } \exists e \in \mathbb{N}_{P^{(m_k^j, m_{k'}^j)}} : \pi \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

To check whether a request is completely mapped on the substrate network, we define a binary function, $\gamma(t, i)$, that returns one or zero if, until time t, the i^{th} request is completely mapped or not mapped, respectively. We can define this function as Equation 3. This equation says that a request is completely mapped, only if a number, except zero, is assigned to all of its VNs related m.

$$\gamma(t, i) = \begin{cases} 1, & \text{if } t \geq t_i \text{ and } \forall j \in \mathbb{N}_{N_i} : m_j^i \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Another function is $\delta(t, i)$, which returns the number of requests that are mapped on the i^{th} SN at time t. In subsection III-A, we will discuss the constraints for mapping the VNs, and based on Constraint 1, two VNs of a same request cannot be mapped on a same SN. Hence, we can say that $\delta(t, i)$ is the number of VNs mapped on the i^{th} SN at time t. This function is calculated according to Equation 4. If the value of α is one, the related VN is mapped on the target SN.

TABLE I
A SUMMARY OF THE RL-BASED VNE SOLUTIONS AND THEIR MAIN CHARACTERISTICS

Model name	Substrate features					Virtual features			Defined actions	Reward parameters	Time complexity
	CPU	Bandwidth	Degree	Distances	Embedding	CPU	Bandwidth	Embedding			
CDRL [17]	●	●	●						Selecting one of the SNs	Covered and consumed amount of resources	$O(\text{SNs}^4)$
RDAM [18]	●	●	●	●					Selecting one of the SNs	Covered and consumed amount of resources	$O(\text{SNs}^4)$
VNEQS [20]	●	●							Selecting one of the SNs	Covered and consumed amount of resources	$O(\text{requests} \times \text{VNs} \times \text{VLs})$
MLRL [21]	●	●		●					Selecting one of the SNs	Covered and consumed amount of resources	$O(\text{VLs} \times \text{SNs}^3)$
GCNNRL [23]	●	●				●	●	●	Selecting one of the SNs	Covered and consumed amount of resources	$O(\text{requests} \times \text{SN} \times \text{VNs})$
A3CGCN [24]	●	●			●	●	●	●	Selecting one of the SNs	The VNs indices	$O(\text{SL} \times \text{VLs})$
DeepViNE [25]	●				●	●	●	●	Deciding to map on the neighbor SNs	Covered and consumed amount of resources	$O(\text{VNs} \times \text{SLs} \times \text{VLs})$
PNVNE [26]	●	●				●	●		Selecting one of the SNs	The consumed bandwidth	$O(\text{SNs} \times \text{SLs} \times \text{VLs})$
A2CRL [27]	●		●	●	●				Deciding whether to accept the current request or not	Number of involved hops and consumed CPU	$O(\text{SNs} \times \text{SLs} \times \text{VLs})$
SIRL (this work)	●	●			●				Selecting one of the SNs	Number of involved hops and maximum number of VNs on a single SN	$O(\text{requests} \times \text{SNs}^3)$

Hence, This equation calculates the sum of α values for all the VNs of all the requests.

$$\delta(t,i) = \sum_{j=1}^{R} \sum_{k=1}^{N_j} \alpha(t,i,j,k) \tag{4}$$

If the number of edges that are involved in mapping the VLs connecting the j^{th} VN of the i^{th} request is shown with $\rho(i,j)$, we have Equation 5. If the value of β is one, the related VL is mapped on the target SL. Hence, calculating the sum of β values results in the number of involved edges.

$$\rho(i,j) = \sum_{l=1}^{N} \sum_{l'=i+1}^{N} \sum_{k=1}^{N_i} \beta(t_j^i, l, l', i, j, k) \tag{5}$$

For calculating the consumed resources, we define two functions as σ and φ. $\sigma(t,i)$ returns the consumed CPU of the i^{th} SN, and $\varphi(t,i,i')$ returns the consumed bandwidth of the link between the i^{th} and i'^{th} SNs at time t. These functions are calculated based on Equation 6 and Equation 7, respectively. In these two equations, we have to calculate the sum of n and l values. If they are multiplied by the values of α and β, the not-mapped nodes and the links are also not considered.

$$\sigma(t,i) = \sum_{j=1}^{R} \sum_{k=1}^{N_j} n_k^j \times \alpha(t,i,j,k) \tag{6}$$

$$\varphi(t,i,i') = \sum_{j=1}^{R} \sum_{k=1}^{N_j} \sum_{k'=k+1}^{N_j} l_{(k,k')}^{j} \times \beta(t,i,i',j,k,k') \tag{7}$$

We also define a binary function, $\tau(i,i',j,j')$, that returns one if the SL between the i^{th} and i'^{th} SN has a shared node with the SL between the j^{th} and j'^{th} SN. Hence, we can calculate this function based on Equation 8. Two links have a shared node, only if one of their endpoints are equal.

$$\tau(i,i',j,j') = \begin{cases} 0, & \text{if } i = i' \text{ or } j = j' \\ 1, & \text{else if } i = j \text{ or } i = j' \text{ or } i' = j \text{ or } i' = j' \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The last defined function is $\lambda(i,i')$, which returns the number of SLs that have a shared node with the link between the i^{th} and i'^{th} SN. The value of this function can be calculated by Equation 9. In this equation, we have to consider all the possible endpoints.

$$\lambda(i,i') = \sum_{j=1}^{N} \sum_{j'=j+1}^{N} \tau(i,i',j,j') \tag{9}$$

A. Mapping constraints

There are some constraints for mapping the requests on the substrate network at each time t.

TABLE II
THE LIST OF NOTATIONS USED FOR MODELING THE NETWORK AND THEIR DESCRIPTION

Notation	Description
$\mathcal{S}$	The substrate network
$\mathcal{N}$	The set of substrate network nodes capacity
$\mathcal{L}$	The matrix of substrate network links capacity
$\mathcal{R}$	The set of requested slices
$\mathcal{R}^t$	The set of requests that last till time t
N	The total number of SNs
n_i	The i^{th} SN capacity
$l_{(i,j)}$	The capacity of the link between the i^{th} and j^{th} SNs
R	The total number of arrived requests during time
r_i	The i^{th} arrived request
t_i	The arrival time of the i^{th} request
d_i	The duration that the i^{th} request lasts in the network
$\mathcal{N}_i$	The set of the i^{th} request VNs capacity
$\mathcal{L}_i$	The matrix of the i^{th} request VLs capacity
$\mathcal{M}_i$	The set of SNs, on which the VNs of the i^{th} request are mapped
N_i	The total number of the i^{th} request VNs
n^i_j	The j^{th} VN capacity of the i^{th} request
$l^i_{(j,k)}$	The capacity of the link between the j^{th} and k^{th} VNs in the i^{th} request
m^i_j	The index of the SN, on which the j^{th} VN of the i^{th} request is mapped
$p^{(i,i')}$	The shortest path between the i^{th} and i'^{th} SNs
$P^{(i,i')}$	The number of edges in the shortest path between the i^{th} and i'^{th} SNs
$p^{(i,i')}_e$	The e^{th} node in the shortest path between the i^{th} and i'^{th} SNs

Constraint 1. *Two VNs in the same request must not be mapped on a shared SN. This constraint is equivalent to Equation 10.*

$$\forall r_i \in \mathcal{R}^t : \nexists j,k \in \mathbb{N}_{N_i} : j \neq k \wedge m^i_j = m^i_k \tag{10}$$

Constraint 2. *The sum of the capacities of all the VNs must not exceed the capacity of their host SN. This constraint is equivalent to Equation 11.*

$$\forall i \in \mathbb{N}_N : \sigma(t,i) \leq n_i \tag{11}$$

Constraint 3. *The sum of the capacities of all the VLs must not exceed the capacity of their host SL. This constraint is equivalent to Equation 12.*

$$\forall i,i' \in \mathbb{N}_N : \varphi(t,i,i') \leq l_{(i,i')} \tag{12}$$

B. Mapping objectives

In this paper, we aim to find an optimized mapping solution that can map as many requests on the substrate network while keeping the effect of a DDoS attack as low as possible. So the objectives of mapping the requests, after considering Constraint 1, Constraint 2, and Constraint 3, are as follows.

Objective 1. *A mapping solution must map the maximum possible number of requests on the substrate network. In other words, we want to increase the number of completely mapped requests, which is equivalent to Equation 13.*

$$\textit{Maximize } \left(\sum_{i=1}^{R} \gamma(t,i) \right) \textit{ for all } t \tag{13}$$

Objective 2. *A mapping solution must map the requests on the substrate network in a way that whenever the adversary performs a DDoS attack against one of the SNs, the minimum possible number of slices are affected. In other words, the maximum number of VNs that are mapped on the SNs must be kept low. This objective is equivalent to Equation 14.*

$$\textit{Minimize } \left(\max_{i \in \mathbb{N}_N} \delta(t,i) \right) \textit{ for all } t \tag{14}$$

In training the agent of an RL model, the time steps are the steps of mapping the VNs, and the requests arrival time step is the next step after the mapping process of the previous request is over. Hence, we have a discrete time VNE problem, where the mapping time of the i^{th} and the j^{th} VNs of the r^{th} request follows $t^r_i < t^r_j$, if and only if $i < j$. Based on this constraint, we have to check if a request is completely mapped, only after mapping its last VN is done. Considering these situations, we can combine Objective 1 and Objective 2 into a function, $obj(t,i,j)$, that must be maximized. This function is presented in Equation 15.

$$\begin{aligned} obj(t,i,j) = &\Psi \left(\frac{m^i_j}{\rho(i,j) \times \max\limits_{k \in \mathbb{N}_N} \delta(t,k)} + m^i_j - 1 \right) + \\ &\Phi \left(\lfloor \frac{j}{N_i} \rfloor (2\gamma(t,i) - 1) \right) \end{aligned} \tag{15}$$

The values of Φ and Ψ depend on the importance of achieving success in complete and partial mapping processes, respectively.

IV. PROPOSED REINFORCEMENT LEARNING MODEL FOR SLICE ISOLATION

A 5G network is always listening for the slice requests and its service performance depends on the slices which are successfully mapped. However, the security performance must be also considered. For example, assume a 5G network that receives five slice requests in two cases. In the first case, all five requests are mapped; however, there are many shared substrate nodes, and by a simple DDoS attack from the adversary, all of them will become out of service. Only four of the requests are mapped in the second case, while the number of shared SNs is reduced, and a DDoS attack causes only one of the slices to become out of service. From the security point of view, the second case is preferable.

In our proposed model, dubbed SIRL, we aim to map as many slice requests as possible while keeping their availability at an acceptable level. In other words, we try to achieve Objective 1 and Objective 2. Our proposed model works based on actor-critic RL concepts. Actor-critic models are a combination of value-based and policy-based RL approaches. The agent in a value-based RL model chooses the action that is experienced to lead in a higher reward for the current state. On the other hand, in policy-based models, the agent attempts to find a function that can probably map the states on to the actions. In actor-critic models, the actor, which is the policy-based part, decides the optimal action to be performed in each situation, and the critic, which is the value-based part, helps it

TABLE III
THE LIST OF DEFINED FUNCTIONS USED FOR MODELING THE NETWORK AND THEIR DESCRIPTION

Function	Description	Equation
$\alpha(t,i,j,k)$	A binary function that returns one if the k^{th} VN of the j^{th} request is mapped on the i^{th} SN at time t, and otherwise, returns zero.	Equation 1
$\beta(t,i,i',j,k,k')$	A binary function that returns one if the link between the k^{th} and k'^{th} VNs of the j^{th} request is mapped on the link between the i^{th} and i'^{th} SNs at time t, and otherwise, returns zero.	Equation 2
$\gamma(t,i)$	A binary function that returns one if the i^{th} request is completely mapped until time t, and otherwise, returns zero.	Equation 3
$\delta(t,i)$	A function that returns the number of VNs that are mapped on the i^{th} SN at time t.	Equation 4
$\rho(i,j)$	A function that returns the number of edges involved in mapping the VLs connected to the j^{th} VN of the i^{th} request.	Equation 5
$\sigma(t,i)$	A function that returns the consumed CPU of the i^{th} SN at time t.	Equation 6
$\varphi(t,i,i')$	A function that returns the consumed bandwidth of the link between the i^{th} and i'^{th} SNs at time t.	Equation 7
$\tau(i,i',j,j')$	A binary function that returns one if the SL between the i^{th} and i'^{th} SN has a shared node with the SL between the j^{th} and j'^{th} SN, and otherwise, returns zero.	Equation 8
$\lambda(i,i')$	A function that returns the number of SLs that have a shared node with the SL between the i^{th} and i'^{th} SN.	Equation 9
$obj(t,i,j)$	The proposed objective function, maximizing which can help in finding the optimal solution of VNE.	Equation 15

by specifying the efficiency of the applied action and how to improve it. Actor-critic RL models have higher performance than value-based and policy based RL models, by addressing some of their individual weaknesses, such as scalability and convergence issues [29]. Actor-critic models are a good candidate for the problems that have both short-term and long-term rewards. The actor finds the optimal action considering the short-term rewards, while the critic warns it about the long-term rewards. In the problem of VNE, the mapping goals are classified into partial and complete mappings. For the partial mapping goal, the focus is on successfully mapping a VN on a SN, and in the complete mapping goal, the focus is to make the whole request successfully mapped. Reaching the first goal requires a short-term reward, while the long-term one is to completely map all the VNs and VLs successfully.

The interaction between the agent and the environment is for learning all its different states and features. In our problem, 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 links weights, and the way they are connected. However, passing the whole S to the agent is not rational, due to the time-consuming processes involved in percepting and learning it, especially for large-scale graphs. As a result, there is a need of extracting appropriate graph features which can generally model the network. Moreover, if the environment passes through some repeated states, the learning performance improves. Because the agent has previously faced that state, and the new predicted action is enhanced by the experiment of the previous one. This idea also reduces the number of states in an environment. The lower number of states are quicker to learn, as the agent spends less time to explore the environment for finding the optimal solution [30]. When the number of states are lower, a wider range of network configurations may result in a same state. Therefore, if a model is trained with a single network, it can generate the optimal solution for many other networks with different conditions. Regarding these points, we have applied two strategies in the proposed learning model: (1) considering appropriate network features helping the agent to find optimal mapping solutions that lead to both the maximum possible number of successfully mapped requests and the minimum possible DDoS damage and, and (2) reducing the number of states of the environment to improve the learning performance.The details of SIRL are described in what follows.

A. Action space

The agent in a RL model explores the environment to realize its states 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 VN and its responsibility is to select one of the SNs to be the host of that VN. Selecting one of the SNs to be the host of the current VN is the action. The agent chooses a number from 1 to N that indicates the index of a SN, on which the current VN must be mapped.

Considering this action space, whenever the number of passed steps reaches the total number of VNs, one episode of the learning phase will be finished. It is worth noting that for a request, whenever the agent maps a VN on a SN, the VLs, the endpoint of which are currently mapped during the previous steps, are also mapped on the shortest path between the related SNs. For example, if the agent has mapped the first three VNs of a request, when the fourth VN is mapped, the links between the fourth VN and the first three ones are also mapped. In the case that one of the VNs or VLs of a request cannot be mapped, the agent unmaps the currently mapped VNs/VLs of that request, and also skips the steps to the other request.

B. Environment states

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 SN
- $\boldsymbol{F_2}$: The sum of the adjacent links remained bandwidth of each SN
- $\boldsymbol{F_3}$: The number of VNs that are currently mapped on each SN

- $\boldsymbol{F_4}$: A binary value for each SN that indicates whether or not that SN can be the host of the current VN
- $\boldsymbol{F_5}$: The importance value of each SN

Some of the existing RL models in this field have considered the first two features (i.e., F_1 and F_2). However, the last three features (i.e., F_3, F_4, and F_5) are suggested by SIRL. The number of currently mapped VNs on each SN helps the agent determine the DDoS vulnerability of that SN. Assume that there are two SNs with the same features, but one of them has 3 and the other has 5 mapped VNs. In this case, when the agent selects the latter SN and receives a low reward according to the DDoS damage, it can figure out the result based on this feature. We have also considered the state of validity of a SN for hosting a VN. It helps the agent figure out which of the SNs are currently valid for hosting. It is worth noting that the validity of hosting in feature F_4 is only about Constraint 1 and Constraint 2. The last feature is the importance value of a SN. We define the importance value of a SN as the sum of the importance values of its adjacent links. The importance value of a SL is also defined as the ratio of the remained CPU capacities of its endpoints to the number of SLs that have a shared SN with that SL (i.e., the value of λ function). The importance value of a link indicates the importance of its existence in increasing the whole resources of the network. For example, if we have a node with a lot of resources, which is connected to only one link, and this links becomes out of resources to map new virtual links, that node's resources are wasted. But if this node is connected to other links as well, the existence of the mentioned link is not of high importance.

Considering that $F = \{F_1, F_2, F_3, F_4, F_5\}$ is the set of mentioned initial features, Algorithm 1 shows how it is calculated. F_1, F_3, and F_4 are calculated in the first loop of this algorithm (line 2 to line 11), where each iteration is dedicated for each of the SNs. The remained CPU capacity of a SN is the difference between its total CPU and the consumed amount (line 3). F_3 stores the number of VNs mapped on each SN, and hence, it is enough to append the value of $\delta(t, i)$ to this set (line 4). If the requested CPU is higher than the SN remained resource, it cannot handle the VN. Hence, zero is appended to F_4 (line 5). Otherwise, we have to first check whether the previous VNs of the same request are mapped on this SN (line 9). If the mapping is valid, one, and otherwise, zero is appended to F_4 (line 10 and line 11). The second loop of the algorithm (line 12 to 17) is for generating F_2 and F_5. A nested loop is required for considering all the edges, by iterating over their endpoints (line 14). The remained bandwidth is calculated as the difference between the total and consumed bandwidth resources (line 15). Then, if there is a link between the two SNs (line 16), the importance value is updated by adding the endpoints remained CPU and dividing it by λ (line 17). It is worth noting that the loops involved in Algorithm 1 are for making the algorithm readable. However, in the implementation phase of SIRL, one can use dynamic programming to reduce its time complexity.

The second strategy of SIRL is to optimize the state size, or in other words, the number of possible states, to improve the learning performance. Here, we give an example to show the importance of state size. Suppose that an environment has four states and two actions. In this condition, the agent has to learn eight different situations in this environment. Now suppose that the modeling of this environment is not optimal and one models it with eight states. Hence, the agent now must learn 16 different situations. The time to train the latter model is much more than the training time of the former model. This example shows that the number of states plays an essential role in the learning performance of a RL model. In the models with continuous state space, the number of possible states the agent may face is directly related to the state size. For example, if a continuous state is modeled with 5 bits, there are 32 different states in that environment. As a result, we try to reduce the state size by presenting the environment states in a new form.

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

```
Require: $S$ (the substrate network)
Require: $t$ (the current time)
Require: $r$ (the index of the current request)
Require: $v$ (the index of the current VN)
Ensure: $F$ (the set of initial features)
    $F_1, F_2, F_3, F_4, F_5 \leftarrow \{\}, \{\}, \{\}, \{\}, \{\}$
 2: for $1 \le i \le N$ do
        $F_1 \leftarrow F_1 + \{n_i - \sigma(t, i)\}$
 4:     $F_3 \leftarrow F_3 + \{\delta(t, i)\}$
        if $n_v^r > F_1[i]$ then $F_4 \leftarrow F_4 + \{0\}$
 6:     else
            $valid \leftarrow 1$
 8:         for $1 \le j < v$ do
                if $m_j^r = i$ then $valid \leftarrow 0$ break
10:         if $valid = 1$ then $F_4 \leftarrow F_4 + \{1\}$
            else $F_4 \leftarrow F_4 + \{0\}$
12: for $1 \le i \le N$ do
        $F_5 \leftarrow F_5 + \{0\}$
14:     for $1 \le i' \le N$ do
            $F_2 \leftarrow F_2 + \{l_(i, i') - \varphi(t, i, i')\}$
16:         if $l_(i, i') \ne 0$ then
                $F_5[i] \leftarrow F_5[i] + (F_1[i] + F_1[i'])/\lambda(i, i')$
18: return $\{F_1, F_2, F_3, F_4, F_5\}$
```

To the best of our knowledge, in the current RL models in the field of mapping VNs on the SNs, the environment is passed to the agent with the numeric form of the features. For example, in the environment of ??, the SNs are 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 SN may vary from 1 to 10, and 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. If we assume 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. If we do not consider the model's generality, it would be useless. Now we explain the new form of representing these numbers. In a network with six SNs, we can rank the nodes by their CPU capacity. So, the SNs can be represented

Algorithm 2 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)
    $S$, $F_i' \leftarrow \{\}, \{\}$
 2: for $1 \leq j \leq N$ do
        $S \leftarrow S + \{\{j, F_i[j]\}\}$
 4: Sort the tuples in $S$ based on the second element
    for $1 \leq j \leq N$ do
 6:     $F_i' \leftarrow F_i' + \{j-1\}$
        if $j \neq 1$ and $S[j][2] = S[j-1][2]$ then
 8:         $F_i'[j] \leftarrow F_i'[j-1]$
    return $F_i'$
```

in the form of $\{4, 2, 2, 5, 0, 1\}$. So, each SN 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 SNs for each of the initial features to generate the final features and create the environment state. This procedure is shown in Algorithm 2, where F_i' is the set of the i^{th} feature values, which is generated based on F_i. In the sorting phase of Algorithm 2, one can use any sorting algorithm to sort the tuples in S ascendingly. We have used merge sort, with the complexity order of $O(nlogn)$, in order to keep the time complexity of our proposed method low.

C. Reward function

When the agent performs an action in a state, it will receive a reward that evaluates the solution found by the agent. The reward function is important in a RL model due to its ability in giving the agent an appropriate direction in solving the problem. For example, a negative reward is like a punishment for the agent to avoid the situations resulting it, and a big positive reward is the sign of a good action in that state. According to our objectives, Objective 1 and Objective 2, we should consider both the number of successfully mapped requests and the DDoS damage in the reward function.

Our agent faces four different situations, based on its selected action:

- Successful partial mapping: If the agent successfully maps a VN and its related VLs, the reward must be a positive value to encourage it for mapping more and more VNs. However, this reward is based on the superiority of agent's action against other possible ones. Since the requests arrive at random times with random features, SIRL does not know the future requests to predict the number of successfully mapped ones. Hence, we can consider the remained resources as a metric that can predict the ability to map the future resources. Different mapping solutions consume the same amount of CPU capacity. Hence, considering it in the reward function does not lead the agent toward the optimal solution. On the other hand, the available bandwidth is not always the same for different mapping solutions. It depends on the number of hops (i.e., edges) that the VLs are mapped on. Moreover, we have to consider the number of VNs that are mapped on each SN, in order to reduce the DDoS damage. We have chosen the maximum number of VNs mapped on the SNs to estimate the DDoS damage in the reward function. We have to design the reward function in a way that the agent is leaded to keep the maximum number of VNs mapped on a same SN and the number of hops for mapping the VLs low. A fraction, such as $1/(hops \times max\_VNs)$, becomes higher when the value of $hops$ and max_VNs are reduced. As a result, if we call the Successful Partial mapping reward of the j^{th} VN of the i^{th} request at time t as $\text{SP}(t, i, j)$, we have Equation 16.

$$\text{SP}(t,i,j) = \frac{+1}{\rho(i,j) \times \max_{k \in \mathbb{N}_N} \delta(t,k)} \tag{16}$$

- Unsuccessful partial mapping: If there is a failure in the partial mapping process, the agent must be punished with a negative reward. For this case, we have considered -1 points.
- Successful overall mapping: Completely mapping a request is much more important than mapping the individual VNs. As a result, whenever the agent succeeds in the overall mapping process, a big positive reward, which is much higher than the successful partial mapping reward, must be received. We have selected +500 points for this situation.
- Unsuccessful overall mapping: If the agent fails in mapping at least one of the VNs or VLs of a request, it does not matter how many of the partial attempts are successful. The whole request must be dropped. Hence, in this situation, the agent receives a big negative value as the punishment, which is bigger than the partial failure punishment, and we have considered -500 points for it.

Based on these four situation, the reward function of mapping the j^{th} VN of the i^{th} request at time t is $\text{Re}(t, i, j)$, and it is calculated as Equation 17.

$$\text{Re}(t,i,j) = \begin{cases} SP(t,i,j), & \text{If } j \neq N_i \text{ and } m_j^i \neq 0 \\ -1, & \text{If } j \neq N_i \text{ and } m_j^i = 0 \\ SP(t,i,j) + 500, & \text{If } j = N_i \text{ and } \gamma(t,i) = 1 \\ SP(t,i,j) - 500, & \text{Else If } m_j^i \neq 0 \\ -501, & \text{Otherwise} \end{cases} \tag{17}$$

We can summarize Equation 17 as Equation 18, where $\lfloor x \rfloor$ is the floor function of x, that returns the biggest integer lower than or equal to x.

$$\text{Re}(t,i,j) = m_j^i(SP(t,i,j) + 1) + 500\lfloor \frac{j}{N_i} \rfloor (2\gamma(t,i) - 1) - 1 \tag{18}$$

It is worth noting that when $\gamma(t, i)$ is one, all the VNs are completely mapped, and hence, we have $m_j^i \neq 0$. This reward function is a special case of our objective function (Equation 15), where the values of Ψ and Φ are 1 and 500, respectively. The process that an agent performs during the training phase is described in Algorithm 3.

Algorithm 3 The procedure of training the agent in SIRL

```
Require: $S$ (the substrate network)
Require: $episodes$ (the number of training episodes)
Ensure: $model$ (the trained model)
    $model \leftarrow$ initialize the actor-critic model
 2: for $1 \leq e \leq episodes$ do
        $moves \leftarrow 0$
 4:     for $1 \leq i \leq R$ do
            $moves \leftarrow moves + N_i$
 6:     for $1 \leq move \leq moves$ do
            $s \leftarrow$ the environment state from Algorithm 2
 8:         $a \leftarrow$ the optimal action from $model$ in $s$
            $r \leftarrow$ the current request index
10:         $v \leftarrow$ the current VN index of the $r^{th}$ request
            Map the $v^{th}$ VN on the $a^{th}$ SN
12:         if $v$ is the last VN of the $r^{th}$ request then
                Map the virtual links on the shortest paths
14:         $reward \leftarrow \text{Re}(move, r, v)$        ▷ Equation 18
            Update $model$ based on $s$, $a$, and $reward$
16: return $model$
```

D. Time Complexity

Algorithm 1 is the naive form of generating the initial features, and its time complexity is not optimized. At the end of this algorithm, we have a nested loop of $O(N^2)$, in which the values of φ and λ are required. Based on Equation 7 and Equation 9, calculating these values is requires a time higher than $O(N^2)$. As a result, the time complexity of Algorithm 1 is at least $O(N^4)$.

For the implementation phase, we can perform some short-cuts and dynamic programmings in order to reduce the time complexity. The implemented algorithm of the proposed method, containing all the processes, is shown in Algorithm 4. In this algorithm, $\{\}$ is an empty array, and $\{0\}^N$, $\{1\}^N$ and $\{\{\}\}^N$ are arrays which are filled with N zeros, ones, and empty arrays, respectively.

First, the variables that are fixed in each episodes are calculated. The values of $lsum[i]$ and $degree[i]$ indicate the total sum of link bandwidth and the number of adjacent links of the i^{th} SN, respectively. $path[i][j]$ stores the SNs in the shortest path between the i^{th} and the j^{th} SNs. These three variables are calculated in a loop in line 2 to line 7. This loop is of $O(N^3)$, because finding the shortest path from the i^{th} SN to the others (line 7), when the links are stored as an adjacency matrix, is of $O(N^2)$ in Dijkstra, and it must be calculated for all the nodes.

The number of links that are adjacent to the link between the i^{th} and the j^{th} SNs is stored in $adjacent[i][j]$, and it is calculated in a nested loop of $O(N^2)$ in line 8 to line 11. Similarly, the initial importance value of the i^{th} SN is stored in $importance[i]$, and it is calculated in a loop of $O(N^2)$ in line 12 to line 14. The value of $index[i]$ indicates the sum of the number of VNs in the first request to the i^{th} one. The loop in line 15 to line 16, which is of $O(N)$, is for generating this variable.

Now, the actor-critic model is initialized and a loop repeats the training processes $episodes$ times (line 17 to line 51). At the beginning of each iteration, the five feature sets are initialized (line 18), and they will be updated during the training phase. FL stores the available bandwidth of each link. The agent maps a single VN on a SN in each step, and then maps the VLs between the VNs that are currently mapped. Hence, there may be some conditions, in which a VN or a VL of a request cannot be mapped, and the previous mappings of the same request must be unmapped. In this case, the agent has to reset the resources to their condition before the mapping of that request. The varibales in line 19 are for this reason. max and r store the maximum number of VNs mapped on a SN and the index of the current request, respectively. $move$ counts the number of steps the agent has taken, and $mapping$ stores the index of the SNs which are hosting the mapped VNs of the current request and the mapping time.

The whole number of VNs indicates the total number of steps, and therefore, a loop is considered in line 21 to 51 to repeat the processes in each step. The variable of this loop is $move$ and at the end of each iteration, it is incremented by one (line 51). v stores the index of the current VN in the current request, and it is calculated in line 22 and line 23. The value of F_4 must be updated ($O(N)$) at the beginning of each step, to check the SNs that do not have enough CPU capacity for hosting the current VN (line 24 and 25). Then, the set of initial features are passed to Algorithm 2 to get the final features. It is done in line 26, and it is of $O(nlogn)$. Now, the agent estimates the optimal action for the current state, and this action indicates the SN that hosts the current VN.

$hops$ stores the number of edges that are involved in mapping the VLs of the current VN, $success$ indicates whether this mapping is successful (line 28). A nested loop is required for mapping the VLs (line 29 to 35). The outer loop is for iterating on one of the endpoints of the involved edges, and the inner one is for considering all the edges in the shortest path. It must be noted that this part maps the VLs with both endpoints successfully mapped. If one of the edges in the involved shortest paths has a lower available bandwidth than that of the VL, we have an unsuccessful mapping. Otherwise, the values of F_2 and FL are updated by reducing the weight of that VL. This nested loop is of $O(N^2)$, because the outer loop iterates $v - 1$ times, which is at most N due to that fact that the number of VNs in a request cannot exceed the total number of SNs, and the inner loop is repeated for at most N times due to the fact that the length of a path in a graph is at most $N - 1$. The value of F_1 is updated by reducing the consumed CPU capacity, and F_3 is incremented (line 36). The value of max is changed, only if F_3 reaches a greater value than before (line 37), and F_5 is also updated based on the new importance values (line 38).

If the agent performs a successful mapping, regarding the mapping of the current VN and related VLs (line 40), the index of the selected SN is appended to $mapping$ and $reward$ becomes a positive value based on the number of involved hops and the maximum number of VNs mapped on the SNs (line 41). Otherwise, the agent has to reset the consumed resources of the unsuccessful mappings (line 43), skip the steps to the next request, and receive a punishment (line 44). In line 45, the end of processing all the VNs in the current request

Algorithm 4 The implemented procedure of SIRL

```
Require: $\mathcal{S}$ (the substrate network)
Require: $episodes$ (the number of training episodes)
Ensure: $model$ (the trained model)
    $lsum$, $degree$, $importance$, $path$, $adjacent$, $index \leftarrow \{0\}^N$, $\{0\}^N$, $\{0\}^N$, $\{\{\}\}^N$, $\{\{\}\}^N$, $\{N_1\}$
 2: for $1 \le i \le N$ do
        for $1 \le j \le N$ do
 4:         $lsum[i] \leftarrow lsum[i] + l_{(i,j)}$
            if $l_{(i,j)} \ne 0$ then $degree[i] \leftarrow degree[i] + 1$
 6:     Fill $path[i]$ with the shortest paths from the $i^{th}$ SN to all other nodes using Dijkstra's algorithm
    for $1 \le i \le N$ do
 8:     for $1 \le j \le N$ do
            if $l_{(i,j)} \ne 0$ then $adjacent[i] \leftarrow adjacent[i] + \{degree[i] + degree[j] - 1\}$
10:         else $adjacent[i] \leftarrow adjacent[i] + \{0\}$
    for $1 \le i \le N$ do
12:     for $1 \le j \le N$ do
            if $adjacent[i][j] \ne 0$ then $importance[i] \leftarrow importance[i] + (n_i + n_j)/adjacent[i][j]$
14: for $2 \le i \le R$ do
        $index \leftarrow index$ + $\{index[i-1] + N_i\}$
16: $model \leftarrow$ initialize the actor-critic model
    for $1 \le e \le episodes$ do
18:     $F_1$, $F_2$, $F_3$, $F_4$, $F_5$, $FL \leftarrow \mathcal{N}$, $lsum$, $\{0\}^N$, $\{1\}^N$, $importance$, $\mathcal{L}$
        $rF_1$, $rF_2$, $rF_3$, $rF_5$, $rL \leftarrow F_1$, $F_2$, $F_3$, $F_5$, $FL$
20:     $max$, $r$, $move$, $mapping \leftarrow$ 0, 1, 1, $\{\}$
        while $move \le index[R]$ do
22:         if $r = 1$ then $v \leftarrow move$
            else $v \leftarrow move - index[r-1]$
24:         for $1 \le i \le N$ do
                if $F_1[i] < n_v^r$ then $F_4[i] \leftarrow 0$
26:         $state \leftarrow$ pass $\{F_1, F_2, F_3, F_4, F_5\}$ to Algorithm 2 to get final features
            $action \leftarrow$ the optimal action from $model$ in $state$
28:         $hops$, $success \leftarrow$ 0, True
            for $1 \le i < v$ do
30:             if $l^r_{(i,v)} \ne 0$ then
                    for $p \in path[action][mapping[i][1]]$ do
32:                     $start$, $end$, $hops \leftarrow$ the first endpoints of $p$, the second endpoints of $p$, $hops + 1$
                        if $l^r_{(i,v)} > FL[start][end]$ then $success \leftarrow$ False break
34:                     $F_2[start]$, $F_2[end] \leftarrow F_2[start] - l^r_{(i,v)}$, $F_2[end] - l^r_{(i,v)}$
                        $FL[start][end]$, $FL[end][start] \leftarrow FL[start][end] - l^r_{(i,v)}$, $FL[end][start] - l^r_{(i,v)}$
36:         $F_1[action]$, $F_3[action] \leftarrow F_1[action] - n_v^r$, $F_3[action] + 1$
            if $max < F_3[action]$ then $max \leftarrow F_3[action]$
38:         for $1 \le i \le N$ do
                if $adjacent[action][i] \ne 0$ then $F_5[action] \leftarrow F_5[action] - n_v^r/adjacent[action][i]$
40:         if $success$ and $F_4[action] \ne 0$ then
                $mapping$, $m_v^r$, $t_v^r$ $reward \leftarrow mapping + \{action, move\}$, $action$, $move$, $1/(hops \times max)$
42:         else
                $F_1$, $F_2$, $F_3$, $F_5$, $FL \leftarrow rF_1$, $rF_2$, $rF_3$, $rF_5$, $rL$
44:             $m_v^r$, $move$, $reward \leftarrow$ 0, $index[r]$, -1
            if $move = index[r]$ then
46:             if size($mapping$) $= N_r$ then $\mathcal{M}_r$, $reward \leftarrow mapping$, $reward$ + 500
                else $reward \leftarrow reward$ - 500
48:             $r$, $F_4$, $mapping$, $rF_1$, $rF_2$, $rF_3$, $rF_5$, $rL \leftarrow r + 1$, $\{1\}^N$, $\{\}$, $F_1$, $F_2$, $F_3$, $F_5$, $FL$
            else $F_4[action] \leftarrow 0$
50:         Update $model$ based on $state$, $action$, and $reward$
            $move \leftarrow move + 1$
52: return $model$
```

is checked. If the agent has reached the end of a request, and the hosts of all the VNs are added to $mapping$, the whole request is completely mapped, and the agent receives a big reward (line 46). Moreover, the value of $\mathcal{M}_r$ (the mapping solution for the current request) is updated. But, on the other hand, if the request is not completely mapped, a big punishment is received (line 47). The values of the variables are also updated based on the next request in line 48. If processing the current request is not done yet, the value of F_4 is updated, because the other VNs cannot be mapped on the selected SN in this step (line 49). Finally, the actor-critic model is updated based on the received reward in the current state by applying the estimated action (line 50).

In each episode of Algorithm 4, there is a loop that iterates at most $\sum_{i=1}^{R} N_i$ (line 21). The value of N_i, for all $1 \leq i \leq R$, does not exceed N, due to the fact that the number of VNs in a request are at most N. Hence, the maximum number of iterations of this loop is $R \times N$. The most time-consuming part of this loop is mapping the links (line 29 to line 35), which is of $O(N^2)$. As a result, we can say that the time complexity of SIRL for training a single episode is $O(R \times N^3)$, where R is the number of requests and N is the number of SNs.

V. Evaluation

To evaluate the performances of our proposed RL model (i.e., SIRL), we have simulated different network topologies in Python, and PyTorch is used for implementing and training the RL models. First, we have evaluated the superiority of SIRL, as an RL method, with some other non-RL methods for solving the same problem. This evaluation helps determining whether the proposed method outperforms the naive solutions. In this regard, the performance of SIRL is compared with four other approaches: random, greedy, ranking, and genetic approaches. In the random approach, the VNs are randomly mapped on the SNs without any extra considerations. In the greedy approach, we aim to balance the number of mapped VNs among different SNs that have enough resources, in order to reduce the DDoS damage. For the ranking approach, we have applied the solution proposed by Cao et al. [7], where the heuristic function uses the defined features of SIRL (i.e., F_1 to F_5). The genetic approach applies a modification of the solution proposed by Zhang et al. [13]. In this modified form, we have used Equation 18 (our proposed reward function) as the fitness function that must be maximized, and the chromosomes structure is based on SIRL's defined features.

Then, we have compared the performance of SIRL against nine of the existing RL models, which are CDRL [17], RDAM [18], VNEQS [20], MLRL [21], DRLVNE [22], GCNNRL [23], A3CGCN [24], DeepViNE [25], PNVNE [26], and A2CRL [27]. Since our focus is on the features and their ability to model the environment, we have 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 episodes). For VNEQS, we have considered the number of VNs that are mapped on each SN as the security level.

Several random network scenarios are considered in the simulations, and the average results are reported. The number of SNs in the simulated networks vary from 10 to 100, and they have random amount of resources, where the CPU weights are from 10 to 30, and the bandwidth weights are from 5 to 45. The probability of having an edge between two SNs is also 0.3. In the simulated scenarios, several random slice requests arrive at random times and last for a random time. However, not more than 100 requests are active at a time. Each request contains at least 2 VNs and at most one quarter of the number of SNs, and each of them request random resources from 1 to 5. While all the features of the requests are random, the same requests are passed to the models to compare them fairly.

For the actor-critic model, we have set the learning and discount rates to 0.001 and 0.99, respectively. The input layer of the actor's neural network takes the environment state, which contains the extracted features, and its output is forwarded to the first hidden layer with 128 neurons. The second and third hidden layers contain 256 neurons, and the output layer predicts the solutions in terms of the probability of the actions that must be selected. The mentioned layers are fully connected and their activation function is ReLU (Rectified Linear Unit). The critic has similar layers, except for the output layer that only evaluates the actor's predicted action.

We have analyzed the acceptance ratio, the DDoS damage, and the learning performance in the simulation results to fairly evaluate the performance of SIRL. The analysis is presented as follows.

A. *Requests acceptance ratio*

The acceptance ratio is the ratio of the number of successfully mapped requests to the total number of arrived requests. In other words, we can calculate the acceptance ratio as Equation 19, where the value of γ function can be calculated by Equation 3.

$$\text{Acceptance Ratio} = \frac{\sum\limits_{i=1}^{R} \gamma(t,i)}{R} \times 100 \tag{19}$$

The value of the acceptance ratio for a method can determine its performance in mapping the requests. The higher values of the acceptance ratio show better performance. The acceptance ratio can show us our goal achievement in reaching Objective 1.

Before comparing SIRL with other RL models, we have to evaluate its performance against non-RL approaches. Figure 2 shows this comparison. This graph shows how RL models outperform the traditional heuristic and metaheuristic approaches. The average results show that the proposed RL model results in an acceptance ratio, which is 70% higher than non-RL approaches. The reason that the acceptance ratio gets lower as the substrate network becomes larger is due to the increase of VNs in each request. In the simulations, the maximum number of VNs in a request is one-quarter of the number of SNs. Another point about these results is the gap between ranking and genetic approaches, which becomes

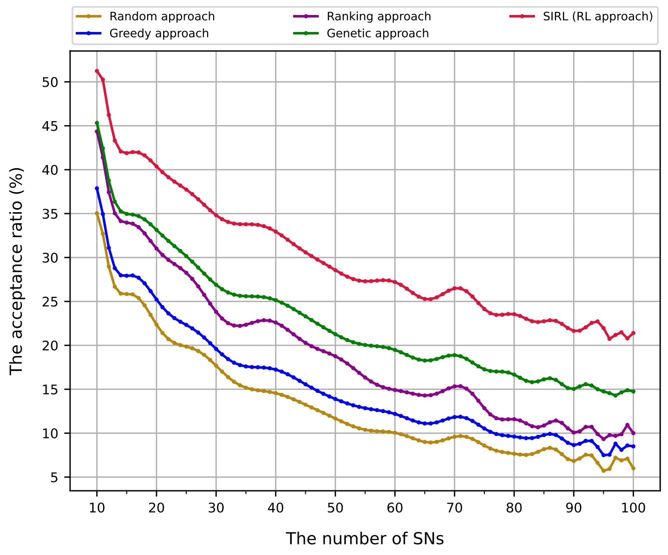


Fig. 2. Comparing SIRL with non-RL approaches in terms of acceptance ratio

larger with the growth of substrate network size. This growth shows the poor performance of heuristic approaches compared to metaheuristic ones in large-scale problems.

Figure 3 illustrates the ability of different RL models in embedding the virtual requests. We can see a big difference between the number of successfully mapped requests in SIRL and the other models. The most significant difference can be seen in the scenarios with larger substrate networks. The acceptance ratio of SIRL is about 54% greater than the average acceptance ratio of the other models. This means that SIRL can map more requests on the SNs than the other models. The model in the next place is A3CGCN, and its acceptance ratio is about 41% lower than that of SIRL. The reason may be that A3CGCN, like SIRL, pays attention to general network features instead of complex details.

B. DDoS damage

When the adversary performs a DDoS attack against the SNs, all the mapped slices on that SN will be affected. Because all the VNs of a request must collaborate, losing one of them leads to the failure of the whole request. As a result, a good mapping solution must map the VNs on the SNs in a way that attacking one of the SNs 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 Objective 2, and its value at time t, considering the adversary's target is the k^{th} SN, can be calculated by Equation 20.

$$\text{DDoS Damage} = \frac{\max\limits_{i \in \mathbb{N}_N} \delta(t,i)}{\sum\limits_{i=1}^{R} \gamma(t,i)} \times 100 \tag{20}$$

We have simulated the networks of all the eight scenarios with the mapping solutions that the models suggest, and then we performed a DDoS attack against the SNs. Some extra nodes are considered out of the substrate network, as the end-hosts,

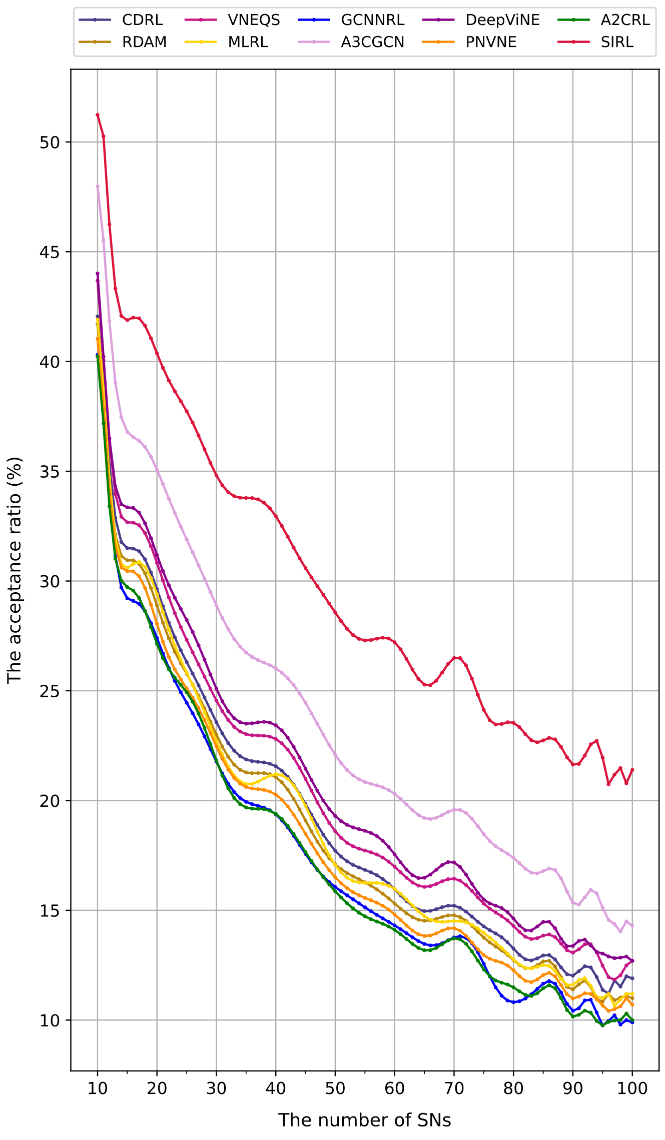


Fig. 3. Comparing SIRL with other RL models in terms of acceptance ratio

which can be compromised by the adversary, and launch a DDoS attack. We have considered that each SN crashed when at least 40 end-host send flooded traffic toward them. In each simulation, the adversary selects a random SN in the network and commands the compromised hosts to launch the attack against it.

For evaluating the DDoS damage, the scenarios with the equal number of accepted requests are classified together. Figure 4 compares SIRL with non-RL approaches in terms of ability to reduce the impact of DDoS attacks. Since the greedy approach tries to balance the number of VNs that are mapped on the SNs, the best results are reported for this approach. There are no other solutions that can reduce the DDoS damage more than the greedy approach. However, among the other approaches shown in this graph, SIRL is the most powerful one. It can reduce the impact of successful DDoS attacks on the embedded requests 19% higher than ranking and genetic approaches. Even though the greedy approach has a good result in terms of reducing the DDoS damage, it is weak

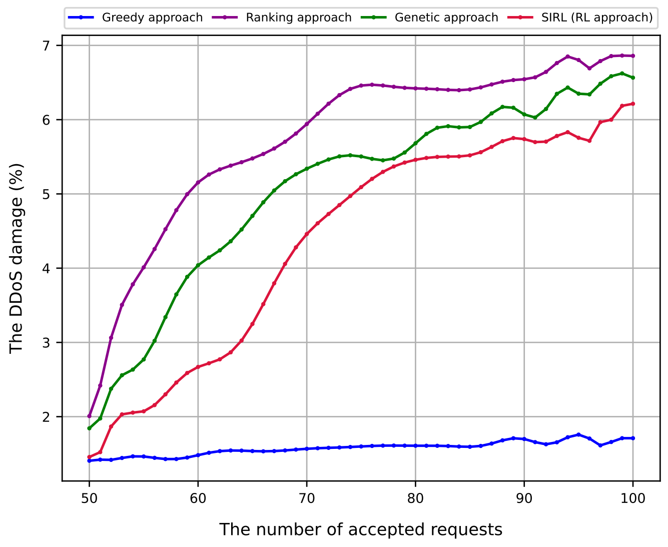


Fig. 4. Comparing SIRL with non-RL approaches in terms of DDoS damage

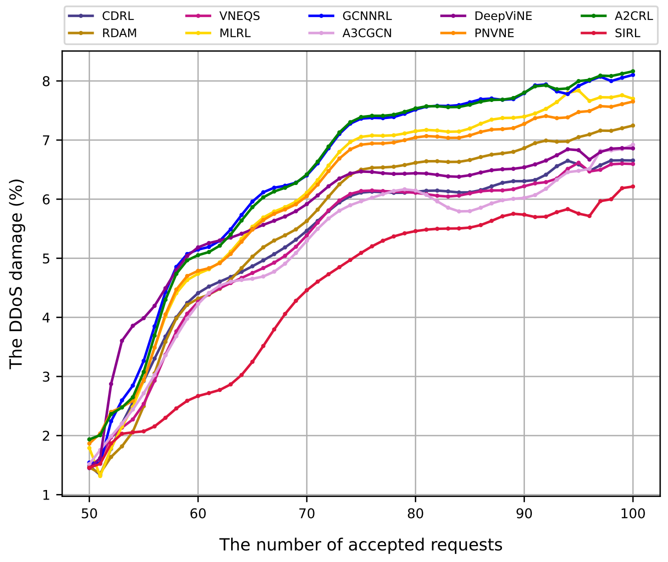


Fig. 5. Comparing SIRL with other RL approaches in terms of DDoS damage

in increasing the number of accepted requests, according to Figure 2. As a result, combining the results reported for both acceptance ratio and DDoS damage metrics, the proposed RL approach outperforms the non-RL techniques.

The average results of the attacks and their effect on the network are reported for different RL models in Figure 5. We can see that the average number of slices that are crashed after the attack utilizing SIRL is lower than in the other cases. The results show that the attacks cause about 23% lower damage to the slices when they are mapped on the substrate network using the suggestions of SIRL. 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 this graph is ascending, because as the number of requests grows, more VNs have to be mapped on a single SN, and it increases the attack damage.

C. *Learning performance*

During the environment exploration, if the agent faces a state that has been explored before, the predicted optimal action will be improved. Hence, the learning speed improves as the number of repeated states in a network becomes higher. Moreover, if the number of same states for modeling two different networks is high, once trained with one of them, the agent can also find a suitable solution for the other. Figure 6 compares the average number of repeats of the states in a network and also between this sample network and another one, where $N = 10$. A complete episode is executed, and then the repeats are counted. In the first scenario, the states are compared with each other in the same network, but in the second one, the states of two different networks are considered. We can see that the number of state repeats in both scenarios for SIRL is greater than the other models. The results of the first and second scenarios for SIRL are about three and six times greater than other models, respectively. The difference between the results of the first and second scenarios in SIRL is lower than that of the others by 36%. This is the sign of SIRL's generality.

Figure 7 shows the acceptance ratio of the models that are trained and tested with the same or different networks. These results claim the generality of SIRL, as its acceptance ratio when trained and tested with different networks is 97% higher than that of the other models.

Figure 8 illustrates the scores that the agent achieves during the training phase of different models in the scenarios with 20 SNs. This graph shows that the agent in SIRL finds the optimal solution after a lower number of episodes than in other models. This is because the number of states to be explored are lower in SIRL. The agent in GCNNRL is slower in terms of the number of episodes for finding the solution. The other point about this graph is the maximum awarded scores of different models. The agent in SIRL can achieve higher scores for solving VNE based on our defined reward function.

VI. Conclusion

Mapping the slice requests on the beyond 5G networks with a high request acceptance ratio is challenging, especially when the security of these slices matters. This paper proposes a novel actor-critic RL model, SIRL, that aims to find the best SNs for hosting the VNs according to the mapping performance and slices protection against DDoS attacks. Five network features are considered to train the agent, three of which are not defined in the previous RL models. Moreover, SIRL reduces the number of possible states of the environment by ranking the SNs. This reduction plays an important role in improving the learning performance. We also conducted several simulations to evaluate and compare the performance of SIRL and nine other RL models. The obtained results showed that SIRL could increase the acceptance ratio by 54% on average. It can also reduce the DDoS damage by 23% on average.

Since the substrate network is a graph, and each graph contains many topological features, we plan to define more powerful topological features that can improve SIRL performance in future research work. Hypergraphs generalize

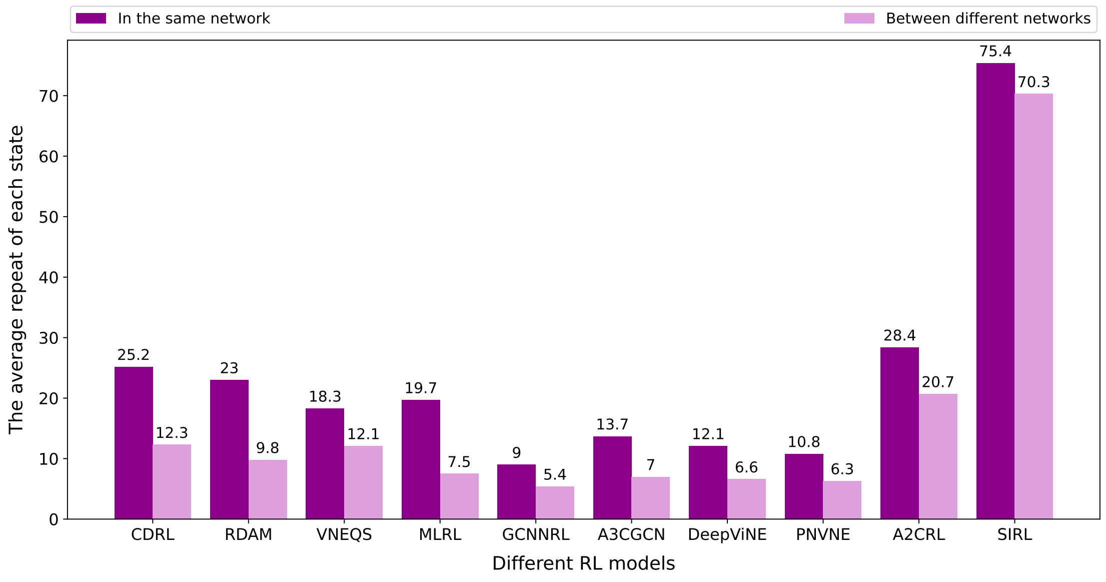


Fig. 6. Comparing the average number of repeated states in different RL models

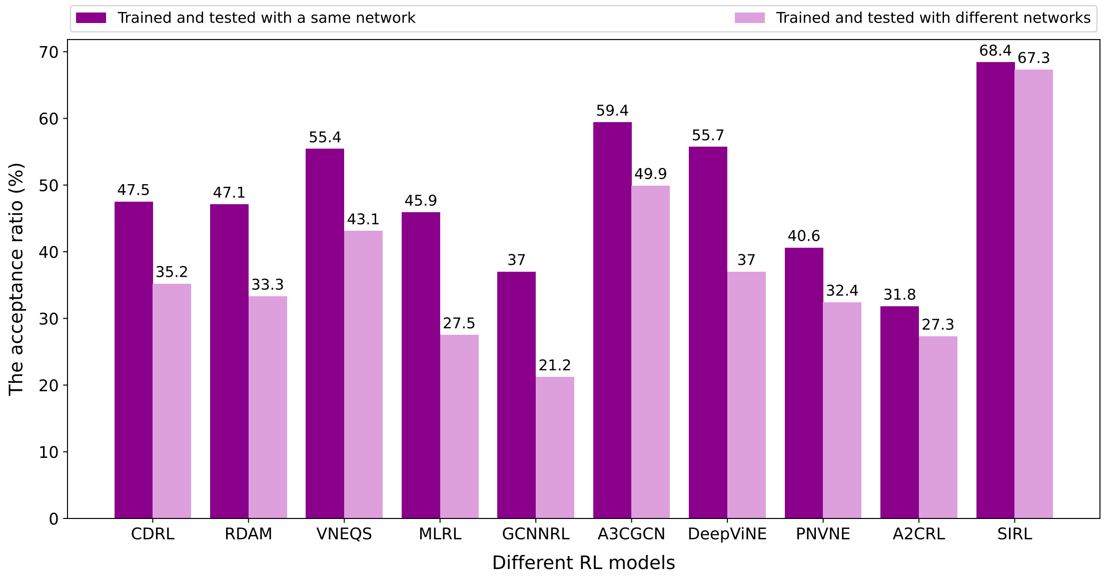


Fig. 7. Comparing the models that are trained and then tested with the same or different networks

the normal graphs, where an edge can connect two sets of vertices. We have planned to model the substrate network with a hypergraph, and then pass its features to the RL agent. Moreover, we intend devising a new algorithm to be executed after the model is trained, which can reduce the DDoS damage. In this algorithm, when the flooding attack is launched, the VNs that are mapped on the target SN, are migrated to another SN considering the resource limitations. This work can also be improved by analyzing different ways of mapping the VLs on the SLs. In this work, we have considered the shortest path between two SNs for mapping the related VLs. However, one can improve the acceptance ratio by mapping the VLs on the other paths, if the shortest path does not have enough resources, and even by dynamically changing the VL mappings in order to avoid congestion.

Acknowledgment

This work was supported in part by the EU's Horizon 2020 research and innovation programme under the INSPIRE-5Gplus project (Grant No. 871808), the EU's HE research and innovation programme HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933), the Academy of Finland IDEA-MILL project under Grant No. 352428, and the Academy of Finland 6Genesis Flagship Project (Grant No. 346208).

References

[1] B. Han, J. Lianghai, and H. D. Schotten, "Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5g networks," *ieee access*, vol. 6, pp. 33 137–33 147, 2018.
[2] R. F. Olimid and G. Nencioni, "5g network slicing: A security overview," *IEEE Access*, vol. 8, pp. 99 999–100 009, 2020.

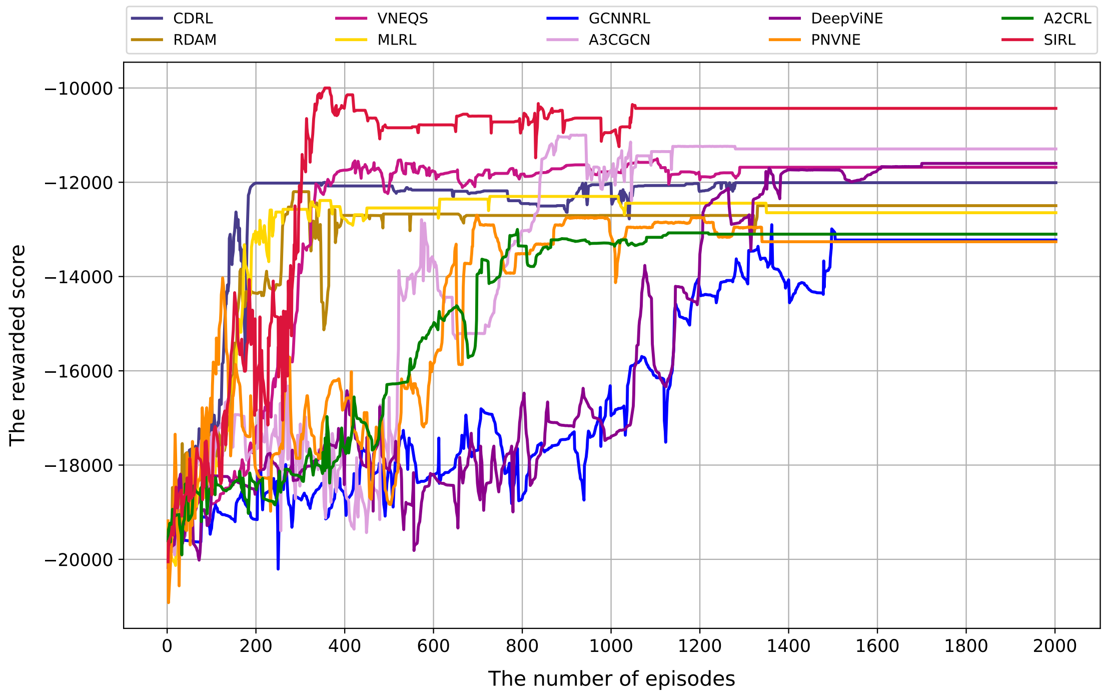


Fig. 8. Comparing the rewarded score of the agent in different RL models

[3] D. Sattar and A. Matrawy, “Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices,” in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 82–90.

[4] V. N. Sathi, M. Srinivasan, P. K. Thiruvasagam, and S. R. M. Chebiyyam, “A novel protocol for securing network slice component association and slice isolation in 5g networks,” in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2018, pp. 249–253.

[5] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, and P. Kankariya, “Secure5g: A deep learning framework towards a secure network slicing in 5g and beyond,” in *2020 10th annual computing and communication workshop and conference (CCWC)*. IEEE, 2020, pp. 0852–0857.

[6] 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.

[7] H. Cao, L. Yang, and H. Zhu, “Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 108–120, 2017.

[8] H. Cao, Y. Guo, Y. Hu, S. Wu, H. Zhu, and L. Yang, “Location aware and node ranking value-assisted embedding algorithm for one-stage embedding in multiple distributed virtual network embedding,” *IEEE Access*, vol. 6, pp. 78 425–78 436, 2018.

[9] A. Hashmi and C. Gupta, “Vne-nr: A node-ranking method for performing topology-aware and resource-driven virtual network embedding,” in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2020, pp. 1–6.

[10] C. Shi, X. Meng, Q. Kang, and X. Han, “Novel node-ranking approach for sdn-based virtual network embedding,” *Mathematical Problems in Engineering*, vol. 2020, 2020.

[11] C. Aguilar-Fuster and J. Rubio-Loyola, “A novel evaluation function for higher acceptance rates and more profitable metaheuristic-based online virtual network embedding,” *Computer Networks*, vol. 195, p. 108191, 2021.

[12] A. Song, W.-N. Chen, T. Gu, H. Yuan, S. Kwong, and J. Zhang, “Distributed virtual network embedding system with historical archives and set-based particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 927–942, 2019.

[13] P. Zhang, X. Pang, G. Kibalya, N. Kumar, S. He, and B. Zhao, “Gcmd: Genetic correlation multi-domain virtual network embedding algorithm,” *IEEE access*, vol. 9, pp. 67 167–67 175, 2021.

[14] L. Zhuang, G. Wang, M. Wang, and K. Zhang, “A virtual network embedding algorithm based on cellular automata genetic mechanism,” in *MATEC Web of Conferences*, vol. 232. EDP Sciences, 2018, p. 01019.

[15] Y. Yuan, Z. Tian, C. Wang, F. Zheng, and Y. Lv, “A q-learning-based approach for virtual network embedding in data center,” *Neural Computing and Applications*, vol. 32, no. 7, pp. 1995–2004, 2020.

[16] H. K. Thakkar, C. K. Dehury, and P. K. Sahoo, “Muvine: Multi-stage virtual network embedding in cloud data centers using reinforcement learning-based predictions,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1058–1074, 2020.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] M. Dolati, S. B. Hassanpour, M. Ghaderi, and A. Khonsari, "Deepvine: Virtual network embedding with deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 879–885.

[26] 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.

[27] 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.

[28] M. Settembre, "A 5g core network challenge: Combining flexibility and security," in *2021 AEIT International Annual Conference (AEIT)*. IEEE, 2021, pp. 1–6.

[29] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[30] A. Asghari and M. K. Sohrabi, "Bi-objective cloud resource management for dependent tasks using q-learning and nsga-3," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–21, 2022.

Amir Javadpour Amir Javadpour obtained his MSc degree in Medical Information Technology Engineering from the University of Tehran, Iran, in 2014. From Guangzhou University, China, he received a Ph.D. in Computer Science (Mathematics/Cybersecurity) in 2020. In addition, he has published papers with his colleagues in highly ranked journals and several ranked conferences on several topics, including Cloud Computing, Software-Defined Networking (SDN), Big Data, Intrusion Detection Systems (IDS), and the Internet of Things (IoT), Moving Target Defence (MTD), Machine Learning (ML), and optimization algorithms. Additionally, he reviewed papers for several reputable venues such as IEEE Transactions on Cloud Computing, IEEE Transactions on Network Science and Engineering, ACM Transactions on Internet Technology, the Journal of Supercomputing, several journals of Springer and Elsevier, etc. and he is also the Technical Program Committee (TCP) member of various conferences.

Forough Ja'fari is a Senior Researcher in cybersecurity and computer science. She received her Bachelor's degree from Sharif University of Technology and her Master's degree in Computer Network Engineering from Yazd University, Iran. She is a visiting scholar researcher at Guangzhou University, China. Software-Defined Networking (SDN), Intrusion Detection Systems (IDS), Internet of Things (IoT), Moving Target Defence (MTD), and Machine Learning are some of her research interests. She is currently a Guest Editor (GE) of Cluster Computing (CLUS) Journal, as well as a reviewer for several journals and conferences.

Tarik Taleb (Senior Member, IEEE) is currently a Professor at the Center of Wireless Communications, The University of Oulu, Finland. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). He is the founder and board chair of ICTFicial Oy. Between Oct. 2014 and Dec. 2021, he has been a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He was then leading the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and Ph.D. degrees in Information Sciences from Tohoku Univ., in 2001, 2003, and 2005, respectively. His research interests lie in the field of telco cloud, network softwarization & network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, & next generation mobile networking. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. Prof. Taleb served on the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Prof. Taleb founded the "IEEE Workshop on Telecommunications Standards: from Research to Standards", a successful event that got awarded "best workshop award" by IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb has also founded and has been the steering committee chair of the IEEE Conf. on Standards for Communications and Networking.

Chafika Benzaïd is currently a senior research fellow at University of Oulu, Finland. Between Nov. 2018 and Dec. 2021, she was senior researcher at Aalto University. Prior to that, she was working as an associate professor at University of Sciences and Technology Houari Boumediene (USTHB). She holds Engineer, Magister and "Doctorat ès Sciences" degrees from USTHB. Her research interests lie in the field of 5G/6G, SDN, Network Security, AI Security, and AI/ML for zero-touch security management. She is an ACM professional member.