A Reinforcement Learning Based Approach for 5G Network Slicing Across Multiple Domains

Godfrey Kibalya*, Joan Serrat*, Juan-Luis Gorricho*, Rafael Pasquini†, Haipeng Yao‡, and Peiying Zhang§
*Dept.Network Engineering, Universitat Politecnica de Catalunya,Barcelona, Spain
†Computing Faculty, Federal University of Uberlandia, Uberlandia, Brazil
‡State key Lab. of Networking and Switching Technology, Beinjing University of Posts and Telecommunications, Beijing, China
§College of Computer and Communication Engineering,China University of Petroleum (East China), Qingdao, China
E-mails: {Godfrey.mirondo.Kibalya@upc.edu, serrat@tsc.upc.edu, juanluis@entel.upc.edu, rafael.pasquini@ufu.br, yaohaipeng@bupt.edu.cn, 25640521@qq.com}

***Abstract*—Network Function Virtualization (NFV) and Machine Learning (ML) are envisioned as possible techniques for the realization of a flexible and adaptive 5G network. ML will provide the network with experiential intelligence to forecast, adapt and recover from temporal network fluctuations. On the other hand, NFV will enable the deployment of slice instances meeting specific service requirements. Moreover, a single slice instance may require to be deployed across multiple substrate networks; however, existing works on multi-substrate Virtual Network Embedding fall short on addressing the realistic slice constraints such as delay, location, etc., hence they are not suited for applications transcending multiple domains. In this paper, we address the multi-substrate slicing problem in a coordinated manner, and we propose a Reinforcement Learning (RL) algorithm for partitioning the slice request to the different candidate substrate networks. Moreover, we consider realistic slice constraints such as delay, location, etc. Simulation results show that the RL approach results into a performance comparable to the combinatorial solution, with more than 99% of time saving for the processing of each request.**

***Index Terms*—Multi-substrate VNE, Reinforcement learning, Multi-domain slicing, 5G.**

I. Introduction

The transition towards full network virtualization will see services being instantiated as Network Slice Instances (NSIs), realized in the form of logical and self-contained networks, consisting of both shared and dedicated resources, including Virtual Network Functions (VNFs) [1], [2]. These VNFs can be launched, placed, and scaled flexibly to meet fluctuating workload demands [2], [3], [4].

A major challenge for the slicing scenario is the deployment of NSIs across multi-provider physical infrastructures. In this case, the NSI is realized as a concatenation of slice parts hosted by different administrative domains [5]. In general, the multi-domain slicing problem can be broken into four sub-problems: Firstly, candidate search which involves identifying a set of Infrastructure Providers (InPs) that can serve the slice request individually or as a combined set; Secondly, slice request partitioning that involves deciding which part of the request each candidate InP will serve in order to maximize a given objective e.g cost while meeting the slice constraints; Thirdly, binding and resource allocation to the slice parts associated with each InP; Fourthly, life-cycle management which involves dynamically adapting the allocated resources and VNFs to the observed real time slice performance.

The multi-domain slicing problem is related to the multi-substrate VNE problem which consists of the resource matching stage, the Virtual Network Request (VNR) partitioning stage, the embedding stage and the binding stage. This problem is studied in [6] - [9]. However, different from these and other existing works, we propose an InP search algorithm which exploits disclosed information such as delay along inter-domain links to reduce the search space of candidate InPs for the subsequent stages, hence significantly reducing the request processing delays. Moreover, the search algorithm is able to identify and reject any unfeasible request, hence avoiding partitioning and embedding delays due to such requests at later stages. Unlike existing works, we precede the partitioning stage by an internal embedding check which eliminates the possibility of request being rejected after partitioning stage due to intra-domain link and node violations. Moreover, we associates each request with realistic constraints such as delay. Reinforcement learning is introduced in the VNE problem in [10]. However, besides being applied on a single substrate network, this work did not address practical VNR constraints in terms of location and delay.

Consequently, our novel contribution is threefold:

1) In order to identify a feasible set of InPs that are capable of serving a request, we propose a Candidate Search Algorithm (CaSA) that jointly considers the network topology attributes and the slice request constraints to identify these InPs. Moreover, the proposed algorithm is able to identify and reject all unfeasible requests.
2) A deep reinforcement learning (DRL) algorithm for selecting an optimal set of InPs among all the feasible candidates in order to maximize the revenue to cost ratio for deploying the slice requests. Simulation results show that the proposed algorithm has low time complexity.

This work has received funding from the European Union's Horizon 2020 re-search and innovation programme under grant agreement No 777067 (NECOS pro-ject). This work is also funded by the national project TEC2015-71329-C2-2-R (MINECO/FEDER).

978-3-903176-24-9

Algorithm 1 Filtering step algorithm

Input:G^U,G^s_n

Output: Filtered Candidate set,$cand^{Req^z}_{filter}$

1: **procedure** VIRTUAL NODE CANDIDATE ASSIGNMENT(LoopN^V)
2: Initialise: $cand^{Req^z}_{filter}$=$\emptyset$ ▷ Initialise lists
3: Assign index to each virtual node.
4: **for** Each virtual node $j \in N^V$ **do**
5: $loc^k_{j,Inp} = \emptyset$ ▷ Initialise location set
6: **for** Each Inp $m \in M$ **do**:
7: **if** $loc(j) + dev(j) \in Rad^m_{Inp}$ **then** ▷ Verify location
8: Add m to $loc^k_{j,Inp}$
9: **end if**
10: **end for**
11: **end for**
12: **for** Each virtual node $j \in N^V$ with Index $K > 0$ **do**
13: **for** Each Inp $y \in M$ in $loc^{k-1}_{i,Inp}$ **do** ▷ Inp for preceding vn
14: **for** Each Inp $m \in M$ in $loc^k_{j,Inp}$ **do** ▷ Inp for current vn
15: path= $findpath(y, m)$ ▷ Find path between y and$m$
16: -Verify if path delay,bandwidth and hops meet constraints
17: on virtual link $i - j$ ▷ check virtual link constraints
18: **end for**
19: -Remove any infeasible InP m from $loc^k_{j,Inp}$
20: **end for**
21: -Remove any infeasible InP y from $loc^{k-1}_{i,Inp}$
22: Append $loc^{k-1}_{i,Inp}$ to $cand^{Req^z}_{filter}$
23: **end for**
24: Return $cand^{Req^z}_{filter}$
25: **end procedure**

3) Unlike other works, we treat the entire multi-substrate slicing problem as a single coordinated problem where the results of one stage are the inputs to the next stage.

The rest of the paper is organized as follows: Section II presents the proposed CaSA and the combinatorial algorithms. Section III presents the DRL algorithm. The performance evaluation is presented in Section IV. Section V presents the conclusion and future work.

II. PROPOSED HEURISTICS

A. Candidate Search Algorithm

The Candidate Search (CaSA) algorithm aims at identifying all InPs that satisfy the end-to-end constraints of the slice request either individually or as a combined set. The algorithm is executed in two steps, first a filtering step and then the intra-domain VNE enumeration step as discussed below.

1) Filtering Step: This step matches the virtual nodes' location constraints to each InP's coverage, and the virtual links constraints to the inter-domain links' attributes. This is done since a virtual node can only be served by an InP whose coverage satisfies the virtual node location requirement. Similarly, for two successive virtual nodes j and k to be served by InP a and InP b, where $a \neg b$, there must exist an inter-domain path between InP a and InP b, that satisfies the constraints on virtual link $j-k$. Any InP not satisfying any of the above conditions is eliminated. This reduces candidate InPs for the subsequent step of intra-domain VNE enumeration.

The pseudocode for the filtering algorithm is shown in Algorithm 1 with the notations indicated in table 1. The request virtual nodes are assigned indices from zero to $N-1$,

TABLE I
NOTATIONS AND VARIABLES

Notation	Description
G^s_n	undirected graph for substrate network of InP m
G^U	Inter-domain connectivity graph
Rad^m_{Inp}	coverage area of InP m
N^S	set of all substrate nodes for of InP m
L^S	set of all substrate links for of InP m
N^V	set of all virtual nodes of a request
L^V	set of all virtual links of a request
$loc(j)$	preferred location of virtual node j
$dev(j)$	Maximum deviation from preferred location of j
$vn\ ID$	virtual node Index
$loc^k_{j,Inp}$	set of all InPs for whose Rad^m_{Inp} includes desired location of vn j
$cand^{Req^z}_{filter}$	Set of candidate InPs for request k after filtering stage

where N is the total number of virtual nodes. Next, for each virtual node j with index k, we generate a location set $loc^k_{j,Inp}$, consisting of all InPs whose coverage radius includes the desired location of j (line 3-7). Next, we match the virtual link constraints as follows: For each virtual node j with index $k > 0$ (i.e starting with second node whose k=1), compute the path (shortest path in this work) between each of the InPs in the set $loc^{k-1}_{i,Inp}$ of the immediate preceding node i to each of the InPs in the set $loc^k_{j,Inp}$ of current node j with index k (line 11-14). For each path, we verify if the path delay, bandwidth and number of external hops satisfy the constraints for virtual link $i-j$ (line 15). In case there is any $InP\ m$ in $loc^{k-1}_{i,Inp}$ or $loc^k_{j,Inp}$ that does not result into any feasible path with all InP combinations in $loc^{k-1}_{i,Inp}$ and $loc^k_{j,Inp}$, then the $InP\ m$ is removed from the candidate set for the corresponding virtual node (line 18-20). This procedure is repeated until all virtual nodes are enumerated with the current virtual node becoming the preceding virtual node in the subsequent round. If any location set is empty, the entire request is rejected.

2) Intra-domain VNE enumeration step: The filtering step associates each virtual node with a set of possible InPs by virtue of link constraints and node location bounds. The VNE enumeration step then verifies if within each of those InPs, there is a feasible substrate node for this virtual node. In case there is no such a node, then this InP is removed from the location set of this virtual node. Moreover, at the filtering step, in case any two successive virtual nodes in the request are associated to the same InP, then there must be a feasible substrate path between the substrate nodes of these virtual nodes within this InP. Upon executing the VNE enumeration, the remaining candidate InPs within the location set of each virtual node constitute the output of the search stage which is the input into the partitioning stage. In case any virtual node can not find a substrate node across all InPs, the entire request is rejected. Observe that the intra-domain VNE enumeration is executed for one InP at a time. Consequently, any of the existing single substrate VNE algorithms such as [11] and [12], can be used for this purpose.

B. Combinatorial VNR partitioning algorithm

From the CaSA algorithm, a single virtual node can be associated with multiple InPs. The partitioning algorithm therefore decides the final InP to serve each virtual node with the aim of optimizing a given objective function while respecting the end-to-end slice constraints [6], [8]. The combinatorial algorithm, whose performance we compare with the DRL algorithm, exploits the results from the CaSA algorithm to generate all possible partitioning alternatives. From these, the feasible solutions meeting the slice constraints are sorted according to the objective to be optimized, and the partitioning with the best value is adopted. As can be expected, such a technique results into the optimal decision at each stage, since it explores all possibilities. At the binding stage, the virtual nodes and links are assigned to the substrate nodes and links selected by the partitioning algorithm .

III. Reinforcement Learning Algorithm

In this section, we discuss the proposed DRL algorithm with focus on the feature extraction, neural network architecture and training. We used the reinforcement learning module to perform the slice partitioning task of the problem.

A. Feature extraction

The features extracted for the policy network reflect the attributes of both the slice request and substrate network. The extracted features are highlighted below:

1) Mapping probability which denotes a fraction of the total request nodes and links that were associated to a given InP during the candidate search stage.
2) Average link bandwidth of the InP's inter-domain links to all candidate InPs of the preceding virtual node.
3) Average link delay of current InP's inter-domain links to the candidate InPs of the preceding virtual node.
4) Average number of hops per link of current InP's inter-domain links to the candidate InPs of the preceding virtual node.
5) Success probability which captures the number of virtual nodes of the request allocated to this InP until now. The greater the number of vns associated with a single InP, the less number of inter-domain connections, hence the less embedding costs.

B. Neural Network architecture

The policy neural network takes as input an $M \times N$ feature matrix, where M is the number of InPs and N is the number of extracted features per InP. This network consists of 4 major layers, that is: input layer, convolutional layer, softmax layer and filtering layer. The convolutional layer performs a convolution between the feature matrix from the input layer and the learnable weight values of the filter to produce an $M - dimensional$ vector vector of values , where M is the number of InPs. These are converted into an $M-dimensional$ vector of probabilities by the softmax layer. The filtering layer filters out all InPs that are not capable of meeting the request constraints. Once such InPs are filtered out, the final InP for a given virtual node is chosen as the one with the highest probability.

C. Training phase

To train the neural network, we used offline demand sets of size 500 requests per epoch with the request delay uniformly distributed between 1 to 200 units. For each training request, we generate the feasible InPs using the CaSA algorithm. Then, for each virtual node for which we want to identify the final InP, we generate its corresponding feature matrix which is fed as an input to the policy network which associates a probability to each InP for embedding this virtual node. However, since the neural network parameters are initially randomly assigned, we perform a trade off between exploration and exploitation during training. For each virtual node, the gradients of the policy neural network are computed using back propagation and resulting gradients stacked. If the entire request is embedded successfully, we compute the resulting revenue to cost ratio as the reward signal, otherwise the stacked gradients are deleted. The final gradient of the entire request is then computed as:

$$g := \alpha.r.g_s \tag{1}$$

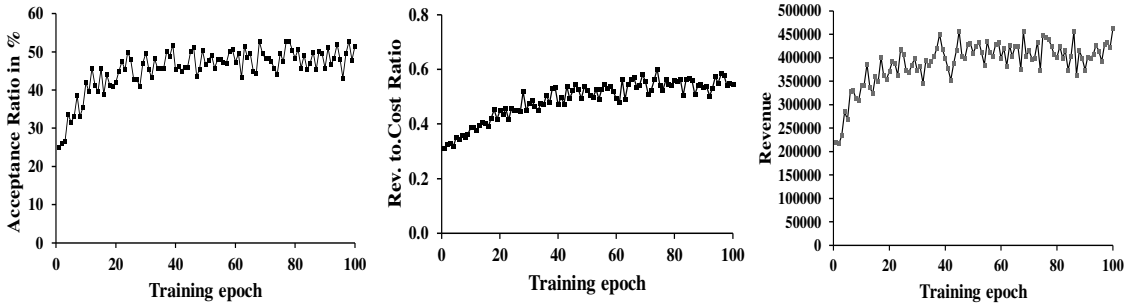
where α is the learning rate, r is the reward and g_s are the stacked gradients. The resulting gradients from the different requests are stacked into buffer until the number of successful requests is equal to batch size. Then all the stacked gradients are jointly applied to the model and the stack buffer is emptied. The performance of the neural network during the training phase is shown in Figure 3 for training duration of 100 epochs.

IV. Performance Evaluation

We compare the performance of the combinatorial and RL schemes that respectively use the combinatorial algorithm and the trained neural network for partitioning the slice request. Both schemes use the same algorithms for candidate search and embedding.

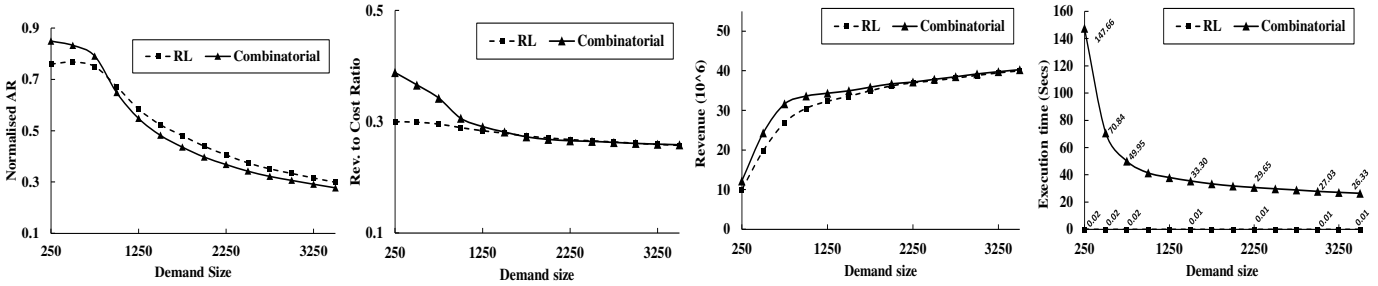
A. Results and discussion

We consider an online scenario where the request arrival follows a Poison distribution with an arrival rate (λ) of 5 requests per 100 unit times, for a total of 70,000 units of time. The performance is analyzed along the different unit times as shown in Figure 4. The values of the different parameters used in simulation are shown in Table II. For both schemes, we assume that whenever a given request is not served, that request leaves the system. From Figure 4(a), the combinatorial scheme initially has a higher Acceptance Ratio (AR) compared to the RL scheme. However, as more requests arrive, the latter gives a better performance. This is so because at each partitioning instant, the combinatorial scheme selects the partitioning with the smallest number of links to minimize embedding costs. This minimizes the link resource usage hence inducing a higher AR but my create bottlenecks on shorter paths,hence low AR in the long run. The RL scheme is able to balance the number of used links and the available



(a) Training acceptance ratio (b) Training revenue to cost ratio (c) Training revenue

Fig. 1. Training Performance



(a) Acceptance ratio (b) Revenue to cost ratio (c) Long term revenue (d) Processing time

Fig. 2. Performance along unit time (λ=5)

TABLE II
SIMULATION PARAMETERS

Substrate Network:	
parameter	Value
No. InPs	8
Nodes per InP	20
Inter-provider bw	unif distrib.[50,200]
Inter-provider delay	unif distrib.[1,100]
Inter-provider link cost	unif distrib.[1,10]
Intra-provider bw	unif distrib.[50,200]
Intra-provider link cost	unif distrib.[1,5]
Intra-provider link delay	unif distrib.[1,20]
Inp Span	30 units
Inp deployment boundaries	250 X 250 square area
Inp connectivity probability	0.4
Node cpu	unif distrib.[50,200]
substrate node connectivity prob	0.5
Slice Request:	
No.Virtual nodes	uni.distrib.[2,10]
Node cpu	uni.distrib.[1,20]
Bandwidth demand	uni.distrib.[1,50]
Bandwidth delay	uni.distrib.[50,200]
Max.hops	uni.distrib.[1,5]
Mean arrival rate	uni.distrib.[2,10]
Life-time	500 (mean)

link resources, which guarantees a long term AR. A similar trend is observed in terms of revenue to cost ratio and long term revenue shown in Figure 4(b) and 4(c) respectively.

From Figure 4(d), the average processing time per admitted request for the combinatorial scheme is shown to decrease with time. This is because as more requests arrive, the candidate InPs for serving the request decrease due to reduced inter-domain link resources, consequently leading to a reduction in the enumeration space for the partitioning stage. For the RL scheme, the processing time is relatively constant since the input features of the policy network is constant irrespective of the demand size.

V. CONCLUSION

This paper addressed the problem of network slicing across multiple domains. Particularly, we have presented a search algorithm that jointly exploits the network attributes and slice request specifications to identify a feasible set of domains onto which to deploy a slice request. Moreover, the algorithm is able to identify and reject all unfeasible requests. Additionally, we proposed an RL algorithm able to partition a slice request in a fraction of a second irrespective of the demand size.

For future work, we consider multi-domain slicing under limited information disclosure. Moreover, we explore the applicability of machine learning to adaptively scale the resources allocated to a slice across different domains basing on the real time resource utilization. We believe that, the integration of self-learning techniques will lead to the realization of networks that are adaptive to temporal network fluctuations.

REFERENCES

[1] M. Leconte, G. Paschos, P. Mertikopoulos, U. Kozat, "A Resource Allocation Framework for Network Slicing", IEEE International Conference on Computer Communications (INFOCOM 2018), Apr. 2018.

[2] Y. Choi and N. Park, "Slice architecture for 5G core network," 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, 2017, pp. 571-575. doi: 10.1109/ICUFN.2017.7993854

[3] K. Samdanis, S. Wright et al. "5G Network Slicing - Part 1: Concepts, Principles, and Architectures," in IEEE Communications Magazine, vol. 55, no. 5, pp. 70-71, May 2017. doi: 10.1109/MCOM.2017.7926919

[4] Nguyen, Van-Giang et al. "SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey." IEEE Communications Surveys and Tutorials 19 (2017): 1567-1602.

[5] Description of Network Slicing Concept,[Online] Available: $https://www.ngmn.org/fileadmin/user\_upload/161010\_NGMN\_Network\_Slicing\_framework\_v1.0.8.pdf$

[6] D. Dietrich, A. Rizk, P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure", IEEE Trans. Netw. Service Manag., vol. 12, no. 2, pp. 188-201, Jun. 2015.

[7] F. Samuel, M. Chowdhury, R. Boutaba, "PolyViNE: Policy-based virtual network embedding across multiple domains", J. Internet Services Appl., vol. 4, no. 1, pp. 6, 2013.

[8] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," Comput. Netw. , vol. 55, no. 4, pp. 1011–1023, 2011.

[9] F. Boutigny, S. Betge-Brezetz et. "Multi-Provider Secure Virtual Network Embedding," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2018, pp. 1-5.doi: 10.1109/NTMS.2018.8328706

[10] H. Yao, X. Chen, M. Li, P. Zhang, and L. Wang, "A novel reinforcement learning algorithm for virtual network embedding," Neurocomputing, vol. 284, pp. 1–9, Apr. 2018.

[11] ViNEYard: Virtual netwok embedding Algorithms with coordinated node and link mapping .used for vnr model

[12]] K. Hejja, X. Hesselbach, Power aware coordinated virtual network embedding with 5g delay constraint, J. Netw. Comput.Appl. (2018), https://doi.org/10.1016/j.jnca.2018.10.005